Cryptosystems for Secure and Efficient Cloud Services

From Key Management, Secure Computing, and Search Functionality

Chen, H.

DOI
10.4233/uuid:6846fff3-cfff-4cf8-8636-4ffc7087a93b

Publication date
2025

Document Version
Final published version

Citation (APA)
Chen, H. (2025). *Cryptosystems for Secure and Efficient Cloud Services: From Key Management, Secure Computing, and Search Functionality*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:6846fff3-cfff-4cf8-8636-4ffc7087a93b

Cryptosystems for Secure and Efficient Cloud Services

From Key Management, Secure Computing, and Search Functionality

Huanhuan Chen

CRYPTOSYSTEMS FOR SECURE AND EFFICIENT CLOUD SERVICES

FROM KEY MANAGEMENT, SECURE COMPUTING, AND SEARCH FUNCTIONALITY

CRYPTOSYSTEMS FOR SECURE AND EFFICIENT CLOUD SERVICES

FROM KEY MANAGEMENT, SECURE COMPUTING, AND SEARCH FUNCTIONALITY

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus *[titles, name]*
chair of the Board for Doctorates
to be defended publicly on
Wednesday 3 September 2025 at 10:00 o'clock

by

Huanhuan CHEN

Master of Science in Pure Mathematics,
Nankai University, China,
born in Nanyang, Henan Province, China

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr.ir. R.L. Lagendijk,	Delft University of Technology, *promotor*
Dr. K. Liang,	Delft University of Technology, *copromotor*
Independent members:	
Prof. dr.ir. F.A. Kuipers	Delft University of Technology
Prof. dr. C.A. Boyd	Norwegian University of Science and Technology, Norway
Prof. dr. L. Chen	University of Surrey, United Kingdom
Dr.ir. J.H. Weber	Delft University of Technology
Prof. dr. G. Smaragdakis	Delft University of Technology (reserve member)

Keywords: Lattice-Based Encryption, Updatable Encryption, Fully Homomorphic Encryption, Searchable Encryption

Printed by: Ipskamp Printing

Cover by: Huanhuan Chen & Xiaodan Zhang. Original image from Canva.com

ISBN 978-94-6473-510-9

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

To my beloved wife.

CONTENTS

SUMMARY

Big data is generated daily from diverse sources and devices, significantly transforming our lives through machine learning. However, it also presents major challenges, particularly for individuals and organizations with limited storage and computational resources. As a result, cloud services have gained increasing popularity over the past decades, enabling users to outsource storage and complex analysis tasks while focusing on data utilization. However, due to the potential curiosity of cloud servers and external attackers, directly uploading private data to the cloud is not a viable option. Instead, sensitive data must be encrypted before being outsourced.

This thesis investigates cryptographic solutions for secure and efficient cloud services, addressing key challenges in security, efficiency, and functionality. We focus on three core areas: updatable encryption (UE) to ensure long-term security for stored data, fully homomorphic encryption (FHE) for efficient computation over encrypted data, and searchable encryption (SE) to maintain search functionality over outsourced encrypted data.

For UE schemes, we bridge existing gaps by clarifying the relationship among various security notions. We also extend prior work by improving adaptive security, CCA-1 security, and post-quantum security. Additionally, we introduce a novel packing technique, enabling the simultaneous encryption and update of multiple messages, significantly reducing token generation overhead.

For FHE, we enhance its efficiency and applicability by developing new batch bootstrapping techniques that optimize both noise growth and computational costs. Our work introduces a basic batch programmable bootstrapping method that evaluates arbitrary univariate functions over multiple ciphertexts while refreshing noise. We further improve this technique with two homomorphic decomposition algorithms, facilitating computations with higher precision.

In the area of SE, we identify two efficient attacks leveraging background knowledge and leakage to recover private information using both passive and active methods. Our passive attack utilizes access and volume leakage patterns to retrieve document and keyword matches with minimal leaked data. Our active attack, performed through file injection, achieves a 100% query recovery rate with fewer injected files compared to current state-of-the-art methods.

While this research primarily addresses secure cloud services, its findings also extend to related cryptographic primitives, such as identity-based encryption, attribute-based encryption, and proxy re-encryption, as well as privacy-preserving machine learning and multi-party computation. Ultimately, our work contributes to advancing the state-of-the-art in cryptography for secure and efficient cloud computing.

SAMENVATTING

Big data wordt dagelijks gegenereerd vanuit diverse bronnen en apparaten, wat ons leven aanzienlijk verandert door middel van machine learning. Het brengt echter ook grote uitdagingen met zich mee, vooral voor individuen en organisaties met beperkte opslag- en rekencapaciteiten. Als gevolg hiervan zijn cloudservices de afgelopen decennia steeds populairder geworden, waarmee gebruikers hun opslag- en complexe analysetaken kunnen uitbesteden en zich kunnen concentreren op het gebruik van data. Echter, vanwege de nieuwsgierigheid van cloudservers en externe aanvallers, is het niet mogelijk om privédata direct naar de cloud te uploaden. In plaats daarvan moeten gevoelige gegevens worden versleuteld voordat ze worden uitbesteed.

Dit proefschrift onderzoekt cryptografische oplossingen voor veilige en efficiënte cloudservices, met nadruk op de belangrijkste uitdagingen op het gebied van veiligheid, efficiëntie en functionaliteit. We richten ons op drie kerngebieden: updatable encryption (UE) voor de waarborging van de langetermijnbeveiliging van opgeslagen gegevens, fully homomorphic encryption (FHE) voor efficiënte berekeningen over versleutelde gegevens en searchable encryption (SE) om de zoekfunctionaliteit van uitbestede versleutelde gegevens te behouden.

Voor UE-schema's overbruggen we bestaande hiaten door de relatie tussen verschillende beveiligingsconcepten te verduidelijken. We breiden ook eerder werk uit door de adaptieve beveiliging, CCA-1-beveiliging en post-quantumbeveiliging te verbeteren. Bovendien introduceren we een nieuwe verpakkingsmethode waarmee meerdere berichten gelijktijdig kunnen worden versleuteld en bijgewerkt, wat de overhead bij het genereren van tokens aanzienlijk vermindert.

Voor FHE verbeteren we de efficiëntie en toepasbaarheid door nieuwe batch bootstrapping-technieken te ontwikkelen die zowel de groei van ruis als de rekenkosten optimaliseren. Ons werk introduceert een basismethode voor batch-gewijs programmeerbare bootstrapping waarmee willekeurige univariate functies over meerdere ciphertexts kunnen worden geëvalueerd, terwijl de ruis wordt ververst. We verbeteren deze techniek verder met twee homomorfe decompositie-algoritmen, waardoor berekeningen met hogere precisie mogelijk worden.

Op het gebied van SE identificeren we twee efficiënte aanvallen die gebruik maken van achtergrondkennis en lekken om privé-informatie te herstellen via zowel passieve als actieve methoden. Onze passieve aanval maakt gebruik van toegang- en volume-lekken om documenten en trefwoordovereenkomsten te achterhalen met minimale gelekte gegevens. Onze actieve aanval, uitgevoerd via bestandinjectie, bereikt een 100

Hoewel dit onderzoek zich primair richt op veilige cloudservices, strekt de bevindingen zich ook uit naar verwante cryptografische primitieve, zoals identity-based encryption, attribute-based encryption en proxy re-encryption, evenals privacy-preservende machine learning en multi-party computation. Uiteindelijk draagt ons werk bij aan de

vooruitgang van de stand van zaken in cryptografie voor veilige en efficiënte cloud computing.

1

INTRODUCTION

In the era of big data, massive amounts of data are being created every day. That boosts the popularity of cloud services in our daily lives. When we use cloud services, such as Dropbox, Google, and Microsoft, we can delegate the management of our data and concentrate solely on using it. Those cloud services support a wide range of activities, including but not limited to data storage, computing and data analysis, collaborating with other authorized users and sharing, etc.

However, concerns are growing about the potential privacy leakages of personal data stored in the cloud, such as financial records, private images, and videos. Attackers could be external hackers or insiders within the cloud services themselves. This thesis aims to propose cryptographic schemes that enable cloud clients to securely use cloud services without having to trust the cloud providers.

1

1.1. THREE ADVANCED ENCRYPTION SCHEMES

To prevent the leakage of raw data to attackers, cloud users encrypt their messages using a secret key through cryptographic techniques, and then send the encrypted data to the cloud. Cryptographic systems ensure two key security properties for the encrypted messages: *Confidentiality* and *Integrity* [1]. Confidentiality ensures that the encrypted data is indistinguishable from random noise to anyone who does not possess the secret key. Only the data owner holds the corresponding key to decrypt the stored encryption and recover the original messages. Integrity ensures that adversaries cannot modify the encrypted data without detection.

Cryptography can be classified into *symmetric-key encryption* and *asymmetric-key encryption* (also known as public key encryption, or PKE). In symmetric-key encryption, the same key is used for both encryption and decryption, and the security level depends on the length of the secret key. In contrast, asymmetric-key encryption uses different keys for encryption and decryption, with security relying on the hardness of mathematical problems, such as the discrete logarithm problem or integer factorization problem. In the past decade, rapid advancements in quantum computing have raised the possibility of attacks based on Grover's and Shor's algorithms in the near future. These algorithms can efficiently solve the hard problems underlying PKE, posing a significant threat to the security of public-key cryptography, including RSA [2], ECDSA [3], and EdDSA [4]. As a result, there is an urgent need to redesign cryptosystems that are resilient to quantum attacks. In fact, NIST has recently announced plans to completely phase out the use of these three schemes by 2035 [5]. Therefore, it is crucial to develop quantum-secure cryptographic schemes for cloud services.

This thesis mainly focuses on three key research topics in advanced cryptography for the secure and efficient use of cloud services:

- *Updatable Encryption* (UE): This addresses the challenge of long-term data storage on the cloud. Regularly updating encryption keys through UE reduces the risk of key compromise and enhances security.
- *Fully Homomorphic Encryption* (FHE): FHE enables secure computation over encrypted data stored in the cloud, allowing service providers to perform operations without exposing the underlying data.
- *Searchable Encryption* (SE): SE is a cryptographic solution that preserves the search functionality of encrypted data without revealing the queried information to the cloud.

In this thesis, we will analyze the security notions related to updatable encryption and propose a quantum-secure UE scheme that meets these security requirements. For FHE, we will present efficient schemes that enable service providers to perform secure and efficient computations over large encrypted datasets. For SE, we propose an efficient passive attack and an efficient active attack to recover private information about underlying keywords and documents.

Our UE and FHE constructions are based on lattice-based cryptosystems, which rely on the presumed hardness of lattice problems such as the Shortest Vector

Problem (SVP). SVP is an NP-hard problem that involves finding the shortest non-zero vector in a lattice. Other related lattice problems, including the Closest Vector Problem (CVP), the Shortest Independent Vectors Problem (SIVP), and Decisional Approximate SVP (GapSVP), also remain difficult to solve efficiently, even with quantum computers, and therefore ensure the lattice-based cryptosystem to be quantum-resistant [6].

UPDATABLE ENCRYPTION.

The privacy of encrypted messages stored in the cloud is guaranteed by the security of cryptography systems. An observation is that this privacy is based on the assumption that the user's private key is not compromised. However, such assumption probably fails to exist for a long-time data storage, as the exposure of a key over an extended period increases the risk of it being compromised. Therefore, a more secure approach for long-term data storage is to periodically change the secret key used to protect the data, and to update the corresponding ciphertext stored in the cloud from the old key to the new one, without altering the plaintext. A similar strategy is employed by TU Delft, which requires all students to change their NetID password every nine months.

Updatable encryption (UE), introduced by Boneh et al. [7], provides a practical solution for periodic key rotation. A naive approach to updating the key would involve downloading the data, decrypting it with the old key, re-encrypting it with the new key, and then re-uploading the new ciphertext. However, this process is computationally expensive for large datasets. In contrast, UE allows a cloud user to generate a short *token* that enables the cloud service to update the data on their behalf. Unlike standard symmetric or asymmetric cryptography, which only involves encryption and decryption algorithms, a UE scheme includes two additional algorithms related to token generation and ciphertext updating. Depending on how the token is generated, there are two types of UE: *ciphertext-independent* (c-i) UE and *ciphertext-dependent* (c-d) UE, as shown in Fig. 1.1.

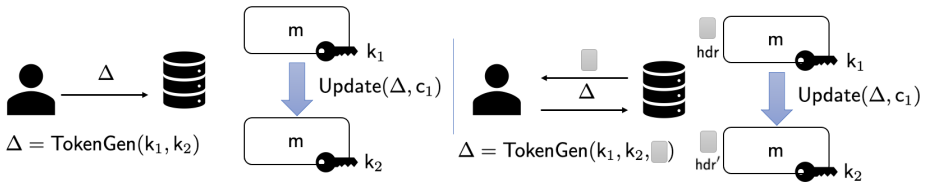


Figure 1.1: An overview of ciphertext-independent UE and ciphertext-dependent UE.

For ciphertext-independent updatable encryption [8–14], the token is independent of the ciphertexts to be updated, meaning a single token can be used to update all old ciphertexts. This approach enables the cloud user to generate the token without downloading any ciphertext from the cloud. The token depends only on the old and new encryption keys. An example of c-i UE, as presented in [8], is shown below:

$$\begin{cases} \mathsf{Enc}(\mathsf{k}_1, \mathsf{m}) = (\pi(N||\mathsf{m}||\mathbf{0}))^{\mathsf{k}_1} \\ \mathsf{TokenGen}(\mathsf{k}_1, \mathsf{k}_2) = \mathsf{k}_2/\mathsf{k}_1 \\ \mathsf{Update}(\Delta, \mathsf{c}_1) = \mathsf{c}_1^{\Delta} \end{cases} \tag{1.1}$$

In this scheme, the encryption function applies a permutation π over the input, which consists of a nonce N, a message m, and a zero string. The term Δ in the update algorithm Update represents the token generated by the token generation algorithm TokenGen. (For simplicity, we omit the decryption algorithm here to focus on the update process.) With the help of a single token Δ, the cloud service can update all ciphertexts that were encrypted under the old key k_1.

Similar to this example, many existing ciphertext-independent UE mechanisms leverage the homomorphic properties of exponentiation during the update process. As a result, the primary computational cost arises from performing exponentiation operations.

For ciphertext-dependent UE [7, 15–17], the token is related to the ciphertext that needs updating, and a small part of the ciphertext, called the ciphertext header, must be downloaded during the token generation process. In this model, a token can only be used to update a single ciphertext. Prior constructions of c-d UE schemes mainly use key-homomorphic pseudorandom functions (KH-PRFs) [7]. An example of such a scheme, presented in [15], is as follows:

$$\begin{cases} \mathsf{Enc}(\mathsf{k}_1, \mathsf{m}) := (\hat{\mathsf{c}}_1, \mathsf{c}_1) = \left(\mathsf{AE.Enc}(\mathsf{k}_1, \mathsf{k}_{\mathsf{prf1}}), F(\mathsf{k}_{\mathsf{prf1}}, i) + \mathsf{m}_i\right) \\ \mathsf{TokenGen}(\mathsf{k}_1, \mathsf{k}_2, \hat{\mathsf{c}}_1) : \begin{cases} \mathsf{k}_{\mathsf{prf1}} \leftarrow \mathsf{AE.Dec}(\mathsf{k}_1, \hat{\mathsf{c}}_1) \\ \mathsf{k}_{\mathsf{prf}}^{\mathsf{up}} \leftarrow \mathsf{k}_{\mathsf{prf2}} - \mathsf{k}_{\mathsf{prf1}} \\ \hat{\mathsf{c}}_2 \leftarrow \mathsf{AE.Enc}(\mathsf{k}_2, \mathsf{k}_{\mathsf{prf2}}) \end{cases} \\ \mathsf{Update}\left((\mathsf{k}_{\mathsf{prf}}^{\mathsf{up}}, \hat{\mathsf{c}}_2), (\hat{\mathsf{c}}_1, \mathsf{c}_1)\right) = \left(\hat{\mathsf{c}}_2, F(\mathsf{k}_{\mathsf{prf}}^{\mathsf{up}}, i) + \mathsf{c}_1\right). \end{cases} \tag{1.2}$$

In this scheme, AE represents an authenticated encryption scheme, and F is a KH-PRF. To update an old ciphertext, its ciphertext header $\hat{\mathsf{c}}_1$ is downloaded to recover the PRF key $\mathsf{k}_{\mathsf{prf1}}$ used in the encryption. The token is computed as the difference between the new PRF key and the old one. The key-homomorphic property guarantees the correctness of the update algorithm as follows:

$$F(\mathsf{k}_{\mathsf{prf2}} - \mathsf{k}_{\mathsf{prf1}}, i) + F(\mathsf{k}_{\mathsf{prf1}}, i) + \mathsf{m}_i = F(\mathsf{k}_{\mathsf{prf2}}, i) + \mathsf{m}_i$$

where $(\mathsf{m}_1, \cdots, \mathsf{m}_l) \leftarrow \mathsf{Encode}(\mathsf{m})$ and $1 \le i \le l$.

We will analyze the challenges in defining and constructing secure updatable encryption in Section 1.2. Intuitively, the tokens and keys should not reveal any useful information to the adversary, and the scheme must also remain secure even if partial keys and tokens are leaked.

FULLY HOMOMORPHIC ENCRYPTION.

Fully Homomorphic Encryption (FHE) allows the cloud to perform arbitrary computations on encrypted data without the need to decrypt it. This capability

enables the cloud to offer its computing power as a service while simultaneously protecting the privacy of sensitive information in the outsourced data. FHE has a wide range of applications, including privacy-preserving machine learning [18–20] and multi-party computation [21–23]. Many FHE schemes have been efficiently implemented in various libraries, such as OpenFHE [24], SEAL [25], Concrete [26], HElib [27], and HEaaN [28]. Our goal is to explore FHE's potential and limitations in these contexts.

In more detail, given encryptions $\mathsf{Enc}(\mathsf{m}_1), \cdots, \mathsf{Enc}(\mathsf{m}_r)$, the cloud can compute an encryption of $f(\mathsf{m}_1, \cdots, \mathsf{m}_r)$ via FHE without gaining access to the secret key of the input ciphertexts. The security of FHE schemes relies on the hardness of the Learning With Errors (LWE) problem and its generalizations (to be defined later), where a ciphertext contains an error term that ensures its security. The decryption of a ciphertext is correct as long as this error term remains small. However, as homomorphic operations are performed, the error increases, which limits the number of homomorphic operations that can be performed on a ciphertext.

The *bootstrapping* technique, introduced by Gentry [29], is known as the unique method to achieve *unbounded* FHE. This technique leverages the homomorphic properties of the scheme to evaluate the decryption function over the ciphertext. The result is still an encryption of the original plaintext, but with reduced error, effectively refreshing the ciphertext for further computations. However, the original bootstrapping construction was impractical, requiring up to half an hour to bootstrap a single bit. Since then, many FHE schemes have been proposed to enhance performance and reduce the time required for bootstrapping.

The second-generation Fully Homomorphic Encryption schemes, such as BGV [30] and its variants [31–34], allow multiple messages to be refreshed simultaneously, achieving a very low *amortized* cost per message. However, these schemes have a downside: the bootstrapping process often leads to quasipolynomial error growth, which necessitates relatively large parameters, such as a superpolynomial size modulus. As a result, these schemes require stronger security assumptions and slower bootstrapping procedures compared to third-generation FHE schemes.

The third-generation FHE schemes, including GSW [35], FHEW [36], and TFHE [37], are collectively referred to as FHEW-like cryptosystems in [38]. These schemes significantly simplify the bootstrapping process, enabling it to be performed within a few milliseconds on a personal computer. Moreover, the bootstrapping process incurs only polynomial noise growth, resulting in weaker security assumptions for lattice-based problems compared to BGV. However, the limitation is that these schemes have a high amortized cost, as a single message can be bootstrapped at a time. In Section 1.2, we will discuss the challenges in proposing an efficient, amortized FHEW-like cryptosystem.

LATTICE-BASED CRYPTOSYSTEM.

The constructions of lattice-based encryption are primarily based on two types of lattice hard problems: the *Short Integer Solution* (SIS) problem and the *Learning With Errors* (LWE) problem. The Short Integer Solution problem was introduced by Ajtai [39] and forms the basis for various lattice-based cryptographic primitives,

including hash functions, digital signatures, and one-way functions. Specifically, the SIS problem involves finding a short nonzero vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{0}$, where $\mathbf{A}$ is a uniformly random integer matrix.

The Learning With Errors problem, proposed by Regev [40] in 2005, asks to find an integer vector $\mathbf{s}$ from the pair $(\mathbf{A}, \mathbf{As} + \mathbf{e} \bmod q)$, where $\mathbf{A}$ is a matrix and $\mathbf{e}$ is an error vector sampled from a particular error distribution.

Under certain parameter settings, the SIS and LWE problems, along with their ring-based variants, are proven to be as hard as solving worst-case lattice problems such as the Shortest Independent Vector Problem (SIVP) and the Gap Shortest Vector Problem (GapSVP) [41–45]. Therefore, cryptographic schemes based on SIS and LWE are considered quantum-secure, assuming that worst-case lattice problems cannot be efficiently solved by quantum computers. Our constructions of Updatable Encryption and Fully Homomorphic Encryption are based on various instantiations of the SIS/LWE problems.

SEARCHABLE ENCRYPTION.

Searchable Encryption (SE) enables a client to outsource private data to an untrusted cloud while preserving the search functionality over encrypted data and keeping the queried information hidden from the cloud. An SE scheme consists of five polynomial-time algorithms: key generation, encryption, decryption, query generation, and search. The key generation and encryption algorithms are run by the user to encrypt a set of documents, which are then stored as ciphertexts on the cloud. The query generation algorithm is also run by the user to create a query for a specific target keyword. The search algorithm, which is deterministic, is run by the cloud using a query from the user. It searches the encrypted documents and returns all (encrypted) documents containing the keyword related to the query. The client can then decrypt the ciphertexts to recover the original documents. Since the initial work by Song et al. [46], many SE schemes have been proposed [47–54]. Today, SE schemes are deployed in various real-world applications, including ShadowCrypt [55] and Mimesis Aegis [56].

As defined in an SE scheme, the interaction typically involves the client sending a query to the server, which then responds with the matching documents. However, this interaction is vulnerable to eavesdropping by an attacker. For example, messages can be intercepted if transmitted over an unsecured channel, or the attacker could be the cloud service provider itself, with access to all search requests and responses. The attacker might attempt to match the query with specific keywords to infer the information stored on the server. This communication of query and response constitutes what is referred to as *leakage*. We will analyze the various attacks on SE schemes based on different leakage levels.

1.2. CHALLENGES IN UE, FHE, AND SE

There are many challenges in both Updatable Encryption and Fully Homomorphic Encryption that must be addressed to enhance the security and efficiency of cloud services. In this section, we analyze the security requirements for UE and examine

the maximum capabilities an adversary can have. This thesis closes the gap of clarifying the relationships among all existing security notions for UE, which is essential for proposing meaningful constructions and enabling a meaningful comparison of existing UE schemes. Additionally, this thesis is the first to propose stronger adaptive security and post-quantum security for ciphertext-dependent UE. Regarding FHE, this section explores the challenges of combining the advantages of the two types of bootstrapping discussed earlier. Furthermore, this thesis extends the goal of optimizing FHE from message space, functionality, and amortized complexity perspectives.

CONFIDENTIALITY FOR UE.

The basic confidentiality one expects from updatable encryption is semantic security, meaning that the ciphertext should not allow an adversary to learn any partial information about the underlying plaintext. Lehmann and Tackmann [11] proposed two confidentiality notions to capture this requirement. The first, denoted as IND-Enc, asks an adversary to distinguish between the ciphertexts of two different plaintexts. They observed that the plaintext can be leaked not only through the encryption algorithm but also through the update algorithm. To address this, they introduced a second notion, IND-Upd, which asks the adversary to submit two ciphertexts and tell which ciphertext the resulting ciphertext was updated from.

However, Boyd et al. [8] demonstrated that neither IND-Enc nor IND-Upd (even in combination) can prevent the leakage of ciphertext 'age'. They captured this security by introducing IND-UE, which guarantees that an adversary cannot distinguish between a fresh encryption and an updated ciphertext. In other words, ciphertexts produced by the encryption algorithm should be indistinguishable from those produced by the update algorithm. They showed that IND-UE is strictly stronger than the combinations of IND-Enc and IND-Upd.

We observe that there are eight variants of IND-UE for UE schemes, based on the direction of key updates and ciphertext updates (see below). However, the relationships among these variants are not fully clear in the literature, and this gap needs to be addressed.

DIRECTION OF UPDATE.

Consider the UE scheme in Eq. (1.1) as an example to explain the direction of updates from the perspectives of ciphertexts and keys.

Direction of Ciphertext Updates. In UE schemes, the primary goal is to upgrade ciphertexts encrypted with old keys to ciphertexts encrypted with new keys using tokens. However, tokens can also potentially downgrade ciphertexts from new keys to old keys, as shown by the relationship $c_2^{1/\Delta} = c_1$ in Eq. (1.1). If such downgrading is possible, the UE scheme is termed bi-directional ciphertext updates. Otherwise, if only upgrading is supported, the UE scheme is called uni-directional ciphertext updates.

Direction of Key Updates. Note that the token is generated by two successive epoch keys via the token generation algorithm, as shown by the expression $\Delta = k_2/k_1$ in Eq. (1.1). This means that the token could potentially allow an adversary to derive

one of the two keys from the other. Jiang [9] proposed three kinds of key update directions: *bi-directional* key updates, where both the old key and the new key can be derived from each other; *uni-directional* key updates, where only the new key can be derived from the old key but not vice versa; and *non-derivable* key updates, where neither key can be derived from the other. Additionally, Nishimaki [12] introduced a new type of uni-directional key update called backward-leak uni-directional key updates, where the update direction is reversed compared to the uni-directional key update in Jiang's work. In this case, the previous key can be inferred from the new key, but not vice versa. This is referred to as the backward-leak uni-directional key update, in contrast to the forward-leak uni-directional key update described in Jiang's framework.

The directions of ciphertext updates and key updates significantly influence the information leakage to an adversary, thereby impacting the security notions of a UE scheme. Understanding the relationships among these directions is critical, as it clarifies which update direction provides stronger security guarantees. Establishing these relationships is essential before constructing secure and effective UE schemes.

INTEGRITY.

Klooß et al. [10] provided the notions of ciphertext (IND-CTXT) and plaintext (IND-PTXT) integrity for UE schemes: IND-CTXT requires that an adversary cannot produce a valid ciphertext that was not derived during the security game and IND-PTXT ensures the adversary cannot work out a valid ciphertext whose underlying message is not queried in the security game.

VARIOUS VARIANTS.

Overall, the set of confidentiality and integrity notions for updatable encryption is $\mathsf{CIS} := \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}, \mathsf{IND\text{-}CTXT}, \mathsf{IND\text{-}PTXT}\}$, where det/rand means the update algorithm is deterministic or randomized, respectively. For each security notion in CIS, there are eight variants

$$(\mathsf{kk}, \mathsf{cc})\text{-}\mathsf{notion},$$

where kk represents the four ways of key update directions $\{\mathsf{bi}, \mathsf{b\text{-}uni}, \mathsf{f\text{-}uni}, \mathsf{no}\}$ and cc denotes the two ways of ciphertext update directions $\{\mathsf{bi}, \mathsf{uni}\}$. We need to investigate the relationships among all those security notions so that we can have a clear goal of UE constructions and a fair comparison among UE schemes.

LIMITATIONS OF C-D UE.

The security of UE should still hold even under the leakage of some tokens and keys, which may be lost during the usage of cloud services. However, prior works on c-d UE [15–17] only capture *selective security*, where certain keys and tokens are provided to the adversary in the start of the security game. It is needed to consider that the adversary can *adaptively* corrupt keys and tokens throughout the experiment, i.e., the *adaptive security* for c-d UE should be considered.

Additionally, we observe that the encryption algorithm of UE has to be randomized to ensure the confidentiality, but the update algorithm can be either randomized or deterministic. However, prior security notions for c-d UE only apply to randomized update algorithms. It is reasonable to consider both types of ciphertext updates.

Furthermore, current UE constructions based on the Decision Diffie-Hellman (DDH) assumption are vulnerable to attacks by powerful quantum computers. It is crucial to deploy UE schemes with post-quantum security while ensuring the desired security requirements.

BATCH BOOTSTRAPPING.

One of the key challenges in fully homomorphic encryption is developing a novel bootstrapping technique that combines the advantages of the two existing types of bootstrapping. Specifically, the goal is to achieve low noise growth, which ensures efficient ciphertext refreshment under weaker assumptions, while also maintaining a low amortized cost, enabling the simultaneous processing of multiple messages.

The first amortized FHEW bootstrapping was introduced by Micciancio and Sorrell [57], enabling the refreshment of n ciphertexts using $O(3^\rho \cdot n^{1+1/\rho})$ FHE multiplications. This approach reduced the amortized cost per message from $O(n)$ to $O(3^\rho \cdot n^{1/\rho})$ for a parameter $\rho > 2$, while still keeping polynomial noise growth. Subsequent work by Guimarães et al. [58] further reduced the amortized cost to $O(\rho \cdot n^{1/\rho})$ FHE operations per ciphertext. More recently, Liu and Wang significantly improved these techniques within a polynomial modulus, achieving an amortized cost of $\widetilde{O}(n^{0.75})$ in [59] and nearly optimal $\widetilde{O}(1)$ in [60].

However, the batch bootstrapping techniques by Liu and Wang are currently limited to binary message spaces, leaving their practicality for larger message spaces uncertain. Moreover, the third-generation FHE schemes is also called *programmable* bootstrapping [61, 62], enabling the evaluation of univariate functions during the bootstrapping process. It remains an open question whether the batch bootstrapping methods in [59, 60] can support this programmable property while maintaining only polynomial noise growth.

ATTACKS ON SE.

In recent years, it has been shown that there are various attacks on SE schemes that exploit SE leakages to recover the information about keywords, using partial information about the documents or the keywords in the documents. The objectives of SE attackers are typically divided into query recovery which denotes matching the the underlying keywords related to the queries that are enquired, and document recovery which is to recover the relationship between plaintext documents and encrypted documents. According to the attacker power, SE attacks can be divided into passive and active attacks.

For passive attacks, the adversary uses leaked queries and their response with auxiliary knowledge to perform query recovery and document recovery. Islam et al. [63] laid the groundwork for passive attacks on SE schemes. They demonstrated that, with adequate auxiliary knowledge, a co-occurrence matrix could be constructed for both the observed leakage and the auxiliary data. This matrix enables the mapping

of queries to keywords by identifying the closest match based on minimal distance. Cash et al. [64] later proposed an attack that matches queries to specific keywords by analyzing their total occurrences in the leaked documents. Blackstone et al. [65] developed the $\text{Subgraph}_{\text{VL}}$ attack, which achieves a relatively high query recovery rate even with a small subset of leaked documents. This attack matches keywords by leveraging unique document volumes, treating them as response patterns. Ning et al. [66] later introduced the LEAP attack, which combines existing techniques, such as co-occurrence analysis and unique occurrence counts. LEAP relies on unique occurrences in the matched documents to achieve this. It effectively utilizes the unique count from the Count attack [64], a co-occurrence matrix from the IKK attack [63], and unique patterns to match keywords and documents.

The active attack, also known as a file-injection attack, involves the attacker selecting specific documents and sending them to the client for injection. The client then encrypts these documents and stores the resulting ciphertexts on the server. Through keyword searches, the attacker can observe the results corresponding to the queries. For example, an attacker could inject files into a user's system by sending special emails. The attacker then observes the returned files, particularly the ones they injected, in response to queries made through the search algorithm. By analyzing the returned (previously injected) files, the attacker can achieve the goal of recovering the queried information. The first file-injection attack is proposed by Zhang et al. [67], which leverages the binary search concept. Each injected file contains exactly half of the keywords from the injected keyword universe K, resulting in a maximum file size of $\lceil \log|K| \rceil$. By analyzing the presence or absence of specific keywords in the returned files, the attacker can determine the queried keyword with 100% accuracy. Based on finite set theory, Wang et al. [68] further enhanced the work in [67] to address the countermeasures involving a threshold for the maximum number of keywords in each file. Their attack requires fewer injected files than the previous one while still achieving a 100% query accuracy rate.

Existing SE attacks have the following limitations. Passive approaches typically exploit only a single leakage pattern, either assess pattern or volume pattern, whereas multiple leakage patterns are possible in SE schemes. In addition to this, the state-of-the-art LEAP attack utilizes the access pattern but does not fully exploit its matching techniques. Additionally, active attacks are less practical in real-world scenarios, particularly when countermeasures like thresholds and padding are implemented to thwart their success. This is especially true when the number of keywords in the dataset is very large.

1.3. PROBLEM STATEMENT

Cloud technology has become an essential tool in modern society. This thesis aims to address challenges in *secure* cloud services over encrypted data, focusing on aspects of security, efficiency, and functionality. The research questions of this thesis are presented in this section and involve three cryptosystems: updatable encryption, fully homomorphic encryption, and searchable encryption.

Our work primarily focuses on applications to cloud services, but it is possible to

extend our results on UE to other cryptosystems that involve a similar process of transferring ciphertexts, such as *proxy re-encryption* [69, 70], *identity-based encryption* [71], and *attribute-based encryption* [72]. Additionally, our FHE techniques can be applied to *privacy-preserving machine learning* and *secure multi-party computation*. Below, we list our research questions.

UPDATABLE ENCRYPTION.

The primary security requirement for UE schemes is that security should be maintained even under the leakage of some keys and tokens. However, certain combinations of leaked information might enable the adversary to trivially win the security game. Thus, some trivial win conditions must be checked at the end of the game. For example, in the CCA security game of a public key encryption (PKE) scheme, it is necessary to verify whether the adversary queries the decryption oracle on the challenge ciphertext. In UE schemes, both the direction of ciphertext updates and key updates affect the information leaked to the adversary, and different security notions involve various types of trivial win conditions. It is challenging to clarify the relationships among all existing security notions across different update settings. Existing work has only explored the relationships among a subset of these notions. Therefore, our first research question is:

Q1: *What are the relationships among all existing security notions for updatable encryption schemes?*

Additionally, we consider potential improvements for ciphertext-dependent updatable encryption (c-d UE) schemes. In a c-d UE scheme, an adversary may adaptively corrupt keys and tokens at any point during the game. It is necessary to "maximize" the adversary's capabilities to fully capture adaptive security. From an efficiency standpoint, c-d UE schemes require downloading a small part of the ciphertext (the ciphertext header) to generate the update token, and each token can only be used to update a single message. A meaningful improvement would allow a single token to update multiple messages, even in the ciphertext-dependent setting. Therefore, our second research question is:

Q2: *How can we design a more secure and more efficient ciphertext-dependent updatable encryption scheme?*

FULLY HOMOMORPHIC ENCRYPTION.

It is desirable to construct a bootstrapping technique that achieves the low amortized cost per message of second-generation FHE schemes while also maintaining the low polynomial noise growth characteristic of third-generation schemes. The state-of-the-art work [59, 60] on batch FHEW-like bootstrapping partially achieves this goal but is limited to a binary message space. Extending this technique to support a larger message space without increasing noise growth or amortized cost is a critical challenge.

Furthermore, FHEW-like bootstrapping offers a *programmable* property, enabling the evaluation of a univariate function simultaneously with ciphertext refreshing. Incorporating this property into batch FHEW-like bootstrapping techniques is essential. These considerations lead to our third research question:

Q3: *Can we design a batch programmable bootstrapping technique for a large message space, within a polynomial modulus?*

SEARCHABLE ENCRYPTION.

We aim to address the limitations in prior works on both passive and active SE attacks, as stated in Section 1.2. Our fourth and fifth questions are summarized as follows:

Q4: *Can we design a passive SE attack by fully exploiting both the volume and access patterns to capture a high recovery rate?*

Q5: *Can we design an active SE attack that is more practical when considering the threshold countermeasure?*

1.4. CONTRIBUTIONS OF THE THESIS

This thesis consists of three independent technical chapters, each based on an individual research paper. Since Chapters 2 and 3 are largely based on published works on Updatable Encryption, and Chapters 5 and 6 on Searchable Encryption, so there may be some overlap in the background and related work sections. We include additional detailed proofs and accessible examples that were previously omitted due to page limitations of publication. The thesis is organized as follows:

CHAPTER 2

EQUIVALENCE OF UPDATABLE ENCRYPTION SCHEMES.

In this chapter, we address the research question Q1 and analyze the relations of all existing security notions. Our main technique is to analyze the relations of the trivial win conditions for each security notion in different key update settings. As in other semantic security definitions, the adversary in the security game for UE is provided access to different oracles to capture realistic attack models. However, it may lead to a trivial win if the adversary queries some combinations of oracles. Therefore, the bookkeeping technique was developed in [10, 11] that tracks the leakage information of tokens, keys, and ciphertexts known to the adversary during the game and checks if those leakages may lead the adversary to trivially win after the adversary submits its guessing bit. The direction of key update affects the computation of leakage information and thus, we analyze the relations among UE schemes by analyzing the relations of trivial win conditions in different key update directions. Our main result presents a surprising finding: for each security notion, no-directional key update UE schemes, which were previously believed to be strictly

stronger than other directional key update schemes, are actually equivalent to those with backward-leak uni-directional key updates.

This chapter is based on the published paper, with minor additions of examples and proofs “No-Directional and Backward-Leak Uni-Directional Updatable Encryption Are Equivalent" by **Chen, H.**, Fu, S. and Liang, K. in European Symposium on Research in Computer Security (2022).

CHAPTER 3

CCA-1 SECURE UPDATABLE ENCRYPTION WITH ADAPTIVE SECURITY.

In this chapter, we address the research question Q2 and make three significant contributions to ciphertext-dependent updatable encryption: First, we introduce stronger security notions compared to previous work, addressing adaptive security and considering the decryption capability of the adversary under adaptive corruption. Second, we propose a novel c-d UE scheme that satisfies these security notions, employing a token generation technique distinct from the traditional Dec-then-Enc structure, while still preventing key leakage. This UE scheme is based on lattice hard problems, ensuring that it achieves quantum security. Finally, we develop a packing technique that enables the encryption and updating of multiple messages within a single ciphertext, reducing the cost of c-d UE by minimizing the need to download partial ciphertexts during token generation.

This chapter is based on the published paper with detailed proofs “CCA-1 Secure Updatable Encryption with Adaptive Security” by **Chen, H.**, Galteland, Y.J. and Liang, K. in International Conference on the Theory and Application of Cryptology and Information Security (2023)

CHAPTER 4

BATCH PROGRAMMABLE BOOTSTRAPPING, WITHIN A POLYNOMIAL MODULUS.

In this chapter, we address the research question Q3 and work on the improvement of the efficiency of fully homomorphic encryption. First, we introduce a novel batch bootstrapping technique within a polynomial modulus that enables noise refreshment over a general message space extending beyond one bit, while maintaining the same amortized cost and noise overhead as the work by Liu and Wang (EUROCRYPT 2023). This batch bootstrapping is also programmable, allowing the evaluation of a univariate function simultaneously with noise refreshment. Second, our approach overcomes a key limitation of third-generation FHE schemes, which require the evaluated function to be negacyclic. In contrast, our method supports arbitrary functions. Additionally, we propose two homomorphic decomposition algorithms, further extending our batch programmable bootstrapping to support larger message spaces. Third, we enhance practicality by demonstrating the evaluation of commonly used activation functions in Convolutional Neural Networks (CNNs), such as ReLU, sign, and max.

This work was partially conducted during a visit to the University of Padua.

CHAPTER 5

VOLUME AND ACCESS PATTERN LEAKAGE-ABUSE SE ATTACK.

In this chapter, we address the research question Q4 and present three contributions to passive SE attacks. First, in addition to exploiting the access pattern, we also leverage volume pattern leakage. By combining both leakage patterns, we match documents based on a unique combination of volume and the number of keywords, enabling us to match nearly all leaked documents to server documents. Second, beyond matching keywords in the identified files, we utilize all leaked documents to analyze unique keyword occurrences, enhancing the keyword matching technique from the LEAP attack. This allows us to maximize keyword matches using the unique occurrence pattern. Third, we evaluate our attack on three different datasets to assess performance, and the results are exceptional, with nearly all leaked documents and a significant number of leaked keywords successfully matched.

This section is based on the published paper "VAL: Volume and Access Pattern Leakage-Abuse Attack with Leaked Documents" by Lambregts, S., **Chen, H.**, Ning, J. and Liang, K. in European Symposium on Research in Computer Security (2022).

CHAPTER 6

FILE-INJECTION ATTACKS ON SE, BASED ON BINOMIAL STRUCTURES.

In this chapter, we address the research question Q5 and present two contributions to active SE attacks. First, we introduce a novel file-injection attack on searchable encryption schemes by leveraging the binomial search concept, which is based on our new definition of a subset family of a finite set. Compared to previous work, this new subset family allows us to include more keywords in the injected files, thereby significantly reducing the number of files required for injection. Second, we propose a mitigation strategy that performs better under a scheme using padding, with minimal trade-offs.

This section is based on the published paper "File-Injection Attacks on Searchable Encryption, Based on Binomial Structures" by Langhout, T., **Chen, H.**, and Liang K. in European Symposium on Research in Computer Security (2024).

CHAPTER 7

DISCUSSION.

This chapter concludes the thesis and discusses the research questions in detail. We propose future work and directions to address the limitations of the work presented in this thesis.

1.4.1. LISTS OF EXCLUDED PUBLICATIONS

In the following, we list the papers published during the Ph.D. period that are not included in the thesis.

- Tjiam, K., Wang, R., **Chen, H.** and Liang, K. Your smart contracts are not secure: investigating arbitrageurs and oracle manipulators in Ethereum. In Proceedings of the 3rd Workshop on Cyber-Security Arms Race (2021).

- Ho, B., **Chen, H.**, Shi, Z. and Liang, K. Similar Data is Powerful: Enhancing Inference Attacks on SSE with Volume Leakages. In European Symposium on Research in Computer Security (ESORICS 2024). File-Injection Attacks on Searchable Encryption, Based on Binomial Structures. InEuropean Symposium on Research in Computer Security (ESORICS 2024).
- Zhang, M., Shi, Z., **Chen, H.** and Liang, K. Inject Less, Recover More: Unlocking the Potential of Document Recovery in Injection Attacks Against SSE. In Computer Security Foundations Symposium (CSF 2024).
- Wang, R., Wang, X., **Chen, H.**, Decouchant, J., Picek, S., Laoutaris, N. and Liang, K. MUDGUARD: Taming Malicious Majorities in Federated Learning using Privacy-Preserving Byzantine-Robust Clustering. In ACM SIGMETRICS (2025).

REFERENCES

[1] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. ISBN: 9781466570269. URL: https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342.

[3] D. Johnson, A. Menezes, and S. A. Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". In: *Int. J. Inf. Sec.* 1.1 (2001), pp. 36–63. DOI: 10.1007/S102070100002.

[4] S. Josefsson and I. Liusvaara. "Edwards-Curve Digital Signature Algorithm (EdDSA)". In: *RFC* 8032 (2017), pp. 1–60. DOI: 10.17487/RFC8032.

[5] NIST. *Transition to Post-Quantum Cryptography Standards: Initial Public Draft*. Tech. rep. NIST IR 8547. U.S. Department of Commerce, 2024. URL: https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf.

[6] C. Peikert. "A Decade of Lattice Cryptography". In: *Found. Trends Theor. Comput. Sci.* 10.4 (2016), pp. 283–424. DOI: 10.1561/0400000074.

[7] D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. "Key homomorphic PRFs and their applications". In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Heidelberg: Springer, 2013, pp. 410–428. DOI: 978-3-642-40041-4_23.

[8] C. Boyd, G. T. Davies, K. Gjøsteen, and Y. Jiang. "Fast and Secure Updatable Encryption". In: *CRYPTO 2020, Part I*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12170. LNCS. Springer, 2020, pp. 464–493. DOI: 10.1007/978-3-030-56784-2_16.

[9] Y. Jiang. "The direction of updatable encryption does not matter much". In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer. Heidelberg, 2020, pp. 529–558. DOI: 10.1007/978-3-030-64840-4_18.

[10] M. Klooß, A. Lehmann, and A. Rupp. "(R)CCA secure updatable encryption with integrity protection". In: *EUROCRYPTO 2019, Part I*. Ed. by Y. Ishai and V. Rijmen. Vol. 11476. LNCS. Springer. Heidelberg, 2019, pp. 68–99. DOI: 10.1007/978-3-030-17653-2_3.

[11] A. Lehmann and B. Tackmann. “Updatable encryption with post-compromise security”. In: *EUROCRYPT 2018, Part III*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10822. LNCS. Springer. Heidelberg, 2018, pp. 685–716. DOI: 10.1007/978-3-319-78372-7_22.

[12] R. Nishimaki. “The Direction of Updatable Encryption Does Matter”. In: *PKC 2022*. Ed. by G. Hanaoka, J. Shikata, and Y. Watanabe. Vol. 13178. LNCS. Cham: Springer, 2022, pp. 194–224. ISBN: 978-3-030-97131-1. DOI: 10.1007/978-3-030-97131-1_7.

[13] D. Slamanig and C. Striecks. *Puncture ’Em All: Updatable Encryption with No-Directional Key Updates and Expiring Ciphertexts*. Cryptology ePrint Archive, Paper 2021/268. https://eprint.iacr.org/2021/268. 2021.

[14] Y. J. Galteland and J. Pan. “Backward-Leak Uni-Directional Updatable Encryption from (Homomorphic) Public Key Encryption”. In: *PKC 2023, Part II*. Ed. by A. Boldyreva and V. Kolesnikov. Vol. 13941. LNCS. Springer, 2023, pp. 399–428. DOI: 10.1007/978-3-031-31371-4_14.

[15] D. Boneh, S. Eskandarian, S. Kim, and M. Shih. “Improving Speed and Security in Updatable Encryption Schemes”. In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Cham: Springer, 2020, pp. 559–589. ISBN: 978-3-030-64840-4. DOI: 10.1007/978-3-030-64840-4_19.

[16] L. Chen, Y. Li, and Q. Tang. “CCA Updatable Encryption Against Malicious Re-encryption Attacks”. In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer, 2020, pp. 590–620. DOI: 10.1007/978-3-030-64840-4_20.

[17] A. Everspaugh, K. Paterson, T. Ristenpart, and S. Scott. “Key rotation for authenticated encryption”. In: *CRYPTO 2017, Part III*. Ed. by J. Katz and H. Shacham. Vol. 10403. LNCS. Springer. Heidelberg, 2017, pp. 98–129. DOI: 10.1007/978-3-319-63697-9_4.

[18] K. Cong, D. Das, J. Park, and H. V. L. Pereira. “SortingHat: Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transciphering”. In: *CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers, and E. Shi. ACM, 2022, pp. 563–577. DOI: 10.1145/3548606.3560702.

[19] R. A. Mahdavi, H. Ni, D. Linkov, and F. Kerschbaum. “Level Up: Private Non-Interactive Decision Tree Evaluation using Levelled Homomorphic Encryption”. In: *CCS 2023*. Ed. by W. Meng, C. D. Jensen, C. Cremers, and E. Kirda. ACM, 2023, pp. 2945–2958. DOI: 10.1145/3576915.3623095.

[20] F. Bourse, M. Minelli, M. Minihold, and P. Paillier. “Fast Homomorphic Evaluation of Deep Discretized Neural Networks”. In: *CRYPTO 2018, Part III*. Ed. by H. Shacham and A. Boldyreva. Vol. 10993. LNCS. Springer, 2018, pp. 483–512. DOI: 10.1007/978-3-319-96878-0_17.

[21] N. P. Smart. “Practical and Efficient FHE-Based MPC”. In: *Cryptography and Coding - 19th IMA International Conference, IMACC 2023, London, UK, December 12-14, 2023, Proceedings*. Ed. by E. A. Quaglia. Vol. 14421. LNCS. Springer, 2023, pp. 263–283. DOI: 10.1007/978-3-031-47818-5_14.

[22] A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N. P. Smart. “Between a Rock and a Hard Place: Interpolating between MPC and FHE”. In: *ASIACRYPT 2013, Part II*. Ed. by K. Sako and P. Sarkar. Vol. 8270. LNCS. Springer, 2013, pp. 221–240. DOI: 10.1007/978-3-642-42045-0_12.

[23] P. Mukherjee and D. Wichs. “Two Round Multiparty Computation via Multi-key FHE”. In: *EUROCRYPT 2016, Part II*. Ed. by M. Fischlin and J. Coron. Vol. 9666. LNCS. Springer, 2016, pp. 735–763. DOI: 10.1007/978-3-662-49896-5_26.

[24] A. A. Badawi, A. Alexandru, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, C. Pascoe, Y. Polyakov, I. Quah, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca. *OpenFHE: Open-Source Fully Homomorphic Encryption Library*. Cryptology ePrint Archive, Paper 2022/915. https://eprint.iacr.org/2022/915. 2022.

[25] *Microsoft SEAL (release 4.1)*. https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA. Jan. 2023.

[26] Zama. *Concrete: TFHE Compiler that converts python programs into FHE equivalent*. https://github.com/zama-ai/concrete. 2022.

[27] S. Halevi and V. Shoup. “Design and implementation of HElib: a homomorphic encryption library”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 1481. URL: https://eprint.iacr.org/2020/1481.

[28] CryptoLab. *HEAAN: Homomorphic Encryption for Arithmetic of Approximate Numbers*. URL: https://heaan.it/.

[29] C. Gentry. “Fully homomorphic encryption using ideal lattices”. In: *ACM STOC 2009*. Ed. by M. Mitzenmacher. ACM, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[30] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *ITIC 2012*. Ed. by S. Goldwasser. ACM, 2012, pp. 309–325. DOI: 10.1145/2090236.2090262.

[31] Z. Brakerski. “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP”. In: *CRYPTO 2012*. Ed. by R. Safavi-Naini and R. Canetti. Vol. 7417. LNCS. Springer, 2012, pp. 868–886. DOI: 10.1007/978-3-642-32009-5_50.

[32] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. “(Leveled) Fully Homomorphic Encryption without Bootstrapping”. In: *ACM Trans. Comput. Theory* 6.3 (2014), 13:1–13:36. DOI: 10.1145/2633600.

[33] Z. Brakerski, C. Gentry, and S. Halevi. “Packed Ciphertexts in LWE-Based Homomorphic Encryption”. In: *PKC 2013*. Ed. by K. Kurosawa and G. Hanaoka. Vol. 7778. LNCS. Springer, 2013, pp. 1–13. DOI: 10.1007/978-3-642-36362-7_1.

[34] J. Fan and F. Vercauteren. “Somewhat Practical Fully Homomorphic Encryption”. In: *IACR Cryptol. ePrint Arch.* (2012), p. 144. URL: http://eprint.iacr.org/2012/144.

[35] C. Gentry, A. Sahai, and B. Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5.

[36] L. Ducas and D. Micciancio. “FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second”. In: *EUROCRYPT 2015, Part I*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 617–640. DOI: 10.1007/978-3-662-46800-5_24.

[37] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. “Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE”. In: *ASIACRYPT 2017, Part I*. Ed. by T. Takagi and T. Peyrin. Vol. 10624. LNCS. Springer, 2017, pp. 377–408. DOI: 10.1007/978-3-319-70694-8_14.

[38] D. Micciancio and Y. Polyakov. “Bootstrapping in FHEW-like Cryptosystems”. In: *WAHC 2021*. WAHC@ACM, 2021, pp. 17–28. DOI: 10.1145/3474366.3486924.

[39] M. Ajtai. “Generating hard instances of lattice problems”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108. URL: https://eccc.weizmann.ac.il/eccc-reports/1996/TR96-007/index.html.

[40] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *ACM 2005*. Ed. by H. N. Gabow and R. Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[41] D. Micciancio and O. Regev. “Worst-Case to Average-Case Reductions Based on Gaussian Measures”. In: *SIAM J. Comput.* 37.1 (2007), pp. 267–302. DOI: 10.1137/S0097539705447360.

[42] D. Aharonov and O. Regev. “Lattice problems in NP cap coNP”. In: *J. ACM* 52.5 (2005), pp. 749–765. DOI: 10.1145/1089023.1089025.

[43] D. Micciancio and C. Peikert. “Hardness of SIS and LWE with Small Parameters”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, 2013, pp. 21–39. DOI: 10.1007/978-3-642-40041-4_2.

[44] C. Peikert. “Public-key cryptosystems from the worst-case shortest vector problem: extended abstract”. In: *STOC 2009*. Ed. by M. Mitzenmacher. ACM, 2009, pp. 333–342. DOI: 10.1145/1536414.1536461.

[45] D. Micciancio and P. Mol. “Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions”. In: *CRYPTO 2011*. Ed. by P. Rogaway. Vol. 6841. LNCS. Springer, 2011, pp. 465–484. DOI: 10.1007/978-3-642-22792-9_26.

[46] D. X. Song, D. A. Wagner, and A. Perrig. “Practical Techniques for Searches on Encrypted Data”. In: *IEEE S&P 2000*. IEEE, 2000, pp. 44–55. DOI: 10.1109/SECPRI.2000.848445.

[47] R. Bost, B. Minaud, and O. Ohrimenko. “Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives”. In: *ACM CCS 2017*. Ed. by B. Thuraisingham, D. Evans, T. Malkin, and D. Xu. ACM, 2017, pp. 1465–1482. DOI: 10.1145/3133956.3133980.

[48] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. “Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation”. In: *NDSS 2014*. The Internet Society, 2014. DOI: 10.14722/ndss.2014.23264.

[49] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. “Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, 2013, pp. 353–373. DOI: 10.1007/978-3-642-40041-4_20.

[50] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. “Searchable symmetric encryption: improved definitions and efficient constructions”. In: *ACM CCS 2006*. Ed. by A. Juels, R. N. Wright, and S. D. C. di Vimercati. ACM, 2006, pp. 79–88. DOI: 10.1145/1180405.1180417.

[51] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. “Public Key Encryption with Keyword Search”. In: *EUROCRYPT 2004*. Ed. by C. Cachin and J. Camenisch. Vol. 3027. LNCS. Springer, 2004, pp. 506–522. DOI: 10.1007/978-3-540-24676-3_30.

[52] R. Zhang and H. Imai. “Combining Public Key Encryption with Keyword Search and Public Key Encryption”. In: *IEICE Trans. Inf. Syst.* 92-D.5 (2009), pp. 888–896. DOI: 10.1587/transinf.E92.D.888.

[53] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. “Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions”. In: *CRYPTO 2005*. Ed. by V. Shoup. Vol. 3621. LNCS. Springer, 2005, pp. 205–222. DOI: 10.1007/11535218_13.

[54] Q. Zheng, S. Xu, and G. Ateniese. “VABKS: Verifiable attribute-based keyword search over outsourced encrypted data”. In: *IEEE INFOCOM 2014*. IEEE, 2014, pp. 522–530. DOI: 10.1109/INFOCOM.2014.6847976.

[55] W. He, D. Akhawe, S. Jain, E. Shi, and D. X. Song. “ShadowCrypt: Encrypted Web Applications for Everyone”. In: *ACM CCS 2014*. Ed. by G. Ahn, M. Yung, and N. Li. ACM, 2014, pp. 1028–1039. DOI: 10.1145/2660267.2660326.

[56] B. Lau, S. P. Chung, C. Song, Y. Jang, W. Lee, and A. Boldyreva. “Mimesis Aegis: A Mimicry Privacy Shield-A System’s Approach to Data Privacy on Public Cloud”. In: *USENIX 2014*. Ed. by K. Fu and J. Jung. USENIX Association, 2014, pp. 33–48. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/lau.

[57] D. Micciancio and J. Sorrell. “Ring Packing and Amortized FHEW Bootstrapping”. In: *ICALP 2018*. Ed. by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 100:1–100:14. DOI: 10.4230/LIPICS.ICALP.2018.100.

[58] A. Guimarães, H. V. L. Pereira, and B. V. Leeuwen. “Amortized Bootstrapping Revisited: Simpler, Asymptotically-Faster, Implemented”. In: *ASIACRYPT 2023, Part VI*. Ed. by J. Guo and R. Steinfeld. Vol. 14443. LNCS. Springer, 2023, pp. 3–35. DOI: 10.1007/978-981-99-8736-8_1.

[59] F. Liu and H. Wang. “Batch Bootstrapping I: - A New Framework for SIMD Bootstrapping in Polynomial Modulus”. In: *EUROCRYPT 2023, Part III*. Ed. by C. Hazay and M. Stam. Vol. 14006. LNCS. Springer, 2023, pp. 321–352. DOI: 10.1007/978-3-031-30620-4_11.

[60] F. Liu and H. Wang. “Batch Bootstrapping II:” in: *EUROCRYPT 2023, Part III*. Ed. by C. Hazay and M. Stam. Vol. 14006. LNCS. Springer, 2023, pp. 353–384. DOI: 10.1007/978-3-031-30620-4_12.

[61] I. Chillotti, D. Ligier, J. Orfila, and S. Tap. “Improved Programmable Bootstrapping with Larger Precision and Efficient Arithmetic Circuits for TFHE”. In: *ASIACRYPT 2021, Part III*. Ed. by M. Tibouchi and H. Wang. Vol. 13092. LNCS. Springer, 2021, pp. 670–699. DOI: 10.1007/978-3-030-92078-4_23.

[62] I. Chillotti, M. Joye, and P. Paillier. “Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks”. In: *CSCML 2021*. Ed. by S. Dolev, O. Margalit, B. Pinkas, and A. A. Schwarzmann. Vol. 12716. LNCS. Springer, 2021, pp. 1–19. DOI: 10.1007/978-3-030-78086-9_1.

[63] M. S. Islam, M. Kuzu, and M. Kantarcioglu. “Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation”. In: *NDSS 2012*. The Internet Society, 2012. URL: https://www.ndss-symposium.org/ndss2012/access-pattern-disclosure-searchable-encryption-ramification-attack-and-mitigation.

[64] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. “Leakage-Abuse Attacks Against Searchable Encryption”. In: *ACM CCS 2015*. Ed. by I. Ray, N. Li, and C. Kruegel. ACM, 2015, pp. 668–679. DOI: 10.1145/2810103.2813700.

[65] L. Blackstone, S. Kamara, and T. Moataz. “Revisiting Leakage Abuse Attacks”. In: *NDSS 2020*. The Internet Society, 2020. DOI: 10.14722/ndss.2020.23103.

[66] J. Ning, X. Huang, G. S. Poh, J. Yuan, Y. Li, J. Weng, and R. H. Deng. “LEAP: Leakage-Abuse Attack on Efficiently Deployable, Efficiently Searchable Encryption with Partially Known Dataset”. In: *ACM CCS 2021*. Ed. by Y. Kim, J. Kim, G. Vigna, and E. Shi. ACM, 2021, pp. 2307–2320. DOI: 10.1145/3460120.3484540.

[67] Y. Zhang, J. Katz, and C. Papamanthou. “All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption”. In: *USENIX 2016*. Ed. by T. Holz and S. Savage. USENIX Association, 2016, pp. 707–720. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/zhang.

[68] G. Wang, Z. Cao, and X. Dong. “Improved File-injection Attacks on Searchable Encryption Using Finite Set Theory”. In: *Comput. J.* 64.8 (2021), pp. 1264–1276. DOI: 10.1093/COMJNL/BXAA161.

[69] E. Kirshanova. “Proxy Re-encryption from Lattices”. In: *PKC 2014*. Ed. by H. Krawczyk. Vol. 8383. LNCS. Springer, 2014, pp. 77–94. DOI: 10.1007/978-3-642-54631-0_5.

[70] K. Sakurai, T. Nishide, and A. Syalim. “Improved proxy re-encryption scheme for symmetric key cryptography”. In: *IWBIS, 2017*. IEEE, 2017, pp. 105–111. DOI: 10.1109/IWBIS.2017.8275110.

[71] S. Yamada. “Adaptively Secure Identity-Based Encryption from Lattices with Asymptotically Shorter Public Parameters”. In: *EUROCRYPT 2016, Part II*. Ed. by M. Fischlin and J. Coron. Vol. 9666. LNCS. Springer, 2016, pp. 32–62. DOI: 10.1007/978-3-662-49896-5_2.

[72] Y. Hsieh, H. Lin, and J. Luo. “Attribute-Based Encryption for Circuits of Unbounded Depth from Lattices”. In: *FOCS 2023*. IEEE, 2023, pp. 415–434. DOI: 10.1109/FOCS57990.2023.00031.

2

EQUIVALENCE OF UPDATABLE ENCRYPTION

Updatable encryption (UE) enables the cloud server to update the previously sourced encrypted data to a new key with only an update token received from the client. Two interesting works have been proposed to clarify the relationships among various UE security notions. Jiang (ASIACRYPT 2020) proved the equivalence of every security notion in the bi-directional and uni-directional key update settings and further, the security notion in the no-directional key update setting is strictly stronger than the above two. In contrast, Nishimaki (PKC 2022) proposed a new definition of uni-directional key update that is called the backward-leak uni-directional key update, and showed the equivalence relation by Jiang does not hold in this setting. We present a detailed comparison of every security notion in the four key update settings and prove that the security in the backward-leak uni-directional key update setting is actually equivalent to that in the no-directional key update setting. Our result reduces the hard problem of constructing no-directional key update UE schemes to the construction of those with backward-leak uni-directional key updates.

This chapter is based on the paper "*No-Directional and Backward-Leak Uni-Directional Updatable Encryption Are Equivalent*" by **Chen, H.**, Fu, S., and Liang, K. in ESORICS (1) 2022: 387-407

2.1. INTRODUCTION

When a client stores encrypted data on a cloud server, a good way of key management is to change keys periodically, so as to resist the risk of key leakage. This process is referred to as key rotation, in which the core of the update relies on how to update the previous encrypted data to be decryptable by a new key. A possible way is to download and decrypt the encrypted data with the old key, and then encrypt the data with the new key and upload the new encrypted data again. But the download and upload process would be extremely expensive if there exists a considerable amount of data.

Updatable encryption (UE) [1] provides a practical solution to the above dilemma. Its core idea is that the client offers the cloud server the ability to update ciphertexts by the update tokens, with a requirement that the update token should not leak any information about the data. There are two flavors of UE depending on if ciphertexts are needed in the generation of the update token. One is called ciphertext-dependent UE [1–4], in which clients need to download partial components of the encrypted data, called ciphertext header, from the cloud to generate the update token, and the token can only update the corresponding ciphertext. The other, more practical than the previous one, is known as ciphertext-independent UE [5–10], in which the token only depends on the old and the new keys, and a single token can update all existing ciphertexts. In this work, we only focus on the latter.

Security Notions. The security of UE schemes should be maintained even under a temporary corruption of keys and update tokens. The original UE construction and its security model against passive adversaries were proposed by Boneh et al. [1], where the adversary in the security game should specify the epoch keys it wishes to know before sending queries to the challenger. The confidentiality notions were further strengthened by [5, 7, 8], which attempts to capture the practical abilities of the adaptive attacker. The adversary can corrupt epoch keys and updated tokens at any time during the game as long as it does not trigger the trivial win conditions, which will be checked after the adversary submits its guessing bit (more details can be found in Section 2.2). A summary of existing confidentiality notions and their major difference is presented in Fig. 2.1.

	Challenge Input	Challenge Output
IND-Enc-notion [8]	$(\bar{m}_0, \bar{m}_1)$	$(\mathsf{Enc}(\bar{m}_0), \mathsf{Enc}(\bar{m}_1))$
IND-Upd-notion [8]	$(\bar{c}_0, \bar{c}_1)$	$\big(\mathsf{Upd}(\bar{c}_0), \mathsf{Upd}(\bar{c}_1)\big)$
IND-UE-notion [5]	$(\bar{m}_0, \bar{c}_1)$	$\big(\mathsf{Enc}(\bar{m}_0), \mathsf{Upd}(\bar{c}_1)\big)$

Figure 2.1: A summary of confidentiality notions, where $\mathsf{notion} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. The adversary in each confidentiality game provides two challenge inputs based on the oracles it has access to and tries to distinguish the challenge outputs.

Boyd et al. [5] proved that IND-UE-notion is strictly stronger even than the combination of the prior two (in Fig. 2.1) defined in [8]. They further proposed an

integrity notion called IND-CTXT. Jiang [6] defined another integrity notion called IND-PTXT. In the CTXT game, the adversary tries to provide a valid ciphertext that is different from the ciphertexts obtained during the game by the challenger; while in the PTXT game, the adversary needs to provide a valid ciphertext, whose underlying plaintext has not been queried during the game. Those two integrity notions are similar to the integrity notions of symmetric encryption schemes, but the adversary is provided with oracles specified in UE and trivial win conditions are also checked after the adversary submits its forgery.

Hereafter by security notions, we mean the set of all confidentiality and integrity notions in [5] and [6]: {detIND-UE-CPA,randIND-UE-CPA,detIND-UE-CCA, randIND-UE-CCA, IND-CTXT, IND-PTXT}, where det/rand denotes the ciphertext updates are deterministic or randomized, respectively.

Key Update Directions. The update token is generated by two successive epoch keys via the token generation algorithm, i.e., $\Delta_{e,e+1} = \mathsf{TokenGen}(k_e, k_{e+1})$ (defined in Section 2.2); therefore the adversary may derive one of the two successive keys from the other if the update token is known. Jiang [6] investigated three key update directions: bi-directional key updates in which both the old key k_e and the new key k_{e+1} can be derived from the other, uni-directional key updates in which only the new key k_{e+1} can be derived from the old key k_e but k_e cannot be derived from k_{e+1}, and no-directional key updates in which no keys in the two successive epoch keys can be derived from the other. The direction of key update affects the computation of leakage information known to the adversary, which in turn affects the computation of trivial win conditions as well as security notions. However, the main result in [6] shows that the security notions in the bi-directional key update setting and in the uni-directional key update setting are equivalence, while the security notions in the no-directional key update setting are strictly stronger.

Nishimaki [9] recently introduced a new definition of uni-directional key update that is called the backward-leak uni-directional key update for distinction, where the update direction is the opposite of the original uni-directional key update in [6] (called the forward-leak uni-directional key update for distinction). That is, the old key k_e can be derived from the new key k_{e+1}, but k_{e+1} cannot be derived from k_e. Nishimaki [9] demonstrated a contrasting conclusion that the security notions in the backward-leak uni-directional key update setting are not equivalent to those in the bi-directional directional key update setting.

But the relations among UE schemes in the four kinds of keys update settings have not been fully investigated yet. Thus, a natural interesting open problem that should be clear before any valuable constructions is as follows:

What are the relations among UE schemes in the bi-directional, forward-leak uni-directional, backward-leak uni-directional, and no-directional key update settings?

Our Contributions. At first glance, one may think that UE schemes with no-directional key updates should be strictly strong than UE with all the other three key update directions, just as proved in [6] that no-directional key updates setting

leaks less information about keys, tokens and ciphertexts than the bi-directional and forward-leak uni-directional key updates. However, our main result provides a surprising result that, for each security notion, no-directional key update UE schemes, which were believed to be strictly stronger than the other directional key update schemes, are actually equivalent to those with backward-leak uni-directional key updates.

Our main technique is to analyze the relations of the trivial win conditions for each security notion in different key update settings. As in other semantic security definitions, the adversary in the security game for UE is provided access to different oracles to capture realistic attack models. However, it may lead to a trivial win if the adversary queries some combinations of oracles. Therefore, the bookkeeping technique was developed in [7, 8] that tracks the leakage information of tokens, keys, and ciphertexts known to the adversary during the game and checks if those leakages may lead the adversary to trivially win after the adversary submits its guessing bit. The direction of key update affects the computation of leakage information and thus, we analyze the relations among UE schemes by analyzing the relations of trivial win conditions in different key update directions, especially the backward-leak uni-directional key update which was not covered by [6].

Based on our result, when analyzing the security notions, we can treat UE schemes with no-directional key updates as those with backward-leak uni-directional key updates. Currently, there are only two no-directional key update UE schemes in the literature: one is built on Ciphertext Puncturable Encryption [10] and the other is built on one-way functions and indistinguishability obfuscation [9]. Our result can eliminate the need for constructing UE schemes with no-directional key dates while also keeping security, since it is sufficient to construct UE schemes with backward-leak uni-directional key updates, which is much easier than the former.

Related Work UE schemes can be built from various cryptographic primitives. The seminal UE scheme BLMR was proposed by [1] as an application of almost key homomorphic pseudorandom functions, which satisfies IND-ENC instead of IND-UPD. An ElGamal-based scheme RISE was introduced by [8] to achieve both security definitions. To provide integrity protection, Klooß et al. [7] constructed two generic schemes based on Encrypt-and-MAC and the Naor-Yung transform [11]. Boyd et al. [5] designed three IND-UE-CPA secure schemes, called SHINE, based on the random-looking permutation. Jiang [6] provided a quantum-resistant scheme based on the decisional LWE [12]. The first UE scheme with backward uni-directional key was presented in Nishimaki [9] based on the Regev PKE scheme, in which a scheme with no-directional key updates is also constructed based on one-way functions [13] and indistinguishability obfuscation [14]. Slamanig and Striecks presented a pairing backward uni-directional scheme and a pairing-based no-directional scheme from ciphertext puncturable encryption [10].

2.2. UPDATABLE ENCRYPTION

We review the syntax of UE and the confidentiality and integrity definitions.

Definition 2.2.1 ([7]). *A* UE *scheme includes a tuple of PPT algorithms* {UE.KG, UE.Enc, UE.Dec, UE.TG, UE.Upd} *that operate in epochs, starting from 0.*

- $\mathsf{UE.KG}(1^\lambda)$*: the key generation algorithm outputs an epoch key* $\mathsf{k_e}$.
- $\mathsf{UE.Enc}(\mathsf{k_e}, \mathsf{m})$*: the encryption algorithm takes as input an epoch key* $\mathsf{k_e}$ *and a message* m *and outputs a ciphertext* $\mathsf{c_e}$.
- $\mathsf{UE.Dec}(\mathsf{k_e}, \mathsf{c_e})$*: the decryption algorithm takes as input an epoch key* $\mathsf{k_e}$ *and a ciphertext* $\mathsf{c_e}$ *and outputs a message* m'.
- $\mathsf{UE.TG}(\mathsf{k_e}, \mathsf{k_{e+1}})$*: the token generation algorithm takes as input two epoch keys* $\mathsf{k_e}$ *and* $\mathsf{k_{e+1}}$ *and outputs a token* Δ_{e+1}.
- $\mathsf{UE.Upd}(\Delta_{e+1}, \mathsf{c_e})$*: the update algorithm takes as input a token* Δ_{e+1} *and a ciphertext* $\mathsf{c_e}$ *and outputs a ciphertext* $\mathsf{c_{e+1}}$.

Correctness for UE means that any valid ciphertext and its updates should be decrypted to the correct message under the appropriate epoch key. The definitions of confidentiality and integrity for UE are given in Definition 2.2.2 and Definition 2.2.3, respectively. In general, the adversary in each security game is provided with access to different oracles, which enables it to obtain information about epoch keys, update tokens and ciphertexts from the challenger. In the challenge phase of the confidentiality game, the adversary submits a challenge message $\bar{\mathsf{m}}$ and a challenge ciphertext $\bar{\mathsf{c}}$ according to the information it already has and receives a ciphertext from the challenger, and its goal is to guess the received ciphertext is an encryption of the message of $\bar{\mathsf{m}}$ or an update of $\bar{\mathsf{c}}$. Then the adversary can continue to query the oracles and eventually provides a guessing bit. In the integrity game, the goal of the adversary is to forge a new valid ciphertext. In both security games, some combinations of oracles may lead to a trivial win of the game for the adversary, so the challenger will check if those trivial win conditions are triggered during the game by a bookkeeping technique developed in [8].

An overview of the oracles that the adversary has access to is shown in Fig. 2.4, how to compute the leakage set and its extension are described in Section 2.2.1, and the trivial win conditions in different security games are presented in Section 2.2.2.

Definition 2.2.2 (Confidentiality, [5]). *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}\}$ *be an updatable encryption scheme. For* $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$*, the* notion *advantage of an adversary* $\mathscr{A}$ *is defined as:* $\mathsf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{notion\text{-}1}}_{\mathsf{UE},\mathscr{A}} = 1] - \Pr[\mathsf{Exp}^{\mathsf{notion\text{-}0}}_{\mathsf{UE},\mathscr{A}} = 1]\right|$*, where the experiment* $\mathsf{Exp}^{\mathsf{notion\text{-}b}}_{\mathsf{UE},\mathscr{A}}$ *is given in Fig. 2.2 and Fig. 2.4, and* det *and* rand *denote the ciphertext update procedure is deterministic and randomized, respectively. We say a UE scheme is* notion *secure if* $\mathsf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) \le \mathsf{negl}(\lambda)$.

Definition 2.2.3 (Integrity, [5, 6]). *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}\}$ *be an updatable encryption scheme. For* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$*, the* notion *advantage of an adversary* $\mathscr{A}$ *is defined as:* $\mathsf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}} = 1]\right|$*, where the experiment*

$\mathsf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathscr{A}}$:

1 : **do Setup**; phase $\leftarrow 0$
2 : $\mathsf{b}' \leftarrow \mathscr{A}^{\mathscr{O}}(1^\lambda)$
3 : **if** $\big((\mathscr{K}^* \cup \mathscr{C}^* \neq \emptyset)$ **or** (xx = det and
4 : $(\tilde{\mathsf{e}} \in \mathscr{T}^*$ **or** $\mathscr{O}.\mathsf{Upd}(\tilde{c})$ is required$))\big)$ **then**
5 : twf $\leftarrow 1$
6 : **if** twf $= 1$ **then**
7 : $\mathsf{b}' \xleftarrow{\$} \{0,1\}$
8 : **return** b'

Figure 2.2: Generic description of the confidentiality experiment $\mathsf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathscr{A}}$ for $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}\}$, $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$ and $\mathsf{b} \in \{0,1\}$. The flag $\mathsf{phase} \in \{0,1\}$ denotes whether or not $\mathscr{A}$ has queried the $\mathscr{O}.\mathsf{Chall}$ oracle, and twf tracks if the trivial win conditions are triggered, and $\mathscr{O} = \mathscr{O}.\{\mathsf{Enc}, \mathsf{Next}, \mathsf{Upd}, \mathsf{Corr}, \mathsf{Chall}, \mathsf{Upd}\tilde{\mathsf{C}}\}$ is the set of oracles $\mathscr{A}$ can access to, which are defined in Fig. 2.4. When $\mathsf{atk} = \mathsf{CCA}$, the decryption oracle $\mathscr{O}.\mathsf{Dec}$ is also added to $\mathscr{O}$. The computation of $\mathscr{K}^*, \mathscr{T}^*, \mathscr{C}^*$ are discussed in Section 2.2.1.

$\mathsf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}$ *is given in Fig. 2.3 and Fig. 2.4. We say a UE scheme is* notion *secure if* $\mathsf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

$\mathsf{Exp}^{\mathsf{IND\text{-}atk}}_{\mathsf{UE},\mathscr{A}}$:

1 : **do Setup**; win $\leftarrow 0$
2 : $\mathscr{A}^{\mathscr{O}}(1^\lambda)$
3 : **if** twf $= 1$ **then**
4 : win $\leftarrow 0$
5 : **return** win

Figure 2.3: Generic description of the confidentiality experiment $\mathsf{Exp}^{\mathsf{IND\text{-}atk}}_{\mathsf{UE},\mathscr{A}}$ for $\mathsf{atk} \in \{\mathsf{CTXT}, \mathsf{PTXT}\}$. The flag win tracks whether or not $\mathscr{A}$ provided a valid forgery, twf tracks if the trivial win conditions are triggered, and $\mathscr{O} = \mathscr{O}.\{\mathsf{Enc}, \mathsf{Next}, \mathsf{Upd}, \mathsf{Corr}, \mathsf{Try}\}$ is the set of oracles $\mathscr{A}$ can access to, which are defined in Fig. 2.4.

2.2.1. LEAKAGE SETS

In security games of UE, the adversary is provided access to various oracles as shown in Fig. 2.4, so it can learn some information about update keys, epoch tokens and ciphertexts during the query phase. Moreover, it can extend the information via

Setup(1^λ):

$k_0 \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$
$\Delta_0 \leftarrow \bot; e, c, \mathsf{twf} \leftarrow 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Enc}(m)$:

$c \leftarrow c + 1$
$c \xleftarrow{\$} \mathsf{UE.ENC}(k_e, m))$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(c, c, e; m)\}$
return c

$\mathcal{O}.\mathsf{Dec}(c)$:

m' **or** $\bot \leftarrow \mathsf{UE.Dec}(k_e, c)$
if $\big($(xx = det **and** $(c, e) \in \tilde{\mathcal{L}}^*$) **or**
(xx = rand **and** $(m', e) \in \tilde{\mathcal{Q}}^*$)$\big)$ **then**
$\mathsf{twf} \leftarrow 1$
return m' **or** $\bot$

$\mathcal{O}.\mathsf{Next}()$:

$e \leftarrow e + 1$
$k_e \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$
$\Delta_e \leftarrow \mathsf{UE.TG}(k_{e-1}, k_e)$
if phase = 1 **then**
$\tilde{c}_e \leftarrow \mathsf{UE.Upd}(\Delta_e, \tilde{c}_{e-1})$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{e})$

if $\hat{e} > e$ **then**
return $\bot$
if inp = key **then**
$\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{e}\}$
return $k_{\hat{e}}$
if inp = token **then**
$\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{e}\}$
return $\Delta_{\hat{e}}$

$\mathcal{O}.\mathsf{Upd}(c_{e-1})$:

if $(j, c_{e-1}, e-1; m) \notin \mathcal{L}$ **then**
return $\bot$
$c_e \leftarrow \mathsf{UE.Upd}(\Delta_e, c_{e-1})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, c_e, e; m)\}$

$\mathcal{O}.\mathsf{Chall}(\bar{m}, \bar{c})$

if phase = 1 **then**
return $\bot$
phase $\leftarrow 1; \tilde{e} \leftarrow e$
if $(\cdot, \bar{c}, e-1; \bar{m}_1) \notin \mathcal{L}$ **then**
return $\bot$
if b = 0 **then**
$\tilde{c}_{\tilde{e}} \leftarrow \mathsf{UE.Enc}(k_{\tilde{e}}, \bar{m})$
else
$\tilde{c}_{\tilde{e}} \leftarrow \mathsf{UE.Upd}(\Delta_{\tilde{e}}, \bar{c})$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{e}\}$
$\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{c}_{\tilde{e}}, \tilde{e})\}$
return $\tilde{c}_{\tilde{e}}$

$\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$

if phase $\neq 1$ **then**
return $\bot$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{e\}$
$\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{c}_e, e)\}$
return $\tilde{c}_e$

$\mathcal{O}.\mathsf{Try}(\bar{c})$

m' **or** $\bot \leftarrow \mathsf{UE.Dec}(k_e, \bar{c})$
if $\big(e \in \mathcal{K}^*$ **or** (atk = CTXT **and** $(\bar{c}, e) \in \mathcal{L}^*$) **or**
(atk = PTXT **and** $(m', e) \in \tilde{\mathcal{Q}}^*$)$\big)$ **then**
$\mathsf{twf} \leftarrow 1$
if $m' \neq \bot$ **then**
$\mathsf{win} \leftarrow 1$

Figure 2.4: Oracles in the UE security games, where the message m_1 is the underlying message of the challenge input ciphertext $\bar{c}$. The leakage sets $\mathcal{L}, \widetilde{\mathcal{L}}, \mathcal{L}^*, \widetilde{\mathcal{L}}^*, \mathcal{C}, \mathcal{K}, \mathcal{K}^*, \mathcal{T}, \mathcal{T}^*, \mathcal{Q}, \mathcal{Q}^*, \tilde{\mathcal{Q}}^*$ are defined in Section 2.2.1.

its known tokens, and the extension depends on the direction of key update and the direction of ciphertext update. We start by describing the leakage sets in [7, 8], and then show how to compute the extended leakage sets in different key update direction settings.

Epoch Leakage Sets. We record the following epoch sets related to epoch keys, update tokens, and challenge-equal ciphertexts.

- $\mathscr{K}$: Set of epochs in which the adversary corrupted the epoch key from $\mathcal{O}$.Corr.
- $\mathscr{T}$: Set of epochs in which the adversary corrupted the update token from $\mathcal{O}$.Corr.
- $\mathscr{C}$: Set of epochs in which the adversary learned a challenge-equal ciphertext (the ciphertext related to the challenge inputs) from $\mathcal{O}$.Chall or $\mathcal{O}$.Upd$\widetilde{\mathsf{C}}$.

The adversary can use its corrupted tokens to extend $\mathscr{K}, \mathscr{T}, \mathscr{C}$ to infer more information. We use $\mathscr{K}^*, \hat{\mathscr{K}}^*, \mathscr{T}^*, \mathscr{C}^*$ as the extended sets respectively, and how to compute $\mathscr{K}^*, \hat{\mathscr{K}}^*, \mathscr{T}^*, \mathscr{C}^*$ are shown later.

Information Leakage Sets. We record the following sets related to ciphertexts known to the adversary.

- $\mathscr{L}$: Set of non-challenge equal ciphertexts $(c, \mathsf{c}, \mathsf{e}; \mathsf{m})$ the adversary learned from $\mathcal{O}$.Enc or $\mathcal{O}$.Upd.
- $\widetilde{\mathscr{L}}$: Set of challenge-equal ciphertexts $(\tilde{\mathsf{c}}_\mathsf{e}, \mathsf{e})$ the adversary learned from $\mathcal{O}$.Chall or $\mathcal{O}$.Upd$\widetilde{\mathsf{C}}$.

The adversary can also use its corrupted tokens to extend $\mathscr{L}, \widetilde{\mathscr{L}}$ to infer more information about ciphertexts. In the deterministic update setting, we denote $\mathscr{L}^*, \widetilde{\mathscr{L}}^*$ as the extended sets of $\mathscr{L}$ and $\widetilde{\mathscr{L}}$, respectively.

In randomized UE schemes, we use $\mathscr{Q}^*, \widetilde{\mathscr{Q}}^*$ to denote respectively the extended sets of $\mathscr{L}, \widetilde{\mathscr{L}}$:

- $\mathscr{Q}^*$: Set of plaintexts (m, e). The adversary learned in the query phase or could create a ciphertext of m in the epoch e.
- $\widetilde{\mathscr{Q}}^*$: Set of challenge plaintexts $\{(\bar{\mathsf{m}}, \mathsf{e}), (\bar{\mathsf{m}}_1, \mathsf{e})\}$, where $(\bar{\mathsf{m}}, \bar{\mathsf{c}})$ is the query input of $\mathcal{O}$.Upd and $\bar{\mathsf{m}}_1$ is the plaintext of $\bar{\mathsf{c}}$. The adversary learned in the query phase or could create a ciphertext of $\bar{\mathsf{m}}$ or $\bar{\mathsf{m}}_1$ in the epoch e.

Inferred Leakage Sets. The adversary can infer more information from $\mathscr{K}$, $\mathscr{L}$ and $\mathscr{C}$ via its corrupted tokens, which are computed as follows.

Key Leakage. Since the update tokens are generated from two successive epoch keys by $\Delta_{\mathsf{e}+1} = \mathsf{UE.TG}(\mathsf{k}_\mathsf{e}, \mathsf{k}_{\mathsf{e}+1})$, one epoch key in $\{\mathsf{k}_\mathsf{e}, \mathsf{k}_{\mathsf{e}+1}\}$ may be inferred by the other via the known token $\Delta_{\mathsf{e}+1}$.

- No-directional key updates: $\mathcal{K}^*_{\mathsf{no}} = \mathcal{K}$. The adversary does have more information about keys except $\mathcal{K}$, since tokens cannot be used to derive keys.
- Forward-leak uni-directional key updates:

$$\mathcal{K}^*_{\mathsf{f\text{-}uni}} = \{\mathsf{e} \in \{0,\ldots,l\} \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\},$$

  where $\mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \Longleftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T})$. The adversary can infer more keys from corrupted tokens and keys in the previous epoch.
- Backward-leak uni-directional key updates:

$$\mathcal{K}^*_{\mathsf{b\text{-}uni}} = \{\mathsf{e} \in \{0,\ldots,l\} \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}, \tag{2.1}$$

  where $\mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \Longleftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T})$. Keys can be inferred from corrupted tokens and keys in the next epoch.
- Bi-directional key updates:

$$\mathcal{K}^*_{\mathsf{bi}} = \{\mathsf{e} \in \{0,\ldots,l\} \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}, \tag{2.2}$$

  where $\mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \Longleftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T})$. Besides the corrupted keys $\mathcal{K}$, the adversary can infer more keys both from key upgrades and downgrades, i.e., $\mathcal{K}^*_{\mathsf{bi}} = \mathcal{K}^*_{\mathsf{f\text{-}uni}} \cup \mathcal{K}^*_{\mathsf{b\text{-}uni}}$.

In the integrity game, a set $\hat{\mathcal{K}}$ is defined to check if the adversary can trivially forge a valid ciphertext as follows.

$$\begin{aligned} \hat{\mathcal{K}}^* &= \{\mathsf{i} \in \{0,\ldots,l\} \mid \mathsf{ForgK}(i) = \mathsf{true}\} \\ &\mathsf{true} \leftarrow \mathsf{ForgK}(i) \Longleftrightarrow (\mathsf{i} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}) \end{aligned} \tag{2.3}$$

Token Leakage. The adversary knows a token by either corrupting or inferring from two successive keys. Then we have

$$\mathcal{T}^*_{\mathsf{kk}} = \{\mathsf{e} \in \{0,\ldots,l\} \mid (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^*_{\mathsf{kk}} \wedge \mathsf{e}-1 \in \mathcal{K}^*_{\mathsf{kk}}\}, \tag{2.4}$$

for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$.

Ciphertext Leakages. Different from the direction of key update, ciphertexts should always be upgraded but are not necessarily downgraded by tokens, so there are only two types of ciphertext directions.

- Uni-directional ciphertext updates:

$$\mathscr{C}^*_{\mathsf{kk,uni}} = \{\mathsf{e} \in \{0,\ldots,l\} \mid \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}, \tag{2.5}$$

  where $\mathsf{ChallEq}(\mathsf{e}) = \mathsf{true} \Longleftrightarrow (\mathsf{e} \in \mathscr{C}) \vee (\mathsf{ChallEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}})$. Besides the learned ciphertext $\mathscr{C}$, the adversary can infer more ciphertexts from corrupted tokens and ciphertexts in the previous epoch.

- Bi-directional ciphertext updates:

$$\mathscr{C}^*_{\mathsf{kk,bi}} = \{\mathsf{e} \in \{0,\ldots,l\} \mid \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}, \tag{2.6}$$

where $\mathsf{ChallEq}(\mathsf{e}) = \mathsf{true} \Leftrightarrow (\mathsf{e} \in \mathscr{C}) \vee (\mathsf{ChallEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathscr{T}^*_{\mathsf{kk}}) \vee (\mathsf{ChallEq}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathscr{T}^*_{\mathsf{kk}})$. Besides the learned ciphertext $\mathscr{C}$, the adversary can infer more ciphertexts both from key upgrades and downgrades.

Remark 2.2.4. *From the definition, the leakage sets have the following relations,*

- $(\tilde{\mathsf{c}}_\mathsf{e}, \mathsf{e}) \in \widetilde{\mathscr{L}} \Longleftrightarrow \mathsf{e} \in \mathscr{C}$,
- $(\tilde{\mathsf{c}}_\mathsf{e}, \mathsf{e}) \in \widetilde{\mathscr{L}}^* \Longleftrightarrow \mathsf{e} \in \mathscr{C}^* \Longleftrightarrow \{(\tilde{\mathsf{m}}, \mathsf{e}), (\tilde{\mathsf{m}}_1, \mathsf{e})\} \in \widetilde{\mathscr{Q}}^*$.

Example. An example is given in Fig. 2.5 to show how to compute leakage sets. We assume the adversary corrupts epoch keys in epochs in $\mathscr{K} = \{\mathsf{e}-5, \mathsf{e}-4, \mathsf{e}-3, \mathsf{e}-1\}$ and corrupts tokens in $\mathscr{T} = \{\mathsf{e}-4, \mathsf{e}-3, \mathsf{e}-1, \mathsf{e}\}$, and queries a non-challenge ciphertext, say $\mathbf{c}_{\mathsf{e}-5}$, in epoch $\mathsf{e}-5$.

In the no-directional key update setting, the adversary cannot infer extra keys and tokens, i.e., $\mathscr{K}^*_{\mathsf{no}} = \mathscr{K}$ and $\mathscr{T}^*_{\mathsf{no}} = \mathscr{T}$. However, it can infer ciphertexts in epoch $\mathsf{e}-4, \mathsf{e}-3$ by using $\mathbf{c}_{\mathsf{e}-5}$ and tokens in epochs $\mathsf{e}-4$ and $\mathsf{e}-3$, but cannot infer the ciphertexts in epochs from $\mathsf{e}-2$ to e, because the token in epoche -2 is unknown to the adversary in the no-directional key update setting.

Epoch	e−5	e−4	e−3	e−2	e−1	e
$\mathscr{K}$	✓	✓	✓	×	✓	×
$\mathscr{T}$	×	✓	✓	×	✓	✓
$\mathscr{K}^*_{\mathsf{no}}$	✓	✓	✓	×	✓	×
$\mathscr{T}^*_{\mathsf{no}}$	×	✓	✓	×	✓	✓
$\mathscr{K}^*_{\mathsf{b\text{-}uni}}$	✓	✓	✓	✓	✓	×
$\mathscr{T}^*_{\mathsf{b\text{-}uni}}$	×	✓	✓	✓	✓	✓
$\mathscr{K}^*_{\mathsf{f\text{-}uni}}$	✓	✓	✓	×	✓	✓
$\mathscr{T}^*_{\mathsf{f\text{-}uni}}$	×	✓	✓	×	✓	✓

Figure 2.5: Example of leakage sets. Marks ✓ and × indicate if an epoch key or epoch token is corrupted. The green mark ✓ indicates an epoch key or epoch token can be inferred from other corrupted keys and tokens.

In the backward uni-directional key update setting, the adversary can infer the key in epoch $\mathsf{e}-2$ from the known token and key in epoch $\mathsf{e}-1$, and further infer the token in the epoch $\mathsf{e}-2$, since the key in epoch $\mathsf{e}-3$ is also corrupted, i.e.,

$\mathcal{K}^*_{\text{b-uni}} = \{\mathsf{e}-5,\ldots,\mathsf{e}-1\}$ and $\mathcal{T}^*_{\text{b-uni}} = \{\mathsf{e}-4,\ldots,\mathsf{e}\}$. Moreover, it can infer the ciphertexts in epoch from $\mathsf{e}-4$ to e by $\mathbf{c}_{\mathsf{e}-5}$ and $\mathcal{T}^*_{\text{b-uni}}$.

In the forward uni-directional key update setting, the adversary cannot infer the token in epoch $\mathsf{e}-2$, since the key in epoch $\mathsf{e}-2$ is unknown to it. But it can infer the key in epoch e via the known key in $\mathsf{e}-1$ and the known token in e, i.e., $\mathcal{K}^*_{\text{f-uni}} = \mathcal{K} \cup \{\mathsf{e}\}$ and $\mathcal{T}^*_{\text{f-uni}} = \mathcal{K}^*$. The ciphertext it can learn is the same as that in the no-directional key update setting.

2.2.2. TRIVIAL WIN CONDITIONS

In the security games of UE, the leaked information probably leads the adversary to trivially win the game. The challenger will check if any trivial win condition is triggered at the end of the game. A summary of trivial win conditions is described in Fig. 2.6, which follows from the analysis in [5–8].

notion	$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$	$\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{\mathsf{c}})$ is queried	$(\mathsf{c},\mathsf{e}) \in \tilde{\mathcal{L}}^*$	$(\mathsf{m}',\mathsf{e}) \in \tilde{\mathcal{Q}}^*$	$\mathsf{e} \in \tilde{\mathcal{K}}^*$	$(\bar{\mathsf{c}},\mathsf{e}) \in \mathcal{L}^*$	$(\mathsf{m}',\mathsf{e}) \in \mathcal{Q}^*$
detIND-UE-CPA	✓	✓	×	×	×	×	×
randIND-UE-CPA	✓	×	×	×	×	×	×
detIND-UE-CCA	✓	✓	✓	×	×	×	×
randIND-UE-CCA	✓	×	×	✓	×	×	×
IND-CTXT	×	×	×	×	✓	✓	×
IND-PTXT	×	×	×	×	✓	×	✓

Figure 2.6: Trivial win conditions in different security games for updatable encryption. ✓ and × indicate whether the security notion considers the corresponding trivial win conditions or not. $\tilde{\mathsf{e}}$ is the challenge epoch, i.e., the epoch the adversary queries $\mathcal{O}.\mathsf{Chall}$, and e represents the current epoch.

We give a detailed explanation for the trivial win conditions in each security game. If $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$, then there is an epoch i that the adversary knows the epoch key k_i and a challenge-equal ciphertext $\mathbf{c}_i$ in the same epoch. Then the adversary can decrypt the challenge-equal ciphertext $\mathbf{c}_i$ with its known key k_i and get the underlying message of $\mathbf{c}_i$. Thus, it can trivially win the game by comparing the underlying message of $\mathbf{c}_i$ with the challenge input message $\tilde{\mathbf{m}}$. This condition should be checked in all confidentiality games.

For deterministic UE schemes, if $\tilde{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}.\mathsf{Upd}(\bar{\mathsf{c}})$ is queried, then the adversary

can obtain the updated ciphertext $\mathbf{c}_1$ of the challenge input ciphertext $\bar{\mathbf{c}}$, and therefore trivially win the game by comparing $\mathbf{c}_1$ with the ciphertext it receives from its challenger.

In the CCA attack, the adversary has the access to the decryption oracle. For deterministic UE schemes, if $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$, the adversary can query the decryption oracle on the challenge-equal ciphertext $\mathbf{c}$ and receive its underlying message. For a randomized UE, it should also be prohibited if the decryption returns a message $\mathbf{m}'$ such that $\mathbf{m}' = \mathbf{m}_0$ or $\mathbf{m}_1$, which is checked by $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$.

If the adversary knows a ciphertext $\mathbf{c}_{\mathsf{e}_0}$ in epoch e_0 and all tokens from epoch e_0 to e, it can forge a valid ciphertext in epoch e by updating $\mathbf{c}_{\mathsf{e}_0}$ via the tokens from e_0 to e. It should be checked in both integrity games if $\mathsf{e} \in \hat{\mathcal{K}}^*$ which is defined as Eq. (2.3). The challenger should also check if $\tilde{\mathbf{c}}$ is a new ciphertext in the CTXT game, and if the adversary knows a ciphertext of $\mathbf{m}'$ in the CTXT game.

We now review some properties of leakage sets as follows.

Definition 2.2.5 (Firewall, [5–8])**.** *An insulated region with firewalls* fwl *and* fwr*, denoted by* $\mathcal{FW}$*, is the consecutive sequence of epochs* $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ *for which:*

- *no key in the sequence of epochs* $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ *is corrupted;*
- *the tokens* Δ_{fwl} *and* $\Delta_{\mathsf{fwr}+1}$ *are not corrupted;*
- *all tokens* $\{\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}}\}$ *are corrupted.*

Denote the union of all firewalls as $\mathcal{IR} := \bigcup_{(\mathsf{fwl},\mathsf{fwr}) \in \mathcal{FW}} \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$. The following lemma shows that $\mathcal{IR}$ is the complementary set of $\mathcal{K}^*_{\mathsf{bi}}$.

Lemma 2.2.6 (Lemma 3.1, [6])**.** *For any* $\mathcal{K}, \mathcal{T} \in \{0, \ldots, l\}$*, we have* $\mathcal{K}^*_{\mathsf{bi}} = \{0, \ldots, l\} \setminus \mathcal{IR}$*, where* l *is the maximal number of updates.*

Corollary 2.2.7. *Since* $\mathcal{K}^*_{\mathsf{bi}} = \mathcal{K}^*_{\mathsf{f\text{-}uni}} \cup \mathcal{K}^*_{\mathsf{b\text{-}uni}}$ *by definition, we have* $\{0, \ldots, l\} = \mathcal{IR} \cup \mathcal{K}^*_{\mathsf{f\text{-}uni}} \cup \mathcal{K}^*_{\mathsf{b\text{-}uni}}$.

Remark 2.2.8. *For an epoch* $\mathsf{e} \in \mathcal{K}^*_{\mathsf{f\text{-}uni}}$*, it holds that either* $\mathsf{e} \in \mathcal{K}$ *or there exists an epoch* e_f *before* e*, such that* $\mathsf{e}_\mathsf{f} \in \mathcal{K}$ *and* $\{\mathsf{e}_\mathsf{f}, \ldots, \mathsf{e}\} \in \mathcal{T}$*; for an epoch* $\mathsf{e} \in \mathcal{K}^*_{\mathsf{b\text{-}uni}}$*, it holds that either* $\mathsf{e} \in \mathcal{K}$ *or there exists an epoch* e_b *after* e*, such that* $\mathsf{e}_\mathsf{b} \in \mathcal{K}$ *and* $\{\mathsf{e}, \ldots, \mathsf{e}_\mathsf{b}\} \in \mathcal{T}$*. That follows directly from the definition.*

2.3. RELATIONS AMONG SECURITY NOTIONS

To capture the security for UE schemes with kk-directional key updates and cc-directional ciphertext updates, we consider the (kk, cc)-variant of each security notion as defined in [6] (where $\mathsf{kk} \in \{\mathsf{bi}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{no}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$), and then compare the relations among all the variants of each security notion.

Definition 2.3.1 ((kk, cc)-variant of confidentiality, [6])**.** *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}\}$ *be an updatable encryption scheme. For* $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA},$

randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA}, *the* (kk, cc)-notion *advantage of an adversary* $\mathscr{A}$ *is defined as*

$$\mathsf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}1}_{\mathsf{UE},\mathscr{A}} = 1] - \Pr[\mathsf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}0}_{\mathsf{UE},\mathscr{A}} = 1]\right|,$$

where the experiment $\mathsf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}\mathsf{b}}_{\mathsf{UE},\mathscr{A}}$ *is the same as the experiment* $\mathsf{Exp}^{\mathsf{notion}\text{-}\mathsf{b}}_{\mathsf{UE},\mathscr{A}}$ *(see Fig. 2.2 and Fig. 2.4), except all leakage sets are computed in the* kk*-directional key update setting and* cc*-directional ciphertext update setting (see Section 2.2.1).*

Definition 2.3.2 ((kk, cc)-variant of integrity, [6])**.** *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}\}$ *be an updatable encryption scheme. For* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$*, the* (kk, cc)-notion *advantage of an adversary* $\mathscr{A}$ *is defined as*

$$\mathsf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\mathscr{A}} = 1]\right|,$$

where the experiment $\mathsf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}$ *is the same as the experiment* $\mathsf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\mathscr{A}}$ *(see Fig. 2.3 and Fig. 2.4), except all leakage sets are computed in the* kk*-directional key update setting and* cc*-directional ciphertext update setting (see Section 2.2.1).*

A general idea to analyze the relation of any two out of the eight variants of each security notion is to construct a reduction $\mathscr{B}$, which runs the security experiment of one variant while simulating all the responses to the queries made by the adversary $\mathscr{A}$ in the security experiment of the other variant and forwards the guess result from $\mathscr{A}$ to its challenger. Recall that if a trivial win condition is triggered, the adversary will lose the game. Therefore, if the reduction $\mathscr{B}$ does not trigger trivial win conditions (when $\mathscr{A}$ does not), the advantage of the reduction $\mathscr{B}$ will be larger than that of the adversary $\mathscr{A}$. Thus, the relation of the two variants depends on the relation of trivial win conditions in each update direction setting.

2.3.1. RELATIONS AMONG CONFIDENTIALITY NOTIONS

The relations of eight variants of confidentiality are shown in Fig. 2.7. We mainly prove the equivalence of each confidentiality notion in the bi-directional key update setting and backward-leak uni-directional key update setting, i.e., (no, bi)-notion $\iff$ (b-uni, bi)-notion and (no, uni)-notion $\iff$ (b-uni, uni)-notion. Then the rest of the relations in Fig. 2.7 can easily follow from the prior work in [6].

The following two lemmas show UE schemes with bi-directional key updates leak more information than those with backward uni-directional key updates, which further leaks more information than those with no-directional key updates.

Lemma 2.3.3. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C}$ *and any* $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$*, we have* $\mathcal{K}^*_{\mathsf{no}} \subseteq \mathcal{K}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{no}} \subseteq \mathcal{T}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{no,cc}} \subseteq \mathcal{C}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \mathcal{C}^*_{\mathsf{bi,cc}}$, $\tilde{\mathcal{L}}^*_{\mathsf{no,cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi,cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{no,cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi,cc}}$, $\mathcal{L}^*_{\mathsf{no,cc}} \subseteq \mathcal{L}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \mathcal{L}^*_{\mathsf{bi,cc}}$ *and* $\mathcal{Q}^*_{\mathsf{no,cc}} \subseteq \mathcal{Q}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi,cc}}$.

Proof. For any $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, the adversary infers more information in the bi-directional key update setting than in the backward uni-directional key update setting. For any $\mathcal{K}, \mathcal{T}, \mathcal{C}$, the inferred leakage sets $\mathcal{K}^*_{\mathsf{b\text{-}uni}}$ and $\mathcal{K}^*_{\mathsf{bi}}$ are computed by Eq. (2.1), (2.2),

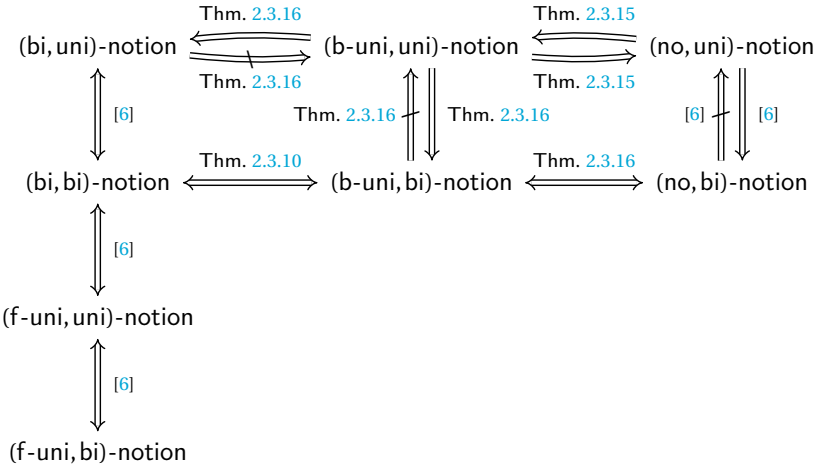


Figure 2.7: Relations among the eight variants on confidentiality for notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA}.

then we have $\mathcal{K}^*_{\mathsf{no}} \subseteq \mathcal{K}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$. By Eq. (2.4), (2.5), (2.6), we have $\mathcal{T}^*_{\mathsf{no}} \subseteq \mathcal{T}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$ and $\mathcal{C}^*_{\mathsf{no,cc}} \subseteq \mathcal{C}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \mathcal{C}^*_{\mathsf{bi,cc}}$. Then we obtain $\tilde{\mathcal{L}}^*_{\mathsf{no,cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi,cc}}$ and $\tilde{\mathcal{Q}}^*_{\mathsf{no,cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi,cc}}$ by Remark 2.2.4. We compute $\mathcal{L}^*$ and $\mathcal{Q}^*$ from $\mathcal{L}$ and $\mathcal{Q}$ with $\mathcal{T}^*$, and then we have $\mathcal{L}^*_{\mathsf{no,cc}} \subseteq \mathcal{L}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \mathcal{L}^*_{\mathsf{bi,cc}}$ and $\mathcal{Q}^*_{\mathsf{no,cc}} \subseteq \mathcal{Q}^*_{\mathsf{b\text{-}uni,cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi,cc}}$, which follows from $\mathcal{T}^*_{\mathsf{no}} \subseteq \mathcal{T}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$. □

Lemma 2.3.4. *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any* $\mathsf{kk} \in \{\mathsf{b\text{-}uni}, \mathsf{bi}\}$*, we have $\mathcal{C}^*_{\mathsf{kk,uni}} \subseteq \mathcal{C}^*_{\mathsf{kk,bi}}$, $\tilde{\mathcal{L}}^*_{\mathsf{kk,uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk,bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk,uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk,bi}}$, $\mathcal{L}^*_{\mathsf{kk,uni}} \subseteq \mathcal{L}^*_{\mathsf{kk,bi}}$ and $\mathcal{Q}^*_{\mathsf{kk,uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk,bi}}$.*

Proof. This follows similarly as in Lemma 2.3.3 and thus we omit the details. □

(bi, bi)-notion $\Longleftrightarrow$ (b-uni, bi)-notion.

We prove the equivalence of (bi, bi)-variant and the (b-uni, bi)-variant in Theorem 2.3.10, which is based on the equivalence of trivial win conditions in Lemmas 2.3.5, 2.3.6, 2.3.8 and 2.3.9. We compare the relations of trivial win conditions in the two settings one by one (see Section 2.2.2).

Lemma 2.3.5. *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$, we have $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \Longleftrightarrow \mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \neq \emptyset$.*

Proof. From Lemma 2.3.3, we know that $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$ and $\mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$, so $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \subseteq \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$. It is sufficient to prove $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \neq \emptyset$. If the adversary never queries any challenge-equal ciphertext in an epoch in $\{0, \ldots, l\} \setminus \mathcal{IR}$, it will not obtain any challenge-equal ciphertext in this set even in the bi-directional ciphertext update setting by Eq. (2.6), i.e., $\mathcal{C}^*_{\mathsf{bi,bi}} \cap \{0, \ldots, l\} \setminus \mathcal{IR} = \emptyset$. This contradicts $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$, since $\mathcal{K}^*_{\mathsf{bi}} = \{0, \ldots, l\} \setminus \mathcal{IR}$ by Lemma 2.2.6. There exists an epoch $\mathsf{e}' \in \{0, \ldots, l\} \setminus \mathcal{IR}$, in which the adversary queries a challenge-equal ciphertext, i.e., $\mathsf{e}' \in \mathcal{C} \cap \mathcal{K}^*_{\mathsf{bi}}$.

Note that $\mathcal{K}^*_{\mathsf{bi}} = \mathcal{K}^*_{\mathsf{b\text{-}uni}} \cup \mathcal{K}^*_{\mathsf{f\text{-}uni}}$. If $\mathsf{e}' \in \mathcal{K}^*_{\mathsf{b\text{-}uni}}$, then $\mathsf{e}' \in \mathcal{K}^*_{\mathsf{b\text{-}uni}} \cup \mathcal{C} \subseteq \mathcal{K}^*_{\mathsf{b\text{-}uni}} \cup \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}}$, so $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cup \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \neq \emptyset$. If $\mathsf{e}' \in \mathcal{K}^*_{\mathsf{f\text{-}uni}}$, then there exists a smaller epoch e'' than e' such that $\mathsf{e}'' \in \mathcal{K}$ and the set $\{\mathsf{e}'', \ldots, \mathsf{e}'\} \subseteq \mathcal{T}$ by Remark 2.2.8. Hence, the adversary can degrade the message from e' to e'' to know $\tilde{c}_{\mathsf{e}''}$ in the bi-directional ciphertext update setting. Therefore, we have $\mathsf{e}'' \in \mathcal{K} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \subseteq \mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}}$, so $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cup \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \neq \emptyset$. □

Lemma 2.3.6. *For any set $\mathcal{K}, \mathcal{T}, \mathcal{C}$, suppose $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,bi}} = \emptyset$ for $\mathsf{kk} \in \{\mathsf{bi}, \mathsf{b\text{-}uni}\}$, then $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{b\text{-}uni}}$.*

Proof. From Lemma 2.3.3, we know that $\mathcal{T}^*_{\mathsf{b\text{-}uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, so if $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{b\text{-}uni}}$, then $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}}$. It is sufficient to prove $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}} \Rightarrow \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{b\text{-}uni}}$. Because the adversary queries the challenge ciphertext in epoch $\tilde{\mathsf{e}}$ (i.e., $\tilde{\mathsf{e}} \in \mathcal{C} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$) and $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, we have $\tilde{\mathsf{e}} \notin \mathcal{K}^*_{\mathsf{bi}}$. $\Delta_{\tilde{\mathsf{e}}}$ cannot be inferred from the successive keys in epochs $\tilde{\mathsf{e}} - 1$ and $\tilde{\mathsf{e}}$. Therefore, if $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}}$, it must be obtained via corrupting, that is $\tilde{\mathsf{e}} \in \mathcal{T}$. Since $\mathcal{T} \subseteq \mathcal{T}^*_{\mathsf{b\text{-}uni}}$, we have $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{b\text{-}uni}}$. □

Remark 2.3.7. *Note that $\mathcal{O}.\mathsf{Upd}(\bar{\mathsf{c}})$ is queried or not is independent of the direction of key and ciphertext updates. Thus it will be the same whether this trivial win condition is triggered or not in all variants.*

Lemma 2.3.8. *For any set $\mathcal{K}, \mathcal{T}, \mathcal{C}$, suppose $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,bi}} = \emptyset$ for $\mathsf{kk} \in \{\mathsf{bi}, \mathsf{b\text{-}uni}\}$, then $(\mathsf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}} \iff (\mathsf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{b\text{-}uni,bi}}$.*

Proof. By Remark 2.2.4 and Lemma 2.3.3, we know that $(\mathsf{c}, \mathsf{e}) \in \tilde{\mathcal{L}} \iff \mathsf{e} \in \mathcal{C}^*$ and $\mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$. So if $(\mathsf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{b\text{-}uni,bi}}$, then $\mathsf{e} \in \mathcal{C}^*_{\mathsf{b\text{-}uni,bi}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$. Thus, we have $(\mathsf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$.

If $(\mathsf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$, that is $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi,bi}}$, then we know $\mathsf{e} \in \mathcal{I}\mathcal{R}$ by the assumption $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$ and the fact that $\mathcal{K}^*_{\mathsf{bi}} = \{0, \ldots, l\} \setminus \mathcal{I}\mathcal{R}$ from Lemma 2.2.6. Suppose $\{\mathsf{fwl}, \ldots, \mathsf{e}\}$ is the last insulated region. If the adversary never queries the challenge-equal ciphertext in the epoch in this set, then it cannot infer any challenge-equal ciphertext in epoch e, which contradicts $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi,bi}}$. Therefore we assume the adversary queries a challenge-equal ciphertext in epoch e', where $\mathsf{e}' \in \{\mathsf{fwl}, \ldots, \mathsf{e}\}$. Since $\{\mathsf{fwl}, \ldots, \mathsf{e}\} \subseteq \mathcal{T}$ even in the backward uni-directional update setting, the adversary can update challenge-equal ciphertext from epoch e' to e, i.e., $\mathsf{e} \in \mathcal{C}^*$. So $(\mathsf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{b\text{-}uni,bi}}$. □

Lemma 2.3.9. *For any set $\mathcal{K}, \mathcal{T}, \mathcal{C}$, suppose $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,bi}} = \emptyset$ for $\mathsf{kk} \in \{\mathsf{bi}, \mathsf{b\text{-}uni}\}$, then $(\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi,bi}} \iff (\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{b\text{-}uni,bi}}$.*

Proof. By Remark 2.3.3, we know that $(\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}} \iff \mathsf{e} \in \mathcal{C}^*$. And the rest of the proof is similar to that of Lemma 2.3.8. □

Theorem 2.3.10. *Let* UE *be an updatable encryption scheme and confidentiality* $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. *For*

any (bi,bi)-notion *adversary* $\mathscr{A}$ *against* UE, *there exists a* (b-uni,bi)-notion *adversary* $\mathscr{B}_1$ *against* UE *such that*

$$\mathsf{Adv}^{\text{(bi,bi)-notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \mathsf{Adv}^{\text{(b-uni,bi)-notion}}_{\mathsf{UE},\mathscr{B}_1}(1^\lambda).$$

Proof. We construct a reduction $\mathscr{B}_1$ who runs the (b-uni,bi)-notion experiment and simulates all responses of the queries made by (bi,bi)-notion adversary $\mathscr{A}$. The reduction $\mathscr{B}_1$ works by sending all the queries of $\mathscr{A}$ to its own challenger and returning the responses to $\mathscr{A}$. At last, $\mathscr{B}_1$ sends the guessing result received from $\mathscr{A}$ to its own challenger. The challenger will check whether the reduction wins or not. If the reduction triggers the trivial win conditions, it will lose the game. The reduction also forwards the experiment result to $\mathscr{A}$.

Notice that Lemmas 2.3.5, 2.3.6, 2.3.8, 2.3.9, and Remark 2.3.7 exactly include all the trivial win conditions in the confidentiality game (see Fig. 2.6). Thus, we can conclude that the trivial win conditions in the (bi,bi)-notion and (b-uni,bi)-notion games are equivalent. If no trivial conditions are triggered by $\mathscr{A}$, there will be no trivial win conditions triggered by $\mathscr{B}$. But if a condition is triggered in the (bi,bi)-notion, the same condition will also be triggered in the (b-uni,bi)-notion. Therefore, the reduction perfectly simulates the (bi,bi)-notion game to $\mathscr{A}$. Then $\mathsf{Adv}^{\text{(bi,bi)-notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \mathsf{Adv}^{\text{(b-uni,bi)-notion}}_{\mathsf{UE},\mathscr{B}_1}(1^\lambda)$. □

(b-uni,uni)-notion $\Longleftrightarrow$ (no,uni)-notion.

We further prove the equivalence of (b-uni,uni)-variant and the (b-uni,uni)-variant in Theorem 2.3.15, which is based on the equivalence of trivial win conditions in Lemmas 2.3.11, 2.3.12, 2.3.13 and 2.3.14.

Lemma 2.3.11. *For any set* $\mathscr{K},\mathscr{T},\mathscr{C}$*, we have* $\mathscr{K}^*_{\text{b-uni}} \cap \mathscr{C}^*_{\text{b-uni,uni}} \neq \emptyset \Longleftrightarrow \mathscr{K}^*_{\text{no}} \cap \mathscr{C}^*_{\text{no,uni}} \neq \emptyset$.

Proof. From Lemma 2.3.3, we know that $\mathscr{K}^*_{\text{no}} \subseteq \mathscr{K}^*_{\text{b-uni}}$ and $\mathscr{C}^*_{\text{no,uni}} \subseteq \mathscr{C}^*_{\text{b-uni,uni}}$, so $\mathscr{K}^*_{\text{no}} \cap \mathscr{C}^*_{\text{no,uni}} \subseteq \mathscr{K}^*_{\text{b-uni}} \cap \mathscr{C}^*_{\text{b-uni,uni}}$. It is sufficient to prove $\mathscr{K}^*_{\text{b-uni}} \cap \mathscr{C}^*_{\text{b-uni,uni}} \neq \emptyset \Rightarrow \mathscr{K}^*_{\text{no}} \cap \mathscr{C}^*_{\text{no,bi}} \neq \emptyset$.

Suppose there exists an epoch $\mathsf{e} \in \mathscr{K}^*_{\text{b-uni}} \cap \mathscr{C}^*_{\text{b-uni,uni}}$. From $\mathsf{e} \in \mathscr{K}^*_{\text{b-uni}}$ and Remark 2.2.8, there is an epoch $\mathsf{e_b}$ after e, satisfying $\mathsf{e_b} \in \mathscr{K}$ and $\{\mathsf{e},\ldots,\mathsf{e_b}\} \in \mathscr{T}$. From $\mathsf{e} \in \mathscr{C}^*_{\text{b-uni,uni}}$ and the definition of $\mathscr{C}^*_{\text{b-uni,uni}}$ in Eq. (2.5), we know that there exists an epoch $\mathsf{e_c}$ before e such that the adversary asks for the challenge ciphertext in epoch $\mathsf{e_c}$ (i.e., $\mathsf{e_c} \in \mathscr{C}$) and $\{\mathsf{e_c},\ldots,\mathsf{e}\} \in \mathscr{T}^*_{\text{b-uni}}$. If the set $\{\mathsf{e_c},\ldots,\mathsf{e}\} \subseteq \mathscr{T}$, then we can upgrade the ciphertext from epoch $\mathsf{e_c}$ to epoch $\mathsf{e_b}$ even in the no-directional key update setting, since $\mathsf{e_c} \in \mathscr{C}$ and $\{\mathsf{e_c},\ldots,\mathsf{e},\ldots,\mathsf{e_b}\} \in \mathscr{T}$. Therefore, we have $\mathsf{e_b} \in \mathscr{K}^*_{\text{no}} \cap \mathscr{C}^*_{\text{no,bi}}$.

If not every epoch in the set $\{\mathsf{e_c},\ldots,\mathsf{e}\}$ is in $\mathscr{T}$ (i.e., there is an epoch $\mathsf{e_s} \in \mathscr{T}^*_{\text{b-uni}} \setminus \mathscr{T}$), then by Eq. (2.4), we know that $\mathsf{e_s}-1$ and $\mathsf{e_s}$ are in $\mathscr{K}^*_{\text{b-uni}}$. Moreover, we have $\mathsf{e_s}-1 \in \mathscr{K}$, because the epoch key in $\mathsf{e_s}-1$ cannot be inferred from the key in $\mathsf{e_s}-1$ in the backward uni-directional key update setting as $\mathsf{e_s} \notin \mathscr{T}$ and the adversary can only learn the epoch key in $\mathsf{e_s}-1$ from querying the corruption oracle. If $\{\mathsf{e_c},\ldots,\mathsf{e_s}-1\} \in \mathscr{T}$, we can upgrade ciphertexts from epoch $\mathsf{e_c}$ to epoch $\mathsf{e_s}-1$ even in the no-directional key update setting, that is $\mathsf{e_s}-1 \in \mathscr{K}^*_{\text{no}} \cap \mathscr{C}^*_{\text{no,bi}}$. Otherwise, we

repeat this step, substitute $e_s - 1$ with a smaller epoch $e_s - j$ in the next iteration for some $j > 1$ such that $e_s - j \in \mathcal{K}$ and $\{e_c, \ldots, e_s - j\} \in \mathcal{T}^*_{\text{b-uni}}$, and check if all epochs in $\{e_c, \ldots, e_s - j\}$ are in $\mathcal{T}$. Since the epoch length is limited, we will stop at an epoch, say $e_s - k$, for some $k > 1$ such that $e_s - k \in \mathcal{K}$ and $\{e_c, \ldots, e_s - k\} \in \mathcal{T}$. We can upgrade ciphertext from epoch e_c to epoch $e_s - k$ even in the no-directional key update setting, that is $e_s - k \in \mathcal{K}^*_{\text{no}} \cap \mathcal{C}^*_{\text{no,bi}}$, so $\mathcal{K}^*_{\text{no}} \cap \mathcal{C}^*_{\text{no,bi}} \neq \emptyset$.

□

Lemma 2.3.12. *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathcal{K}^*_{\text{kk}} \cap \mathcal{C}^*_{\text{kk,uni}} = \emptyset$ *for* $\text{kk} \in \{\text{b-uni}, \text{no}\}$*, then* $\tilde{e} \in \mathcal{T}^*_{\text{b-uni}} \iff \tilde{e} \in \mathcal{T}^*_{\text{no}}$.

Proof. The proof is similar to that of Lemma 2.3.6. From Lemma 2.3.3, we know that $\mathcal{T}^*_{\text{no}} \subseteq \mathcal{T}^*_{\text{b-uni}}$, so if $\tilde{e} \in \mathcal{T}^*_{\text{no}}$, then $\tilde{e} \in \mathcal{T}^*_{\text{b-uni}}$. Notice that $\tilde{e} \notin \mathcal{K}^*_{\text{b-uni}}$, because the adversary queries the challenge ciphertext in the epoch $\tilde{e}$ and $\mathcal{K}^*_{\text{b-uni}} \cap \mathcal{C}^*_{\text{b-uni,bi}} = \emptyset$. Then $\Delta_{\tilde{e}}$ cannot be inferred from the successive keys in epochs $\tilde{e} - 1$ and $\tilde{e}$. Therefore, if $\tilde{e} \in \mathcal{T}^*_{\text{b-uni}}$, then it must be obtained from corrupting, that is $\tilde{e} \in \mathcal{T}$. Since $\mathcal{T} = \mathcal{T}^*_{\text{no}}$, we have $\tilde{e} \in \mathcal{T}^*_{\text{no}}$. □

Lemma 2.3.13. *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathcal{K}^*_{\text{kk}} \cap \mathcal{C}^*_{\text{kk,uni}} = \emptyset$ *for* $\text{kk} \in \{\text{b-uni}, \text{no}\}$*, then* $(c, e) \in \tilde{\mathcal{L}}^*_{\text{b-uni}} \iff (c, e) \in \tilde{\mathcal{L}}^*_{\text{no}}$.

Proof. The proof is similar to that of Lemma 2.3.8. By Remark 2.2.4 and Lemma 2.3.3 , we know that $(\mathbf{c}, e) \in \tilde{\mathcal{L}} \iff e \in \mathcal{C}^*$ and $\mathcal{C}^*_{\text{no,uni}} \subseteq \mathcal{C}^*_{\text{b-uni,uni}}$. So if $(\mathbf{c}, e) \in \mathcal{L}^*_{\text{no,uni}}$, then $e \in \mathcal{C}^*_{\text{no,uni}} \subseteq \mathcal{C}^*_{\text{b-uni,uni}}$. Thus, we have $(\mathbf{c}, e) \in \tilde{\mathcal{L}}^*_{\text{b-uni}}$.

If $(\mathbf{c}, e) \in \tilde{\mathcal{L}}^*_{\text{b-uni}}$, then $e \in \mathcal{C}^*_{\text{b-uni,uni}} \subseteq \mathcal{IR}$. From the definition of $\mathcal{C}^*_{\text{b-uni,uni}}$ in Eq. (2.5), we know that there is an epoch e_c before e, satisfying the adversary queries the challenge ciphertext in epoch e_c (i.e., $e_c \in \mathcal{C}$) and $\{e_c, \ldots, e\} \in \mathcal{T}^*_{\text{b-uni}}$, which implies $\{e_c, \ldots, e\} \in \mathcal{C}^*_{\text{b-uni}}$. From the assumption that $\mathcal{K}^*_{\text{b-uni}} \cap \mathcal{C}^*_{\text{b-uni,uni}} = \emptyset$, then we have $\{e_c, \ldots, e\} \notin \mathcal{K}^*_{\text{b-uni}}$. To meet the condition $\{e_c, \ldots, e\} \in \mathcal{T}^*_{\text{b-uni}}$, all tokens in epochs in $\{e_c, \ldots, e\}$ can only be obtained by corrupting, that is $\{e_c, \ldots, e\} \in \mathcal{T} = \mathcal{T}^*_{\text{no}}$. We can upgrade the ciphertext from epoch e_c to epoch e_b even in the no-directional key update setting. Therefore, we have $e \in \mathcal{C}^*_{\text{no,uni}} \subseteq \mathcal{IR}$ and further $(\mathbf{c}, e) \in \tilde{\mathcal{L}}^*_{\text{no}}$.

□

Lemma 2.3.14. *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathcal{K}^*_{\text{kk}} \cap \mathcal{C}^*_{\text{kk,uni}} = \emptyset$ *for* $\text{kk} \in \{\text{b-uni}, \text{no}\}$*, then* $(m', e) \in \tilde{\mathcal{Q}}^*_{\text{b-uni,uni}} \iff (m', e) \in \tilde{\mathcal{Q}}^*_{\text{no,uni}}$.

Proof. By Remark 2.2.4, we know that $(m', e) \in \tilde{\mathcal{Q}} \iff e \in \mathcal{C}^*$. The rest of the proof is similar to that of Lemma 2.3.13.

□

Theorem 2.3.15. *Let* UE *be an updatable encryption scheme and confidentiality* $\text{notion} \in \{\text{detIND-UE-CPA}, \text{randIND-UE-CPA}, \text{detIND-UE-CCA}, \text{randIND-UE-CCA}\}$*. For any* (b-uni, uni)-notion *adversary* $\mathcal{A}$ *against* UE*, there exists a* (no, uni)-notion *adversary* $\mathcal{B}_2$ *against* UE *such that*

$$\text{Adv}^{\text{(b-uni,uni)-notion}}_{\text{UE},\mathcal{A}}(1^\lambda) = \text{Adv}^{\textit{(no,uni)-notion}}_{\text{UE},\mathcal{B}_2}(1^\lambda)$$

Proof. The proof is similar to that of Theorem 2.3.10. We construct a reduction $\mathscr{B}_2$ who runs the (b-uni, uni)-notion experiment and simulates all responses of the queries made by (no, uni)-notion adversary $\mathscr{A}$. The reduction $\mathscr{B}_2$ works by sending all the queries of $\mathscr{A}$ to its own challenger and forwarding its received responses to $\mathscr{A}$. In the end, $\mathscr{B}_2$ sends the guessing result from $\mathscr{A}$ to its own challenger. The challenger will check if the reduction wins. The reduction also forwards the experiment result to $\mathscr{A}$. If the trivial win conditions were triggered, the reduction will be regarded as losing the game.

From Lemmas 2.3.11, 2.3.12, 2.3.13, 2.3.14 and Remark 2.3.7, we obtain the trivial win conditions in the (b-uni, uni)-notion and (no, uni)-notion games are equivalent. If there is a trivial win condition that is triggered by $\mathscr{A}$, then the same trivial win condition will be triggered by $\mathscr{B}$, and vice versa. Therefore, the reduction perfectly simulates the (no, uni)-notion game to $\mathscr{A}$. Then, we have $\mathsf{Adv}^{\text{(b-uni,uni)-notion}}_{\mathsf{UE},\mathscr{A}}(1^\lambda) = \mathsf{Adv}^{\text{(no,uni)-notion}}_{\mathsf{UE},\mathscr{B}_2}(1^\lambda)$. □

Theorem 2.3.16. *For* notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA}, *Fig. 2.7 is the relations among the eight variants on the same confidentiality notion.*

Proof. We conclude the relations among the eight variants on confidentiality from Theorems 2.3.10 and 2.3.15, together with the previous conclusions in [6], which proved that a UE scheme with bi-directional key updates is equivalent to the one with forward-leak uni-directional key updates, shown in Fig. 2.8.

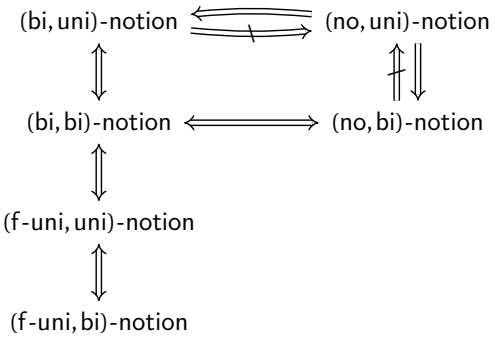


Figure 2.8: Relations among the six variants of confidentiality for notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA} in [6].

Thus, we have the relations among the eight variants on confidentiality in Fig. 2.7 by the equivalences: (bi, bi)-notion $\iff$ (b-uni, bi)-notion from Theorem 2.3.10 and (b-uni, uni)-notion $\iff$ (no, uni)-notion from Theorem 2.3.15 and Fig. 2.8. □

2.3.2. RELATIONS AMONG INTEGRITY NOTIONS

The relations of the eight variants on integrity are illustrated in Fig. 2.9. We first prove two equivalence of trivial win conditions in Lemmas 2.3.17, 2.3.19 and 2.3.21.

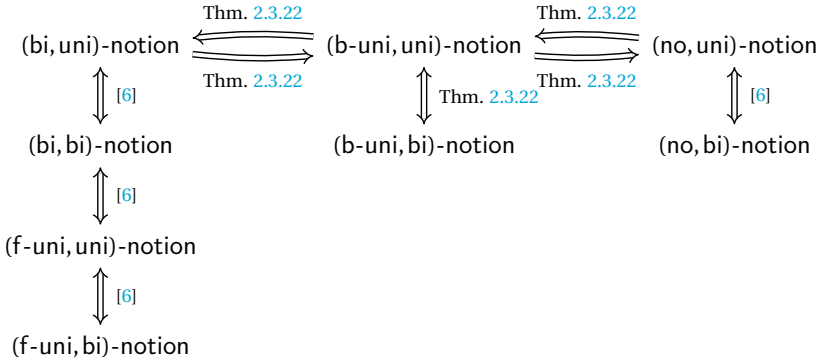


Figure 2.9: Relations among the eight variants of integrity for notion $\in$ {IND-CTXT, IND-PTXT}.

Lemma 2.3.17. *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, we have* $\mathsf{e} \in \hat{\mathcal{K}}^*_{\mathsf{b\text{-}uni}} \iff \mathsf{e} \in \hat{\mathcal{K}}^*_{\mathsf{kk}}$ *for* $\mathsf{kk} \in \{\mathsf{f\text{-}uni}, \mathsf{bi}, \mathsf{no}\}$.

Proof. From Eq. (2.3), we know that the computation of the extended set $\hat{\mathcal{K}}^*$ is independent of the direction of key updates. □

Lemma 2.3.18 ([6], Lemma 3.11)**.** *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathsf{e} \notin \hat{\mathcal{K}}^*$*, then we have* $(\mathsf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \iff (\mathsf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk}',\mathsf{cc}'}$ *for any* $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{f\text{-}uni}, \mathsf{bi}, \mathsf{no}\}$ *and* $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Lemma 2.3.19. *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathsf{e} \notin \hat{\mathcal{K}}^*$*, then* $(\mathsf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{b\text{-}uni},\mathsf{cc}} \iff (\mathsf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk}',\mathsf{cc}'}$ *for any* $\mathsf{kk}' \in \{\mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}, \mathsf{no}\}$ *and* $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Proof. It follows directly from Lemma 2.3.18 and $\mathcal{L}^*_{\mathsf{no},\mathsf{cc}} \subseteq \mathcal{L}^*_{\mathsf{b\text{-}uni},\mathsf{cc}} \subseteq \mathcal{L}^*_{\mathsf{bi},\mathsf{cc}}$ by Lemma 2.3.3 for any $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$. □

Lemma 2.3.20 ([6], Lemma 3.12)**.** *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathsf{e} \notin \hat{\mathcal{K}}^*$*, then we have* $(\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{cc}} \iff (\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{kk}',\mathsf{cc}'}$ *for any* $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{f\text{-}uni}, \mathsf{bi}, \mathsf{no}\}$ *and* $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Lemma 2.3.21. *For any set* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, suppose* $\mathsf{e} \notin \hat{\mathcal{K}}^*$*, then we have* $(\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{b\text{-}uni},\mathsf{cc}} \iff (\mathsf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{kk}',\mathsf{cc}'}$ *for any* $\mathsf{kk}' \in \{\mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}, \mathsf{no}\}$ *and* $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Proof. It follows directly from Lemma 2.3.20 and $\mathcal{Q}^*_{\mathsf{no},\mathsf{cc}} \subseteq \mathcal{Q}^*_{\mathsf{b\text{-}uni},\mathsf{cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi},\mathsf{cc}}$ by Lemma 2.3.4 for any $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$. □

Theorem 2.3.22. *Let* UE *be an updatable encryption scheme, the integrity notion* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$*. For any* $(\mathsf{b\text{-}uni}, \mathsf{cc})$-notion *adversary* $\mathcal{A}$ *against* UE*, there exists a* $(\mathsf{kk}', \mathsf{cc}')$-notion *adversary* $\mathcal{B}_4$ *against* UE *such that*

$$\mathsf{Adv}^{(\mathsf{b\text{-}uni},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\mathcal{A}}(1^\lambda) = \mathsf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\mathcal{B}_4}(1^\lambda)$$

for any $\mathsf{kk}' \in \{\mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}, \mathsf{no}\}$ *and* $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Proof. The proof is similar to that of Theorem 1. We construct a reduction $\mathscr{B}_4$ which runs the $(\mathsf{kk}', \mathsf{cc}')$-notion game and simulates all responses to the queries made by the $(\mathsf{b\text{-}uni}, \mathsf{cc})$-notion adversary $\mathscr{A}$. If there is a trivial win condition that is triggered by $\mathscr{A}$, the same trivial win condition will be triggered by $\mathscr{B}$, and vice versa, which follows from Lemmas 2.3.17, 2.3.19 and 2.3.21. Thus, the reduction perfectly simulates the game to A, and the advantages are equal. □

Theorem 2.3.23. *For* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$*, Fig. 2.9 shows the relations among the eight variants on the same integrity notion.*

Proof. We conclude the relations from Theorem 2.3.22, together with the previous conclusions in [6] that $(\mathsf{kk}, \mathsf{cc})$-$\mathsf{notion} \Longleftrightarrow (\mathsf{no}, \mathsf{cc}')$-$\mathsf{notion}$ for $\mathsf{notion} \in \{\mathsf{IND\text{-}CTXT}, \mathsf{IND\text{-}PTXT}\}$, $\mathsf{kk} \in (\mathsf{bi}, \mathsf{f\text{-}uni})$ and $\mathsf{cc}, \mathsf{cc}' \in (\mathsf{bi}, \mathsf{uni})$. □

2.4. CONCLUSION

The relations among various security notions for UE should be clearly investigated before any valuable constructions. We provided a detailed comparison of every security notion in the four key update settings, and our results showed that the UE schemes in the no-directional key update setting, which were believed to be strictly stronger than others, are equivalent to those in the backward-leak uni-directional key update setting. As future work, we intend to develop an efficient UE scheme with backward-leak uni-directional key updates.

REFERENCES

[1] D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. “Key homomorphic PRFs and their applications”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Heidelberg: Springer, 2013, pp. 410–428. DOI: 978-3-642-40041-4_23.

[2] D. Boneh, S. Eskandarian, S. Kim, and M. Shih. “Improving Speed and Security in Updatable Encryption Schemes”. In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Cham: Springer, 2020, pp. 559–589. ISBN: 978-3-030-64840-4. DOI: 10.1007/978-3-030-64840-4_19.

[3] L. Chen, Y. Li, and Q. Tang. “CCA Updatable Encryption Against Malicious Re-encryption Attacks”. In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer, 2020, pp. 590–620. DOI: 10.1007/978-3-030-64840-4_20.

[4] A. Everspaugh, K. Paterson, T. Ristenpart, and S. Scott. “Key rotation for authenticated encryption”. In: *CRYPTO 2017, Part III*. Ed. by J. Katz and H. Shacham. Vol. 10403. LNCS. Springer. Heidelberg, 2017, pp. 98–129. DOI: 10.1007/978-3-319-63697-9_4.

[5] C. Boyd, G. T. Davies, K. Gjøsteen, and Y. Jiang. “Fast and Secure Updatable Encryption”. In: *CRYPTO 2020, Part I*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12170. LNCS. Springer, 2020, pp. 464–493. DOI: 10.1007/978-3-030-56784-2_16.

[6] Y. Jiang. “The direction of updatable encryption does not matter much”. In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer. Heidelberg, 2020, pp. 529–558. DOI: 10.1007/978-3-030-64840-4_18.

[7] M. Klooß, A. Lehmann, and A. Rupp. “(R)CCA secure updatable encryption with integrity protection”. In: *EUROCRYPTO 2019, Part I*. Ed. by Y. Ishai and V. Rijmen. Vol. 11476. LNCS. Springer. Heidelberg, 2019, pp. 68–99. DOI: 10.1007/978-3-030-17653-2_3.

[8] A. Lehmann and B. Tackmann. “Updatable encryption with post-compromise security”. In: *EUROCRYPT 2018, Part III*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10822. LNCS. Springer. Heidelberg, 2018, pp. 685–716. DOI: 10.1007/978-3-319-78372-7_22.

[9] R. Nishimaki. “The Direction of Updatable Encryption Does Matter”. In: *PKC 2022*. Ed. by G. Hanaoka, J. Shikata, and Y. Watanabe. Vol. 13178. LNCS. Cham: Springer, 2022, pp. 194–224. ISBN: 978-3-030-97131-1. DOI: 10.1007/978-3-030-97131-1_7.

[10] D. Slamanig and C. Striecks. *Puncture 'Em All: Updatable Encryption with No-Directional Key Updates and Expiring Ciphertexts*. Cryptology ePrint Archive, Paper 2021/268. https://eprint.iacr.org/2021/268. 2021.

[11] M. Naor and M. Yung. "Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks". In: *STOC 1990*. Ed. by H. Ortiz. Baltimore, Maryland, USA: ACM, 1990, pp. 427–437. DOI: 10.1145/100216.100273.

[12] O. Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *ACM 2005*. Ed. by H. N. Gabow and R. Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[13] O. Goldreich, S. Goldwasser, and S. Micali. "How to Construct Random Functions". In: *J. ACM* 33.4 (1986), pp. 792–807. ISSN: 0004-5411. DOI: 10.1145/6490.6503.

[14] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. "On the (im)possibility of obfuscating programs". In: *J. ACM* 59.2 (2012), pp. 1–48. DOI: 10.1145/2160158.2160159.

3

CCA-1 Secure Updatable Encryption with Adaptive Security

Updatable encryption (UE) enables a cloud server to update ciphertexts using client-generated tokens. There are two types of UE: ciphertext-independent (c-i) and ciphertext-dependent (c-d). In terms of construction and efficiency, c-i UE utilizes a single token to update all ciphertexts. The update mechanism relies mainly on the homomorphic properties of exponentiation, which limits the efficiency of encryption and updating. Although c-d UE may seem inconvenient as it requires downloading parts of the ciphertexts during token generation, it allows for easy implementation of the Dec-then-Enc structure. This methodology significantly simplifies the construction of the update mechanism. Notably, the c-d UE scheme proposed by Boneh et al. (ASIACRYPT'20) has been reported to be 200 times faster than prior UE schemes based on DDH hardness, which is the case for most existing c-i UE schemes. Furthermore, c-d UE ensures a high level of security as the token does not reveal any information about the key, which is difficult for c-i UE to achieve. However, previous security studies on c-d UE only addressed selective security; the studies for adaptive security remain an open problem.

In this study, we make three significant contributions to ciphertext-dependent updatable encryption (c-d UE). Firstly, we provide stronger security notions compared to previous work, which capture adaptive security and also consider the adversary's decryption capabilities under the adaptive corruption setting. Secondly, we propose a new c-d UE scheme that achieves the proposed security notions. The token generation technique significantly differs from the previous Dec-then-Enc structure, while still preventing key leakages. At last, we introduce a packing technique that enables the

This chapter is based on the paper "*CCA-1 Secure Updatable Encryption with Adaptive Security*" by **Chen, H.**, Galteland, Y.J., and Liang, K. in ASIACRYPT (5) 2023: 374-406

simultaneous encryption and updating of multiple messages within a single ciphertext. This technique helps alleviate the cost of c-d UE by reducing the need to download partial ciphertexts during token generation.

3.1. INTRODUCTION

Regularly changing encryption keys is widely recognized as an effective approach to mitigate the risk of key compromise, especially when outsourcing encrypted data to a semi-honest cloud server. Updatable encryption (UE), introduced by Boneh et al. [1], offers a practical solution to this challenge. In UE schemes, in addition to the usual KG, Enc, Dec algorithms, two core algorithms, TokenGen and Update, are employed. Essentially, TokenGen takes the old and new encryption keys, along with *possibly* a small fraction of the ciphertext, and generates an update token on the client side. This token is then sent to the cloud server, which utilizes the Update algorithm to convert ciphertexts from the old keys to the new keys.

c-d/c-i UE. Depending on if a part of ciphertext (called ciphertext header) is needed in the token generation algorithm TokenGen, UE schemes have two variants: *ciphertext-independent* (c-i) UE [2–7] and *ciphertext-dependent* (c-d) UE [1, 8–10]. In the former, tokens are independent of ciphertexts, and a single update token is used to update all old ciphertexts. In the latter, update tokens depend on the specific ciphertext to be updated and a tiny part of the ciphertexts is downloaded by the client when generating the update tokens.

In this paper, we specifically focus on ciphertext-dependent UE (c-d UE) due to its notable advantages in terms of efficiency and security. First of all, c-d UE schemes have been reported to be more efficient than ciphertext-independent (c-i) constructions. For instance, the nested c-d UE construction presented in [8], which relies solely on symmetric cryptographic primitives, approaches the performance of AES. In contrast, c-i UE schemes imply the use of public key encryption, as proven by Alamati et al. [11], and most c-i constructions require costly exponentiation operations to update ciphertexts. With regard to UE security, Jiang [3] demonstrated that there are no c-i UE schemes stronger than those with no-directional key updates, as defined in Section 3.3. However, constructing such schemes remains an open problem, primarily due to the requirement that update tokens should not reveal any information about either the old key or the new key. Consequently, only two c-i UE schemes with no-directional key updates have been proposed thus far. One is presented by Slamanig [7], which is based on the SXDH assumption, thus necessitating expensive exponential operations. The other is introduced by Nishimaki [6], relying on the existence of indistinguishability obfuscation, but remains purely theoretical. On the other hand, for c-d UE, the construction of no-directional key update schemes is considerably easier and practical. In fact, the token generation algorithm in all existing c-d UE constructions [1, 8–10] benefits from a “Dec-then-Enc” process. This involves decrypting the ciphertext header using the old key to recover the secret information, and then computing the token by encrypting the secret information using the new key. As a result, the old key remains independent of the token, while the new key is safeguarded by the underlying encryption scheme. The update token does not divulge any information about the old and new keys.

Security Notions (c-d UE). The primary security objective of UE is to ensure the

confidentiality of ciphertexts even when the keys are exposed. Extensive research on this topic has been conducted in [8–10]. Previous security models provide guarantees that adversaries cannot differentiate between a freshly generated ciphertext in the current epoch and an updated ciphertext rotated to the current epoch. In practical scenarios, this property safeguards the confidentiality of the *age* of ciphertext, i.e., the number of times it has been updated, from being leaked to an adversary. For instance, consider a situation where a client stores its encrypted medical records with a cloud provider. The existing security notions ensure that the adversary observing the records cannot determine which records are new and which ones are old, thereby preserving the sort of privacy.

Limitation. Unfortunately, prior work on c-d UE has the following limitations:

1. The current security notions for c-d UE solely capture selective security, where the adversary is provided with certain keys at the beginning of the security experiment. It is needed to introduce a stronger notion of adaptive security, which guarantees security in the model where the adversary can corrupt keys throughout the experiment.

2. Prior notions for c-d UE only apply to randomized ciphertext updates (see Sect. 3.2.1 for details), whereas the ciphertext update procedure can be also deterministic[1], which can be seen in our construction in Sect. 3.5.1. It is still an open problem how could we capture confidentiality for both types of ciphertext updates.

3. The current security notions for c-d UE are complex, requiring multiple simulations of oracles that the adversary has access to in the security analysis. A simpler and more compact notion can help one simplify the proof.

3.1.1. RELATED WORK

CONSTRUCTIONS OF UE.

Since the introduction of updatable encryption by Boneh et al. [1], various constructions have been proposed. All c-d UE schemes in [1, 8–10] benefit from a *Dec-then-Enc* structure in token generation, whether they are treated in a symmetric manner to deploy double encryption or rely on *key-homomorphic PRFs*. As a consequence, tokens only contain the ciphertext under the new key, avoiding the issue of leaking neither old nor new key.

By comparison, all c-i UE schemes in [2, 4, 5] are based on the DDH or SXDH assumption and rely on the homomorphic properties of exponentiation to rotate ciphertexts. Tokens are the division of the new key and the old key; therefore, one of the two successive keys key can be inferred if the other is leaked. Such a leakage limitation is also applied to the scheme proposed by Jiang [3], because tokens are

[1] Note this case does not require the server to generate randomness for ciphertext updates, which is required in the former case.

	Schemes	Dir. Key	s/a	Prob.	Enc.	Token Gen.	Update
c-i UE	*SHINE [2]	bi	a	DDH	1 exp.	1 division	1 exp.
	LWEUE [3]	bi	a	LWE	(n, m, l)	1 subtract.	(n, m, l)
	UNIUE [12]	bk.	a	LWE	$(n, m, l) \times l_e$	l_e subtract.	$(n, m, l) \times l_e$
	Nishimaki [6]	bk.	a	LWE	$(1, m, l)$	$(nk, m, n+l)$	$(1, m, n+l)$
	Nishimaki [6]	no	a	IO, OWF	–	–	–
	SS [7]	no	a	SXDH	–	–	–
c-d UE	KSS [10]	no	s	Symmetry	1 AE.Enc	1 AE.Enc	1 vect. addition
	ReCrypt [10]	no	s	KH-PRF	1 KH-PRF	1 KH-PRF	1 KH-PRF
	ReCrypt$^+$ [9]	no	s	HomHash	1 HomHash	1 HomHash	1 HomHash
	Nested [8]	no	s	Symmetry	1 AE.Enc	1 AE.Enc	1 AE.Enc
	BEKS [8]	no	s	KH-PRF	1 KH-PRF	1 AE.Enc	1 KH-PRF
	TDUE (3.5.1)	no	a	LWE	$(1, n, \bar{m}+2l)$	$\bar{m}+2l$ sampling	$(1, \bar{m}+2l, \bar{m}+2l)$
	Packing UE (3.5.4)	no	a	LWE	$(1, n, \bar{m}+l+Nl)$	$\bar{m}+l+Nl$ sampling	$(1, \bar{m}+l+Nl, \bar{m}+l+Nl)$

Figure 3.1: A comparison of c-i UE which can avoid the leakage of "ciphertext age" and all existing c-d UE.

the subtraction of new and old keys, even though this scheme avoids the expensive exponentiation but is instead lattice-based.

Two promising c-i UE schemes have been proposed to overcome this leakage limitation. Nishimaki [6] presented a construction that utilizes *indistinguishability obfuscation (IO)* for an update circuit, which operates as a Dec-then-Enc process taking a ciphertext as input. This scheme relies on an assumption that there exists a practical IO. Slamanig and Striecks [7] gave a pairing-based scheme and defined an expiry model: each ciphertext is associated with an expiry epoch, after which the updated ciphertext cannot be decrypted anymore. Their scheme consumes expensive group operations and moreover, the key size increases linearly to the maximum number of updates.

In Fig. 3.1, we provide a comparison of UE schemes in terms of security and efficiency. The second column set states the direction of key updates, achieved security, and the underlying assumptions, where bk. stands for the backward directional key updates, s and a represent the selective and adaptive security, respectively, and KH-PRF, IO, OWF, HomHash represent key-homomorphic PRF, indistinguishability obfuscation, one-way function, and homomorphic hash function respectively. The third column set shows the computational efficiency in terms of the most expensive cost of encryption, token generation, and update for one-block ciphertext ([6] and [7] are omitted here as the first is theoretical and the second is built on a different expiry model). For lattice-based schemes, (a, b, c) denotes the major computation cost by the multiplication of two matrices of size $a \times b$ and $b \times c$, and m and n denote the size of the matrix generated on $\mathbb{Z}_q$ in the setup, for which $m = O(nk)$, $k = \lceil q \rceil$, message bit length $l = nk$, $\bar{m} = O(nk)$, and the maximum number of updates l_e. In our UE schemes, tokens are generated with multiple calls to a preimage sampling oracle, and N is a power of 2 that defines the associated cyclotomic ring. AE represents authenticated encryption, [9] instantiates HomHash from DDH groups, and KH-PRFs are constructed from the Ring-LWE problem in [8].

Our c-d UE constructions offer several advantages compared to c-i UE schemes. Regarding security, we achieve no-directional key updates to protect keys being derived by tokens, in comparison to the difficulty in constructing such c-i UE, as discussed above. In terms of efficiency, we utilize lattice encryption to circumvent expensive group operations that are used in [2, 7]. Compared to lattice-based c-i UE Schemes, our works are the first to achieve CCA-1 security, and TDUE exhibits equivalent complexity to that of [6] for both the encryption and update algorithms, an improvement by a factor n over the algorithms in [3][2]. Our packing UE further reduces the encryption and update algorithms' complexity of TDUE by a factor N, leading to more efficient encryption and update compared to [3, 6, 12]. Nevertheless, it is worth noting that [3] provides the most efficient token generation using a simple vector subtraction. In comparison with c-d UE schemes, our constructions ensure the confidentiality that hides ciphertext age, whereas [9, 10] only capture message confidentiality and re-encryption indistinguishability. Note that Boneh et al. [8] demonstrated that, for c-d UE, even the combination of the above two

[2]Note that, for lattice-based schemes, the cost is determined by the multiplication of two matrices, which takes $O(nml)$ for matrices of size $n \times m$ and $m \times l$ by a naive multiplication, for example.

notions cannot prevent the leakage of ciphertext age (see Fig. 3.3 for more details). Moreover, our schemes are the first c-d UE schemes to achieve adaptive security.

RELATIVE PRIMITIVES.

Proxy Re-encryption (PRE) and Homomorphic Encryption (HE) are two highly related primitives to updatable encryption.

Proxy Re-encryption (PRE) enables a ciphertext to be decryptable by the new key after re-encryption. Compared to UE, it does not necessarily require the updated ciphertext to be indistinguishable from fresh encryption, thereby not covering the confidentiality requirement inherent in UE. However, PRE schemes have served as a source of inspiration for the construction of UE due to the similar ciphertext update process, for example, the ElGamal-based proxy re-encryption scheme is adapted to RISE [5] and Sakurai et al.[13] to SHINE [2].

The PRE scheme proposed by Kirshanova [14] is based on lattices and only uses the old secret key (serving as the trapdoor) to sample a matrix as the update token to rotate ciphertexts, which are LWE samples. Such a matrix leaks neither the old nor the new keys, since it does not involve any function of old and new key (recall the key leakage caused the division or subtraction of two keys in the token of c-i UE schemes). However, Fan and Liu [15] pointed out a mistake in the security proof of [14] that the simulated game in the proof is not indistinguishable from the real game. In this work, a new UE scheme that leverages a part of the techniques in [14] is constructed with a detailed reduction proof.

Fully Homomorphic Encryption (FHE) develops a key-switching technique [16, 17] that takes as input the old ciphertext and the encryption of the old key under the new key (called the key-switching key) and outputs a new ciphertext that is decryptable by the new key. Such a technique has been used in [4, 12] to construct UE schemes with so-called backward directional key updates. In our UE construction, the matrix in the token is called a key-switching matrix as it achieves the same functionality as the key-switching key in FHE.

3.1.2. OUR APPROACHES

We propose new UE schemes that achieve the new confidentiality notion. To achieve this, we first build a new PKE scheme inspired by [18] that utilizes lattice trapdoor techniques as the underlying encryption scheme. For the UE construction, we leverage the "re-encryption key generation" process in [14] to generate a key-switching matrix, which is used to update ciphertexts from the old to the new key. However, the key-switching matrix alone is not sufficient to achieve our confidentiality notion, which will be discussed later in this section. A detailed proof of our construction is presented in Sect. 3.5.3.

A New PKE Scheme. This scheme is based on lattice trapdoor techniques. The public key is a 1×3 block matrix $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_1] \in \mathbb{Z}^{\bar{m}+2nk}$ where $\mathbf{A}_0$ and $\mathbf{A}_1$ are two random matrices, and $\mathbf{H}_\mu$ is an invertible matrix. The secret key is the trapdoor $\mathbf{R}$ for the first two block matrices of $\mathbf{A}_\mu$, which allows for an efficient

algorithm for inverting LWE samples related to $\mathbf{A}_\mu$ (see Sect. 3.4.1 for more details). The ciphertext is a tuple $c = (\mathbf{H}_\mu, \mathbf{b})$ where $\mathbf{b}$ is a LWE sample as follows:

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \tag{3.1}$$

for a proper encoding algorithm encode, error items $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$, and integer q. To decrypt a ciphertext, one first recovers $\mathbf{s}$ and $(\mathbf{e}_0, \mathbf{e}_1)$ from the first two blocks of $\mathbf{b}$ using the trapdoor $\mathbf{R}$ and an inversion algorithm. Then $\mathbf{m}$ and $\mathbf{e}_2$ are recovered from the last block of $\mathbf{b}$ with the recovered $\mathbf{s}$ and the inverse of encode.

3

Key-switching Matrix. The key-switching matrix enables the transition of a ciphertext in Eq. (3.1) to a new ciphertext with the same form, denoted as $c' = (\mathbf{H}_\mu, \mathbf{b}')$, where

$$\mathbf{b}'^t = \mathbf{s}^t\mathbf{A}'_\mu + \mathbf{e}'^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \tag{3.2}$$

for new public matrix $\mathbf{A}'_\mu$ and new error items $\mathbf{e}'$. This matrix is essentially the transition matrix from $\mathbf{A}_\mu$ to $\mathbf{A}'_\mu$, with the last row block matrix $[\mathbf{0}\ \mathbf{0}\ \mathbf{I}]$, i.e., $\mathbf{A}_\mu \cdot \mathbf{M} = \mathbf{A}'_\mu$. The old ciphertext c is updated by multiplying $\mathbf{b}^t$ and $\mathbf{M}$, that is

$$\begin{aligned}\mathbf{b}^t\mathbf{M} = &\ \mathbf{s}^t\mathbf{A}_\mu \cdot \mathbf{M} + \mathbf{e}^t \cdot \mathbf{M} + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \cdot \mathbf{M} \\ = &\ \mathbf{s}^t\mathbf{A}'_\mu + \mathbf{e}'^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod q.\end{aligned}$$

which matches the desired form in Eq. (3.2). The matrix $\mathbf{M}$ can be efficiently generated by the trapdoor (secret key) $\mathbf{R}$ and the preimage sampling algorithm, as presented in Sect 3.2.2.

Challenges and a New UE Scheme. We state that there are two technical challenges in directly using the key-switching matrix as the update token to construct a secure UE scheme satisfying our confidentiality notion, which requires the indistinguishability between "fresh" and updated ciphertexts. The first observation is that $\mathbf{H}_\mu$, as part of the ciphertext, is never rotated in the update process. The adversary can distinguish the challenge ciphertexts by comparing $\mathbf{H}_\mu$ extracted from the challenge output and input. Beyond that, $\mathbf{s}$ is also never changed during the update process. With the known $\mathbf{s}$ used in the challenge input ciphertext, the adversary may attempt to decrypt the challenge output (note that the last step in the decryption algorithm in PKE only requires $\mathbf{s}$). If it fails, then the adversary knows the challenge output is a fresh encryption of the challenge input message. Otherwise, that is an update of the challenge input ciphertext.

Our solution to address the challenges is to change the invertible matrix $\mathbf{H}_\mu$ and the variable $\mathbf{s}$ in each update. Specifically, a new invertible matrix $\mathbf{H}'_\mu$ and a fresh encryption of message $\mathbf{0}$ under the new key with $\mathbf{H}'_\mu$, denoted by $\mathbf{b}_0$, are generated in the token generation algorithm to improve the randomness. In summary, the update token is a triple $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$, and the update of ciphertext $c = (\mathbf{H}_\mu, \mathbf{b})$ works by multiplying $\mathbf{b}$ by $\mathbf{M}$ and then adding $\mathbf{b}_0$. That is, $c' = (\mathbf{H}'_u, \mathbf{b}')$, where

$$\begin{aligned}(\mathbf{b}')^t = &\ \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \\ = &\ \left[\mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t\right]\mathbf{M} + (\mathbf{s}')^t\mathbf{A}'_\mu + (\mathbf{e}')^t \\ = &\ (\mathbf{s} + \mathbf{s}')^t\mathbf{A}'_\mu + \left(\mathbf{e}^t\mathbf{M} + (\mathbf{e}')^t\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q.\end{aligned}$$

Thus, the updated ciphertext shares the same form as the old ciphertext, but has a new independent invertible matrix and new random factor $\mathbf{s}+\mathbf{s}'$, thereby avoiding the two problems mentioned above. Note that even if an adversary corrupts the update token and the old key (or new key), it can only recover $\mathbf{A}'_\mu$ (or $\mathbf{A}_\mu$, resp.) that is actually public. Therefore, the UE scheme does not leak any information about secret keys, and its token generation process is different from the previously commonly used Dec-then-Enc method.

Regarding the CCA-1 security, at a high level, the decryption procedure allows the adversary to recover at most $\mathbf{A}_\mu$ before the challenge phase, while ensuring that the secret key $\mathbf{R}$ (i.e., the trapdoor) remains statistically hidden from the adversary. We state that the scheme cannot achieve CCA-2 security as the decryption of a challenge ciphertext with extra small noise, which is also a valid ciphertext, reveals the information of the challenge plaintext.

A Packing UE. Our packing UE scheme allows for the simultaneous encryption and update of multiple messages in a single ciphertext. It is based on our UE scheme with the main difference in the encoding algorithm as follows:

$$\mathsf{encode}(\mathbf{m}_0,\ldots,\mathbf{m}_{N-1}) = \mathsf{encode}(\mathbf{m}_0) + \mathsf{encode}(\mathbf{m}_1)X + \cdots + \mathsf{encode}(\mathbf{m}_{N-1})X^{N-1},$$

for messages $\mathbf{m}_0,\ldots,\mathbf{m}_{N-1}$, where the encode in the right side is the same as that in the PKE scheme. Multiple messages blocks are encrypted into one single ciphertext, which can then be recovered degree by degree. This packing scheme enhances efficiency by requiring only one ciphertext header to be downloaded during the update process.

3.1.3. SUMMARY OF CONTRIBUTIONS

We strengthen the confidentiality notions for c-d UE to address the above limitations 1-3 of existing work and provide efficient UE schemes that achieve the confidentiality we define. First, we simplify the description of the confidentiality model by reducing the number of oracles available to the adversary, while maintaining the same level of security. This simplification facilitates the security analysis of UE schemes. Our new definition "maximizes" the capability of the adversary, including the ability to corrupt keys in an adaptive manner and gain access to the decryption oracle, thus providing a stronger security than prior work. We then propose a new construction that is the first c-d UE to achieve adaptive security under the LWE assumption. It is built on our lattice-based PKE scheme, and rotates ciphertext with a key-switching matrix, which differs from the Dec-then-Enc structure used in existing c-d UE schemes. We also propose a new packing method to further enhance the efficiency of c-d UE. Our approach enables multiple messages to be encrypted and updated simultaneously, reducing the overhead associated with downloading ciphertext headers during the update process.

3.2. PRELIMINARIES

We use upper-case and lower-case bold letters to denote matrices and column vectors, respectively. For a vector $\mathbf{x}$, we denote the 2-norm of $\mathbf{x}$ by $\|\mathbf{x}\|$ and the infinity norm by $\|\mathbf{x}\|_\infty$. The largest singular value of a matrix $\mathbf{B}$ is denoted by $s_1(\mathbf{B}) := \max_{\mathbf{u}} \|\mathbf{B}^t\mathbf{u}\|$, where the maxima is taken over all unit vectors $\mathbf{u}$ and $\mathbf{B}^t$ is the transpose of $\mathbf{B}$. For two matrices $\mathbf{A}$ and $\mathbf{B}$, $[\mathbf{A} \mid \mathbf{B}]$ denotes the concatenation of the columns of $\mathbf{A}$ and $\mathbf{B}$. We also use standard asymptotical notations such as ω, Ω and O.

3

3.2.1. UPDATABLE ENCRYPTION

We briefly review the syntax of ciphertext-dependent UE and prior confidentiality notions for c-d UE.

Definition 3.2.1 ([1, 8, 10])**.** *A ciphertext-dependent UE scheme includes a tuple of PPT algorithms* {KG, Enc, Dec, TokenGen, Update} *that operate in epochs starting from 0.*

- $\mathsf{KG}(1^\lambda)$*: the key generation algorithm outputs an epoch key* $\mathsf{k_e}$.
- $\mathsf{Enc}(\mathsf{k_e}, \mathsf{m})$*: the encryption algorithm takes as input an epoch key* $\mathsf{k_e}$ *and a message* m *and outputs a ciphertext header* $\hat{\mathsf{ct}}_\mathsf{e}$ *and a ciphertext body* $\mathsf{ct_e}$*, i.e.,* $\mathsf{ct} = (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$.
- $\mathsf{Dec}(\mathsf{k_e}, (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e}))$*: the decryption algorithms takes as input an epoch key* $\mathsf{k_e}$ *and a ciphertext* $(\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$ *and outputs a message* m' *or* $\perp$.
- $\mathsf{TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}}, \hat{\mathsf{ct}}_\mathsf{e})$*: the token generation algorithm takes as input two epoch keys* $\mathsf{k_e}$ *and* $\mathsf{k_{e+1}}$ *and a ciphertext header* $\hat{\mathsf{ct}}_\mathsf{e}$*, and outputs an update token* $\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}_\mathsf{e}}$ *or* $\perp$.
- $\mathsf{Update}(\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}_\mathsf{e}}, (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e}))$*: the update algorithm takes as input a token* $\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}}$ *related to the ciphertext* $(\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$*, and outputs an updated ciphertext* $(\hat{\mathsf{ct}}_{\mathsf{e+1}}, \mathsf{ct_{e+1}})$ *or* $\perp$.

In an updatable encryption scheme, there are two ways to generate a ciphertext: either via the encryption algorithm to produce the fresh ciphertext, or via the update algorithm to produce an updated ciphertext. The *correctness* of a UE scheme requires both types of ciphertexts to decrypt correctly to the underlying message, except with a low failure probability.

Prior Notions of Confidentiality. To capture the security under key leakage, the challenger in prior confidentiality games [8, 10] provides the adversary some selective keys in the setup phase. In the query phase, the adversary is given access to query the algorithms involved in UE schemes, including {Enc, TokenGen, Update}, to obtain the encryption of messages, update tokens, and updates of ciphertexts, respectively. The adversary then submits two challenge inputs in the challenge phase based on the information it has acquired and receives the challenge output

from the challenger. The goal of the adversary is to guess which challenge input the challenge output is related to (encrypted or updated from). The adversary can continue querying those oracles as long as the combination of queries would not lead to a trivial win, and eventually submits a guess bit.

Prior confidentiality notions have three variants with the only difference in challenge inputs: UP-IND [10] has inputs of two messages $(\bar{m}_0, \bar{m}_1)$ to capture the security of fresh encryptions, UP-REENC [10] uses inputs of two ciphertexts $(\bar{c}_0, \bar{c}_1)$ to protect the confidentiality after updating, and Confidentiality [8], which is stronger the former two, takes one message and one ciphertext as input $(\bar{m}_0, \bar{c}_1)$ to protect against the leakage of the *age* of ciphertext, i.e., the number of update times, to the adversary. We rewrite the confidentiality game of Confidentiality in Fig. 3.2 with two modifications.

First, we describe oracles that operate in consecutive epochs $\{\ldots, e-1, e, e+1, \ldots\}$, which is more consistent with the practical periodic updating of ciphertexts and differs from the node-based oracles originating from proxy re-encryption in prior work. Second, we introduce a new lookup table in the game to track non-challenge ciphertexts (as defined in Definition 3.2.2) to address the insufficient analysis of trivial win conditions for deterministic UE schemes in prior work [1, 8–10]. Our main observation is that for UE schemes with deterministic updates, the adversary should be prevented from querying $\mathcal{O}_{\mathsf{Update}}$ and $\mathcal{O}_{\mathsf{TokenGen}}$ on the challenge input ciphertext in the challenge epoch before querying the challenge oracle, as this would enable the adversary to know one of the possible challenge output ciphertexts in advance due to the determinism of the update. Such conditions are not analyzed in prior notions, which are therefore only applicable to UE with randomized updates; however, the update algorithm can be deterministic as in our construction, even though the encryption algorithm must be randomized.

Definition 3.2.2. *A ciphertext is called challenge-equal ciphertext, if the adversary learns it via querying the challenge oracle $\mathcal{O}_{\mathsf{Chall}}$, or obtains it by updating the challenge ciphertext using $\mathcal{O}_{\mathsf{Update}}$ or tokens acquired from $\mathcal{O}_{\mathsf{TokenGen}}$. Any ciphertext that is not obtained through these methods is referred to as a non-challenge ciphertext.*

The functionalities and restrictions of oracles used in the Confidentiality game in Fig. 3.2 are as follows.

- $\mathcal{O}_{\mathsf{Enc}}$: returns an encryption of a message.
- $\mathcal{O}_{\mathsf{Update}}$: returns an update of a valid (lines 1-3) ciphertext, recorded by $\mathsf{TC}_{\mathsf{chall}}$ (line 9) or $\mathsf{TC}_{\mathsf{non}}$ (line 12) according to the input. But the update of challenge-equal ciphertexts in epochs with known epoch keys is not allowed (line 6).
- $\mathcal{O}_{\mathsf{TokenGen}}$: returns a token related to a valid ciphertext, and updates $\mathsf{TC}_{\mathsf{chall}}$ (line 7) or $\mathsf{TC}_{\mathsf{non}}$ (line 11). But tokens related to challenge-equal ciphertexts in epochs with known epoch keys are not allowed to be acquired (lines 1-2).
- $\mathcal{O}_{\mathsf{Chall}}$: returns the challenge output, either a fresh encryption of the input message or an update of input valid ciphertext (lines 2-4). However, this oracle

$\text{Expt}^{\text{Confidentiality}}_{\text{UE}}(\lambda, l, \mathscr{A}, b)$:

$k_1, \ldots, k_l \leftarrow \text{KG}(1^\lambda)$
$b' \leftarrow \mathscr{A}^{\mathcal{O}}(\mathcal{K})$
return $b' = b$

$\mathcal{O}_{\text{Enc}}(e, m)$:

$(\hat{ct}, ct) \leftarrow \text{Enc}(k_e, m)$
$\text{TC}_{\text{non}}[e, \hat{ct}] \leftarrow ct$
return $(\hat{ct}, ct)$

$\mathcal{O}_{\text{Update}}(e, (\hat{ct}, ct))$:

if $\text{TC}_{\text{chall}}[e-1, \hat{ct}] = \perp$ and
  $\text{TC}_{\text{non}}[e-1, \hat{ct}] = \perp$ **then**
  return $\perp$
$\Delta_{e,\hat{ct}} \leftarrow \text{TokenGen}(k_{e-1}, k_e, \hat{ct})$
if $\text{TC}_{\text{chall}}[e-1, \hat{ct}] \neq \perp$ **then**
  if $e \in \mathcal{K}$ **return** $\perp$
  else $ct \leftarrow \text{TC}_{\text{chall}}[e-1, \hat{ct}]$
    $(\hat{ct}', ct') \leftarrow \text{Update}(\Delta_{e,\hat{ct}}, (\hat{ct}, ct))$
    $\text{TC}_{\text{chall}}[e, \hat{ct}'] \leftarrow ct'$
else $ct \leftarrow \text{TC}_{\text{non}}[e-1, \hat{ct}]$
    $(\hat{ct}', ct') \leftarrow \text{Update}(\Delta_{e,\hat{ct}}, (\hat{ct}, ct))$
    $\text{TC}_{\text{non}}[e, \hat{ct}'] \leftarrow ct'$
return $(\hat{ct}', ct')$

$\mathcal{O}_{\text{TokenGen}}(e, \hat{ct})$:

if $e \in \mathcal{K}$ and $\text{TC}_{\text{chall}}[e-1, \hat{ct}] \neq \perp$
  return $\perp$
$\Delta_{e,\hat{ct}} \leftarrow \text{TokenGen}(k_{e-1}, k_e, \hat{ct})$
if $e \notin \mathcal{K}$ and $\text{TC}_{\text{chall}}[e-1, \hat{ct}] \neq \perp$
  $ct \leftarrow \text{TC}_{\text{chall}}[e-1, \hat{ct}]$
  $(\hat{ct}', ct') \leftarrow \text{Update}(\Delta_{e,\hat{ct}}, (\hat{ct}, ct))$
  $\text{TC}_{\text{chall}}[e, \hat{ct}'] \leftarrow ct'$
elseif $\text{TC}_{\text{non}}[e-1, \hat{ct}] \neq \perp$
  $ct \leftarrow \text{TC}_{\text{non}}[e-1, \hat{ct}]$
  $(\hat{ct}', ct') \leftarrow \text{Update}(\Delta_{e,\hat{ct}}, (\hat{ct}, ct))$
  $\text{TC}_{\text{non}}[e, \hat{ct}'] \leftarrow ct'$
else return $\perp$
return $\Delta_{e,\hat{ct}}$

$\mathcal{O}_{\text{Chall}}(e, m, (\hat{ct}, ct))$:

if $e \in \mathcal{K}$ **return** $\perp$
$(\hat{ct}'_0, ct'_0) \leftarrow \text{Enc}(k_e, m)$
if $(\hat{ct}'_0, ct'_0) = \perp$ or $\text{TC}_{\text{non}}[e-1, \hat{ct}] \neq ct$
  return $\perp$
$\Delta_{e,\hat{ct}} \leftarrow \text{TokenGen}(k_{e-1}, k_e, \hat{ct})$
$(\hat{ct}'_1, ct'_1) \leftarrow \text{Update}(\Delta_{e,\hat{ct}}, (\hat{ct}, ct))$
if $|\hat{ct}'_0| \neq |\hat{ct}'_1|$ or $|ct'_0| \neq |ct'_1|$
  return $\perp$
if $(xx = \text{det}$ and $\text{TC}_{\text{non}}[e, \hat{ct}'_1] = ct'_1)$
  return $\perp$
$\text{TC}_{\text{chall}}[e, \hat{ct}'_b] \leftarrow ct'_b$
return $(\hat{ct}'_b, ct'_b)$

Figure 3.2: Security game for Confidentiality. The adversary in the startup is provided with selective keys whose epochs are recorded by the set $\mathcal{K}$, and the other keys are kept private from the adversary. Initially set to be empty, the table T_{chall} (or T_{non}) maps an epoch and challenge-equal (or non-challenge, respectively) ciphertext header pair to the corresponding challenge-equal (or non-challenge, respectively) ciphertext body. $xx = \text{det}$ means the update algorithm is deterministic.

should not be queried in epochs with known epoch keys (line 1), and for deterministic UE, the input ciphertext should not be updated in advance (lines 9-10).

In fact, the adversary may infer more ciphertexts, tokens, and keys from corrupted information, aside from the recorded sets, and the extended leakages cannot be tracked (but can be computed) by look-up tables. For example, a token can be inferred if two successive epoch keys are known. We will show in theorems 3.3.4, 3.3.6 and 3.3.7 that trivial win conditions on recorded leakages and extended leakages are actually the same for no-directional UE (Def. 3.3.1). Therefore, it is sufficient to check the above restrictions on recorded look-up tables to avoid trivial win.

3.2.2. GAUSSIANS AND LATTICES

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we first review the Learning With Errors (LWE) and Short Integer Solution (SIS) problems as follows:

- $\mathsf{LWE}_{q,\alpha}$: for arbitrary $\mathbf{s} \in \mathbb{Z}_q^n$ and error $\mathbf{e}$ from the discrete Gaussian distribution $D_{\mathbb{Z}^m,\alpha q}$ (Def. 3.2.6), let $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$. The *search*-$\mathsf{LWE}_{q,\alpha}$ is to find $\mathbf{s}$ and $\mathbf{e}$ from $(\mathbf{A},\mathbf{b})$; the *decision*-$\mathsf{LWE}_{q,\alpha}$ is to distinguish between $\mathbf{b}$ and a uniformly random sample from $\mathbb{Z}_q^m$.
- $\mathsf{SIS}_{q,\beta}$: find a nonzero $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\| \le \beta$.

When $\mathbf{A}$ is a uniformly random matrix, solving the above two problems is computationally intractable under some parameter settings [19, 20]. However, for a random matrix $\mathbf{A}$ with a $\mathbf{G}$-trapdoor (Def. 3.2.3), those two problems can be solved immediately (Lemma 3.2.4 and Lemma 3.2.5).

For the rest of the paper, let $q \ge 2$ be an integer modulus with $k = \lceil \log_2 q \rceil$, and $\mathbf{G}$ is defined as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n\times nk}$, i.e.,

$$\mathbf{G} = diag(\mathbf{g}^t, \ldots, \mathbf{g}^t),$$

where $\mathbf{g}^t = [1\ 2\ 4\ \ldots\ 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$ and integer $n \ge 1$.

Definition 3.2.3 ($\mathbf{G}$-trapdoor)**.** *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for some $m \ge nk \ge n$. A $\mathbf{G}$-trapdoor for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}_q^{(m-nk)\times nk}$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{HG}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$.*

As an example in [18], $\mathbf{R}$ is a $\mathbf{G}$-trapdoor for a random matrix $\mathbf{A} = [\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{HG}]$, where $\mathbf{A}_0$ is a uniform matrix in $\mathbb{Z}_q^{n\times m}$, $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ is an invertible matrix and $\mathbf{R}$ is chosen from a distribution over $\mathbb{Z}_q^{m\times nk}$.

Lemma 3.2.4 ([18], Theorem 5.4)**.** *Given a $\mathbf{G}$-trapdoor $\mathbf{R}$ for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and an* LWE *instance $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, if $\left\|[\mathbf{R}^t\ \mathbf{I}] \cdot \mathbf{e}\right\|_\infty \le q/4$, then there is an efficient algorithm called* $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R},\mathbf{A},\mathbf{H},\mathbf{b})$ *that recovers $\mathbf{s}$ and $\mathbf{e}$ from the $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$.*

Lemma 3.2.5 ([18], Theorem 5.5)**.** *Given a $\mathbf{G}$-trapdoor $\mathbf{R}$ for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with invertible matrix $\mathbf{H}$ and any $\mathbf{u} \in \mathbb{Z}_q^n$, there is an efficient algorithm called* $\mathsf{SampleD}^{\mathcal{O}}(\mathbf{R},\mathbf{A},\mathbf{H},\mathbf{u},s)$

that samples a Gaussian vector $\mathbf{x}$ *from* $D_{\mathbb{Z}^m,s}$ *such that* $\mathbf{Ax}=\mathbf{u}$*, where* s *can be as small as* $\sqrt{s_1(\mathbf{R})^2+1}\cdot\sqrt{s_1(\Sigma_\mathbf{G})+1}\cdot\omega(\sqrt{\log n})$ *and* $s_1(\Sigma_\mathbf{G})$ *is a constant for given* $\mathbf{G}$*(equal to 4 if* q *is a power of 2, and 5 otherwise).*

Definition 3.2.6 ([21])**.** *For a positive real* s*, the discrete Gaussian distribution over a countable set* A *is defined by the density function*

$$D_{A,s}(x) := \frac{\rho_s(x)}{\sum_{y\in A}\rho_s(y)},$$

where $\rho_s(x)=\exp\left(-\pi\|x\|^2/s^2\right)$.

Lemma 3.2.7 ([22], Lemma 1.5)**.** *Let* $c\geq 1, C=c\cdot\exp\left(\left(1-c^2\right)/2\right)$*. For any real* $s>0$ *and any integer* $n\geq 1$*, we have that*

$$\Pr_{\mathbf{e}\leftarrow D_{\mathbb{Z}^n,s}}\left[\|\mathbf{e}\|\geq cs\sqrt{\frac{n}{2\pi}}\right]\leq C^n.$$

In particular, letting $c=\sqrt{2\pi}$ *and* $C<1/4$ *, we have*

$$\Pr_{\mathbf{e}\leftarrow D_{\mathbb{Z}^n,s}}\left[\|\mathbf{e}\|\geq s\sqrt{n}\right]<2^{-2n}$$

Lemma 3.2.8 ([18], Lemma 2.9)**.** *Let* $\mathrm{X}\in\mathbb{R}^{n\times m}$ *be a* δ*-subgaussian random matrix with parameter* s*. There exists a universal constant* $C>0$ *such that for any* $t\geq 0$*, we have* $s_1(\mathbf{X})\leq C\cdot s\cdot(\sqrt{m}+\sqrt{n}+t)$ *except with probability at most* $2\exp(\delta)\exp\left(-\pi t^2\right)$.

Lemma 3.2.9 ([23], Fact 6)**.** *For any* $m,n,s>0$*, let* $\mathbf{R}\in D_{\mathbb{Z},s}^{n\times m}$*, we have* $s_1(\mathbf{R})\leq s\cdot O(\sqrt{n}+\sqrt{m})$*, except with probability* $2^{-\Omega(n+m)}$.

The following lemma bounds the maximal singular value of the product and addition of two matrices, which follows directly from the definition.

Lemma 3.2.10. *Let* $\mathbf{A}\in\mathbb{R}^{m\times n}$*,* $\mathbf{B}\in\mathbb{R}^{n\times m}$*, then* $s_1(\mathbf{AB})\leq s_1(\mathbf{A})s_1(\mathbf{B})$ *and* $s_1(\mathbf{A}+\mathbf{B})\leq s_1(\mathbf{A})+s_1(\mathbf{B})$.

Lemma 3.2.11 (Leftover Hash Lemma)**.** *Let* $\mathscr{P}$ *be a distribution over* $\mathbb{Z}_q^n$ *with min-entropy* k*. For any* $\epsilon>0$ *and* $l\leq(k-2\log(1/\epsilon)-O(1))/\log(q)$*, the joint distribution of* $(\boldsymbol{C},\boldsymbol{Cs})$ *is* ϵ*-close to the uniform distribution over* $\mathbb{Z}_q^{l\times n}\times\mathbb{Z}_q^l$*, where* $\boldsymbol{C}\xleftarrow{\$}\mathbb{Z}_q^{l\times n}$ *and* $s\leftarrow\mathscr{P}$.

Lemma 3.2.12 ([24], Fact 2.2)**.** *Let* $X_1,\ldots,X_n$ *be independent mean-zero subgaussian random variables with parameter* s*, and let* $u\in\mathbb{R}^n$ *be arbitrary. Then* $\sum_k(a_kX_k)$ *is subgaussian with parameter* $s\|u\|$.

3.3. NEW CONFIDENTIALITY NOTIONS FOR UPDATABLE ENCRYPTION

To simplify the security notion given in [8], we define a new confidentiality notion called sConfidentiality, where we replace $\mathscr{O}_{\mathsf{TokenGen}}$ and $\mathscr{O}_{\mathsf{Update}}$ in the security game

with a single $\mathcal{O}_{\mathsf{sUpd}}$ that returns both the update token and updated ciphertext to the adversary simultaneously. We prove in Theorem 3.3.3 that sConfidentiality and Confidentiality are equal for UE schemes with no-directional key updates.

Meanwhile, to provide the adversary with maximum power, we introduce a new stronger confidentiality notion than sConfidentiality, called xxIND-UE-atk[3], where the adversary is given extra access to $\mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_{\mathsf{Corr}}$, which enables it to corrupt epoch keys at any time during the game. To avoid making the security game trivial, we fully analyze the conditions for any trivial win in this game model. A brief comparison of the proposed notions with those of prior work is presented in Fig. 3.3.

Notions	Oracles	Key	Challenge Input	Update
UP-IND [10]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	S.	$(\bar{m}_0, \bar{m}_1)$	rand
UP-REENC [10]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	S.	$(\bar{c}_0, \bar{c}_1)$	rand
Confidentiality [8]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	S.	$(\bar{m}_0, \bar{c}_1)$	rand
sConfidentiality Sect. 3.3.2	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}$	S.	$(\bar{m}_0, \bar{c}_1)$	rand
xxIND-UE-CPA Sect. 3.3.3	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}},$	A.	$(\bar{m}_0, \bar{c}_1)$	xx
xxIND-UE-CCA Sect. 3.3.3	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}}, \mathcal{O}_{\mathsf{Dec}}$	A.	$(\bar{m}_0, \bar{c}_1)$	xx

Figure 3.3: A summary of confidentiality notions, where $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}\}$ represents the update procedure can be either randomized or deterministic. The abbreviations 'S.' and 'A.' represent the methods of key compromising, specifically 'selective' and 'adaptive,' respectively. The adversary in each confidentiality game provides two challenge inputs based on the oracles it has access to and tries to distinguish the challenge outputs. Confidentiality is proven stronger than both UP-IND and UP-REENC in [8], and $\mathcal{O}_{\mathsf{sUpd}}$ is defined in Sect. 3.3.2. Chen et al. [9] proposed strengthened UP-IND and UP-REENC to capture malicious update security, with the modification in the oracle $\mathcal{O}_{\mathsf{Update}}$ that enables the adversary to query the update of maliciously generated ciphertexts, instead of only honestly generated ciphertexts as in [10].

3.3.1. UE SCHEMES WITH NO-DIRECTIONAL KEY UPDATES

In c-i UE schemes, update tokens are generated by two successive epoch keys: $\Delta = \mathsf{TokenGen}(k_e, k_{e+1})$, e.g., $\Delta = k_{e+1}/k_e$ in [2] or $\Delta = k_{e+1} - k_e$ in [3]); therefore,

[3]The same notion for the c-i UE scheme was proposed in [2]. We aim to unify the notions for c-i/c-d UE that both capture adaptive security and prevent the leakage of ciphertext age. Note that, as analyzed in the introduction, there are intrinsic differences between c-i UE and c-d UE. The disparity is evident in the confidentiality notion, specifically in the approach to recording leakage sets.

one key may be derived by the other if the token is known by the adversary. However, in c-d UE schemes, tokens are also determined by the ciphertext header: $\Delta = \mathsf{TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}}, \hat{\mathsf{ct}}_\mathsf{e})$, so keys may not be derived via corrupted tokens. We generalize the definition of no-directional key updates from c-i UE to c-d UE as follows.

Definition 3.3.1. *A UE scheme, either ciphertext-independent or ciphertext-dependent, is said to have no-directional key updates if epoch keys cannot be inferred from known tokens.*

Jiang [3] proposed the open problem of constructing no-directional c-i UE schemes. However, all known c-d UE schemes in [1, 8–10] (as well as our construction in Sect. 3.5) have no-directional key updates, which benefit from a Dec-then-Enc process as discussed in the introduction. In contrast, there are only two c-i UE schemes with no-directional key update: one is not practical [4] and the other is less efficient [7]. In the following, we focus on c-d UE schemes with no-directional key updates.

3.3.2. A Simplified Confidentiality Notion

Based on our refinement on Confidentiality, we now define a new simplified confidentiality notion by substituting the oracles $\mathcal{O}_{\mathsf{TokenGen}}$ and $\mathcal{O}_{\mathsf{Upd}}$ in the Confidentiality game with a single $\mathcal{O}_{\mathsf{sUpd}}$ that returns both the token and update simultaneously. We call this new notion sConfidentiality. In Theorem 3.3.3, we prove sConfidentiality is equivalent to Confidentiality for UE schemes with no-directional key updates, as defined in [8].

Definition 3.3.2 (sConfientiality)**.** *Let* $\mathsf{UE} = \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update}\}$ *be an updatable encryption scheme. For a security parameter* λ*, an integer* l*, an adversary* $\mathcal{A}$*, and a binary bit* $b \in \{0,1\}$*, we define the confidentiality experiment* $\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, l, \mathcal{A}, b)$ *and oracles* $\mathcal{O} = \{\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Chall}}\}$ *as described in Fig. 3.4. The experiment maintains two look-up tables* $\mathsf{TC}_{\mathsf{non}}$ *and* $\mathsf{TC}_{\mathsf{chall}}$ *that record non-challenge and challenge-equal ciphertexts known to the adversary, respectively, and an epoch set* $\mathcal{K}$ *in which epoch keys are provided to the adversary in setup.*

We say that an updatable encryption scheme UE *satisfies* sConfidentiality *if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\mathcal{K} \subseteq [0, \ldots, l]$ *and efficient adversaries* $\mathcal{A}$*, we have*

$$\left|\Pr\left[\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, l, \mathcal{A}, 0) = 1\right] - \Pr\left[\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, l, \mathcal{A}, 1) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

Theorem 3.3.3. *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update})$ *be an updatable encryption scheme with no-directional key updates. For any* sConfidentiality *adversary* $\mathcal{A}$ *against* UE*, there is a* Confidentiality *adversary* $\mathcal{B}$ *against* UE *such that*

$$\mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathcal{B}}(1^\lambda). \tag{3.3}$$

In addition, for any Confidentiality *adversary* $\mathcal{B}$ *against* UE*, there is a* sConfidentiality *adversary* $\mathcal{A}$ *against* UE *such that*

$$\mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathcal{B}}(1^\lambda) = \mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathcal{A}}(1^\lambda).$$

$\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, l, \mathscr{A}, b)$:

1 : $\mathsf{k}_1, \ldots, \mathsf{k}_l \leftarrow \mathsf{KG}(1^\lambda)$
2 : $b' \leftarrow \mathscr{A}^{\mathscr{O}}(\mathscr{K})$
3 : **return** $b' = b$

$\mathscr{O}_{\mathsf{Enc}}(\mathsf{e}, \mathsf{m})$:

1 : $(\hat{\mathsf{ct}}, \mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{k}_\mathsf{e}, \mathsf{m})$
2 : $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}] \leftarrow \mathsf{ct}$
3 : **return** $(\hat{\mathsf{ct}}, \mathsf{ct})$

$\mathscr{O}_{\mathsf{sUpd}}(\mathsf{e}, (\hat{\mathsf{ct}}, \mathsf{ct}))$:

1 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ and
2 : $\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ **then**
3 : $\quad$ **return** $\perp$
4 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}})$
5 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$ **then**
6 : $\quad$ **if** $\mathsf{e} \in \mathscr{K}$ **return** $\perp$
7 : $\quad$ **else** $\mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
8 : $\quad\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
9 : $\quad\quad \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
10 : **else** $\mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
11 : $\quad\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
12 : $\quad\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
13 : **return** $(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}', \mathsf{ct}'))$

$\mathscr{O}_{\mathsf{Chall}}(\mathsf{e}, \mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct}))$:

1 : **if** $\mathsf{e} \in \mathscr{K}$ **return** $\perp$
2 : $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) \leftarrow \mathsf{Enc}(\mathsf{k}_\mathsf{e}, \mathsf{m})$
3 : **if** $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) = \perp$ or $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \mathsf{ct}$
4 : $\quad$ **return** $\perp$
5 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}})$
6 : $(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
7 : **if** $|\hat{\mathsf{ct}}'_0| \neq |\hat{\mathsf{ct}}'_1|$ or $|\mathsf{ct}'_0| \neq |\mathsf{ct}'_1|$
8 : $\quad$ **return** $\perp$
9 : **if** $(\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'_1] = \mathsf{ct}'_1)$
10 : $\quad$ **return** $\perp$
11 : $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'_b] \leftarrow \mathsf{ct}'_b$
12 : **return** $(\hat{\mathsf{ct}}'_b, \mathsf{ct}'_b)$

Figure 3.4: Security game for sConfidentiality in Definition 3.3.2.

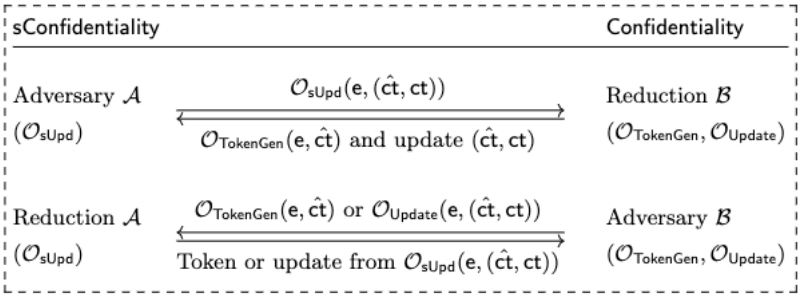


Figure 3.5: Reductions in the proof of Theorem 3.3.3. When the adversary makes queries to specific oracles, indicated above the arrow, the reduction forwards to the adversary the corresponding responses from its own challenger, marked below the arrow.

Proof. In general, we construct a reduction that runs the Confidentiality (or sConfidentiality) game and simulates all responses to the queries of the adversary in

the sConfidentiality (or Confidentiality game, respectively), as shown in Fig. 3.5. The details are presented as follows.

We provide reductions for both sConfidentiality and Confidentiality games. For any sConfidentiality adversary $\mathscr{A}$, we construct a reduction $\mathscr{B}$ that runs the Confidentiality game and simulates all responses to the queries of $\mathscr{A}$ as the first line shown in Fig. 3.5. The reduction sends all its known keys to $\mathscr{A}$, and all $\mathscr{A}$'s queries except $\mathcal{O}_{\mathsf{sUpd}}$ to its challenger, and returns the challenger's responses to $\mathscr{A}$.

When $\mathscr{A}$ queries the oracle $\mathcal{O}_{\mathsf{sUpd}}$ on some input, the reduction $\mathscr{B}$ submits the same input to $\mathcal{O}_{\mathsf{TokeGen}}$. If the response of the challenger is $\perp$, $\mathscr{B}$ also sends $\perp$ to $\mathscr{A}$; otherwise, $\mathscr{B}$ calculates the updated ciphertext by the received update token and forwards the update token, together with the updated ciphertext, to the adversary $\mathscr{A}$. At last, $\mathscr{B}$ forwards $\mathscr{A}$'s guess to its challenger.

We check the trivial win conditions of $\mathscr{A}$ and $\mathscr{B}$ one by one. Suppose $\mathscr{A}$ does not query an update of challenge-equal ciphertext in an epoch in which it knows the key (via $\mathcal{O}_{\mathsf{sUpd}}$). From the proof of Lemma 3.3.4, then we know $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$, and therefore the epochs in which $\mathscr{A}$ knows a challenge-equal ciphertext are the same as $\mathscr{B}$. Therefore, $\mathscr{B}$ will not query the token related to challenge-equal ciphertexts in epochs in which it knows the epoch key (since $\mathscr{A}$ and $\mathscr{B}$ have the same known keys). There are no additional restrictions for randomized UE. For deterministic UE, the adversary should not learn the update of challenge input ciphertext in advance before the challenge. Note that the look-up table $\mathsf{TC}_{\mathsf{non}}$ for $\mathscr{A}$ and $\mathscr{B}$ is the same. If $\mathscr{A}$ does not query $\mathcal{O}_{\mathsf{sUpd}}(\tilde{\mathsf{e}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ before querying the challenge oracle, the same applies to $\mathscr{B}$ (recall the ciphertexts acquired before the challenge are all recorded in $\mathsf{TC}_{\mathsf{non}}$). Therefore, $\perp$ will also not be returned from the challenger of $\mathscr{B}$ when $\mathscr{A}$ does not trigger trivial win conditions. Thus $\mathscr{B}$ has at least the same advantage as $\mathscr{A}$, i.e., the Inequality (3.3).

Similarly, for any Confidentiality adversary $\mathscr{B}$, we construct a reduction $\mathscr{A}$ that runs the sConfidentiality game and simulates all the responses to the queries of the given $\mathscr{B}$ as shown in the second line of Fig. 3.5. The reduction $\mathscr{A}$ sends all its known keys to $\mathscr{B}$ and all $\mathscr{B}$'s queries except those on $\mathcal{O}_{\mathsf{TokenGen}}$ and $\mathcal{O}_{\mathsf{Upd}}$ to its challenger, and returns its challenger's responses to $\mathscr{B}$. When $\mathscr{B}$ queries the oracle $\mathcal{O}_{\mathsf{TokenGen}}$ (or $\mathcal{O}_{\mathsf{Upd}}$) on some input, the reduction $\mathscr{A}$ submits the same input to the $\mathcal{O}_{\mathsf{sUpd}}$ oracle, and returns the update token (or updated ciphertext, respectively) received from its challenger to $\mathscr{B}$ if the response is not $\perp$; otherwise, $\mathscr{A}$ returns $\perp$ to $\mathscr{B}$.

Therefore, the reduction $\mathscr{A}$ simulates all responses to $\mathscr{B}$'s queries. Suppose $\mathscr{B}$ does not query an update of challenge-equal ciphertext in an epoch in which it knows the epoch key. By proof of Lemma 3.3.4 again, we know $\mathscr{A}$ does not query $\mathsf{O}_{\mathsf{sUpd}}$ in an epoch in which it knows the epoch key. In addition, the analysis for deterministic is almost the same as above. We omit the details. Thus, we have

$$\mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathscr{B}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathscr{A}}(1^\lambda).$$

In combination with (3.3), we conclude the advantage of $\mathscr{A}$ is equal to that of $\mathscr{B}$. □

3.3.3. A STRONGER CONFIDENTIALITY NOTION

We now provide a stronger confidentiality notion, called xxIND-UE-atk for c-d UE in Definition 3.3.8, which provides the adversary with more power than the notion of sConfidentiality in Sect. 3.3.2. All available oracles that the adversary has access to are described in Fig. 3.8. The stronger notion allows the adversary to corrupt keys at any time during the game by querying $\mathcal{O}_{\mathsf{Corr}}$, instead of selecting the compromised keys in the setup phrase. In addition, the adversary is provided with an extra ability to query the decryption oracle compared with sConfidentiality. Prior to defining xxIND-UE-atk, we first analyze the conditions that lead the adversary to trivially win the game through a combination of queries, which therefore should be excluded from the game.

Leakage Information. To track the information leaked to the adversary, we similarly record two look-up tables $\mathsf{TC}_{\mathsf{non}}$ and $\mathsf{TC}_{\mathsf{chall}}$ as defined in Sect. 3.3.2, and an epoch set $\mathcal{K}$ in which the epoch key is corrupted via $\mathcal{O}_{\mathsf{Corr}}$. We define $\mathsf{TC}_{\mathsf{chall}}[0]$ as the set of epochs in which the adversary learns a challenge-equal ciphertext, and $\mathcal{T}$ as the set of epochs in which the adversary learns a token corresponding to a challenge-equal ciphertext, which are exactly the epochs stored in $\mathsf{TC}_{\mathsf{chall}}$ and $\Delta_{\mathsf{e},\hat{\mathsf{ct}}}$, respectively. A summary of notations is shown in Table 3.1.

Table 3.1: Summary of leakage set notations

Notations	Descriptions
$\mathsf{TC}_{\mathsf{non}}$	Look-up table recording leaked non-challenge ciphertexts
$\mathsf{TC}_{\mathsf{chall}}$	Look-up table recording leaked challenge-equal ciphertexts
$\mathsf{TC}_{\mathsf{chall}}[0]$	Set of epochs in which a challenge-equal ciphertext is learned
$\mathcal{K}$	Set of epochs in which the adversary learned the epoch key
$\mathcal{T}$	Set of epochs in which a token w.r.t. a challenge-equal ct is learned

Leakage Extension. Note that the adversary possibly extends its corrupted information $\mathsf{TC}_{\mathsf{non}}, \mathsf{TC}_{\mathsf{chall}}, \mathcal{K}$ via corrupted tokens, and the former leakages may also in turn help to corrupt more tokens. We denote $\mathsf{TC}^*_{\mathsf{chall}}[0], \mathcal{K}^*, \mathcal{T}^*$ as the extended sets of $\mathsf{TC}_{\mathsf{chall}}[0], \mathcal{K}, \mathcal{T}$, respectively. Following the analysis in [5], the extended leakage sets are computed as follows:

$$\mathcal{K}^* = \mathcal{K} \text{ (no-directional key updates)}, \tag{3.4}$$

$$\mathcal{T}^* = \{\mathsf{e} \in \{0,\dots,l\} \mid (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^* \wedge \mathsf{e}-1 \in \mathcal{K}^*\}, \tag{3.5}$$

$$\begin{aligned}\mathsf{TC}^*_{\mathsf{chall}}[0] = \{\mathsf{e} \in \{0,\dots,l\} \mid (\mathsf{e} \in \mathsf{TC}_{\mathsf{chall}}[0]) \vee (\mathsf{e}-1 \in \mathsf{TC}_{\mathsf{chall}}[0] \wedge \mathsf{e} \in \mathcal{T}^*) \vee \\ (\mathsf{e}+1 \in \mathsf{TC}_{\mathsf{chall}}[0] \wedge \mathsf{e}+1 \in \mathcal{T}^*)\}.\end{aligned} \tag{3.6}$$

An example is shown in Fig. 3.6. Assume the adversary queries $\mathcal{O}_{\mathsf{sUpd}}$ only in epoch $\mathsf{e}-5$ and corrupts epoch keys in epochs $\mathsf{e}-5$ and $\mathsf{e}-4$. Even though it cannot learn the token in epoch $\mathsf{e}-4$ by $\mathcal{O}_{\mathsf{sUpd}}$, it can infer that token via corrupted keys in $\mathsf{e}-5$ and $\mathsf{e}-4$, which further infers the ciphertexts in $\mathsf{e}-4$.

Epoch	…	$\mathsf{e}-5$		$\mathsf{e}-4$	…
$\mathsf{TC}_{\mathsf{chall}}[0]$		✓		×	
$\mathscr{K}$		✓		✓	
$\mathscr{T}$	✓		×		
$\mathsf{TC}^*_{\mathsf{chall}}[0]$		✓		✓	
$\mathscr{K}^*$		✓		✓	
$\mathscr{T}^*$	✓		✓		

Figure 3.6: Example of leakage sets. Marks ✓ and × indicate if an epoch key/token is corrupted. The green mark ✓ indicates an epoch key/token can be inferred from other corrupted keys and tokens.

Trivial Win Conditions. We follow the analysis of trivial win conditions for c-i UE in [2–5], as shown in Fig. 3.7. Our analysis for c-d UE in theorems 3.3.4, 3.3.6 and 3.3.7 shows that it is sufficient to check trivial win conditions on recorded leakages $\mathscr{K}, \mathsf{TC}_{\mathsf{chall}}, \mathscr{T}$, eliminating the need to calculate extended leakages $\mathscr{K}^*, \mathsf{TC}^*_{\mathsf{chall}}, \mathscr{T}^*$ and check trivial win conditions on them.

I. Trivial win by keys and ciphertexts

If the adversary knows the epoch key and a valid challenge-equal ciphertext in the same epoch, it can recover the underlying message by a direct decryption with its corrupted key and therefore win the game. Namely, we should ensure $\mathscr{K}^* \cap \mathsf{TC}^*_{\mathsf{chall}}[0] = \emptyset$. The following lemma shows this condition is equal to $\mathscr{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$ for c-d UE with no-directional key updates.

Lemma 3.3.4. *For c-d UE schemes with no-directional key updates, we have* $\mathscr{K}^* \cap \mathsf{TC}^*_{\mathsf{chall}}[0] = \emptyset \iff \mathscr{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$.

Proof. By the definition of no-directional key updates, we have $\mathscr{K}^* = \mathscr{K}$. In addition, we have $\mathsf{TC}_{\mathsf{chall}}[0] \subseteq \mathsf{TC}^*_{\mathsf{chall}}[0]$. Therefore, we only need to prove $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$ when $\mathscr{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$.

Suppose $\mathsf{TC}_{\mathsf{chall}}[0] = \cup\{\mathsf{e}_{start}, \ldots, \mathsf{e}_{end}\}$. We prove that the adversary cannot learn a challenge-equal ciphertext in epoch e_{end+1} either by querying or inferring. First, the adversary cannot learn a challenge-equal ciphertext in epoch e_{end+1} via querying $\mathcal{O}_{\mathsf{sUpd}}$, since e_{end} is the last epoch in the epoch continuum; otherwise the received

Abilities	Trivial Win Conditions
Keys and ciphertexts	$\mathcal{K}^* \cap \mathsf{TC}^*_{\mathsf{chall}}[0] \neq \emptyset$
Updates	rand-UE : – det-UE : $\bar{e} \in \mathcal{T}^*$ or $\mathcal{O}_{\mathsf{sUpd}}(\bar{e}, (\hat{ct}, ct))$ is queried (line 8-9 of $\mathcal{O}_{\mathsf{Chall}}$)
Decryptions	rand-UE : $e \in \mathsf{TC}^*_{\mathsf{chall}}[0]$ and $(m' = m$ or $m_1)$ (line 3-4 of $\mathcal{O}_{\mathsf{Dec}}$) det-UE : $\mathsf{TC}^*_{\mathsf{chall}}[e, \hat{ct}] = ct$ (line 2 of $\mathcal{O}_{\mathsf{Dec}}$)

Figure 3.7: A summary of trivial win conditions, where $\bar{e}$ is the challenge epoch, $(\hat{ct}, ct)$ is the challenge input ciphertext whose underlying message is m_1, m is the challenge input message, and m' is the returned message of decryption algorithm. Oracles are given in Fig. 3.8.

updated ciphertext will be recorded in the table $\mathsf{TC}_{\mathsf{chall}}$, which conflicts with the condition that e_{end} is the last epoch in the epoch continuum. Alternatively, it can update challenge-equal ciphertext in epoch e_{end} with its inferred token as Eq. (3.6). But from $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$, we know the epoch key k_{end} is unknown to the adversary, which is needed to infer the token in e_{end+1} (see Eq. (3.5)).

The proof is the same for the challenge-equal ciphertext in epoch e_{start}. Therefore, the adversary cannot learn a challenge-equal ciphertext in any epoch outside of the set $\mathsf{TC}_{\mathsf{chall}}[0]$, which implies that $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$. □

Remark 3.3.5. *Lemma 3.3.4 shows that the adversary cannot infer a challenge-equal ciphertext in an epoch that is not recorded in the look-up table, i.e.,* $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$. *But that does not mean all the ciphertexts known to the adversary are stored in the table* $\mathsf{TC}_{\mathsf{chall}}$, *or equally* $\mathsf{TC}_{\mathsf{chall}} = \mathsf{TC}^*_{\mathsf{chall}}$, *which is only true for deterministic UE. For randomized UE schemes, the adversary can create an arbitrary number of valid challenge-equal ciphertexts in any epoch in* $\mathsf{TC}_{\mathsf{chall}}[0]$ *by performing the update with its known ciphertexts and tokens.*

II. Trivial win by updates

For UE schemes with randomized updates, there are no restrictions on the update oracle. However, for UE schemes with deterministic updates, the adversary can learn one of the possible challenge outputs by querying the oracle $\mathcal{O}_{\mathsf{sUpd}}$ on the challenge input $(\hat{ct}, ct)$, or infer the update of $(\hat{ct}, ct)$ if $\bar{e} \in \mathcal{T}^*$, in advance before the challenge phase. In the first case, all known ciphertext leakages before the challenge are recorded by $\mathsf{TC}_{\mathsf{non}}$, so that we can set lines 8-9 in challenge oracle to check for this, as shown in Fig. 3.8. In the second case, if $\bar{e} \in \mathcal{T} (\subseteq \mathcal{T}^*)$, i.e., the token is learned by querying $\mathcal{O}_{\mathsf{sUpd}}$, it goes back to the first case ($\mathcal{O}_{\mathsf{sUpd}}$ also returns the updated

ciphertext, which is recorded in $\mathsf{TC}_{\mathsf{non}}$). If $\bar{e} \in \mathcal{T}^* \setminus \mathcal{T}$, the following lemma shows the impossibility.

Lemma 3.3.6. *For c-d UE schemes with no-directional key updates, if $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$, then the challenge epoch $\bar{e} \notin \mathcal{T}^* \setminus \mathcal{T}$.*

Proof. Note that since the adversary queries the challenge oracle in $\bar{e}$, then $\bar{e} \in \mathsf{TC}_{\mathsf{chall}}[0]$. Due to $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$, we know the epoch key $k_{\bar{e}}$ is unknown to the adversary, which is necessary to infer $\Delta_{\bar{e},\hat{ct}}$ (see Eq. (3.5)). □

$\mathcal{O}_{\mathsf{Enc}}(e, m)$:

1 : $(\hat{ct}, ct) \leftarrow \mathsf{Enc}(k_e, m)$
2 : $\mathsf{TC}_{\mathsf{non}}[e, \hat{ct}] \leftarrow ct$
3 : **return** $(\hat{ct}, ct)$

$\mathcal{O}_{\mathsf{sUpd}}(e, (\hat{ct}, ct))$:

1 : **if** $\mathsf{TC}_{\mathsf{chall}}[e-1, \hat{ct}] = \perp$ and
2 : $\quad \mathsf{TC}_{\mathsf{non}}[e-1, \hat{ct}] = \perp$ **then**
3 : $\quad$ **return** $\perp$
4 : $\Delta_{e,\hat{ct}} \leftarrow \mathsf{TokenGen}(k_{e-1}, k_e, \hat{ct})$
5 : $(\hat{ct}', ct') \leftarrow \mathsf{Update}(\Delta_{e,\hat{ct}}, (\hat{ct}, ct))$
6 : **if** $\mathsf{TC}_{\mathsf{chall}}[e-1, \hat{ct}] \neq \perp$
7 : $\quad \mathsf{TC}_{\mathsf{chall}}[e, \hat{ct}'] \leftarrow ct'$
8 : **else** $\mathsf{TC}_{\mathsf{non}}[e, \hat{ct}'] \leftarrow ct'$
9 : **return** $(\Delta_{e,\hat{ct}}, (\hat{ct}', ct'))$

$\mathcal{O}_{\mathsf{Corr}}(e)$:

1 : $\mathcal{K} = \mathcal{K} \cup \{e\}$
2 : **return** k_e

$\mathcal{O}_{\mathsf{Dec}}(e, (\hat{ct}, ct))$:

1 : m' or $\perp \leftarrow \mathsf{Dec}(k_e, (\hat{ct}, ct))$
2 : **if** (xx = det and $\mathsf{TC}_{\mathsf{chall}}[e, \hat{ct}] = ct$) or
3 : $\quad$ ((xx = rand and $e \in \mathsf{TC}_{\mathsf{chall}}[0]$) and
4 : $\quad$ ($m' = m$ or m_1)) **then**
5 : $\quad$ **return** $\perp$
6 : **return** $\mathsf{Dec}(k_e, (\hat{ct}, ct))$

$\mathcal{O}_{\mathsf{Chall}}(\bar{e}, m, (\hat{ct}, ct))$:

1 : $(\hat{ct}'_0, ct'_0) \leftarrow \mathsf{Enc}(k_{\bar{e}}, m)$
2 : **if** $(\hat{ct}'_0, ct'_0) = \perp$ or $\mathsf{TC}_{\mathsf{non}}[\bar{e}-1, \hat{ct}] \neq ct$
3 : $\quad$ **return** $\perp$
4 : $\Delta_{\bar{e},\hat{ct}} \leftarrow \mathsf{TokenGen}(k_{\bar{e}-1}, k_{\bar{e}}, \hat{ct})$
5 : $(\hat{ct}'_1, ct'_1) \leftarrow \mathsf{Update}(\Delta_{\bar{e},\hat{ct}}, (\hat{ct}, ct))$
6 : **if** $|\hat{ct}'_0| \neq |\hat{ct}'_1|$ or $|ct'_0| \neq |ct'_1|$
7 : $\quad$ **return** $\perp$
8 : **if** (xx = det and $\mathsf{TC}_{\mathsf{non}}[\bar{e}, \hat{ct}'_1] = ct'_1$)
9 : $\quad$ **return** $\perp$
10 : $\mathsf{TC}_{\mathsf{chall}}[\bar{e}, \hat{ct}'_b] \leftarrow ct'_b$
11 : **return** $(\hat{ct}'_b, ct'_b)$

Figure 3.8: An overview of the oracles that the adversary has access to in Definition 3.3.8. In the decryption oracle, m is the challenge input message and m_1 is the underlying message of the challenge input ciphertext.

III. Trivial win by decryptions

Table $\mathsf{TC}^*_{\mathsf{chall}}$ records all the challenge-equal ciphertexts known to the adversary in the game. By remark 3.3.5, we first have the following lemma.

Lemma 3.3.7. *For c-d UE schemes with no-directional key updates, if* $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$, *then* $\mathsf{TC}^*_{\mathsf{chall}} = \mathsf{TC}_{\mathsf{chall}}$ *for deterministic UE, and* $\mathsf{TC}^*_{\mathsf{chall}}[0] = \mathsf{TC}_{\mathsf{chall}}[0]$ *for randomized UE.*

For UE schemes with deterministic ciphertext updates, table $\mathsf{TC}_{\mathsf{chall}}$ records all leaked challenge-equal ciphertexts in the game. The adversary can trivially win the game by querying the decryption oracle on the challenge-equal ciphertexts recorded on the table $\mathsf{TC}_{\mathsf{chall}}$ (line 2 in $\mathcal{O}_{\mathsf{Dec}}$, Fig. 3.8).

For UE schemes with randomized ciphertext updates, the epoch set $\mathsf{TC}_{\mathsf{chall}}[0]$ records all the epochs in which the adversary can generate a valid challenge-equal ciphertext. The adversary can trivially win the game if the returned message of the decryption oracle in epochs in $\mathsf{TC}_{\mathsf{chall}}[0]$ is the challenge message or the plaintext of the challenge input ciphertext (lines 3-4).

In summary, the above analysis shows trivial win conditions for c-d UE can be checked immediately based on the recorded leakages during the confidentiality game, without the need for extra calculations and further checks of the extended leaked sets of keys, tokens and ciphertext as in previous work for c-i UE in [2, 3, 5]. After all the queries, if $\perp$ is not returned, only one condition remains to be checked: $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$. This advantage is due to both the no-directional key update setting and the proper ways of recording leakage information via look-up tables. Finally, we introduce the definition of xxIND-UE-atk.

Definition 3.3.8 (xxIND-UE-atk)**.** *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update})$ *be a ciphertext-dependent updatable encryption scheme with no-directional key updates. For an adversary* $\mathcal{A}$ *and* $\mathsf{b} \in \{0,1\}$, *we define the confidentiality experiment* $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ *in Fig. 3.9 for* $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ *and* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$.

We say UE *meets the* xxIND-UE-atk *confidentiality if there is a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{xxIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$, *where*

$$\mathsf{Adv}^{\mathsf{xxIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}1}}_{\mathsf{UE},\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}1}}_{\mathsf{UE},\mathcal{A}} = 0\right]\right|.$$

Future Extensions. In our security model, the adversary is only allowed to query the update oracle with "correctly" generated ciphertexts throughout the experiment. An interesting future work is to investigate security notions that capture both adaptive security and protection against malicious update.

3.3.4. FIREWALL TECHNIQUES

Firewall Technique. In c-i UE, the firewall technique was developed in [4, 5] to facilitate the security proof by separating epochs into different regions. Inside an insulated region, the simulation in the proof should appropriately respond to the queries of the adversary, since it corrupts all tokens within this region. While outside, the simulation can generate tokens and epoch keys freely.

In c-d UE, we similarly define the insulated region, inside which all tokens related to challenge-equal ciphertexts (called challenge-equal tokens) are corrupted but no epoch key is corrupted.

$\mathsf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathscr{A}}$:

1 : $(\mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct})) \leftarrow \mathscr{A}^{\mathscr{O}_1}(1^\lambda)$ // setup phase

2 : $\mathscr{A}$ queries $\mathscr{O}_{\mathsf{chall}}$ on $(\mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ // challenge phase

3 : $\mathsf{b}' \leftarrow \mathscr{A}^{\mathscr{O}_2}(1^\lambda)$ // response phase

4 : **if** $(\mathscr{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] \neq \emptyset)$ **then**

5 : $\quad \mathsf{b}' \xleftarrow{\$} \{0,1\}$

6 : **return** b'

Figure 3.9: The confidentiality game $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathscr{A}}$ where $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ indicates the type of UE scheme (deterministic or randomized) and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$ indicates the type of attack model. In the game, the adversary is given access to a set of oracles, denoted by $\mathscr{O}_1$ and $\mathscr{O}_2$ which are shown in Fig. 3.8 and Fig. 3.10. During the setup phase, the adversary generates a challenge plaintext and a challenge ciphertext using the oracles in $\mathscr{O}_1$, and submits them to the challenger in the challenge phase. The adversary continues to query the oracles in $\mathscr{O}_2$ and eventually provides a guess bit. The only condition for the adversary to lose the game is $\mathscr{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] \neq \emptyset$.

atk	$\mathscr{O}_1$	$\mathscr{O}_2$
CPA	$\mathscr{O}_{\mathsf{Enc}}, \mathscr{O}_{\mathsf{sUpd}}, \mathscr{O}_{\mathsf{Corr}}$	$\mathscr{O}_{\mathsf{Enc}}, \mathscr{O}_{\mathsf{sUpd}}, \mathscr{O}_{\mathsf{Corr}}$
CCA-1	$\mathscr{O}_{\mathsf{Enc}}, \mathscr{O}_{\mathsf{sUpd}}, \boxed{\mathscr{O}_{\mathsf{Dec}}}, \mathscr{O}_{\mathsf{Corr}}$	$\mathscr{O}_{\mathsf{Enc}}, \mathscr{O}_{\mathsf{sUpd}}, \mathscr{O}_{\mathsf{Corr}}$
CCA	$\mathscr{O}_{\mathsf{Enc}}, \mathscr{O}_{\mathsf{sUpd}}, \boxed{\mathscr{O}_{\mathsf{Dec}}}, \mathscr{O}_{\mathsf{Corr}}$	$\mathscr{O}_{\mathsf{Enc}}, \mathscr{O}_{\mathsf{sUpd}}, \boxed{\mathscr{O}_{\mathsf{Dec}}}, \mathscr{O}_{\mathsf{Corr}}$

Figure 3.10: Oracles that the adversary has access to before and after the challenge phase in the confidentiality game for different attacks. It can corrupt keys at any time during the game in all attacks via querying $\mathsf{O}_{\mathsf{Corr}}$, but is not allowed to query the decryption oracle in the CPA attack, limited to query the decryption oracle before the challenge in the CCA-1 attack, and free to query the decryption oracle in the CCA attack.

Definition 3.3.9 (Firewall)**.** *In ciphertext-dependent UE schemes, an insulated region with firewalls* fwl *and* fwr*, denoted by* $\mathscr{FW}$*, is a consecutive sequence of epochs* $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ *for which:*

- *no key in the sequence of epochs* $\{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$ *is corrupted;*
- *no challenge-equal tokens in epochs* fwl *and* $\mathsf{fwr}+1$ *is corrupted;*
- *all challenge-equal tokens in epochs* $\{\mathsf{fwl}+1, \ldots, \mathsf{fwr}\}$ *are corrupted.*

Suppose an xxIND-UE-atk adversary $\mathscr{A}$ queries the challenge oracle in the epoch $\bar{e}$ and does not trigger trivial win conditions in the game, and $\mathsf{TC}_{\mathsf{chall}}[0] = \cup\{e_{start}, \ldots, e_{end}\}$. The proof of Lemma 3.3.4 shows $\mathscr{A}$ cannot update a ciphertext from the epoch e_{end} to the start epoch e'_{start} of the next continuum. Thus, we have $\mathsf{TC}_{\mathsf{chall}}[0] = \{e_{start}, \ldots, e_{end}\}$, meaning that the epoch set in which $\mathscr{A}$ knows a challenge-equal ciphertext is only a consecutive continuum starting from the challenge epoch ($e_{start} = \bar{e}$), and ending in the epoch e_{end}, the last epoch that the adversary queries the update oracle $\mathscr{O}_{\mathsf{sUpd}}$ on the challenge-equal ciphertext. The epoch keys and tokens in the epoch in $\mathsf{TC}_{\mathsf{chall}}[0]$ have the following properties.

- $\mathscr{A}$ does not know the challenge-equal token in epochs e_{start} and $e_{end}+1$, following from the proof of Lemma 3.3.4;
- $\mathscr{A}$ knows all challenge-equal tokens in epochs in $\{e_{start}+1, \ldots, e_{end}\}$, obtained when $\mathscr{A}$ queries the updates of challenge-equal ciphertexts via $\mathscr{O}_{\mathsf{sUpd}}$;
- $\mathscr{A}$ does not know any key in epochs in $\{e_{start}, \ldots, e_{end}\}$, as $\mathscr{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$;

We thus have the Lemma 3.3.10, following from the discussion above, and Lemma 3.3.11, as a corollary of Lemma 3.3.10, both of which provide important tools in the confidentiality proof for c-d UE.

Lemma 3.3.10. *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Update}, \mathsf{Decrypt})$ *be a c-d UE scheme with no-directional key updates, and* $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ *and* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$. *For an* xxIND-UE-atk *adversary* $\mathscr{A}$ *against* UE, *the set of epochs in which* $\mathscr{A}$ *knows a challenge-equal ciphertext is an insulated region (Def. 3.3.9), starting from the challenge epoch and ending at the last epoch in which the adversary queries the* $\mathscr{O}_{\mathsf{sUpd}}$.

Lemma 3.3.11. *For a c-d UE with no-directonal key updates, if the* xxIND-UE-atk *adversary knows a challenge-equal ciphertext in epoch* e, *then* e *must be in an insulated region.*

3.4. A CCA-1 Secure PKE Scheme

In this section, we propose a new PKE scheme called TDP, which is based on the lattice trapdoor techniques. We will use this scheme in Sect. 3.5 as the underlying encryption scheme to build our UE scheme.

3.4.1. A New PKE Scheme

Our overall idea is to construct a 1×3 block matrix $\mathbf{A}_\mu$ in the encryption algorithm, with the secret key serving as the trapdoor for the first two blocks of $\mathbf{A}_\mu$ to ensure the correctness of decryption.

We introduce some parameters involved in the construction in Fig. 3.11, where we use standard asymptotic notations of O, Ω, ω. Let λ be the security parameter, $\omega(\sqrt{\log n})$ is a fixed function that grows asymptotically faster than $\sqrt{\log n}$, and $\Lambda(\mathbf{G}^t)$ is the lattice generated by $\mathbf{G}^t$.

Notations	Functionalities
$\mathbf{G} = \mathbb{Z}_q^{n\times nk}$ (Sect. 3.2.2) $k = \lceil \log_2 q \rceil = O(\log n)$, $q = \mathsf{poly}(\lambda)$	Make oracles $\mathsf{Invert}^{\mathcal{O}}$ and $\mathsf{SampleD}^{\mathcal{O}}$ efficient for the random matrix with a $\mathbf{G}$-trapdoor
$\bar{m} = O(nk)$, $\mathscr{D} = D_{\mathbb{Z}^{\bar{m}\times nk},\omega(\sqrt{\log n})}$	Ensure $(\mathbf{A}, \mathbf{AR})$ is $\mathsf{negl}(\lambda)$-far from uniform for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}$ and $\mathbf{R} \leftarrow \mathscr{D}$, due to leftover hash lemma
$\mathsf{encode}: \{0,1\}^{nk} \to \Lambda(\mathbf{G}^t)$ by $\mathsf{encode}(\mathbf{m}) = \mathbf{Bm} \in \mathbb{Z}^{nk}$, and $\mathbf{B}$ is any basis of $\Lambda(\mathbf{G}^t)$	Ensure an efficient decoding for decryption
LWE error rate α such that: $1/\alpha = 4 \cdot O(nk) \cdot \omega(\sqrt{\log n})$	Control the magnitude of error in ciphertext

Figure 3.11: A summary of notations used in PKE construction and their functionalities.

The PKE scheme TDP is described as follows. On a first reading, we suggest readers to neglect the error parameter settings that are used to control the error bound within the decryption capability, in order to have a simpler view at a high level.

- $\mathsf{TDP.KG}(1^\lambda)$: choose $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}, \mathbf{R}_1, \mathbf{R}_2 \xleftarrow{\$} \mathscr{D}$ and let $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \mathbf{A}_2] = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 \mid -\mathbf{A}_0\mathbf{R}_2] \in \mathbb{Z}_q^{n\times m}$ where $m = \bar{m} + 2nk$. The public key is $\mathsf{pk} = \mathbf{A}$ and the secret key is $\mathsf{sk} = \mathbf{R}_1$.

- $\mathsf{TDP.Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m} \in \{0,1\}^{nk})$: choose an invertible matrix $\mathbf{H}_\mu \in \mathbb{Z}_q^{n\times n}$, and let $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. Choose a random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and an error vector $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \in D_{\mathbb{Z}^{\bar{m}},\alpha q} \times D_{\mathbb{Z}^{nk},d} \times D_{\mathbb{Z}^{nk},d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m}\cdot(\alpha q)^2)\cdot\omega(\sqrt{\log n})^2$. Let

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \tag{3.7}$$

  where the first $\mathbf{0}$ has dimension $\bar{m}$ and the second has dimension nk. Output the ciphertext $c = (\mathbf{H}_\mu, \mathbf{b})$. Notice that $\mathbf{R}_1$ is a trapdoor for $[\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G}]$.

- $\mathsf{TDP.Dec}(\mathsf{sk} = \mathbf{R}_1, c = (\mathbf{H}_\mu, \mathbf{b}))$: let $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. The decryption first recovers $\mathbf{s}$ from the first two blocks via the invert algorithm and then the

message $\mathbf{m}$ from the third block by decoding (when $\mathbf{s}$ is known):

$$(\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2)^t = \mathbf{s}^t[\mathbf{A}_0 \mid \mathbf{A}_1+\mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2] + (\mathbf{e}_0,\mathbf{e}_1,\mathbf{e}_2)^t + (\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}))^t \mod q.$$

1. If c or $\mathbf{b}$ does not parse, or $\mathbf{H}_\mu = \mathbf{0}$, output $\perp$. Otherwise parse $\mathbf{b}^t = (\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2)^t$.
2. **Recover s.** Call $\mathsf{Invert}^{\mathscr{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid \mathbf{A}_1+\mathbf{H}_\mu\mathbf{G}], [\mathbf{b}_0,\mathbf{b}_1], \mathbf{H}_\mu)$ by Lemma 3.2.4, which returns $\mathbf{s}$ and $(\mathbf{e}_0,\mathbf{e}_1)$ such that
$$(\mathbf{b}_0,\mathbf{b}_1)^t = \mathbf{s}^t[\mathbf{A}_0 \mid \mathbf{A}_1+\mathbf{H}_\mu\mathbf{G}] + (\mathbf{e}_0,\mathbf{e}_1)^t \mod q.$$
If $\mathsf{Invert}^{\mathscr{O}}$ fails, output $\perp$. Invert $\mathbf{b}_2^t - \mathbf{s}^t\mathbf{A}_2$ again and find the unique solution $\mathbf{u},\mathbf{e}_2$ to the equation
$$\mathbf{b}_2^t - \mathbf{s}^t\mathbf{A}_2 = \mathbf{u}^t\mathbf{G} + \mathbf{e}_2^t \mod q,$$
3. If $\|\mathbf{e}_0\| \geq \alpha q\sqrt{\bar{m}}$ or $\|\mathbf{e}_j\| \geq \alpha q\sqrt{2\bar{m}nk}\cdot\omega(\sqrt{\log n})$ for $j=1,2$, output $\perp$ (Lemma 3.2.7).
4. **Recover the plaintext.** Output the following result
$$\mathsf{encode}^{-1}\left(\mathbf{b}_2^t - \mathbf{s}^t\mathbf{A}_2 - \mathbf{e}_2^t\right) \in \mathbb{Z}_2^{nk},$$
if it exists, otherwise output $\perp$.

3.4.2. CORRECTNESS AND SECURITY

We provide a full proof of the correctness (Lemma 3.4.1) and security (Lemma 3.4.2) of the updatable encryption scheme TDUE in Sect. 3.5, which is based on TDP as a subcase of TDUE.

Lemma 3.4.1. *Our* TDP *decrypts correctly except with* $2^{-\Omega(n)}$ *failure probability.*

Proof. The proof is the same as that of Lemma 3.5.1, except the bound for the error vectors. The secret key $\mathbf{R}$ serves as the trapdoor for the first two blocks of $\mathbf{A}_\mu$, which ensures the proper recovery of $\mathbf{s}$ in Step 2 as long as the error bound is within the capability of Invert. That is $\left\|\mathbf{e}^t\binom{\mathbf{R}}{\mathbf{I}}\right\| \leq q/4$ by Lemma 3.2.4. By Lemma 3.2.9, we have $s_1(\mathbf{R}) = \omega(\sqrt{\log n})\cdot O(\sqrt{nk})$. By Lemma 3.2.7, we have $\|\mathbf{e}_0\| \leq \alpha q\sqrt{\bar{m}}$ and $\|\mathbf{e}_i\| \leq \alpha q\sqrt{2\bar{m}nk}\cdot\omega(\sqrt{\log n})$ for $j=1,2$, except with negligible probability $2^{-\Omega(n)}$, where $\bar{m} = O(nk)$. Therefore,

$$\begin{aligned}\left\|\begin{pmatrix}\mathbf{e}_0\\ \mathbf{e}_1\end{pmatrix}^t\begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}\right\|_\infty &\leq \left\|\begin{pmatrix}\mathbf{e}_0\\ \mathbf{e}_1\end{pmatrix}^t\begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}\right\| \\ &\leq \|\mathbf{e}_0^t\mathbf{R}\| + \|\mathbf{e}_1\| \\ &\leq \alpha q\cdot O(nk)\cdot\omega(\sqrt{\log n})\end{aligned}$$

which is further smaller than $q/4$ since $1/\alpha = 4\cdot O(nk)\cdot\omega(\sqrt{\log n})$, and $\|\mathbf{e}_2\|_\infty \leq q/4$ for the same reason, which ensures the correct recovery of $\mathbf{s}$, $\mathbf{u}$ and $\mathbf{m}$. □

Lemma 3.4.2. *Our PKE scheme* TDP *is* CCA-1 *secure if the* LWE *problem is hard.*

Proof. We provide a detailed CCA-1 proof for our UE scheme in Theorem 3.5.2. Note that, if the adversary is disallowed to query the token generation and update algorithm, the CCA-1 game for UE is exactly the standard CCA-1 game for the underlying PKE. Therefore, CCA-1 security of TDP follows from Theorem 3.5.2. □

3.5. A CCA-1 SECURE UPDATABLE ENCRYPTION SCHEME

Based on our PKE scheme in Sect. 3.4, we construct a new UE scheme, which is IND-UE-CCA-1 secure under the assumption of the LWE hardness.

3.5.1. CONSTRUCTION

Our UE scheme uses the same encryption and decryption algorithm in TDP, i.e., the ciphertext of a plaintext $\mathbf{m}$ is of the form $(\hat{\mathsf{ct}}, \mathsf{ct}) = (\mathbf{H}_\mu, \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m})^t)$. To update a ciphertext, at a high level, the update algorithm first generates a key-switching matrix $\mathbf{M}$ with the last row block matrix $[\mathbf{0} \quad \mathbf{0} \quad \mathbf{I}]$, such that $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ for the aimed $\mathbf{A}'_\mu$ in the new ciphertext. This step is feasible since the secret key is the trapdoor for the first two blocks of $\mathbf{A}_\mu$, ensuring an efficient preimage sampling algorithm (Lemma 3.2.5). To increase the randomness of $\mathbf{s}$, then we add a fresh encryption of message $\mathbf{0}$ to the ciphertext. Fig. 3.12 shows an overview of the ciphertext update.

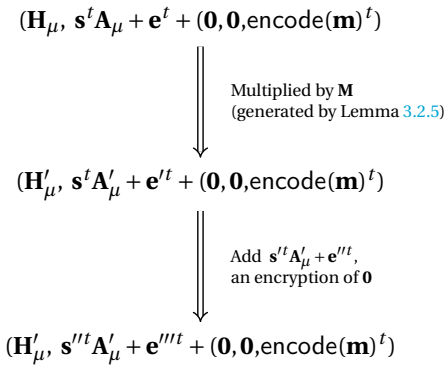


Figure 3.12: An overview of ciphertext update in our UE construction. The first step mainly updates $\mathbf{A}_\mu$ to $\mathbf{A}'_\mu$, and the second step refreshes the randomness $\mathbf{s}$.

We use the same parameters as in Sect. 3.4.1 except the following. We also suggest readers on the first reading to neglect the parameter setting for error items which are used to control the updated error bound.

- $1/\alpha = 4l \cdot \omega(\sqrt{\log n})^{2l+2} O(\sqrt{nk})^{3l+3}$ where l is the maximal number of update that the scheme can support.

- $\tau = \sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{s_1(\Sigma_{\mathbf{G}}) + 1} \cdot \omega(\sqrt{\log n})$ is smallest Gaussian parameter for the discrete Gaussian distribution from which the sampling algorithm $\mathsf{SampleD}^{\mathcal{O}}$ can sample vectors, where $s_1(\Sigma_{\mathbf{G}}) = 5$ by Theorem 3.2.4.

The UE scheme TDUE is described as follows.

- $\mathsf{TDUE.KG}(1^\lambda)$: output $\mathsf{TDP.KG}(1^\lambda)$.

- $\mathsf{TDUE.Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m} \in \{0,1\}^{nk})$: output $\mathsf{TDP.Enc}(\mathbf{A}, \mathbf{m})$.

- $\mathsf{TDUE.Dec}(\mathsf{sk} = \mathbf{R}_1, c = (\mathbf{H}_\mu, \mathbf{b}))$: output $\mathsf{TDP.Dec}(\mathbf{R}_1, (\mathbf{H}_\mu, \mathbf{b}))$.

- $\mathsf{TDUE.TokenGen}(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}', \mathbf{H}_\mu)$: parse $\mathsf{pk} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \mathbf{A}_2] = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 \mid -\mathbf{A}_0\mathbf{R}_2]$, $\mathsf{sk} = \mathbf{R}_1$, and $\mathsf{pk}' = [\mathbf{A}'_0 \mid \mathbf{A}'_1 \mid \mathbf{A}'_2]$.
  1. Generate a random invertible matrix $\mathbf{H}'_\mu$ and let $\mathbf{A}'_\mu = [\mathbf{A}'_0 \mid \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G} \mid \mathbf{A}'_2]$. We first generate a transition matrix $\mathbf{M}$ for which $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ in the following steps 2, 3, 4, and then compute the encryption of the message $\mathbf{0}$ under $\mathbf{A}'_\mu$ in step 5.
  2. Call $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_\mu, \mathbf{A}'_0, \tau)$ (Lemma 3.2.5 and $\mathbf{R}_1$ is a trapdoor for $[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}]$), which returns an $(\bar{m} + nk) \times \bar{m}$ matrix, parsed as $\mathbf{X}_{00} \in \mathbb{Z}^{\bar{m} \times \bar{m}}$ and $\mathbf{X}_{10} \in \mathbb{Z}^{nk \times \bar{m}}$ with Gaussian entries of parameter τ, satisfying
  $$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} = \mathbf{A}'_0. \tag{3.8}$$
  3. Call $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_\mu, \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G}, \tau\sqrt{\bar{m}/2})$, which returns $\mathbf{X}_{01} \in \mathbb{Z}_q^{\bar{m} \times nk}$ and $\mathbf{X}_{11} \in \mathbb{Z}_q^{nk \times nk}$ with Gaussian entries of parameter $\tau\sqrt{\bar{m}/2}$ such that
  $$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{X}_{01} \\ \mathbf{X}_{11} \end{bmatrix} = \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G}. \tag{3.9}$$
  4. Continue calling the sample oracle $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_1, \mathbf{A}'_2 - \mathbf{A}_2, \tau\sqrt{\bar{m}/2})$ and obtain $\mathbf{X}_{02} \in \mathbb{Z}_q^{\bar{m} \times nk}$ and $\mathbf{X}_{12} \in \mathbb{Z}_q^{nk \times nk}$ with Gaussian entries of parameter $\tau\sqrt{\bar{m}/2}$ such that
  $$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = \mathbf{A}'_2 - \mathbf{A}_2. \tag{3.10}$$

  Let $\mathbf{M}$ be the key-switching matrix as follows:
  $$\mathbf{M} = \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{3.11}$$

Note that $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu \mathbf{G} \mid \mathbf{A}_2]$. Then we have $\mathbf{A}_\mu \mathbf{M} = \mathbf{A}'_\mu$ from Equations (3.8) to (3.10).

5. Let $\mathbf{b}_0$ be the ciphertext of message $\mathbf{m} = \mathbf{0}$ under the public key pk' with the invertible matrix $\mathbf{H}'_\mu$ generated in step 1. That is,

$$\mathbf{b}_0^t = (\mathbf{s}')^t \mathbf{A}'_\mu + (\mathbf{e}')^t \mod q.$$

6. Output the update token $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$.

- $\mathsf{TDUE.Update}(\Delta, c = (\mathbf{H}_\mu, \mathbf{b}))$: parse $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$ and compute

$$(\mathbf{b}')^t = \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \mod q,$$

and output $c' = (\mathbf{H}'_\mu, \mathbf{b}')$.

No-directional Key Updates. TDUE has no-directional key updates since one can only learn from the update token about the value of $\mathbf{A}'_\mu$ (or $\mathbf{A}_\mu$) through $\mathbf{A}_\mu \mathbf{M} = \mathbf{A}'_\mu$ even if sk (or sk', resp.) is corrupted, whereas $\mathbf{A}'_\mu$ and $\mathbf{A}_\mu$ are random due to the leftover hash lemma and the distribution of the secret key. Therefore, the adversary cannot infer any information about the secret key from the update tokens.

3.5.2. CORRECTNESS

We prove that the decryption algorithm in our scheme can perform correctly with overwhelming probability. Note that the second component in the ciphertext generated by the update algorithm (updated ciphertext) is as follows:

$$\begin{aligned}(\mathbf{b}')^t &= \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \\ &= \left[\mathbf{s}^t \mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t\right] \mathbf{M} + (\mathbf{s}')^t \mathbf{A}'_\mu + (\mathbf{e}')^t \\ &= (\mathbf{s} + \mathbf{s}')^t \mathbf{A}'_\mu + \left(\mathbf{e}^t \mathbf{M} + (\mathbf{e}')^t\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q. \end{aligned} \tag{3.12}$$

The third equation holds because $\mathbf{A}_\mu \mathbf{M} = \mathbf{A}'_\mu$ and the last nk rows in $\mathbf{M}$ is $[\mathbf{0}\ \mathbf{0}\ \mathbf{I}]$. Therefore the item $(\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t$ stays the same when multiplied by $\mathbf{M}$. Then the updated ciphertext shares the same form with the fresh ciphertext (generated by the encryption algorithm), except that the update algorithm enlarges the error terms by $\mathbf{e}^t \mathbf{M} + (\mathbf{e}')^t$, which may cause the failure in the invert algorithm $\mathsf{Invert}^{\mathcal{O}}$ and further influence the correctness of the decryption algorithm. In the following, we show that the decryption algorithm can tolerate the accumulated errors in the updated ciphertexts by choosing an appropriate value for the parameter α.

Lemma 3.5.1. *Our UE scheme* TDUE *decrypts correctly except with* $2^{-\Omega(n)}$ *failure probability.*

Proof. Since the decryption on the fresh ciphertext (from Enc) is a subcase of that on the updated ciphertext (from Update), we choose to prove that the decryption

algorithm can output a correct plaintext after performing l updates from epoch 0, where l is the maximum update number.

Let $(\mathsf{pk}_\mathsf{e}, \mathsf{sk}_\mathsf{e} = \mathbf{R}_\mathsf{e})_{0\le \mathsf{e}\le l} \leftarrow \mathsf{KG}(1^n)$ be the public and secret key in epoch e. For a random plaintext $\mathbf{m} \in \{0,1\}^{nk}$, let c_e be the ciphertext of $\mathbf{m}$ in epoch e, which is updated from $c_0 = \mathsf{Enc}(\mathbf{m}) = (\mathbf{H}_{\mu,0}, \mathbf{s}_0^t\mathbf{A}_{\mu,0} + \mathbf{e}_0^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t)$. For $1 \le i \le l$, let the token in epoch i be $\Delta_i = (\mathbf{M}_i, \mathbf{b}_{0,i}, \mathbf{H}_{\mu,i})$, where $\mathbf{b}_{0,i}$ is the fresh ciphertext of message $\mathbf{0}$ in epoch i, i.e., $\mathbf{b}_{0,i}^t = \mathbf{s}_i^t\mathbf{A}_{\mu,i} + \mathbf{e}_i^t$ in which $\mathbf{A}_{\mu,i} = [\mathbf{A}_{0,i} \mid \mathbf{A}_{1,i} + \mathbf{H}_{\mu,i}\mathbf{G} \mid \mathbf{A}_{2,i}]$. Iteratively by Eq. (3.12), we know the updated ciphertext of $\mathbf{m}$ in epoch l is $c_l = (\mathbf{H}_{\mu,l}, \mathbf{b}_l)$ where

$$\mathbf{b}_l^t = \left(\sum_{i=0}^{l} \mathbf{s}_i\right)^t \mathbf{A}_{\mu,l} + \sum_{i=0}^{l}\left(\mathbf{e}_i^t \prod_{j=i+1}^{l} \mathbf{M}_j\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t.$$

Let $\sum_{i=0}^{l}(\mathbf{e}_i^t \prod_{j=i+1}^{l} \mathbf{M}_j) = (\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)}, \mathbf{e}_2^{(l)})^t = (\mathbf{e}^{(l)})^t$. In the end of the proof, we provide the upper bound for the error $\mathbf{e}^{(l)}$ such that

$$\left\| (\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)}, \mathbf{e}_2^{(l)})^t \cdot \begin{bmatrix} \mathbf{R}_l \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \right\|_\infty < q/4 \quad \text{and} \quad \left\| \mathbf{e}_2^{(l)} \right\|_\infty < q/4, \tag{3.13}$$

except with probability $2^{-\Omega(n)}$ via the appropriate parameter selection for the scheme. Let $\mathbf{b}_l^t = (\mathbf{b}_0^{(l)}, \mathbf{b}_1^{(l)}, \mathbf{b}_2^{(l)})^t$. Then by Lemma 3.2.4, the call to $\mathsf{Invert}^{\mathcal{O}}$ made by $\mathsf{Dec}(\mathsf{sk}_l, (\mathbf{H}_{\mu,l}, \mathbf{b}_l))$ returns $\mathbf{s}$ $(=\sum_{i=0}^{l}\mathbf{s}_i)$ and $(\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)})$ correctly, for which

$$(\mathbf{b}_0^{(l)}, \mathbf{b}_1^{(l)})^t = \mathbf{s}^t[\mathbf{A}_{0,l} \mid \mathbf{A}_{1,l} + \mathbf{H}_{\mu,l}\mathbf{G}] + (\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)})^t \mod q.$$

It follows that

$$(\mathbf{b}_2^{(l)})^t - \mathbf{s}^t\mathbf{A}_{2,l} = (\mathbf{e}_2^{(l)})^t + \mathsf{encode}(\mathbf{m})^t, \tag{3.14}$$

where $\left\|\mathbf{e}_2^{(l)}\right\| < q/4$ by Inequality (3.13) and $\mathsf{encode}(\mathbf{m})^t = \mathbf{u}^t\mathbf{G}$ for some $\mathbf{u} \in \mathbb{Z}_q^{nk}$ by the definition of encode. Inverting $(\mathbf{b}_2^{(l)})^t - \mathbf{s}^t\mathbf{A}_{2,l}$, we can find the unique solution $\mathbf{e}_2^{(l)}$ and $\mathbf{u}$ to Eq. (3.14). Finally, we have

$$\mathsf{encode}^{-1}\left((\mathbf{u}^t\mathbf{G})^t\right) = \mathsf{encode}^{-1}(\mathsf{encode}(\mathbf{m})) = \mathbf{m}.$$

Therefore, the decryption algorithm Dec outputs $\mathbf{m}$ as desired.

Proof of Inequality (3.13). We start from the error $\mathbf{e}_0$ generated in the fresh encryption in epoch 0 and estimate the bound on $\mathbf{e}_0^t \cdot \prod_{j=1}^{l}\mathbf{M}_j$. Errors generated in the update algorithm in later epochs have the same distribution as $\mathbf{e}_0$, but are multiplied fewer times than $\mathbf{e}_0$ by the transition matrix $\{\mathbf{M}_j\}$.

Step 1. Let $\mathbf{e}_0^t = (\mathbf{e}_{0,0}, \mathbf{e}_{1,0}, \mathbf{e}_{2,0})^t$ and $\mathbf{M}^{(s)}$ be the s products of $\{\mathbf{M}_j\}_{j=1}^{s}$, denoted as follows:

$$\mathbf{M}^{(s)} = \prod_{t=1}^{s}\mathbf{M}_t = \begin{bmatrix} \mathbf{X}_{00}^{(s)} & \mathbf{X}_{01}^{(s)} & \mathbf{X}_{02}^{(s)} \\ \mathbf{X}_{10}^{(s)} & \mathbf{X}_{11}^{(s)} & \mathbf{X}_{12}^{(s)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \text{ and } \mathbf{M}_j = \begin{bmatrix} \mathbf{X}_{00,j} & \mathbf{X}_{01,j} & \mathbf{X}_{02,j} \\ \mathbf{X}_{10,j} & \mathbf{X}_{11,j} & \mathbf{X}_{12,j} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix},$$

for $s \in \{1, \ldots, l\}$.

We first estimate the bound on the maximal singular values of $\mathbf{M}_j$ and $\mathbf{M}^{(s)}$. For any $j \in \{1,\ldots,l\}$, we have $s_1(\mathrm{X}_{kt,j}) \le \tau \cdot O(\sqrt{nk})$ for $kt \in \{00,10\}$, and $s_1(\mathrm{X}_{kt,j}) \le \tau\sqrt{\bar{m}/2} \cdot O(\sqrt{nk})$ for $kt \in \{01,11,02,12\}$ by Lemma 3.2.9. Let $\mu = \sqrt{\bar{m}/2}$. For $s = 2$, since $\mathbf{X}_{00}^{(2)} = \mathbf{X}_{00,1} \cdot \mathbf{X}_{00,2} + \mathbf{X}_{01,1} \cdot \mathbf{X}_{10,2}$, we have

$$\begin{aligned} s_1(\mathbf{X}_{00}^{(2)}) &\le s_1(\mathbf{X}_{00,1}) \cdot s_1(\mathbf{X}_{00,2}) + s_1(\mathbf{X}_{01,1}) \cdot s_1(\mathbf{X}_{10,2}) \\ &\le \tau^2(1+\mu) \cdot O(\sqrt{nk})^2, \end{aligned}$$

by Lemma 3.2.10. Similarly, we give the bound on the maximal singular value of other block matrices in $\mathbf{M}^{(2)}$ (except the last nk rows, which equal $[\mathbf{0}, \mathbf{0}, \mathbf{I}]$ for all $\mathbf{M}$) as follows (element-wise comparison):

$$\begin{aligned} &\begin{bmatrix} s_1(\mathbf{X}_{00}^{(2)}) & s_1(\mathbf{X}_{01}^{(2)}) & s_1(\mathbf{X}_{02}^{(2)}) \\ s_1(\mathbf{X}_{10}^{(2)}) & s_1(\mathbf{X}_{11}^{(2)}) & s_1(\mathbf{X}_{12}^{(2)}) \end{bmatrix} \\ &\quad \le \begin{bmatrix} \tau^2(1+\mu) & \tau^2\mu(1+\mu) & \tau^2\mu[(1+\mu)+1] \\ \tau^2(1+\mu) & \tau^2\mu(1+\mu) & \tau^2\mu[(1+\mu)+1] \end{bmatrix} \cdot O(\sqrt{nk})^2. \end{aligned}$$

Iteratively, we have the bound on the maximal singular value of each block matrix in $\mathbf{M}^{(l)}$, which is

$$\begin{aligned} &\begin{bmatrix} s_1(\mathbf{X}_{00}^{(l)}) & s_1(\mathbf{X}_{01}^{(l)}) & s_1(\mathbf{X}_{02}^{(l)}) \\ s_1(\mathbf{X}_{10}^{(l)}) & s_1(\mathbf{X}_{11}^{(l)}) & s_1(\mathbf{X}_{12}^{(l)}) \end{bmatrix} \\ &\quad \le \begin{bmatrix} \tau^l(1+\mu)^{l-1} & \tau^l\mu(1+\mu)^{l-1} & \tau^l\mu\sum_{k=0}^{l-1}(1+\mu)^k \\ \tau^l(1+\mu)^{l-1} & \tau^l\mu(1+\mu)^{l-1} & \tau^l\mu\sum_{k=0}^{l-1}(1+\mu)^k \end{bmatrix} \cdot O(\sqrt{nk})^l. \end{aligned} \tag{3.15}$$

Now we can estimate the bound on the updated $\mathbf{e}_0$ in the last epoch. Notice that

$$\begin{aligned} \mathbf{e}_0^t \cdot \mathbf{M}^{(l)} \cdot \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} &= (\mathbf{e}_{0,0}, \mathbf{e}_{1,0}, \mathbf{e}_{2,0})^t \begin{bmatrix} \mathbf{X}_{00}^{(l)} & \mathbf{X}_{01}^{(l)} & \mathbf{X}_{02}^{(l)} \\ \mathbf{X}_{10}^{(l)} & \mathbf{X}_{11}^{(l)} & \mathbf{X}_{12}^{(l)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \\ &= (\mathbf{e}_{0,0}^t\mathbf{X}_{00}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{10}^{(l)})\mathbf{R}_{1,l} + \mathbf{e}_{0,0}^t\mathbf{X}_{01}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{11}^{(l)}. \end{aligned} \tag{3.16}$$

By Lemma 3.2.7, we have $\|\mathbf{e}_{0,0}\| < \alpha q\sqrt{\bar{m}}$ and $\|\mathbf{e}_i\| < \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $\mathbf{e}_i \in \{\mathbf{e}_{1,0}, \mathbf{e}_{2,0}\}$, except with probability $2^{-\Omega(n)}$. Combining this conclusion with Eq. (3.15), we obtain

$$\begin{cases} \left\|\mathbf{e}_{0,0}^t\mathbf{X}_{00}^{(l)}\mathbf{R}_{1,l}\right\| \le \|\mathbf{e}_{0,0}\| \cdot s_1(\mathbf{X}_{00}^{(l)}) \cdot s_1(\mathbf{R}_{1,l}) < \alpha q\sqrt{\bar{m}} \cdot \tau^l(1+\mu)^{l-1}O(\sqrt{nk})^l \cdot \sigma, \\ \left\|\mathbf{e}_{1,0}^t\mathbf{X}_{10}^{(l)}\mathbf{R}_{1,l}\right\| < (\alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})) \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l \cdot \sigma, \\ \left\|\mathbf{e}_{0,0}^t\mathbf{X}_{01}^{(l)}\right\| < \alpha q\sqrt{\bar{m}} \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l, \\ \left\|\mathbf{e}_{1,0}^t\mathbf{X}_{11}^{(l)}\right\| < (\alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})) \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l, \end{cases} \tag{3.17}$$

where σ is the upper bound for $s_1(\mathbf{R}_{1,l})$ by Corollary 3.2.9, i.e., $\sigma := \omega(\sqrt{\log n}) \cdot O(\sqrt{nk})$.

Substituting σ, τ and μ in Inequality (3.17), we get the upper bound for the norm of the updated $\mathbf{e}_0^t$ in Eq.(3.16) by the triangle inequality as follows:

$$\left\|(\mathbf{e}_{0,0}^t\mathbf{X}_{00}^{(l)}+\mathbf{e}_{1,0}^t\mathbf{X}_{10}^{(l)})\mathbf{R}_{1,l}+\mathbf{e}_{0,0}^t\mathbf{X}_{01}^{(l)}+\mathbf{e}_{1,0}^t\mathbf{X}_{11}^{(l)}\right\| < \alpha q\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}, \tag{3.18}$$

and similarly, the bound for the norm of the last nk coordinates of $\mathbf{e}_0^t\cdot\mathbf{M}^{(l)}$ is

$$\left\|\mathbf{e}_{0,0}^t\mathbf{X}_{02}^{(l)}+\mathbf{e}_{1,0}^t\mathbf{X}_{12}^{(l)}+\mathbf{e}_{2,0}^t\right\| < \alpha q\omega(\sqrt{\log n})^{2l+1}O(\sqrt{nk})^{3l+2}. \tag{3.19}$$

Since the infinity norm is smaller than the 2-norm, we have

$$\left\|(\mathbf{e}_0^t\cdot\prod_{j=1}^{l}\mathbf{M}_j)\begin{bmatrix}\mathbf{R}_{1,l} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\right\|_\infty \le \alpha q\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}, \tag{3.20}$$

by combining the Inequality (3.18) and Inequality (3.19).

Step 2. Errors generated from epoch 1 to $l-1$ have the same distribution with $\mathbf{e}_0$, but are multiplied fewer times by the key-switching matrices $\{\mathbf{M}_j\}$, which therefore yields a lower bound than $\mathbf{e}_0$.

Step 3. For $\mathbf{e}_l$, it is not multiplied by any transition matrix. Let $\mathbf{e}_l^t=(\mathbf{e}_{0,l},\mathbf{e}_{1,l},\mathbf{e}_{2,l})$. Then we have

$$\mathbf{e}_l^t\cdot\begin{bmatrix}\mathbf{R}_1 & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}=(\mathbf{e}_{0,l}^t\mathbf{R}_1+\mathbf{e}_{1,l}^t,\mathbf{e}_{0,l}^t\mathbf{R}_2+\mathbf{e}_{2,l}^t).$$

The bound on updated $\mathbf{e}_0$ in Inequality (3.20) also holds for $\mathbf{e}_l$.

Finally, we conclude that the upper bound on the infinity norm of the sum of updated errors is $\alpha ql\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}$ by triangle inequality, except with probability $2^{-\Omega(n)}$. That is

$$\left\|\sum_{i=0}^{l}(\mathbf{e}_i^t\cdot\prod_{j=i+1}^{l}\mathbf{M}_j)\cdot\begin{bmatrix}\mathbf{R}_{1,l} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\right\|_\infty \le \alpha ql\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}.$$

Since $1/\alpha = 4l\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}$, we have the desired property of error vectors, i.e., the Inequality (3.13).

□

3.5.3. Security Proof

In this section, we show that our scheme is IND-UE-CCA-1 secure under the hardness assumption of LWE.

Theorem 3.5.2. *For any* IND-UE-CCA-1 *adversary* $\mathscr{A}$ *against* TDUE, *there exists an adversary* $\mathscr{B}$ *against* $\mathsf{LWE}_{n,q,\alpha}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1}}_{\mathsf{TDUE},\mathscr{A}}(1^\lambda) \le 2(l+1)^3\cdot\Big[&(l+2)\cdot\mathsf{negl}(\lambda)\\ &+(n_{\mathsf{Dec}}+n_{\mathsf{sUpd}})\cdot 2^{-\Omega(n)}+\mathsf{Adv}^{\mathsf{LWE}}_{n,q,\alpha}(\mathscr{B})\Big],\end{aligned}$$

where l is the maximum number of ciphertext updates that the scheme TDUE *supports, and n_{Dec} and n_{sUpd} are the number of queries to the oracles $\mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_{\mathsf{sUpd}}$, respectively.*

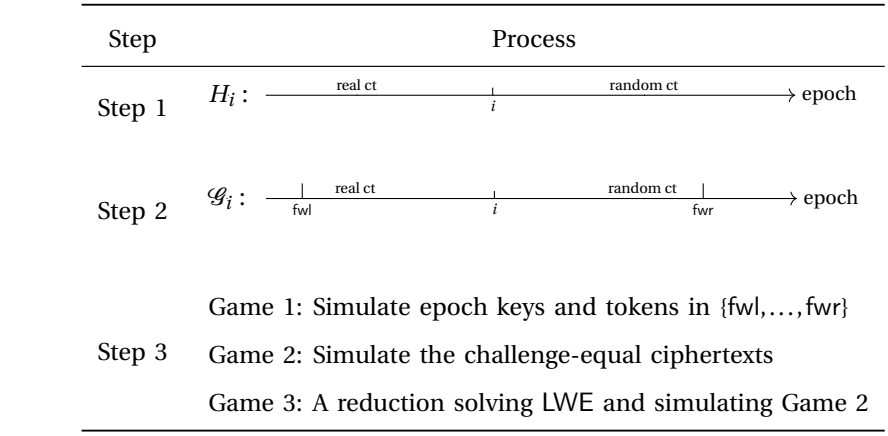


Figure 3.13: Steps in the security proof of TDUE. Within an insulated region, the reduction should appropriately respond to all the queries made by the adversary. Outside the region, the reduction can generate epoch keys and tokens freely. ct is the abbreviation of ciphertext.

Proof. In general, we take three steps, see Fig. 3.13, to bound the advantage of the adversary. In the first step, we build a hybrid game H_i for each epoch i, following [3, 6]. To the left of i, the game H_i returns the real challenge-equal ciphertexts and real generated tokens to respond to $\mathcal{O}_{\mathsf{Chall}}$ and $\mathcal{O}_{\mathsf{sUpd}}$ queries; while, to the right of i, H_i returns random ciphertexts and tokens as responses. To distinguish games H_i and H_{i+1}, we assume the adversary queries a challenge-equal ciphertext in epoch i, otherwise the response of both games will be the same. Therefore, the epoch i must be in an insulated region by Lemma 3.3.11. In Step 2, we then set up a modified game of H_i, called $\mathcal{G}_i$ that is the same as H_i except for the two randomly chosen epochs $\mathsf{fwr}, \mathsf{fwl}$ to simulate the insulated region around epoch i: if the adversary queries keys inside the region $[\mathsf{fwr}, \cdots, \mathsf{fwl}]$ or challenge-equal tokens in epochs fwr or $\mathsf{fwl}+1$, $\mathcal{G}_i$ aborts. In the last step, we play three games to bound the advantage of distinguishing games $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$. In Game 1, we simulate keys inside the insulated region, which are unknown to the adversary, and show how to simulate the response to queries on challenge-equal and non-challenge ciphertexts with the simulated keys. We then simulate the challenge-equal ciphertext in the second game, which allows for the construction of a reduction that solves the LWE by simulating the second game to the adversary. The details are provided as follows.

Denote the challenge input as $(\bar{\mathbf{m}}, \bar{c})$. We proceed via the following three steps.

Step 1. Consider a sequence of hybrid experiments $H_0^{\mathsf{b}}, \ldots, H_{l+1}^{\mathsf{b}}$ for $\mathsf{b} \in \{0,1\}$. The

game H_i^{b} is the same as the IND-UE-CCA-1 game except when the adversary queries a challenge-equal ciphertext via $\mathcal{O}_{\mathsf{Chall}}$ or $\mathcal{O}_{\mathsf{sUpd}}$ in epoch j:

- if $j < i$, return an honestly generated challenge-equal ciphertext to $\mathcal{O}_{\mathsf{Chall}}$. That is, return the encryption $\mathsf{Enc}(\mathsf{pk}_j, \bar{\mathbf{m}})$ if $\mathrm{b} = 0$, or the updated ciphertext $\mathsf{Upd}(\Delta_j, \bar{\mathrm{c}})$ if $\mathrm{b} = 1$. For the query to $\mathcal{O}_{\mathsf{sUpd}}$, return the real generated token and the true update of challenge-equal ciphertexts.
- if $j \geq i$, return a random ciphertext to $\mathcal{O}_{\mathsf{Chall}}$. For the query to $\mathcal{O}_{\mathsf{sUpd}}$, return a randomly generated token and the update of ciphertext by TDUE.Update.

We see that H_{l+1}^{b} is the same as $\mathsf{Exp}_{\mathsf{UE},\mathscr{A}}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1\text{-}b}}$, and $H_0^0 = H_0^1$ since all challenge responses are the same. Then we have

$$\begin{aligned}\mathsf{Adv}_{\mathsf{UE},\mathscr{A}}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1\text{-}b}}(1^\lambda) &= \left|\Pr\left[H_{l+1}^1 = 1\right] - \Pr\left[H_{l+1}^0 = 1\right]\right| \\ &\leq \sum_{i=0}^{l}\left|\Pr\left[H_{i+1}^1 = 1\right] - \Pr\left[H_i^1 = 1\right]\right| \\ &\quad + \sum_{i=0}^{l}\left|\Pr\left[H_{i+1}^0 = 1\right] - \Pr\left[H_i^0 = 1\right]\right|.\end{aligned}$$

Our goal is to prove $\left|\Pr\left[H_{i+1}^{\mathrm{b}} = 1\right] - \Pr\left[H_i^{\mathrm{b}} = 1\right]\right| \leq \mathsf{negl}(\lambda)$ for any i and b.

Step 2. Since the responses in all epochs except i will be the same in both games H_{i+1}^{b} and H_i^{b} for $\mathrm{b} \in \{0,1\}$, we assume the adversary who tries to distinguish the two games asks for a challenge-equal ciphertext in epoch i. Therefore, there exists an insulated region $[\mathsf{fwl}, \mathsf{fwr}]$ around epoch i such that no epoch keys in $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ and no tokens related to challenge-equal ciphertexts in epochs fwl and $\mathsf{fwr}+1$ are corrupted by Lemma 3.3.11.

We then define a new game $\mathscr{G}_i^{\mathrm{b}}$ which is the same as H_i^{b} except that the game chooses two random numbers $\mathsf{fwl}, \mathsf{fwr} \leftarrow \{0, \ldots, l\}$. If the adversary corrupts any epoch keys in $[\mathsf{fwl}, \mathsf{fwr}]$, or any token related to challenge-equal ciphertexts in epochs fwl and $\mathsf{fwr}+1$, the game aborts. The guess is correct with probability $1/(l+1)^2$. Then we have

$$\left|\Pr\left[H_{i+1}^{\mathrm{b}} = 1\right] - \Pr\left[H_i^{\mathrm{b}} = 1\right]\right| \leq (l+1)^2 \cdot \left|\Pr\left[\mathscr{G}_{i+1}^{\mathrm{b}} = 1\right] - \Pr\left[\mathscr{G}_i^{\mathrm{b}} = 1\right]\right|.$$

Our next goal is to prove $\left|\Pr\left[\mathscr{G}_{i+1}^{\mathrm{b}} = 1\right] - \Pr\left[\mathscr{G}_i^{\mathrm{b}} = 1\right]\right| \leq \mathsf{negl}(\lambda)$ for any i and b.

Step 3. For $\mathrm{b} \in \{0,1\}$ let $\mathscr{A}_i$ be an adversary who tries to distinguish $\mathscr{G}_{i+1}^{\mathrm{b}}$ from $\mathscr{G}_i^{\mathrm{b}}$. To provide an upper bound for the advantage of $\mathscr{A}_i$, we define a sequence of games as follows, and w.l.o.g. we assume $i = \mathsf{fwl}$.

Game 0:

For a random number $d \xleftarrow{\$} \{0,1\}$, if $d = 0$ the game plays $\mathscr{G}_i^b$ to $\mathscr{A}_i$; otherwise it plays $\mathscr{G}_{i+1}^b$ to $\mathscr{A}_i$. Denote E_j be the event that the adversary succeeds in guessing d in the **Game** j for $j \in \{0,1,2,3\}$. Then we have

$$\Pr[E_0] = \left|\Pr\left[\mathscr{G}_{i+1}^{\mathrm{b}} = 1\right] - \Pr\left[\mathscr{G}_i^{\mathrm{b}} = 1\right]\right|.$$

Game 1:
We consider a modified game, which is the same as **Game 0**, except the way that the epoch keys and tokens are generated in epochs from i to fwr. The overall idea to change the key in epoch i with a special form that ensures the embedding of LWE samples to the challenge ciphertexts in epoch i. But this form of the public key in epoch i may cause the failure to generate tokens in epochs from $i+1$ to fwr. We will show how to simulate tokens in epochs from $i+1$ to fwr.

1. At the start of the game, we pre-generate random invertible matrices $\{\mathbf{H}^*_{\mu,j}\}_{j=i}^{\mathsf{fwr}} \in \mathbb{Z}_q^{n\times n}$ that will be used to generate Δ_j and pk_j.
2. We generate all keys from 0 to l as **Game 0** by running the key generation algorithm, except for $\mathsf{pk}_i, \ldots, \mathsf{pk}_{\mathsf{fwr}}$.
3. **Public key in epoch i.** We choose random $\mathbf{A}_{0,i} \in \mathbb{Z}_q^{n\times \bar{m}}$ and secret key $\mathbf{R}_{1,i} \in \mathscr{D}$, and let the public key be

$$\mathsf{pk}_i = \left[\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}\right].$$

Notice that pk_i is still $\mathsf{negl}(\lambda)$-far from the uniform for any choice of $\mathbf{H}^*_{\mu,i}$. This design is to generate challenge ciphertexts of the form in the following Eq. (3.24) and further facilitate the simulation of challenge-equal ciphertext in epoch i in **Game 2**.

Public key in epoch $i+1$. In epoch $i+1$, we choose two matrices $\mathbf{X}_{00,i+1} \in \mathbb{Z}_q^{\bar{m}\times nk}$ and $\mathbf{X}_{10,i+1} \in \mathbb{Z}_q^{nk\times nk}$ from a Gaussian distribution with parameter τ and let

$$\mathbf{A}_{0,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \cdot \begin{bmatrix}\mathbf{X}_{00,i+1}\\ \mathbf{X}_{10,i+1}\end{bmatrix}, \tag{3.21}$$

which is still a $\mathsf{negl}(\lambda)$-far from the uniform. Then we choose a random matrix $\mathbf{R}_{1,i+1} \in \mathbb{Z}^{\bar{m}\times nk}$ whose entry equals to 0 with probability 1/2 and ± 1 with probability 1/4 each, and two random matrices $\mathbf{X}_{02,i+1}, \mathbf{X}_{12,i+1} \in \mathbb{Z}_q^{\bar{m}\times nk} \times \mathbb{Z}_q^{nk\times nk}$ with Gaussian entries of parameter $\tau\sqrt{\bar{m}/2}$. Compute

$$\mathbf{A}_{1,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \begin{bmatrix}\mathbf{X}_{00,i+1}\\ \mathbf{X}_{10,i+1}\end{bmatrix} \cdot \mathbf{R}_{1,i+1}, \tag{3.22}$$

$$\mathbf{A}_{2,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \begin{bmatrix}\mathbf{X}_{02,i+1}\\ \mathbf{X}_{12,i+1}\end{bmatrix} - \mathbf{A}_{2,i}, \tag{3.23}$$

where $\mathbf{A}_{2,i} = -\mathbf{A}_{0,i}\mathbf{R}_{2,i}$. Let the public key in epoch $i+1$ be

$$\mathsf{pk}_{i+1} = [\mathbf{A}_{0,i+1} \mid \mathbf{A}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G} \mid \mathbf{A}_{2,i+1}],$$

and secret key be $\mathsf{sk}_{i+1} = \mathbf{R}_{1,i+1}$, where $\mathbf{H}^*_{\mu,i+1}$ is generated in process 1.

Remark 3.5.3. *By Lemma 3.2.11, we know $(\boldsymbol{A}_{0,i+1}, \boldsymbol{A}_{0,i+1}\boldsymbol{R}_{1,i+1})$ is $\mathsf{negl}(n)$-close to the uniform if $\bar{m} \geq n\log(q) + 2\log(nk/\delta)$ for some small $\delta = \mathsf{negl}(n)$.*

Remark 3.5.4. *Every entry of* $\begin{bmatrix} \boldsymbol{X}_{00,i+1} \\ \boldsymbol{X}_{10,i+1} \end{bmatrix} \cdot \boldsymbol{R}_{1,i+1}$ *is an inner product of a* $\bar{m}$*-vector from a Gaussian distribution with parameter* τ *and a* $\{0,1,-1\}^{\bar{m}}$ *vector with half of the coordinates equal to 0, which is therefore a vector from a Gaussian distribution with parameter* $\tau\sqrt{\bar{m}/2}$ *by Lemma 3.2.12, i.e., the same distribution as the second-column block matrix of the token in* ***Game 0****, which is the same distribution as in the real game.*

Remark 3.5.5. *We do not require the last block matrix* $\boldsymbol{A}_{2,i+1}$ *to be presented in the form of* $\boldsymbol{A}_{0,i+1}\boldsymbol{R}_{2,i+1}$ *for some matrix* $\boldsymbol{R}_{2,i+1}$ *as in the real game. However, we will show this does not affect the responses to the queries.*

Public key in epochs from $i+2$ **to** fwr**.** For any epoch j in $\{i+2,\ldots,\mathsf{fwr}\}$, we iteratively generate the public key pk_j and secret key sk_j in a way as generating pk_{i+1} and sk_{i+1}, respectively.

4. **Simulating challenge-equal queries.** An overview of the oracles that the adversary has access to, related to challenge-equal ciphertexts, are summarised in Fig. 3.14. We should respond to the update of and token w.r.t challenge-equal ciphertexts in epochs from i to fwr, but do not need to answer decryption queries on challenge-equal ciphertexts.

Ciphertext. For challenge-equal ciphertexts in epoch i, notice that a random invertible matrix should be generated when updating ciphertexts and encrypting messages; here we use the special pre-generated invertible matrix $\mathbf{H}^*_{\mu,i}$ in both of the two algorithms, which is randomly generated in process 1 and is unknown to the adversary conditioned on the public keys we sampled in process 3. Then the challenge ciphertext in epoch i generated either from updating or from fresh encryption is in the form $c_i = (\mathbf{H}^*_{\mu,i}, \mathbf{b}_i)$ where (for simplicity, we skip the modular arithmetic operation in the ciphertexts)

$$\begin{aligned}\mathbf{b}_i^t &= \mathbf{s}^t[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}^*_{\mu,i}\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}_\mathsf{b}))^t \\ &= \mathbf{s}^t[\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}_\mathsf{b}))^t, \end{aligned} \tag{3.24}$$

for some $\mathbf{s},\mathbf{e}$ and $\mathsf{b} \in \{0,1\}$, where $\mathbf{m}_0 = \bar{\mathbf{m}}$ and $\mathbf{m}_1$ is the plaintext of the challenge ciphertext $\tilde{c}$. Similarly, we use the pre-generated invertible matrix $\mathbf{H}^*_{\mu,j}$ in the update algorithm for epoch $j \in \{i+1,\ldots,\mathsf{fwr}\}$. And the challenge-equal ciphertext is in the form $c_j = (\mathbf{H}^*_{\mu,j}, \mathbf{b}_j)$ where

$$\begin{aligned}\mathbf{b}_j^t &= \mathbf{s}^t[\mathbf{A}_{0,j} \mid (-\mathbf{A}_{0,j}\mathbf{R}_{1,j} - \mathbf{H}^*_{\mu,j}\mathbf{G}) + \mathbf{H}^*_{\mu,j}\mathbf{G} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}_\mathsf{b}))^t \\ &= \mathbf{s}^t[\mathbf{A}_{0,j} \mid -\mathbf{A}_{0,j}\mathbf{R}_{1,j} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}_\mathsf{b}))^t, \end{aligned} \tag{3.25}$$

for some $\mathbf{s},\mathbf{e}$.

Token. For challenge-equal tokens in epoch $i+1$, we set

$$\begin{bmatrix} \mathbf{X}_{01,i+1} \\ \mathbf{X}_{11,i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{X}_{00,i+1} \\ \mathbf{X}_{10,i+1} \end{bmatrix} \cdot \mathbf{R}_{1,i+1}, \tag{3.26}$$

Simulation	Challenge-equal	
	i	$i+1$
Public Key	$\mathbf{A}_{0,i}$ $-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}$ $-\mathbf{A}_{0,i}\mathbf{R}_{2,i}$	$\mathbf{A}_{0,i+1}$ $-\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G}$ $\mathbf{A}_{2,i+1}$
Enc/$\mathbf{A}_\mu$	$\mathbf{A}_{0,i}$ $-\mathbf{A}_{0,i}\mathbf{R}_{1,i}$ $-\mathbf{A}_{0,i}\mathbf{R}_{2,i}$	$\mathbf{A}_{0,i+1}$ $-\mathbf{A}_{0,i}\mathbf{R}_{1,i+1}$ $\mathbf{A}_{2,i+1}$
Dec	–	–
TokenGen	Eq. (3.28)	
Update	$\mathbf{b}^t_{i+1} = \mathbf{b}^t_i \Delta_{i+1} + \mathbf{b}^t_0$	

Figure 3.14: Simulation of the responses to the queries on challenge-equal ciphertexts. When the adversary queries the oracle $\mathcal{O}_{\mathsf{sUpd}}$, the simulation returns the output of TokenGen and Update simultaneously. Public keys and ciphertexts (only $\mathbf{A}_\mu$ is shown) are represented by the block matrices.

and

$$\mathbf{M}_{i+1} = \begin{bmatrix} \mathbf{X}_{00,i+1} & \mathbf{X}_{01,i+1} & \mathbf{X}_{02,i+1} \\ \mathbf{X}_{10,i+1} & \mathbf{X}_{11,i+1} & \mathbf{X}_{12,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{3.27}$$

and $\mathbf{b}_{0,i+1}$ is the ciphertext of message $\mathbf{0}$ under pk_{i+1} with the invertible matrix $\mathbf{H}^*_{\mu,i+1}$, i.e., $\mathbf{b}_{0,i+1} = \mathbf{s}^t\mathbf{A}_{\mu,i+1} + \mathbf{e}^t$ for some $\mathbf{s}$ and $\mathbf{e}$. Based on Equations (3.21) to (3.23), we know that $\mathbf{M}_{i+1}$ is the key-switching matrix from epoch i to $i+1$: $\mathbf{A}_{\mu,i}\mathbf{M}_{i+1} = \mathbf{A}_{\mu,i+1}$, and moreover has the distribution $\mathsf{negl}(n)$-far from the distribution of the token in **Game 0** by Remark 3.5.4. Then we have

$$\Delta_{i+1} = (\mathbf{M}_{i+1}, \mathbf{b}_{0,i+1}, \mathbf{H}^*_{\mu,i+1}), \tag{3.28}$$

is a valid challenge-equal token in epoch $i+1$. Similarly, we generate challenge-equal tokens in epochs from $i+2$ to fwr.

5. **Simulating non-challenge queries.** An overview of the oracles that the adversary has access to on non-challenge ciphertexts is summarised in Fig. 3.15. We should respond to the queries on the encryption, update, and decryption in all epochs. Since keys outside the insulated region are truly generated in process 2, we focus on the simulation inside the region.

Simulation	Non-challenge	
	i	$i+1$
Public Key	$\mathbf{A}_{0,i}$	$\mathbf{A}_{0,i+1}$
	$-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}$	$-\mathbf{A}_{0,i+1}\mathbf{R}_{2,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G}$
	$-\mathbf{A}_{0,i}\mathbf{R}_{2,i}$	$\mathbf{A}_{2,i+1}$
Enc/$\mathbf{A}_\mu$	$\mathbf{A}_{0,i}$	$\mathbf{A}_{0,i+1}$
	$-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G} + \mathbf{H}_{\mu,i}\mathbf{G}$	$-\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G} + \mathbf{H}_{\mu,i+1}\mathbf{G}$
	$-\mathbf{A}_{0,i}\mathbf{R}_{2,i}$	$\mathbf{A}_{2,i+1}$
Dec	SimDec	
TokenGen	$\Delta_{i+1} = \mathsf{TokenGen}(\,)$	
Update	$\mathbf{b}^t_{i+1} = \mathbf{b}^t_i \Delta_{i+1} + \mathbf{b}^t_0$	

Figure 3.15: Simulation of the queries on non-challenge ciphertexts. The algorithm SimDec is defined below, which has the same decryption ability as TDUE.Dec if $\mathbf{H}_{\mu,j} \neq \mathbf{H}^*_{\mu,j}$ for $j \in \{i+1, \mathsf{fwr}\}$. When the adversary queries the oracle $\mathcal{O}_{\mathsf{sUpd}}$, the simulation returns the output of TokenGen and Update simultaneously.

Ciphertext. For non-challenge ciphertexts in epoch i, we perform the encryption and update algorithm as **Game 0** by generating random invertible matrices $\mathbf{H}_{\mu,i}$. The resulting non-challenge ciphertexts in epoch i are in the

form $c_i = (\mathbf{H}_{\mu,i}, \mathbf{b}_i)$ where

$$\mathbf{b}_i^t = \mathbf{s}^t[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t, \tag{3.29}$$

for some $\mathbf{s}, \mathbf{e}$. Similarly, for non-challenge ciphertexts in any epoch $j \in \{i+1, \ldots, \mathsf{fwr}\}$, we randomly generate invertible matrices $\mathbf{H}_{\mu,j}$ in the update algorithm and the encryption algorithm. The ciphertext is in the form $c_j = (\mathbf{H}_{\mu,j}, \mathbf{b}_j)$ where

$$\mathbf{b}_j^t = \mathbf{s}^t[\mathbf{A}_{0,j} \mid (-\mathbf{A}_{0,j}\mathbf{R}_{1,j} - \mathbf{H}^*_{\mu,j}\mathbf{G}) + \mathbf{H}_{\mu,j}\mathbf{G} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t. \tag{3.30}$$

Decryption. To aid with update and decryption queries, we choose an arbitrary (not necessarily short) $\hat{\mathbf{R}}_i \in \mathbb{Z}_q^{\bar{m} \times nk}$ such that $-\mathbf{A}_{0,i}\hat{\mathbf{R}}_i = -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}$. Then we know $\hat{\mathbf{R}}_i$ is a trapdoor for $\mathbf{A}_{\mu,i,01} = [\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G}]$. We use the algorithm SimDec described below to simulate the decryption algorithm for non-challenge ciphertexts in epoch i, and the simulated decryption algorithm can also be applied to non-challenge ciphertexts in epoch from $i+1$ to fwr.

- $\mathsf{SimDec}(\mathsf{sk} = \mathbf{R}_{1,i}, c = (\mathbf{H}_{\mu,i}, \mathbf{b})):$
  1. If c or $\mathbf{b}$ does not parse, or $\mathbf{H}_{\mu,i} = \mathbf{0}$ or $\mathbf{H}^*_{\mu,i}$, output $\perp$. Otherwise parse $\mathbf{b}^t = (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)^t$.
  2. Call $\mathsf{Invert}^{\mathcal{O}}(\hat{\mathbf{R}}_i, \mathbf{A}_{\mu,i,01}, (\mathbf{b}_0, \mathbf{b}_1) \bmod q, \mathbf{H}_{\mu,i})$ to get $\mathbf{z}$ and $\mathbf{e}^t = (\mathbf{e}_0, \mathbf{e}_1)^t$ such that $(\mathbf{b}_0, \mathbf{b}_1)^t = \mathbf{z}^t\mathbf{A}_{\mu,i,01} + (\mathbf{e}_0, \mathbf{e}_1)^t \bmod q$ (Lemma 3.2.4). If $\mathsf{Invert}^{\mathcal{O}}$ fails, output $\perp$.
  3. Let $\mathbf{u}, \mathbf{e}_2$ be the unique solution to the equation
  $$\mathbf{b}_2^t - \mathbf{z}^t\mathbf{A}_{2,i} = \mathbf{u}^t\mathbf{G} + \mathbf{e}_2^t \mod q,$$
  if they exist; otherwise output $\perp$. Let $\mathbf{m} = \mathsf{encode}^{-1}(\mathbf{u}^t\mathbf{G} \bmod q)$.
  4. If $\|\mathbf{e}_0\| \geq \alpha q\sqrt{\bar{m}}$ or $\|\mathbf{e}_j\| \geq \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $j = 1, 2$, output $\perp$. Otherwise output $\mathbf{m}$.

Whenever $\mathbf{H}_{\mu,i} \neq \mathbf{H}^*_{\mu,i}$ which is the case with probability $2^{-\Omega(n)}$, the call to the invert algorithm returns $\mathbf{z}$ and $\mathbf{u}$ properly if they exist, and SimDec has the same decryption ability as TDUE.Dec.

Token. By construction, the matrix $\hat{\mathbf{R}}_i$ is the trapdoor for the $[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G}]$. We can use the real token generation algorithm to generate matrices $\{\mathbf{X}_{i,00}, \mathbf{X}_{i,01}, \mathbf{X}_{i,02}, \mathbf{X}_{i,10}, \mathbf{X}_{i,11}, \mathbf{X}_{i,12}\}$ with the same distributions as in **Game 0** by calling the invert algorithm on $[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G}]$

with the trapdoor $\hat{\mathbf{R}}_i$ such that

$$[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G} \mid \mathbf{A}_{2,i}] \begin{bmatrix} \mathbf{X}_{00,i+1} & \mathbf{X}_{01,i+1} & \mathbf{X}_{02,i+1} \\ \mathbf{X}_{10,i+1} & \mathbf{X}_{11,i+1} & \mathbf{X}_{12,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$
$$= [\mathbf{A}_{0,i+1} \mid (-\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G}) + \mathbf{H}_{\mu,i+1}G \mid \mathbf{A}_{2,i+1}]. \tag{3.31}$$

Let $\mathbf{b}_{0,i+1}$ be the ciphertext of message $\mathbf{0}$ under pk_{i+1} with the random invertible matrix $\mathbf{H}_{\mu,i}$, $\mathbf{M}_{i+1}$ be the transition matrix from $\mathbf{A}_{\mu,i}$ to $\mathbf{A}_{\mu,i+1}$ in Eq. (3.31), and $\Delta_{i+1} = (\mathbf{M}_{i+1}, \mathbf{b}_{0,i+1}, \mathbf{H}^*_{\mu,i+1})$. Based on Equations (3.29) to (3.31), we know that Δ_{i+1} is a valid non-challenge token in epoch $i+1$. Since this process only requires the property that $\hat{\mathbf{R}}_i$ is a trapdoor, it can be applied to any epoch $j \in \{i+1 \ldots \mathsf{fwr}\}$ by choosing an matrix $\hat{\mathbf{R}}_j$ with the same property, which works as long as $\mathbf{H}_{\mu,j} \neq \mathbf{H}^*_{\mu,j}$.

Overall, we conclude that **Game 1** and **Game 0** are indistinguishable, which follows from

$$\begin{aligned} |\Pr[E_1] - \Pr[E_0]| &\leq (n_{\mathsf{Dec}} + n_{\mathsf{sUpd}}) \cdot \Pr\left[\mathbf{H} = \mathbf{H}^*\right] + \mathsf{negl}(\lambda) \cdot (l+1) \\ &= (n_{\mathsf{Dec}} + n_{\mathsf{sUpd}}) \cdot 2^{-\Omega(n)} + \mathsf{negl}(\lambda) \cdot (l+1), \end{aligned}$$

where n_{Dec} and n_{sUpd} are the number of queries to decryption and update, respectively. $\Pr[\mathbf{H} = \mathbf{H}^*]$ is the probability that two random invertible matrices are equal, and $l+1$ is the maximum length of the firewall.

Game 2:
Compared to **Game 1**, we only change challenge-equal ciphertexts in epoch from i to fwr, while keeping the other simulations the same, especially the challenge-equal token. In epoch i, we modify the last two nk-coordinates of the challenge ciphertext and keep the first $\bar{m}$ coordinates unchanged. That is,

$$\begin{aligned} \mathbf{b}_0^t &= \mathbf{s}^t\mathbf{A}_{0,i} + \hat{\mathbf{e}}_0^t, \\ \mathbf{b}_1^t &= -\mathbf{b}_0^t \cdot \mathbf{R}_{1,i} + \hat{\mathbf{e}}_1^t, \\ \mathbf{b}_2^t &= -\mathbf{b}_0^t \cdot \mathbf{R}_{2,i} + \hat{\mathbf{e}}_2^t + \mathsf{encode}(\mathbf{m}_\mathrm{b})^t, \end{aligned}$$

where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\hat{\mathbf{e}}_0 \leftarrow D_{\mathbb{Z}^{\bar{m}},\alpha q}$ and $\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2 \leftarrow D_{\mathbb{Z}^{nk},\alpha q\sqrt{m}\cdot\omega(\sqrt{\log n})}$. By substitution, we have $\mathbf{b}_1^t = -\mathbf{s}^t\mathbf{A}_{0,i} \cdot \mathbf{R}_{1,i} - \hat{\mathbf{e}}_0^t \cdot \mathbf{R}_{1,i} + \hat{\mathbf{e}}_1^t$. According to the Corollary 3.10 in [20], the distribution of $-\hat{\mathbf{e}}_0^t \cdot \mathbf{R}_{1,i} + \hat{\mathbf{e}}_1^t$ is $\mathsf{negl}(n)$-far from $D_{\mathbb{Z}^{nk},d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m} \cdot \alpha q) \cdot \omega(\sqrt{\log n})^2$, which is the error distribution of $\mathbf{b}_1$ in **Game 1**. Therefore, the distribution of $\mathbf{b}_1$ is within $\mathsf{negl}(n)$-distance from that in **Game 1**. The same applies to $\mathbf{b}_2$. Then we change the ciphertext in epoch $i+1$ by updating $(\mathbf{H}^*_{\mu,i}, (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2))$ using the challenge-equal token in Eq. (3.28), which is therefore a valid token, and similarly change challenge-equal ciphertexts in epochs from $i+2$ to fwr. Thus, we have $|\Pr[E_2] - \Pr[E_1]| \leq \mathsf{negl}(\lambda)$.

Game 3:
We consider a modified game that is the same as **Game 2**, except that we change the first $\bar{m}$ coordinates of the challenge ciphertext in epoch i, letting it be the challenge ciphertext to the adversary $\mathscr{A}_i$: either uniformly random or $\mathbf{s}^t\mathbf{A}_{0,i} + \widehat{\mathbf{e}}_0^t$, which should be hard to distinguish under the $\mathsf{LWE}_{n,q,\alpha}$ problem. Then we also update this modified challenge-equal ciphertext and change the ciphertext in epochs from $i+1$ to fwr as in **Game 2** to make sure that the challenge-equal tokens still serve as valid tokens. Thus, we have $|\Pr[E_3] - \Pr[E_2]| \le \mathsf{Adv}^{\mathsf{LWE}}_{n,q,\alpha}$.

The rest we need to prove now is that $|\Pr[E_3]| = 1/2$. It follows from $(\mathbf{A}_{0,i}, \mathbf{b}_0, \mathbf{b}_0 \cdot \mathbf{R}_{1,i}, \mathbf{b}_0 \cdot \mathbf{R}_{2,i})$ is $\mathsf{negl}(\lambda)$-far from uniform by the leftover hash lemma for random $\mathbf{A}_{0,i} \in \mathbb{Z}_q^{n \times \bar{m}}$, $\mathbf{b}_0 \in \mathbb{Z}_q^{\bar{m}}$, and $\mathbf{R}_{1,i}, \mathbf{R}_{2,i} \in \mathscr{D}$.

We thus complete the proof.

□

3.5.4. A PACKING UE

We now introduce a packing method to further improve the efficiency of c-d UE, which allows us to encrypt multiple messages into one ciphertext and execute ciphertext updates simultaneously.

Let N be a power of 2, $\mathscr{R} = \mathbb{Z}[X]/(X^N + 1)$, and $\mathscr{R}_q = \mathscr{R}/(q\mathscr{R})$ be the residue ring of $\mathscr{R}$ modulo q. Any polynomial $p(X)$ in $\mathscr{R}$ can be represented by $p(X) = \sum_{i=0}^{N-1} p_i X^i$ with degree less than N, which is associated with its coefficient vector $\{p_0, \ldots, p_{N-1}\} \in \mathbb{Z}^N$. For a distribution $\mathscr{X}$, when we say $p(X) \xleftarrow{\$} \mathscr{X}$, we mean the coefficient of $p(X)$ is chosen from $\mathscr{X}$. We use the same notations as in Sect. 3.5.1

Encoding. Prior to the packing construction, we first present an efficient encoding algorithm that encodes multiple messages $\mathbf{m}_0, \ldots, \mathbf{m}_{N-1} \in \mathbb{Z}_2^{nk}$ as an element in $\mathscr{R}_q$ with coefficients in $\Lambda(\mathbf{G}^t)$ as follows:

$$\mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) = \mathbf{B} \cdot \left(\mathbf{m}_0 + \mathbf{m}_1 x + \cdots + \mathbf{m}_{N-1} x^{N-1}\right),$$

where $\mathbf{B} \in \mathbb{Z}^{nk \times nk}$ is any basis of $\Lambda(\mathbf{G}^t)$. Note that it can be efficiently decoded.

At a high level, multiple message blocks are encrypted in the following form:

$$\begin{aligned}(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2(x))^t = \mathbf{s}^t [\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu \mathbf{G} \mid \mathbf{A}_2(x)] \\ + (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2(x))^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}))^t \mod q. \end{aligned} \tag{3.32}$$

Compared to TDUE, the major modification in this approach is in the third block that uses polynomial matrices and vectors. The secret key $\mathbf{R}$ is still the trapdoor for $[\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G}]$. Therefore, the decryption procedure is able to properly recover $\mathbf{s}$ from the first two blocks in Eq. (3.32) as TDUE.Dec, and then call $\mathsf{Invert}^{\mathscr{O}}$ over $\mathbf{b}_2(x) - \mathbf{s}^t\mathbf{A}_2(x)$ degree by degree to recover every message. Moreover, the token generation is feasible due to a generalized preimage sampling algorithm in Lemma 3.5.6.

Lemma 3.5.6. *Given a* $\mathbf{G}$*-trapdoor* $\mathbf{R}$ *for* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *with invertible matrix* $\mathbf{H}$ *and any polynomial vector* $\mathbf{u}(X) \in \mathscr{R}_q^n$*, there is an efficient algorithm called* $\mathsf{GSampleD}^{\mathscr{O}}(\mathbf{R}, \mathbf{A}, \mathbf{H}, \mathbf{u}(X), s)$ *that samples a Gaussian polynomial vector* $\mathbf{p}(X) \in \mathscr{R}_q^m$ *with coefficients from* $D_{\mathbb{Z},s}$ *such that* $\mathbf{A}\cdot\mathbf{p}(X) = \mathbf{u}(X)$*, where* s *is the smallest Gaussian parameter defined in Lemma 3.2.5.*

Proof. Calling the oracle $\mathsf{SampleD}^{\mathscr{O}}$ on each coefficient vector of $\mathbf{u}(X)$ returns a vector $\mathbf{p}_i$ such that

$$\mathbf{A}\mathbf{p}_i = \mathbf{u}_i,$$

for $0 \le i \le N$ and $\mathbf{u}(X) = \sum_{i=0}^{N-1} \mathbf{u}_i X^i$. Denote $\mathbf{p}(X) = \sum_{i=0}^{N-1} \mathbf{p}_i X^i$, then we know $\mathbf{A}\mathbf{p}(X) = \mathbf{u}(X)$. □

Packing UE. Our packing UE scheme is described as follows.

- $\mathsf{KG}(1^\lambda)$: choose $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}, \mathbf{R}_1, \mathbf{R}_2(X) \xleftarrow{\$} \mathscr{D}$ and let $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \mathbf{A}_2] = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 \mid -\mathbf{A}_0\mathbf{R}_2(X)] \in \mathscr{R}_q^{n\times m}$ where $m = \bar{m} + 2nk$. The public key is $\mathsf{pk} = \mathbf{A}$ and the secret key is $\mathsf{sk} = \mathbf{R}_1$.

- $\mathsf{Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m}_0, \ldots, \mathbf{m}_{N-1} \in \{0,1\}^{nk})$: choose an invertible matrix $\mathbf{H}_\mu \in \mathbb{Z}_q^{n\times n}$, and let $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. Choose a random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and an error vector $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2(X)) \in D_{\mathbb{Z}^{\bar{m}}, \alpha q} \times D_{\mathbb{Z}^{nk}, d} \times D_{\mathbb{Z}^{nk}, d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m}\cdot(\alpha q)^2)\cdot \omega(\sqrt{\log n})^2$. Let

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1})^t \mod q, \tag{3.33}$$

  where $\mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) = \mathbf{B}\cdot(\mathbf{m}_0 + \mathbf{m}_1 X + \cdots + \mathbf{m}_{N-1}X^{N-1})$.

- $\mathsf{Dec}(\mathsf{sk} = \mathbf{R}_1, c = (\mathbf{H}_\mu, \mathbf{b}))$: Recover $\mathbf{s}$ as the steps 1 to 3 in the decryption algorithm of TDP. Parse $\mathbf{b}^t = (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2(X))^t$, invert $\mathbf{b}_2(X)^t - \mathbf{s}^t\mathbf{A}_2$ degree by degree, and find the unique solution $\mathbf{u}_i, \mathbf{e}_{2,i}$ to the equation

$$\mathbf{b}_{2,i}^t - \mathbf{s}^t\mathbf{A}_{2,i} = \mathbf{u}_i^t\mathbf{G} + \mathbf{e}_{2,i}^t \mod q,$$

  by Lemma 3.2.4 if they exist, where $\mathbf{b}_2(X) = \sum \mathbf{b}_{2,i}X^i$ and $\mathbf{A}_2 = \sum \mathbf{A}_{2,i}X^i$. Output the following result as $\mathbf{m}_i$ if it exists,

$$\mathsf{encode}^{-1}\left((\mathbf{u}_i^t\mathbf{G})^t\right) \in \mathbb{Z}_2^{nk},$$

  for $0 \le i \le N-1$, otherwise output $\perp$.

- $\mathsf{TokenGen}(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}', \mathbf{H}_\mu)$: Generate block matrices $\mathbf{M}_{00}$, $\mathbf{M}_{01}$, $\mathbf{M}_{10}$, $\mathbf{M}_{11}$ of $\mathbf{M}$ as in steps 2 and 3 of TDUE.TokenGen, and call the algorithm $\mathsf{GSampleD}^{\mathscr{O}}$ in Lemma 3.5.6 to find $\mathbf{M}_{02}(X) \in \mathscr{R}_q^{\bar{m}\times nk}, \mathbf{M}_{12}(X) \in \mathscr{R}_q^{nk\times nk}$ such that

$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{M}_{02}(X) \\ \mathbf{M}_{12}(X) \end{bmatrix} = \mathbf{A}_2' - \mathbf{A}_2.$$

Generate $\mathbf{b}_0$ a fresh encryption of message 0. Output the update token $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$.

- TDUE.Update$(\Delta, c = (\mathbf{H}_\mu, \mathbf{b}))$: parse $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$ and compute

$$(\mathbf{b}')^t = \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \mod q,$$

and output $c' = (\mathbf{H}'_\mu, \mathbf{b}')$.

Remark. The correctness and IND-UE-CCA-1 security of packing UE is analogous to those of TDUE (as shown in Lemma 3.5.1 and Theorem 3.5.2). We omit the details. For a message of bit length Nnk, packing UE, compared to TDUE, reduces the number of ciphertexts by a factor of N, and only one ciphertext header is required to be downloaded in the token generation procedure.

3.6. CONCLUSION AND FUTURE WORK

In this paper, we propose a stronger confidentiality notion than prior work for ciphertext-dependent updatable encryption, which captures adaptive security and is applied to both types of UE schemes: deterministic and randomized updates. We also provide a new public key encryption scheme, based on which we construct our updatable encryption scheme. Moreover, we propose a cost-effective packing UE scheme that is able to execute ciphertext updates simultaneously.

Future Work. The first FHE scheme introduced by Gentry [25] and all its subsequent works require a "circular security" assumption, namely that it is safe to encrypt old secret keys with new keys. Such an idea has inspired the UE construction with backward directional key updates. In turn, we suggest an open problem that if no-directional updatable encryption, which is able to update ciphertext without revealing old and new keys, can be used to construct FHE that does not rely on the assumption.

REFERENCES

[1] D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. "Key homomorphic PRFs and their applications". In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Heidelberg: Springer, 2013, pp. 410–428. DOI: 978-3-642-40041-4_23.

[2] C. Boyd, G. T. Davies, K. Gjøsteen, and Y. Jiang. "Fast and Secure Updatable Encryption". In: *CRYPTO 2020, Part I*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12170. LNCS. Springer, 2020, pp. 464–493. DOI: 10.1007/978-3-030-56784-2_16.

[3] Y. Jiang. "The direction of updatable encryption does not matter much". In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer. Heidelberg, 2020, pp. 529–558. DOI: 10.1007/978-3-030-64840-4_18.

[4] M. Klooß, A. Lehmann, and A. Rupp. "(R)CCA secure updatable encryption with integrity protection". In: *EUROCRYPTO 2019, Part I*. Ed. by Y. Ishai and V. Rijmen. Vol. 11476. LNCS. Springer. Heidelberg, 2019, pp. 68–99. DOI: 10.1007/978-3-030-17653-2_3.

[5] A. Lehmann and B. Tackmann. "Updatable encryption with post-compromise security". In: *EUROCRYPT 2018, Part III*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10822. LNCS. Springer. Heidelberg, 2018, pp. 685–716. DOI: 10.1007/978-3-319-78372-7_22.

[6] R. Nishimaki. "The Direction of Updatable Encryption Does Matter". In: *PKC 2022*. Ed. by G. Hanaoka, J. Shikata, and Y. Watanabe. Vol. 13178. LNCS. Cham: Springer, 2022, pp. 194–224. ISBN: 978-3-030-97131-1. DOI: 10.1007/978-3-030-97131-1_7.

[7] D. Slamanig and C. Striecks. *Puncture 'Em All: Updatable Encryption with No-Directional Key Updates and Expiring Ciphertexts*. Cryptology ePrint Archive, Paper 2021/268. https://eprint.iacr.org/2021/268. 2021.

[8] D. Boneh, S. Eskandarian, S. Kim, and M. Shih. "Improving Speed and Security in Updatable Encryption Schemes". In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Cham: Springer, 2020, pp. 559–589. ISBN: 978-3-030-64840-4. DOI: 10.1007/978-3-030-64840-4_19.

[9] L. Chen, Y. Li, and Q. Tang. "CCA Updatable Encryption Against Malicious Re-encryption Attacks". In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer, 2020, pp. 590–620. DOI: 10.1007/978-3-030-64840-4_20.

[10] A. Everspaugh, K. Paterson, T. Ristenpart, and S. Scott. “Key rotation for authenticated encryption”. In: *CRYPTO 2017, Part III*. Ed. by J. Katz and H. Shacham. Vol. 10403. LNCS. Springer. Heidelberg, 2017, pp. 98–129. DOI: 10.1007/978-3-319-63697-9_4.

[11] N. Alamati, H. Montgomery, and S. Patranabis. “Symmetric Primitives with Structured Secrets”. In: *CRYPTO 2019, Part I*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11692. LNCS. Springer, 2019, pp. 650–679. DOI: 10.1007/978-3-030-26948-7_23.

[12] Y. J. Galteland and J. Pan. “Backward-Leak Uni-Directional Updatable Encryption from (Homomorphic) Public Key Encryption”. In: *PKC 2023, Part II*. Ed. by A. Boldyreva and V. Kolesnikov. Vol. 13941. LNCS. Springer, 2023, pp. 399–428. DOI: 10.1007/978-3-031-31371-4_14.

[13] K. Sakurai, T. Nishide, and A. Syalim. “Improved proxy re-encryption scheme for symmetric key cryptography”. In: *IWBIS, 2017*. IEEE, 2017, pp. 105–111. DOI: 10.1109/IWBIS.2017.8275110.

[14] E. Kirshanova. “Proxy Re-encryption from Lattices”. In: *PKC 2014*. Ed. by H. Krawczyk. Vol. 8383. LNCS. Springer, 2014, pp. 77–94. DOI: 10.1007/978-3-642-54631-0_5.

[15] X. Fan and F. Liu. “Proxy Re-Encryption and Re-Signatures from Lattices”. In: *ACNS 2019*. Ed. by R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung. Vol. 11464. LNCS. Springer, 2019, pp. 363–382. DOI: 10.1007/978-3-030-21568-2_18.

[16] Z. Brakerski and V. Vaikuntanathan. “Efficient Fully Homomorphic Encryption from (Standard) LWE”. In: *FOCS, 2011*. Ed. by R. Ostrovsky. IEEE Computer Society, 2011, pp. 97–106. DOI: 10.1109/FOCS.2011.12.

[17] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *ITIC 2012*. Ed. by S. Goldwasser. ACM, 2012, pp. 309–325. DOI: 10.1145/2090236.2090262.

[18] D. Micciancio and C. Peikert. “Trapdoors for lattices: Simpler, tighter, faster, smaller”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.

[19] M. Ajtai. “Generating hard instances of lattice problems”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108. URL: https://eccc.weizmann.ac.il/eccc-reports/1996/TR96-007/index.html.

[20] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *ACM 2005*. Ed. by H. N. Gabow and R. Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[21] S. Agrawal, C. Gentry, S. Halevi, and A. Sahai. “Discrete Gaussian Leftover Hash Lemma over Infinite Domains”. In: *ASIACRYPT 2013, Part I*. Ed. by K. Sako and P. Sarkar. Vol. 8269. LNCS. Springer, 2013, pp. 97–116. DOI: 10.1007/978-3-642-42033-7_6.

[22] W. Banaszczyk. “New bounds in some transference theorems in the geometry of numbers”. In: *Mathematische Annalen* 296.1 (1993), pp. 625–635.

[23] L. Ducas and D. Micciancio. “Improved Short Lattice Signatures in the Standard Model”. In: *CRYPTO 2014, Part I*. Ed. by J. A. Garay and R. Gennaro. Vol. 8616. LNCS. Springer, 2014, pp. 335–352. DOI: 10.1007/978-3-662-44371-2_19.

[24] J. Alwen and C. Peikert. “Generating Shorter Bases for Hard Random Lattices”. In: *Theory Comput. Syst.* 48.3 (2011), pp. 535–553. DOI: 10.1007/s00224-010-9278-3.

[25] C. Gentry. “Fully homomorphic encryption using ideal lattices”. In: *ACM STOC 2009*. Ed. by M. Mitzenmacher. ACM, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

4

Batch Programmable Bootstrapping, within a Polynomial Modulus

To improve the efficiency of fully homomorphic encryption (FHE), Liu and Wang (EUROCRYPT 2023) proposed a bootstrapping technique within a polynomial modulus that refreshes n ciphertexts at once with an asymptotic cost of $\widetilde{O}(n^{0.75})$ FHE multiplications in amortization. Despite the low amortized cost, it remains unclear whether their technique is practical for larger message spaces beyond a single bit. In this work, we introduce a novel batch bootstrapping technique within a polynomial modulus that enables noise refreshment over a general message space extending beyond one bit, while maintaining the same amortized cost and noise overhead. Our batch bootstrapping is also programmable, enabling the evaluation of a univariate function simultaneously with noise refreshment. Furthermore, our approach overcomes a key limitation of third-generation FHE schemes, which require the evaluated function to be negacyclic; in contrast, our method supports arbitrary functions. Additionally, we propose two homomorphic decomposition algorithms, extending our batch programmable bootstrapping to support larger message spaces. To further enhance practicality, we demonstrate the evaluation of commonly used activation functions in Convolutional Neural Networks (CNNs), such as ReLU, sign, *and* max.

4.1. INTRODUCTION

Fully homomorphic encryption (FHE) enables arbitrary computation over encrypted data without the need to decrypt it. Currently, Gentry's *bootstrapping* technique [1] is known as the unique method to achieve FHE, refreshing ciphertexts after multiple homomorphic computations and allowing further operations. Recent advancements in FHE have reduced bootstrapping time to milliseconds, inspiring numerous applications like privacy-preserving machine learning and multiparty computation. However, some open problems still limit the application of FHE, which we discuss below according to two primary bootstrapping techniques.

The first type of bootstrapping is represented by BGV and its variants [2–5], designed to reduce the amortized bootstrapping cost per message by refreshing multiple messages at once. This approach supports single instruction multiple data (SIMD) operations. However, its major limitation lies in quasi-polynomial noise growth, necessitating large storage for homomorphic evaluation due to the requirement of a quasi-polynomial modulus. Additionally, its security is based on a stronger assumption of worst-case lattice problems (i.e., with superpolynomial approximation factors).

The second simpler one is FHEW-like bootstrapping [6–9], which encrypts a single message into a ciphertext. These works require only tens of milliseconds for bootstrapping, and they attain only a polynomial noise. Thus they achieve a polynomial size modulus and the ideal assumption of the worst-case lattice problem (i.e., with polynomial approximation factors). FHEW-like bootstrapping is also often called *programmable bootstrapping* (PBS) [10, 11], which means that a univariate function can be evaluated during the bootstrapping procedure. However, they refresh one single ciphertext each bootstrapping, which results in a higher amortized cost than the above. Moreover, the function to be evaluated should be *negacyclic* (to be defined later).

Recent works aim to combine the advantages of both types of bootstrapping. Micciancio and Sorrell [12] proposed the first amortized FHEW bootstrapping technique capable of bootstrapping n FHEW ciphertexts using $O(3^\rho \cdot n^{1+1/\rho})$ FHE multiplications. This results in an amortized cost of $O(3^\rho \cdot n^{1/\rho})$ FHE operations per ciphertext, where $1/\rho$ is a small parameter close to 0. Subsequently, Guimarães et al. [13] further improved this approach, reducing the amortized cost to $O(\rho \cdot n^{1/\rho})$ FHE operations. Liu and Wang [14] achieved even greater improvement, attaining an amortized cost of $\widetilde{O}(n^{0.75})$.

Two nearly optimal batched bootstrapping techniques were proposed by [15] and [16], both achieving an asymptotic cost of $\widetilde{O}(1)$ homomorphic multiplications per ciphertext. However, Liu and Wang [15] utilized the SIMD feature of BFV/BGV for evaluating the decryption of packed LWE ciphertexts, leading to superpolynomial approximation factors, despite supporting arbitrary function evaluation.

Regarding the work by Liu and Wang [14, 16], their work is limited by its binary message space, rendering it impractical for applications such as privacy-preserving machine learning, where inputs, such as handwritten images, typically have higher bit depths, such as 8 bits. It remains unclear whether this work is practical for larger message spaces and whether it supports the homomorphic evaluation of functions

over such spaces. We present a comparison to previous schemes in Fig. 4.1.

We refer to this bootstrapping procedure, which evaluates a univariate function over multiple ciphertexts simultaneously while refreshing the noise, as *batch programmable bootstrapping*.

Schemes	Amortized Cost	Approximation Factors	Programmable
FHEW-like [6, 7]	$O(n)$	Poly	Yes
MS18 [12]	$\widetilde{O}(3^{\rho} \cdot n^{1/\rho})$	Poly	No
GPV23 [13]	$O(\rho \cdot n^{1/\rho})$	Poly	Yes
LW23 [14]	$\widetilde{O}(n^{0.75})$	Poly	No
LW23 [16]	$\widetilde{O}(1)$	Poly	No
LW23 [15]	$\widetilde{O}(1)$	Superpoly	Yes
Our work	$\widetilde{O}(n^{0.75})$	Poly	Yes

Figure 4.1: Comparison of batch bootstrapping schemes. The columns represent the amortized bootstrapping running time per ciphertext (n is the number of ciphertexts), approximation factors for the based worst-case lattice problems, and whether the scheme is programmable, respectively.

4.1.1. OUR RESULT

In this paper, we make two contributions to *batch programmable bootstrapping* (batch PBS). Firstly, we propose a basic batch PBS over a more general message space $\mathbb{Z}_t$ for $t > 2$, which incurs only a polynomial noise growth. It eliminates the requirement of negacyclicity for the function to evaluate compared to PBS, resulting in a precision one bit more than that in PBS (limited to 4-5 bits). Secondly, we optimize the basic batch PBS by introducing two homomorphic decomposition algorithms, which allow us to homomorphically decompose a large plaintext into several ciphertexts encrypting small chunks of the input plaintext. For each chunk, we are able to evaluate arbitrary functions by making use of the basic batch PBS as a black box, leading to a further improvement of the precision. As an application, we show an accurate evaluation of the activation functions commonly used in Convolutional Neural Networks, involving $\mathsf{Sign}, \mathsf{ReLU}$, and max in a batch way.

Basic Batch PBS. We split the evaluation of a univariate function over multiple ciphertexts into a linear and nonlinear step, the former of which corresponds to linear operation in the decryption algorithm, such as subtraction and inner product, and the latter deals with nonlinear parts, including division and rounding. We reflect the target function into a look-up table, which will be evaluated during bootstrapping with no additional cost. Various techniques are provided to optimize the performance.

- We reduce the amortized cost of the homomorphic inner product by adopting the packing technique in [14]. Roughly, this requires $\widetilde{O}(n)$ FHE multiplication for $O(n^{1/4})$ ciphertexts, resulting in an amortized cost of $\widetilde{O}(n^{0.75})$ compared to $O(n)$ for the naive way. This is the major computation cost in our batch PBS method.

- To eliminate the need for negacyclicity in function to bootstrap, we work with ciphertexts in the general cyclotomic rings, instead of cyclotomic rings of two's power used by PBS [6, 7]. We observe that this limitation comes from the fact that $x^{m+N} = -x^m$ for any message m over the power-of-two cyclotomic ring $\mathbb{Z}[x]/(x^N+1)$. However, ξ_q^i, ξ_q^j in prime cyclotomic ring $\mathbb{Z}[\xi_q]$ are independent for $i \neq j$, overcoming the limitation. Our general cyclotomic ring uses $\mathbb{Z}[\xi_q]$ for a subring. For more details, please refer to Section 4.3.1.

- We designed a new look-up table that is suitable for our batch PBS mechanism, especially in the setting of general cyclotomic rings. We embed this look-up table in the beginning (instead of at the end) of the first step, leading to a noise growth independent of the look-up table, by taking advantage of the asymmetric error growth of GSW (defined later) ciphertexts.

Decomposition. Eliminating the need for negacyclicity enables our batch PBS to support one more bit of precision compared to PBS, which is at most 5 bits. However, our batch PBS technique suffers from the same limitation as PBS that the run time complexity increases linearly with the ciphertext modulus. For higher precisions, we provide two decomposition algorithms to construct batch PBS methods that scale logarithmically with the ciphertext modulus, using the basic batch PBS as a black box.

- **Decomposition and Removal.** This algorithm works for evaluating functions like Sign, where the value of the function is determined by only a few most significant bits of the message. Take the Sign function for an example, its value is equal to the most significant bit of a message. The high-level idea decomposition and removal algorithm is adopted from [17] decomposing a large-precision ciphertext into digits and constructing a HomFloor algorithm which is repeatedly executed to clear the last digit until the modulus is small enough that we can directly use the basic PBS. The most efficient HomFloor algorithm in [17] requires 2 invocations of PBS to homomorphically clear the last digit of a single ciphertext. We provide a more efficient HomFloor algorithm that requires only 1 invocation of basic PBS to homomorphically clear the last digit of multiple ciphertexts.

- **Decomposition and Reconstruction.** This algorithm is more suitable for functions whose value is not determined by only a few bits. The first step is still a decomposition of a large-precision ciphertext into small digits. Instead of removing the last digit, we repeatedly evaluate a newly designed function related to the target function via batch PBS and reconstruct the function value over input plaintext by summing all the evaluations on each last digit. Take

$m = \sum m_j \cdot B^j$ with basis B as an example, we homomorphically evaluate a new function $\widetilde{f}_j : x \to f(x \cdot B^j)$ over the last digit m_j and reconstruct $f(m)$ by summing $\widetilde{f}_j(m_j)$.

4.1.2. RELATED WORKS

To reduce the limitation of negacyclicity, Chillotti et al. [10] developed a programmable bootstrapping technique without padding. This approach requires two invocations of programmable bootstrapping and an additional LWE ciphertext multiplication to refresh an encryption of a message without padding. In contrast, our method can refresh multiple such messages with a single invocation of our batch programmable bootstrapping.

Summary of Contributions. We propose a batch programmable bootstrapping method within a polynomial modulus, which requires an amortized cost of $\widetilde{O}(n^{0.75})$ FHE multiplications per ciphertext and supports large precision evaluation in practice. Our scheme eliminates the need for negacyclicity in function to be evaluated, which is required by programmable bootstrapping schemes.

4.2. PRELIMINARIES

In this work, column vectors and matrices are denoted by lower-case and upper-case bold letters, respectively. For a vector $\mathbf{x}$, we denote $\|\mathbf{x}\|$ as the 2-norm of $\mathbf{x}$, $\|\mathbf{x}\|_\infty$ as the infinity norm, and we use non-boldface letters to refer to its entries: $\mathbf{x} = (x_1, \cdots, x_n)$. For a matrix $\mathbf{A}$, we use $\mathbf{A}^\top$ to denote its transpose.

4.2.1. ALGEBRAIC NUMBER THEORY

In this section, we provide a brief overview of the algebraic number theory concepts necessary for understanding the batch bootstrapping procedure outlined in [14]. Additional details can be found in [18, 19].

Cyclotomic Number Fields. For a positive integer m, let $\xi_m = e^{2\pi i/m}$ be an mth root of unity, and let $K = \mathbb{Q}(\xi_m)$ be the mth *cyclotomic field*, obtained by adjoining ξ_m to $\mathbb{Q}$. Since the minimal polynomial of ξ_m over $\mathbb{Q}$ has degree $n = \phi(m)$ [18], where ϕ is Euler's totient function, we can view K as an n-dimensional vector space with a basis $1, \xi_m, \cdots, \xi_m^{n-1}$. The ring of integers of the field K is $\mathbb{Z}[\xi_m]$.

It is known that $K = \mathbb{Q}(\xi_m)$ is a Galois extension over $\mathbb{Q}$, and the set of automorphisms of K that fix every element of $\mathbb{Q}$ forms a group under composition, called the *Galois group* of K, denoted by $\mathrm{Gal}(K/\mathbb{Q})$. It can be expressed as [20]:

$$\mathrm{Gal}(K/\mathbb{Q}) = \{\sigma_i : \gcd(i, m) = 1\},$$

where σ_i is defined by $\sigma_i(\xi_m) = \xi_m^i \in \mathbb{C}$. We observe that σ_i is the complex conjugate of σ_{m-i}: $\sigma_i = \overline{\sigma_{m-i}}$. Then, the canonical embedding $\sigma : K \to \mathbb{C}^{\phi(m)}$ is defined as [19]:

$$\sigma(a) = \{\sigma_i(a)\}_{i \in \mathbb{Z}_m^*},$$

and the $\mathbb{Q}$-linear *trace* function $\mathrm{Tr} = \mathrm{Tr}_{K/\mathbb{Q}} : K \to \mathbb{Q}$ is defined as [19]:

$$\mathsf{Tr}(a) = \sum_{\sigma \in \mathsf{Gal}(K/\mathbb{Q})} \sigma(a) = \sum_i \sigma_i(a),$$

for $a \in K$. Moreover, we have

$$\mathsf{Tr}(a \cdot b) = \sum_i \sigma_i(a) \cdot \sigma_i(b) = \langle \sigma(a), \overline{\sigma(b)} \rangle.$$

which is a symmetric bilinear form that enables us to define duality as follows:

Duality. For any $\mathbb{Q}$-basis $B = \{\mathsf{b}_i\}$ of K, its dual basis is denoted by $B^\vee = \{\mathsf{b}_i^\vee\}$, characterized by $\mathsf{Tr}(\mathsf{b}_i \cdot \mathsf{b}_j^\vee) = 1$ if $i = j$, and 0 otherwise.

The trace function and duality can be defined similarly over any Galois extension $[E : F]$. For a prime number m, the extension field $\mathbb{Q}(\xi_m)$ has the following properties.

Lemma 4.2.1 ([18], Proposition 4.2.5)**.** *When m is a prime, then $n = m - 1$, and the minimal polynomial of ξ_m over $\mathbb{Q}$ is given by $\Phi_m(x) = 1 + x + \cdots + x^{m-1}$.*

Corollary 4.2.2. *For a prime m, the integer solutions to the equation $(1, \xi_m, \cdots, \xi_m^{m-1}) \cdot \mathbf{x}^\top = 0$ are only of the form $\mathbf{x} = (a, a, \cdots, a)$ for $a \in \mathbb{Z}$.*

4.2.2. FHEW-LIKE CRYPTOSYSTEMS

Micciancio and Polyakov [9] refer to FHEW [6, 21] and TFHE [7, 22] collectively as FHEW-like cryptosystem. The bootstrapping procedure in FHEW-like cryptosystem requires to work with LWE ciphertext, and RLWE and RGSW ciphertexts in the power-of-two cyclotomic ring $\mathcal{R} = \mathbb{Z}[x]/(x^N + 1)$ where N is a power of 2. We define the quotient ring $\mathcal{R}_Q := \mathcal{R}/Q\mathcal{R}$ for the modulus Q and give a review of FHEW-like cryptosystem below.

- **LWE**. Let t, n and q be three positive integers, $\mathscr{X}$ the key distribution over $\mathbb{Z}_q^n$, and $\mathscr{X}'$ the error distribution over $\mathbb{Z}_q$. The LWE encryption of a message $\mu \in \mathbb{Z}_t$ under the secret key $\mathbf{s} \in \mathscr{X}$ is defined as $\mathrm{LWE}_{\mathbf{s},q}(\mu) = (b, \mathbf{a}) = (\langle \mathbf{a}, \mathbf{s} \rangle + e + \lfloor \frac{q}{t} \cdot \mu \rceil, \mathbf{a})$, where $\mathbf{a}$ is a random vector from $\mathbb{Z}_q^n$ and the error e is sampled from $\mathscr{X}'$.

- **RLWE**. The set of RLWE encryptions of a polynomial message $\mu \in \mathcal{R}_t$ is defined as

$$\mathsf{RLWE}_{s,Q}(\mu) := \left\{ \left(sa + e + \left\lfloor \frac{Q}{t} \right\rceil \mu, a \right) \in \mathcal{R}_Q^2 \right\},$$

  where the polynomial a is sampled uniformly from $\mathcal{R}_Q$ and the secret key s (e, resp.) is from a key distribution $\mathscr{D}$ (error distribution $\mathscr{D}'$, resp.) over $\mathcal{R}_Q$.

- **RGSW.** Use the same notations as in RLWE, and assume that $Q = B_g^{d_g}$ for a base B_g and some integer d_g. Denote the gadget vector as $\mathbf{g}^\top = (1, B_g, \cdots, B_g^{d_g - 1})$

and the gadget matrix as $\mathbf{G} = \mathbf{g}^\top \otimes \mathbf{I}_2$. The set of RGSW ciphertexts encrypting a message $\mu \in \mathscr{R}$ is given by:

$$\mathsf{RGSW}_{s,Q}(\mu) := \left\{ \begin{pmatrix} s\mathbf{a}^t + \mathbf{e}^t \\ \mathbf{a}^t \end{pmatrix} + \mu\mathbf{G} \in \mathscr{R}_Q^{2\times 2l} \right\},$$

where the polynomial vector $\mathbf{a}$ is random in $\mathscr{R}_Q^{2l}$, and the secret key s and polynomial error vector $\mathbf{e}$ are sampled from $\mathscr{D}$ and $\mathscr{D}'$, respectively.

FHEW-like Bootstrapping. The bootstrapping procedure is based on the so-called external product $\odot$:

$$\odot : \mathrm{RLWE} \times \mathrm{RGSW} \to \mathrm{RLWE},$$

which takes as input a RLWE encryption of a polynomial μ_1 and a RGSW encryption of a polynomial μ_2, and outputs a RLWE encryption of $\mu_1 \cdot \mu_2$.

FHEW-like bootstrapping is often called *programmable bootstrapping* [11], noted as PBS, meaning that a univariate function $f : \mathbb{Z}_t \to \mathbb{Z}_t$ can be evaluated at the same time as the noise is reduced at the bootstrapping procedure. The function f is encoded as *look-up table* (LUT) in a so-called test polynomial $\mathsf{testP} \in \mathscr{R}$ (of degree N), whose coefficients are as follows

$$\underbrace{\underbrace{f(0),\ldots,f(0)}_{N/2t \text{ elements}},\underbrace{f(1),\ldots,f(1)}_{N/t \text{ elements}},\ldots,\underbrace{f(t-1),\ldots,f(t-1)}_{N/t \text{ elements}},\underbrace{f(0),\ldots,f(0)}_{N/2t \text{ elements}}}_{t+1 \text{ chunks}}$$

The goal of bootstrapping procedures is to output the correct element in the above LUT, and it works by first *blindly* rotating the table (since the table is encrypted), and then extract the aimed value from the encrypted rotated table as follows:

- $\mathsf{BlindRotate}$. This algorithm initializes a noiseless RLWE ciphertext, called accumulator and noted as $\mathsf{acc}_0 = (0, x^{-b} \cdot \mathsf{testP})$, and updates the accumulator by computing
$$\mathsf{acc}_i = \mathsf{acc}_{i-1} \odot \left((x^{a_i} - 1) \cdot \mathsf{BK}_i\right) + \mathsf{acc}_{i-1}, \tag{4.1}$$
for $i \in [1, n]$, where the bootstrapping key is a RGSW encryption of the binary secret key $\mathbf{s}$ of the LWE ciphertext $(b, \mathbf{a})$ to be evaluated, i.e., $\mathsf{BK}_i = \mathsf{RGSW}_{sk,Q}(s_i)$. The final accumulator should be $\mathsf{acc}_n \in \mathsf{RLWE}_{sk,Q}(\mathsf{testP} \cdot x^{-b+\langle \mathbf{a},\mathbf{s}\rangle})$.
- $\mathsf{Extract}$. This algorithm extracts a LWE encryption of the constant item of the polynomial $\mathsf{testP} \cdot x^{-b+\langle \mathbf{a},\mathbf{s}\rangle}$ from acc_{n-1}, that is the aimed $\mathsf{LWE}(f(m))$.

PBS is very efficient compared to the second generation FHE, but has limitations:

1. PBS is not SIMD. Only one message can be bootstrapped each time.
2. PBS is only efficient for messages with a small precision (limited to 4-5 bits). For higher precision, the run time increases exponentially with ciphertext modulus bit size, as noted by Liu, Micciancio and Polyakov [17].
3. The univariate function f to be evaluated has to be negacyclic, i.e., $f(x + q/2) = -f(x)$ as noted by Micciancio and Polyakov [9]. Then the most significant bit of the message has to be zero or at least known [10]. More discussion will be given in Section 4.3.1.

4.2.3. BATCH BOOTSTRAPPING

Liu and Wang [14, 16] addressed the above limitation 1 of PBS by providing an efficient batch framework that supports the bootstrapping of multiple ciphertexts, simultaneously. They worked with a general cyclotomic ring $\mathscr{R}$ (instead of power-of-two) and presented the batch homomorphic computation based on the decomposability of the ring $\mathscr{R}$: $\mathscr{R} = \mathscr{R}_1 \otimes \mathscr{R}_2 \otimes \mathscr{R}_3$. The first ring $\mathscr{R}_1$ is the message space, and the other two rings $\mathscr{R}_2, \mathscr{R}_3$ are designed for packing computations.

Notations. For an integer $m = q\rho'\tau'$ for co-prime integers q, ρ', τ', let mth cyclotomic field $K := \mathbb{Q}(\xi_m) \cong \mathbb{Q}(\xi_q) \otimes \mathbb{Q}(\xi_{\rho'}) \otimes \mathbb{Q}(\xi_{\tau'}) := K_1 \otimes K_2 \otimes K_3$. The prime q is equal to the modulus of the LWE ciphertext to be bootstrapped, and ρ', τ' are powers of constant primes: $\rho' = p_1^{d_1}, \tau' = p_2^{d_2}$. Let $\rho = \phi(\rho'), \tau = \phi(\tau')$.

Denote K_{12} and K_{13} as $K_1 \otimes K_2$ and $K_1 \otimes K_3$, respectively. Let $\mathscr{R}, \mathscr{R}_1, \mathscr{R}_2, \mathscr{R}_3, \mathscr{R}_{12}, \mathscr{R}_{13}$ be the rings of integers of fields $K, K_1, K_2, K_3, K_{12}, K_{13}$, respectively. It follows from [18] that

$$\begin{cases} \mathscr{R} = \mathbb{Z}[\xi_m], \mathscr{R}_1 = \mathbb{Z}[\xi_q], \mathscr{R}_2 = \mathbb{Z}[\xi_{\rho'}], \mathscr{R}_3 = \mathbb{Z}[\xi_{\tau'}] \\ \mathscr{R} \cong \mathscr{R}_1 \otimes \mathscr{R}_2 \otimes \mathscr{R}_3, \mathscr{R}_{12} \cong \mathscr{R}_1 \otimes \mathscr{R}_2, \mathscr{R}_{13} \cong \mathscr{R}_1 \otimes \mathscr{R}_3 \\ \rho \text{ and } \tau \text{ are the degrees of the rings } \mathscr{R}_2 \text{ and } \mathscr{R}_3, \text{ respectively.} \end{cases} \tag{4.2}$$

Let $(\mathsf{v}_1, \cdots, \mathsf{v}_\rho)$ and $(\mathsf{w}_1, \cdots, \mathsf{w}_\tau)$ be the $\mathbb{Z}$-bases of $\mathscr{R}_2$ and $\mathscr{R}_3$, respectively. Denote their corresponding $\mathbb{Z}$-bases of the dual space $\mathscr{R}_2^\vee$ and $\mathscr{R}_3^\vee$ as $(\mathsf{v}_1^\vee, \cdots, \mathsf{v}_\rho^\vee)$ and $(\mathsf{w}_1^\vee, \cdots, \mathsf{w}_\tau^\vee)$. Liu and Wang [14] chose short enough basis $\{\mathsf{v}_i\}_{i \in \rho}$ and $\{\mathsf{w}_i\}_{i \in \tau}$ such that

- $\|\mathsf{v}_i\|_\infty = 1$, and $\|\mathsf{w}_i\|_\infty = 1$,
- $\left\|\mathsf{v}_i^\vee\right\|_\infty \leq 2(p_1 - 1)/\rho'$, and $\left\|\mathsf{w}_i^\vee\right\|_\infty \leq 2(p_2 - 1)/\tau'$.

Denote the trace function over Galois extensions $[K : K_{12}]$ and $[K : K_{13}]$ as

$$\mathsf{Tr}_{K/K_{12}} : K \to K_{12} \text{ and } \mathsf{Tr}_{K/K_{13}} : K \to K_{13}.$$

An important conclusion is that $\mathsf{Tr}_{K/K_{12}}(f \cdot g \cdot \mathsf{w}_i \mathsf{w}_j^\vee) = \delta_i^j \cdot f \cdot g$ and $\mathsf{Tr}_{K/K_{13}}(f \cdot \mathsf{v}_i \mathsf{v}_j^\vee \cdot h) = \delta_i^j \cdot f \cdot h$ for any $f \in \mathscr{R}_1, g \in \mathscr{R}_2, h \in \mathscr{R}_3$, where $\delta_i^j = 1$ if $i = j$ or 0 else. In the rest of the work, we use the same notations as above.

RLWE/RGSW Ciphertexts. Liu and Wang [14] presented the RLWE and RGSW encryption schemes as in Section 4.2.2 by moving the setting of algebraic structure from power-of-two to the general cyclotomic ring $\mathscr{R}$. They also described a similar product $\boxdot$ between RGSW ciphertexts, as well as an external product $\boxtimes$ between a RLWE and a RGSW ciphertext, with a refined analysis of the error behavior. For more details, we refer to [14].

Ciphertexts Packing. Based on the above tensor decomposition of rings, Liu and Wang [14] showed how to pack RGSW and RLWE ciphertexts and how to perform

batch homomorphic computation over packed ciphertexts. For r GSW ciphertexts $\{\mathbf{C}_i = \mathsf{RGSW}_s(\mu_i) \in \mathscr{R}^{2\times 2l}\}_{1\le i\le r}$ where $\mu_i \in \mathscr{R}_1$ and $r = min(\rho,\tau)$ the maximal number of packing, the RGSW ciphertexts packing algorithm RGSW-pack has four options of packing models:

- "$\mathscr{R}_{12}$": output $\sum_i \mathbf{C}_i \cdot \mathsf{v}_i \in \mathscr{R}^{2\times 2l}$, that is a RGSW encryption of $\sum_i \mu_i \cdot \mathsf{v}_i$[1];
- "$\mathscr{R}_{13}$": output $\sum_i \mathbf{C}_i \cdot \mathsf{w}_i \in \mathscr{R}^{2\times 2l}$, that is a RGSW encryption of $\sum_i \mu_i \cdot \mathsf{w}_i$;
- "$\mathscr{R}_{12} \to \mathscr{R}_{13}$": output $\sum_i \mathbf{C}_i \cdot \mathsf{v}_i^\vee \mathsf{w}_i \in (\mathscr{R}_1 \otimes \mathscr{R}_2^\vee \otimes \mathscr{R}_3)^{2\times 2l}$, that is a RGSW encryption of $\sum_i \mu_i \cdot \mathsf{v}_i^\vee \mathsf{w}_i$;
- "$\mathscr{R}_{13} \to \mathscr{R}_{12}$": that is $\sum_i \mathbf{C}_i \cdot \mathsf{v}_i \mathsf{w}_i^\vee \in (\mathscr{R}_1 \otimes \mathscr{R}_2 \otimes \mathscr{R}_3^\vee)^{2\times 2l}$, that is a RGSW encryption of $\sum_i \mu_i \cdot \mathsf{v}_i \mathsf{w}_i^\vee$;

The index of the model is also attached in the packing output, i.e. the packing output is $(\mathbf{C}, \mathsf{mode})$. Similarly, there is a RLWE ciphertext packing algorithm RLWE-pack that takes as input r RLWE ciphertexts $\{\mathbf{c}_i = \mathsf{RLWE}_s(\mu_i) \in \mathscr{R}^2\}_{1\le i\le r}$ and the aimed mode mode, and outputs $(\mathbf{c}, \mathsf{mode})$.

Packing Operations. We now describe how to homomorphically perform the batch computation over packed ciphertexts, using a homomorphic evaluation of the trace function in Lemma 4.2.3, denoted as Eval-Tr.

- Add: For two packed RGSW ciphertexts $(\mathbf{C}_x, \mathsf{mode})$, $(\mathbf{C}_y, \mathsf{mode})$ of the message vectors $\mathbf{x} = (x_1, \cdots, x_r) \in \mathscr{R}_1^r$ and $\mathbf{y} = (y_1, \cdots, y_r) \in \mathscr{R}_1^r$, respectively, then $(\mathbf{C}_x + \mathbf{C}_y, \mathsf{mode})$ is a packed RGSW ciphertext of the message vector $\mathbf{x}+\mathbf{y}$ with the same mode. The conclusion is the same for two packed RLWE ciphertexts.
- Batch-Mult: For two packed RGSW ciphertexts $(\mathbf{C}_x, \mathsf{mode}_x)$, $(\mathbf{C}_y, \mathsf{mode}_y)$ of the message vectors $\mathbf{x} \in \mathscr{R}_1^r$ and $\mathbf{y} \in \mathscr{R}_1^r$, respectively, then Batch-Mult outputs a packed RGSW encryption of the message vector $(x_1 y_1, \cdots, x_r y_r)$ in two cases: If mode_x = "$\mathscr{R}_{12}$", mode_y = "$\mathscr{R}_{12} \to \mathscr{R}_{13}$" or vice versa, the output ciphertext is $\text{Eval-Tr}_{K/K_{13}}(\mathbf{C}_x \boxdot \mathbf{C}_y)$ and the output mode is "$\mathscr{R}_{12}$"; If mode_x = "$\mathscr{R}_{13}$", mode_y = "$\mathscr{R}_{13} \to \mathscr{R}_{12}$" or vice versa, the output is $\text{Eval-Tr}_{K/K_{12}}(\mathbf{C}_x \boxdot \mathbf{C}_y)$ with mode "$\mathscr{R}_{12}$". For other cases, output $\perp$.
- Ext-Prod: For a packed RGSW ciphertext $(\mathbf{C}_x, \mathsf{mode}_x)$ of the message vector $\mathbf{x} \in \mathscr{R}_1^r$ and a packed RLWE ciphertext $(\mathbf{c}_y, \mathsf{mode}_y)$ of the message vector $\mathbf{y} \in \mathscr{R}_1^r$, then Ext-Prod outputs a packed RLWE encryption of the message vector $(x_1 y_1, \cdots, x_r y_r)$ in two cases as Batch-Mult, and only the multiplication $\mathbf{C}_x \boxdot \mathbf{C}_y$ in Batch-Mult should be changed to $\mathbf{C}_x \boxtimes \mathbf{c}_y$
- UnPack$_i$: For a packed RGSW ciphertext $(\mathbf{C}_x, \mathsf{mode}_x)$ of the message vector $\mathbf{x} \in \mathscr{R}_1^r$, output $\mathbf{C}_i \in \mathsf{RGSW}(x_i)$ for $1 \le i \le r$. If mode_x = "$\mathscr{R}_{12}$", then output $\text{Eval-Tr}_{K/K_{13}}(\mathbf{C}_x \cdot \mathsf{v}_i^\vee)$. If mode_x = "$\mathscr{R}_{13}$", then output $\text{Eval-Tr}_{K/K_{12}}(\mathbf{C}_x \cdot \mathsf{w}_i^\vee)$. The conclusion is the same for a packed RLWE ciphertext.

[1] We can see that it is necessary to choose short bases as above, to reduce the noise growth caused by multiplication with bases.

Lemma 4.2.3 (Eval-Tr)**.** *Given evaluation key* evk *as input, the homomorphic trace algorithm* Eval-Tr *transforms a RLWE ciphertext* $c = \mathsf{RLWE}_s(\mu) \in (\mathscr{R}_1 \otimes \mathscr{R}_2^v \otimes \mathscr{R}_3)^2$ *of a message* $m \in \mathscr{R}_1 \otimes \mathscr{R}_2^v \otimes \mathscr{R}_3$ *into a RLWE encryption of* $c' \in \mathsf{RLWE}_s(\mathsf{Tr}_{K/K_{13}}(\mu))$ *with error* $\mathsf{Err}(c') = \mathsf{Tr}_{K/K_{13}}(\mathsf{Err}(c)) + e'$ *where* $\|e'\|_\infty$ *is a fresh noise independent with* c *and bounded by a sub-Gaussian with parameter* $3dB$.

Thus, one can check the correctness of Ext-Prod:

$$\begin{aligned}\mathsf{Eval\text{-}Tr}_{K/K_{13}}(\mathbf{C}_x \boxtimes \mathbf{c}_y) &= \mathsf{Eval\text{-}Tr}_{K/K_{13}}\big(\mathsf{RLWE}(\textstyle\sum(x_i \cdot \mathsf{v}_i) \cdot \sum(y_i \cdot \mathsf{v}_i^{\mathsf{V}} \mathsf{w}_i))\big) \\ &= \mathsf{RLWE}\big(\mathsf{Eval\text{-}Tr}_{K/K_{13}}(\textstyle\sum(x_i \cdot \mathsf{v}_i) \cdot \sum(y_i \cdot \mathsf{v}_i^{\mathsf{V}} \mathsf{w}_i))\big) \\ &= \mathsf{RLWE}(\textstyle\sum(x_i y_i \cdot \mathsf{w}_i))\end{aligned}$$

and the same for Batch-Mult.

Batch FHEW-like Bootstrapping. Given r LWE ciphertexts $\{(b_l, \mathbf{a}_l) \in \mathbb{Z}_q^{n+1}\}_{1 \le l \le r}$ under the same key $\mathbf{s}$ to be bootstrapped, the batch bootstrapping introduced by Liu and Wang [14] mainly consists of the following two steps:

- Batch-BR. This batch blind rotation algorithm initializes an accumulator acc_0 as a packed noiseless RLWE ciphertext[2]:

$$\mathsf{acc}_0 = \mathsf{RLWE\text{-}pack}\Big((0, \xi_q^{-b_1}), \cdots, (0, \xi_q^{-b_l}), \text{“}\mathscr{R}''_{12}\Big) = \mathsf{RLWE}\Big(\sum_j \xi_q^{-b_j} \cdot \mathsf{v}_j\Big),$$

  and updates the accumulator by computing

$$\begin{aligned}\mathsf{acc}_i &= \mathsf{Ext\text{-}Prod}\Big(\mathsf{acc}_{i-1}, \sum_j \big((\xi_q^{a_j^i} - 1) \cdot \mathsf{BK}_i + \mathbf{G}\big) \cdot \mathsf{v}_j^{\mathsf{V}} \mathsf{w}_j\Big) \\ &= \mathsf{Ext\text{-}Prod}\Big(\mathsf{acc}_{i-1}, \mathsf{RGSW}\Big(\sum_j \xi_q^{a_j^i \cdot s_j} \cdot \mathsf{v}_j^{\mathsf{V}} \mathsf{w}_j\Big)\Big),\end{aligned}$$

  for odd $i \in [0, n-1]$, and for even i, the item $\mathsf{v}_j^{\mathsf{V}} \mathsf{w}_j$ should be changed to $\mathsf{v}_j \mathsf{w}_j^{\mathsf{V}}$, in order to support a successive external product. The bootstrapping key BK_i is also a RGSW encryption of the binary secret key $\mathbf{s}$ like FHEW-like Bootstrapping, i.e., $\mathsf{BK}_i = \mathsf{RGSW}_{sk,Q}(s_i)$. The final accumulator should be $\mathsf{acc}_n \in \mathsf{RLWE}_{sk,Q}(\sum_j \xi_q^{-b_j + \langle \mathbf{a}_j, \mathbf{s} \rangle} \cdot \mathsf{v}_j) = \mathsf{RLWE}_{sk,Q}(\sum_j \xi_q^{-\frac{q}{2} \cdot \mu_j + e_j} \cdot \mathsf{v}_j)$ [3].

- Batch-Extract. This algorithm constructs a test polynomial testP with coefficients as

$$\underbrace{0, 1, 0, 1 \cdots, 1, 0}_{q \text{ elements}},$$

  and extracts a RLWE encryption $\mathsf{RLWE}(\sum \mu_j \cdot \mathsf{v}_j)$ from $\mathsf{RLWE}_{sk,Q}(\sum_j \mathsf{testP} \cdot \xi_q^{-\frac{q}{2} \cdot \mu_j + e_j} \cdot \mathsf{v}_j)$ by Lemma 4.2.4.

[2] We omit some constants for simplicity of presentation. Moreover, recall that $(1, \xi_q, \cdots, \xi_q^{q-1})$ is a basis of $\mathscr{R}_1$.

[3] We assume n is even without loss of generality.

Lemma 4.2.4. *For any $\zeta = \xi_q^z$, we have $1+\zeta+\zeta^2+\cdots+\zeta^{q-1} = q$ if $z = 0 \bmod q$, or 0 else, where q is a prime and $z \in \mathbb{Z}_q$. Since $\mathsf{Tr}_{K/K_{23}}(\zeta) = \zeta+\zeta^2+\cdots+\zeta^{q-1}$, we furthermore have $1+\mathsf{Tr}_{K/K_{23}}(\zeta) = q$ if $z = 0$, or 0 otherwise.*

The batch FHEW-like bootstrapping [14] overcomes limitation 1 and enables to fresh multiple messages by a single bootstrapping, reducing the cost of amortized cost per message. However, it also suffers from limitation 2 and even worse, since it only supports binary messages. Moreover, it is unclear whether it is *programmable*, meaning that it allows a univariate function to be evaluated during bootstrapping. If so, are there any restrictions on the function, such as negacyclicity, as in limitation 3 for PBS? Our work in Section 4.3 addresses these questions.

4.3. BATCH EVALUATIONS OF ARBITRARY FUNCTION WITHIN A POLYNOMIAL MODULUS

In this section, we overcome the above-mentioned limitations of PBS and Batch Bootstrapping. First, we generalize the batch bootstrapping so that it can evaluate messages in $\mathbb{Z}_t$ instead of only one bit. Furthermore, we proposed a Batch PBS that can evaluate an arbitrary function, not limited to being negacyclic, during bootstrapping, thus overcoming limitation 3 in a batch way. Therefore, our approach does not require the MSB of messages to be zero and is practical when messages are 5-6 bits (compared to 4-5 bits for PBS). As an application, we propose a batch *decomposition* of multiple messages from a single ciphertext into several ciphertexts encrypting small chunks of input messages.

4.3.1. BATCH PBS

Given r LWE ciphertexts $\{(b_j, \mathbf{a}_j) \in \mathsf{LWE}(\mu_j) : b_j = \langle \mathbf{a}_j, \mathbf{s}\rangle + e_j + \lfloor \frac{q}{t} \cdot \mu_j \rceil$ and $\mu_j \in \mathbb{Z}_t\}_{i\in[r]}$ and a function $f : \mathbb{Z}_q \to \mathbb{Z}_q$, our goal is to simultaneously compute ciphertexts $c'_j \in \mathsf{LWE}(f(\mu_j))$ with polynomial noise overhead. We use the same notations as instantiations in Section 4.2.3, and without loss of generality, we assume the dimension of LWE ciphertexts is even.

Recall that the decryption algorithm on a LWE ciphertext $\mathsf{LWE}_{\mathbf{s}}(b, \mathbf{a})$ is as follows:

$$\mathsf{Dec}_{\mathbf{s}}(b, \mathbf{a}) := \left\lfloor \frac{t}{q} \cdot (b - \langle \mathbf{a}, \mathbf{s}\rangle) \right\rceil.$$

Therefore, the homomorphic evaluation of the decryption circuit contains two steps: (1) batch inner product and subtraction, and (2) division and rounding. Our final goal is to compute a LWE encryption of $f(\mu)$, so we need to carefully design a look-up table related to the function f, which is rotated during the batch bootstrapping and output the correct element in the table in the final.

Batch Inner Product and Subtraction. To compute $b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle$ for $j \in [r]$, we adopt

the batch blind rotation in [14]. We compute:

$$\begin{cases} \mathsf{acc}_0 = \mathsf{RLWE\text{-}pack}\left((0,\xi_q^{-b_1}),\cdots,(0,\xi_q^{-b_r}),\text{“}\mathscr{R}''_{12}\right) \\ \mathbf{B}_1 = \mathsf{RGSW\text{-}pack}\left(((\xi_q^{a_1^1}-1)\cdot\mathsf{BK}_1+\mathbf{G}),\cdots,((\xi_q^{a_r^1}-1)\cdot\mathsf{BK}_1+\mathbf{G}),\text{“}\mathscr{R}_{12}\to\mathscr{R}''_{13}\right) \\ \mathbf{B}_2 = \mathsf{RGSW\text{-}pack}\left(((\xi_q^{a_1^2}-1)\cdot\mathsf{BK}_2+\mathbf{G}),\cdots,((\xi_q^{a_r^2}-1)\cdot\mathsf{BK}_2+\mathbf{G}),\text{“}\mathscr{R}_{13}\to\mathscr{R}''_{12}\right) \\ \vdots \\ \mathbf{B}_n = \mathsf{RGSW\text{-}pack}\left(((\xi_q^{a_1^n}-1)\cdot\mathsf{BK}_n+\mathbf{G}),\cdots,((\xi_q^{a_r^n}-1)\cdot\mathsf{BK}_n+\mathbf{G}),\text{“}\mathscr{R}_{12}\to\mathscr{R}''_{13}\right) \end{cases} \tag{4.3}$$

where the bootstrapping key BK_i is a RGSW encryption of the i-th entry of the secret key $\mathbf{s}$. The packing mode is changed between “$\mathscr{R}_{13}\to\mathscr{R}''_{12}$ and “$\mathscr{R}_{12}\to\mathscr{R}''_{13}$ alternatively in order to support the successive external product Ext-prod between packed RLWE and RGSW ciphertexts[4]. We update the accumulator by

$$\mathsf{acc}_i = \mathsf{Ext\text{-}Prod}(\mathsf{acc}_{i-1},\mathbf{B}_i),$$

for $1\le i\le n$. For each i, $(\xi_q^{a_j^i}-1)\cdot\mathsf{BK}_i+\mathbf{G}$ is a RGSW sample of message $\xi_q^{a_j^i\cdot s_i}$, which can be verified by replacing s_i with values 0 and 1. Thus, $\mathbf{B}_i$ is packed RGSW ciphertext of message vector

$$(\xi_q^{a_1^i\cdot s_i},\xi_q^{a_2^i\cdot s_i},\cdots,\xi_q^{a_r^i\cdot s_i}).$$

Since the external products in fact compute the point-wise multiplication, the final accumulator acc_n is a packed RLWE encryption of message vector

$$(\xi_q^{-b_1+\langle a_1,\mathbf{s}\rangle},\xi_q^{-b_2+\langle a_2,\mathbf{s}\rangle},\cdots,\xi_q^{-b_r+\langle a_r,\mathbf{s}\rangle}),$$

i.e., $\mathsf{acc}_n\in\mathsf{RLWE}\left(\sum_j\xi_q^{-b_j+\langle a_j,\mathbf{s}\rangle}\cdot\mathsf{w}_j\right)$.

Look-up Table. We embed the division and rounding process in decryption circuit, as well as the the target function f, into the following look-up table

$$\underbrace{f\left(\left\lfloor\frac{t}{q}\cdot 0\right\rceil\right),f\left(\left\lfloor\frac{t}{q}\cdot 1\right\rceil\right),f\left(\left\lfloor\frac{t}{q}\cdot 2\right\rceil\right),\cdots,f\left(\left\lfloor\frac{t}{q}\cdot(q-1)\right\rceil\right)}_{q\text{ elements}}, \tag{4.4}$$

by setting the so-called test polynomial $\mathsf{testP}=\sum_{y\in\mathbb{Z}_q}f\left(\left\lfloor\frac{t}{q}\cdot y\right\rceil\right)\cdot\xi_q^y$. Multiplying testP with acc_n, we have a packed RLWE encryption of

$$\sum_j\sum_{y\in\mathbb{Z}_q}\left[f\left(\left\lfloor\frac{t}{q}\cdot y\right\rceil\right)\cdot\xi_q^y\cdot\xi_q^{-b_j+\langle a_j,\mathbf{s}\rangle}\right]\cdot\mathsf{w}_j.$$

[4]Recall Ext-prod only supports two ciphertexts with modes (“$\mathscr{R}''_{12}$, “$\mathscr{R}_{12}\to\mathscr{R}''_{13}$), or (“$\mathscr{R}''_{13}$, “$\mathscr{R}_{13}\to\mathscr{R}''_{12}$)

For each $j \in [r]$, the constant item of $\sum_{y\in\mathbb{Z}_q}\left[f\left(\left\lfloor\frac{t}{q}\cdot y\right\rceil\right)\cdot\xi_q^y\cdot\xi_q^{-b_j+\langle a_j,\mathbf{s}\rangle}\right]$ is $f\left(\left\lfloor\frac{t}{q}\cdot(b_j-\langle a_j,\mathbf{s}\rangle)\right\rceil\right)$, which is exactly equal to the aimed $f(\mu_i)$. We extract this constant item by the homomorphic trace algorithm Eval-Tr as on line 7 in Algorithm 1.

Enhancement. Directly multiplying testP with acc_n will increase the noise in the resulting ciphertext by a factor of $\|f\|_\infty$. Therefore, we instead set $\mathsf{acc}_0 \leftarrow \mathsf{testP}\cdot\mathsf{acc}_0$, which does not affect the correctness but leads to an error in acc_n independent of f, following from the asymmetric noise growth property of the external product Ext-Prod between the packed RLWE ciphertext acc_0 and the RGSW ciphertext $\mathbf{B}_1$. We now present our batch PBS in Algorithm 1 and its analysis in Theorem 4.3.1.

Algorithm 1: Batch PBS

Input: An arbitrary non-constant function $f:\mathbb{Z}_q\to\mathbb{Z}_q$,
r LWE samples $(b_i,\mathbf{a}_i)\in\mathsf{LWE}_{\mathbf{s}}(\mu_i)$ with the same secret key $\mathbf{s}$ for $i\in[r]$,
Bootstrapping key: $\{\mathsf{BK}_i\in\mathsf{RGSW}^Q_{s'}(s_i)\}$ of the secret key $\mathbf{s}=(s_1,\cdots,s_n)$,
Evaluation key evk for the homomorphic trace Eval-Tr (see Lemma 4.2.3)

Output: A packed RLWE ciphertext of message vector $(f(\mu_1),\cdots,f(\mu_r))$

1 Let $\mathsf{testP}=\sum_{y\in\mathbb{Z}_q}f\left(\left\lfloor\frac{t}{q}\cdot y\right\rceil\right)\cdot\xi_q^y$ and $\mathbf{B}_1,\ldots,\mathbf{B}_n$ be as defined in Eqs. (4.4) and (4.3), respectively.
2 Let $\mathsf{acc}_0=\mathsf{testP}\cdot\mathsf{RLWE\text{-}pack}\left((t^{-1}\xi_q^{-b_1},0),\cdots,(t^{-1}\xi_q^{-b_r},0),\text{“}\mathscr{R}''_{12}\right)$
3 **for** $k=1$ ***to*** n **do**
4 $\quad\mathsf{acc}_k\leftarrow\mathsf{Ext\text{-}Prod}(\mathsf{acc}_{k-1},\mathbf{B}_k)$
5 **end**
6 Set $\mathbf{d}=\left(\sum_{j\in[r]}t^{-1}\cdot\left[\sum_{y\in\mathbb{Z}_q}f\left(\left\lfloor\frac{ty}{q}\right\rceil\right)\cdot\mathsf{w}_j\right],0\right)$
7 **return** $\mathbf{c}=\mathbf{d}+\mathsf{Eval\text{-}Tr}_{K/K_{23}}(\mathsf{acc}_n)$

Theorem 4.3.1. *Let $(b_i,\mathbf{a}_i)$ be the LWE encryption of message $\mu_i\in\mathbb{Z}_t$ for $i\in[r]$. Then Algorithm 1 outputs a fresh packed encryption $\mathbf{c}$ in $\mathsf{RLWE}_{Q,s'}(\sum f(\mu_i)\cdot\mathsf{w}_i)$ Moreover, $\|\mathsf{Err}(\mathbf{c})\|_\infty$ is bounded by a sub-Gaussian variable with parameter $O(\gamma)$ with $\gamma\le rnN\sqrt{N\log Q}E$, where E is the upper bound of errors in all Bootstrapping and evaluation keys, and the amortized complexity per input ciphertext is $\tilde{O}(n^{0.75})$ external products.*

Proof. Correctness. We first analyze the correctness of our batch programmable bootstrapping algorithm. On line 2, we initialize the accumulator to be a trivial (noiseless) RLWE encryption of $\mathsf{testP}\cdot\sum_{j\in[r]}q^{-1}\cdot\xi_q^{b_j}\cdot\mathsf{v}_j$ [5]. By the correctness of Ext-Prod, we know that the final accumulator Acc_n on line 5 is a packed RLWE encryption of $\mathsf{testP}\cdot\sum_{j\in[r]}q^{-1}\cdot\xi_q^{b_j-\langle\mathbf{a}_j,\mathbf{s}\rangle}\cdot\mathsf{w}_j$. Note that, due to corollary 4.2.2,

[5] Note that the RLWE ciphertext $(t^{-1}m,0)=(q/t\cdot q^{-1}m,0)$, therefore $(t^{-1}m,0)$ is a noiseless encryption of $q^{-1}m$, due to the scaling factor q/t in the encryption. The same applies to $\mathbf{d}$ on line 6 of the algorithm.

testP $\not\equiv 0$ under our assumption that f is not a constant function. Then we compute the output ciphertext in the last line by substituting $\mathbf{d}$, test, and Acc_n:

$$\begin{aligned}
\mathbf{c} &= \mathbf{d} + \mathsf{Eval\text{-}Tr}(\mathsf{testP} \cdot \mathsf{Acc}_n) \\
&= \mathsf{RLWE}\left(\sum_{j\in[r]} q^{-1} \cdot \left[\sum_{y\in\mathbb{Z}_q} f(\lfloor ty/q \rceil)\right] \cdot \mathsf{w}_j\right) + \mathsf{RLWE}\left(\mathsf{Tr}(\mathsf{testP} \cdot \sum_{j\in[r]} q^{-1}\xi_q^{b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle} \cdot \mathsf{w}_j)\right) \\
&= \mathsf{RLWE}\left(\sum_{j\in[r]} q^{-1} \cdot \left[\sum_{y\in\mathbb{Z}_q} f(\lfloor ty/q \rceil) + \mathsf{Tr}(\mathsf{testP} \cdot \xi_q^{b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle})\right] \cdot \mathsf{w}_j\right) \\
&= \mathsf{RLWE}\left(\sum_{j\in[r]} q^{-1} \cdot \left[\sum_{y\in\mathbb{Z}_q} \left(f(\lfloor ty/q \rceil) + f(\lfloor ty/q \rceil) \cdot \mathsf{Tr}(\sum_{y\in\mathbb{Z}_q} \xi_q^{-y} \cdot \xi_q^{b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle})\right)\right] \cdot \mathsf{w}_j\right) \\
&= \mathsf{RLWE}\left(\sum_{j\in[r]} q^{-1} \cdot \left[\sum_{y\in\mathbb{Z}_q} f(\lfloor ty/q \rceil) \cdot (1 + \mathsf{Tr}(\xi_q^{-y} \cdot \xi_q^{b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle}))\right] \cdot \mathsf{w}_j\right), \qquad (4.5)
\end{aligned}$$

where the second equality holds due to the correctness of Eval-Tr. By Lemma 4.2.4, we have $1 + \mathsf{Tr}(\xi_q^{-y} \cdot \xi_q^{b_i - \langle \mathbf{a}_i, \mathbf{s}\rangle}) = q$ if $b_i - \langle \mathbf{a}_i, \mathbf{s}\rangle = y$ for some y, or 0 otherwise. Therefore, we know the polynomial factor in the above underlying messages is

$$\sum_{y\in\mathbb{Z}_q} f(\lfloor ty/q \rceil) \cdot (1 + \mathsf{Tr}(\xi_q^{-y} \cdot \xi_q^{b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle})) = f(\lfloor t/q \cdot (b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle) \rceil) \cdot q. \qquad (4.6)$$

Substituting it into Eq. (4.5), we finally get

$$\begin{aligned}
\mathbf{c} &= \mathsf{RLWE}\left(\sum_{j\in[r]} q^{-1} \cdot f(\lfloor t/q \cdot (b_j - \langle \mathbf{a}_j, \mathbf{s}\rangle) \rceil) \cdot q \cdot \mathsf{w}_j\right) \\
&= \mathsf{RLWE}\left(\sum_{j\in[r]} q^{-1} \cdot f(\mu_i) \cdot q \cdot \mathsf{w}_j\right) \\
&= \mathsf{RLWE}\left(\sum_{j\in[r]} f(\mu_i) \cdot \mathsf{w}_j\right). \qquad (4.7)
\end{aligned}$$

Noise Overhead. We begin by proving, by induction on k, that the accumulator exhibits the same noise growth as that in the batch bootstrapping of [14]. For $k = 0$, this is obvious since Acc_0 in both works are error-less ciphertexts. For each $k \geq 1$, we note from the definition that the Ext-Prod in the for loop calls the external product of RGSW encryption of $s_k \cdot (-a_{ik})$ and Acc_{k-1}, i.e., $\mathsf{Acc}_k = \mathsf{Eval\text{-}Tr}(\mathsf{BK}_k \cdot \mathbf{g}^{-1}(\mathsf{Acc}_{k-1}))$ by definition.

Let e_k be the error of Acc_k, m_k the message of BK_k, and e'_k the noise from key-switching procedure in Eval-Tr. By Lemma 4.2.3, we have

$$\begin{aligned}
e_k &= e'_k + \mathsf{Tr}(\mathsf{BK}_k \cdot \mathbf{g}^{-1}(\mathsf{Acc}_{k-1})) \\
&= e'_k + \mathsf{Tr}(\mathsf{Err}(\mathsf{BK}_k) \cdot \mathbf{g}^{-1}(\mathsf{Acc}_{k-1})) + \mathsf{Tr}(m_k \cdot e_{k-1}). \qquad (4.8)
\end{aligned}$$

The key-switching error e'_k is independent of the message of BK_k and Acc_{k-1}. In addition, by Fact 12 in FHEW [6], the error of $\mathsf{Err}(\mathsf{BK}_k)\cdot\mathbf{g}^{-1}(\mathsf{Acc}_{k-1})$ is a subgaussian of parameter $O(\mathsf{Err}(\mathsf{BK}_k)\cdot B_g\sqrt{Nd_g l})$ where l is the number of bootstrapping keys. Obviously, this error is also independent of the message of Acc_{k-1}, which is the main difference between [14] and our work. Then, we conclude that Acc_n has the same noise growth as that in [14], which is bounded by a subgaussian with parameter less than $O(nr^3\sqrt{N\log Q}E)$ by the proof of Theorem 6.2 in [14].

In the last step of Algorithm 1, the Eval-Tr procedure is applied, which increases the error by q at most. So we conclude that the final error is bounded by a subgaussian with parameter less than $O(nr^3q\sqrt{N\log Q}E) = O(rnN\sqrt{N\log Q}E)$ due to the parameter setting $N = r^2q$. Let $\beta = rnN\sqrt{N\log Q}E$ for simplicity.

Amortized Cost. We compare the efficiency of batch evaluation of arbitrary functions with the batch technique in [14]. Notice that bootstrapping r ciphertexts in both works requires n Batch-Upd and 1 homomorphic trace evaluation, resulting in the same amortized cost. That is, $\widetilde{O}(\lambda^{0.75})$ external products per input ciphertext when we use the same parameter setting as [14]: security parameter λ, $n = O(\lambda), q = \widetilde{O}(\sqrt{n})$, $N = O(n)$, and $r \approx \sqrt{N/q} = O(\lambda^{1/4-o(1)})$. □

Remark 4.3.2. *Compared Batch PBS to PBS, it shall be noticed that*

(1) negacyclicity. PBS works with the power-of-two cyclotomic ring $\mathbb{Z}[x]/(x^N+1)$, in which $1+x^N = 0$. Therefore, we have

$$\left(x^{\frac{q}{p}\cdot m+e} + x^{\frac{q}{p}\cdot(m+\frac{p}{2})+e}\right)\cdot\mathsf{testP} = 0, \tag{4.9}$$

for any message m since $q = 2N$. Recall that the extraction process in PBS extracts the constant term of the polynomial $x^{\frac{q}{p}\cdot m+e}\cdot\mathsf{testP}$, which equals $f(m)$. Thus, the constant term in Eq. (4.9) is $f(m)+f\left(m+\frac{p}{2}\right)$. Consequently, we have $f(m)+f\left(m+\frac{p}{2}\right) = 0$ as a restriction on the function to bootstrap.

Arbitrary. However, in our setting as $\mathscr{R}_1 = \mathbb{Z}[\xi_q]$, we know from Lemma 4.2.1 that ξ_q^i, ξ_q^j are independent for $i \neq j$. Therefore we conclude that $\mathsf{testP}\cdot\xi^{\frac{q}{t}\cdot\mu_i+e}$ and $\mathsf{testP}\cdot\xi^{\frac{q}{t}\cdot\mu_j+e}$ are independent for two different messages μ_i, μ_j, unless $\mathsf{testP} = 0$. That means $f(\mu_i)$ and $f(\mu_j)$ are independent for $\mu_i \neq \mu_j$. Therefore, the target function in batch PBS can be arbitrary. Recall $\mathsf{testP} = \sum_{y\in\mathbb{Z}_q} f\left(\left\lfloor\frac{t}{q}\cdot y\right\rceil\right)\cdot\xi_q^y$, the only restriction is that f cannot be a constant function, i.e., $f \not\equiv c$ for some constant c; otherwise, $\mathsf{testP} \equiv 0$ by Corollary 4.2.2.

Thus, we solve Limitation 3. On one hand, it enables our scheme to be efficient over one bit more than the precision of PBS, up to 6 bits. On the other hand, we will show its advantage in application in Section 4.4. To evaluate a complex function by PBS, one needs to carefully design negacyclic functions, and normally requires multiple invocations of PBS. However, it is quite direct

and efficient by using batch PBS.

(2) complexity. For Limitation 1, we note that our batch PBS reduces the amortized cost per ciphertext from $O(n)$ to $O(n^{0.75})$.

We write $\mathsf{Batch\text{-}Boot}[f](\mathsf{ct}_1, \mathsf{ct}_2, \cdots, \mathsf{ct}_r)$ as the result of batch-bootstrapping a given integer function f over ciphertexts $(\mathsf{ct}_1, \mathsf{ct}_2, \cdots, \mathsf{ct}_r)$. The output model of the Batch-Boot can also be "$\mathscr{R}_{13}$" if we pack those error less RLWE ciphertext on line 2 with model "$\mathscr{R}_{13}$". We use $\mathsf{Batch\text{-}Boot}[f](\mathbf{c}, \mathsf{model})$ to emphasize the output with model.

For a constant function $f : \mathbb{Z}_q \to \mathbb{Z}_q$, we know $\mathsf{testP} = 0$ by Lemma 4.2.1, which leads to an encryption of 0 if we directly use Batch-Boot to evaluate f over any ciphertexts. Instead, we can evaluate f by adding the evaluation of functions f_1, and f_2: $\mathsf{Batch\text{-}Boot}[f_1] + \mathsf{Batch\text{-}Boot}[f_2]$, where $f_1(x) = f(x)$ except at $x = 0$ and $f_2(x) = f(x)$ only when $x = 0$.

4.3.2. APPLICATIONS

We can now efficiently batch-evaluate activation functions in CNNs using Batch-Boot from Algorithm 1. Activation functions like Sign, ReLU, and max are not polynomials. Typically, these functions are approximated by polynomials, which can degrade accuracy when applying FHE to privacy-preserving CNNs. We demonstrate how to exactly evaluate them, particularly in a batch manner.

Batch-HomSign. Let us formally define this problem. Given LWE encryptions of messages $\{\mu_i\}$ as input, the goal of Batch-HomSign is to obtain an RLWE encryption of $\{\sum_i \mathsf{Sign}(m_i) \cdot \mathsf{w}_i\}$. The key observation here is that Sign is an integer function,

$$\mathsf{Sign} : \mathbb{Z}_t \to \{1, 0\},$$

which has a value 1 if $m_i \geq 1$ or 0 otherwise. Therefore, we can directly apply the result from Section 4.3.1, yielding $\mathsf{Batch\text{-}HomSign} = \mathsf{Batch\text{-}Boot}[\mathsf{sign}]$.

Batch-HomReLU. Taking LWE encryptions $\mathbf{c} = (\mathbf{c}_i)_{i \in [r]}$ of messages $\{m_i\}$ as input, Batch-HomReLU aims to produce an RLWE encryption of $\sum_i \mathsf{ReLU}(m_i) \cdot \mathsf{w}_i$ where v_i is the basis of $\mathscr{R}_2$. The ReLU function $\mathsf{ReLU}(x)$ is defined as:

$$\mathsf{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{else.} \end{cases}$$

Since ReLU is also an integer function over the message space, we can directly utilize Algorithm 1 to evaluate the ReLU function, thus $\mathsf{Batch\text{-}ReLU} = \mathsf{Batch\text{-}Boot}[\mathsf{ReLU}]$.

Batch-HomMax. Given two sequences of LWE encryptions $\mathbf{c}_0 = (\mathbf{c}_{0i})_{i \in [r]}$ and $\mathbf{c}_1 = (\mathbf{c}_{1i})_{i \in [r]}$ of messages $\{m_{0,i}\}$ and $\{m_{1,i}\}$, respectively, the objective of

Batch-HomMax is to obtain an RLWE encryption of $\sum_i \mathsf{max}(m_{0,i}, m_{1,i}) \cdot \mathsf{w}_i$. The max function can be decomposed as follows:

$$\mathsf{max}(x, y) = \frac{x+y}{2} + \frac{|x-y|}{2}.$$

To evaluate the max function, we homomorphically compute each part of this equation separately, i.e., evaluating $f_1(\mu) = \mu/2$ over $\mathbf{c}_0 + \mathbf{c}_1$ and $f_2(\mu) = |\mu|/2$ over $\mathbf{c}_0 - \mathbf{c}_1$, and then sum the evaluations. Therefore, we have:

$$\mathsf{Batch\text{-}HomMax}(\mathbf{c}_0, \mathbf{c}_1) = \mathsf{Batch\text{-}Boot}[f_1](\mathbf{c}_0 + \mathbf{c}_1) + \mathsf{Batch\text{-}Boot}[f_2](\mathbf{c}_0 - \mathbf{c}_1).$$

The correctness of this approach can be easily verified.

4.4. LARGE PRECISION

In this section, we present approaches to batch homomorphically evaluate the sign functions over large-precision ciphertexts. The challenge is that due to the requirement that the modulo of LWE ciphertext to bootstrap should be equal to the order of polynomial ring $\mathcal{R}_1$, the computation complexity increases linearly with the ciphertext modulus, similar to Limitation 2 for PBS.

We reduce the runtime complexity to be logarithmical with the ciphertext modulus, thus supporting large precision in practice. Our techniques adopt the idea of Liu, Micciancio and Polyakov in [17] but incorporate two improvements as discussed later. We also generalize this procedure to the evaluation of ReLU/Max over large-precision.

4.4.1. DECOMPOSITION AND REMOVAL

First, let us formally define this problem and its solution by PBS in [17]. Given a LWE ciphertext $(b, \mathbf{a})$ with modulus Q larger than the modulus q supported by PBS, we want to obtain an LWE encryption $\mathsf{LWE.Enc}(\mathsf{sign}(\mu)) \in \mathbb{Z}_Q^{n+1}$. Recall that PBS takes a LWE ciphertext $(b, \mathbf{a})$ with modulus q and a negacyclic function f as input, and outputs a ciphertext $\mathsf{LWE.Enc}(f(\mu))$ with modulus Q and fresh error β. We denote $\mathsf{Boot}[f](b, \mathbf{a})$ as the procedure of PBS.

PBS Technique [17]. To evaluate the sign function is to find the most significant bit (MSB) of an encrypted message. Over large precision, Liu et al. [17] develop a homomorphic floor function that clears the least significant digits (LSB) from a ciphertext by using PBS, as shown in Fig. 4.2, followed by modulus switching that reduces the ciphertext modulus Q but keeps MSB unchanged. Iteratively repeating this procedure until the modulus is smaller than q, then the sign function can be evaluated by a primal FHEW-like bootstrapping.

- HomFloor. As shown in Fig. 4.2, let c_1 be the input LWE ciphertext of message μ with modulus Q. Assume error $e_1 < \beta \leq \alpha/4$. The ciphertext $c_2 := c_1 \bmod q$ computes an encryption of the $\log q - \log \alpha$ least significant bits of μ, denoted

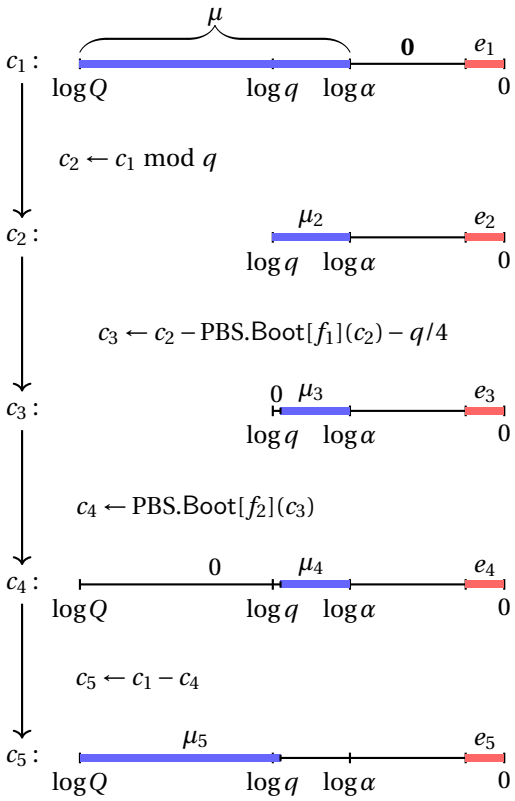


Figure 4.2: Workflow of HomFloor algorithm, where the blue-marked and red-marked bits represent messages and errors, respectively, the unmarked bits are 0, and f_1, f_2 is defined in Eq. (4.10).

as μ_2. Since c_2 has the modulus q, we can evaluate a negacyclic function f_1 by PBS over it, resulting in a ciphertext with modulus Q.

$$f_1(x) = \begin{cases} -q/4 & \text{if } 0 \le x < q/2 \\ q/4 & \text{else} \end{cases} \quad \text{and} \quad f_2(x) = \begin{cases} x & \text{if } x < q/2 \\ q/2 - x & \text{else} \end{cases} \tag{4.10}$$

Let $c_3 = c_2 - \text{PBS.Boot}[f_1](c_2) - q/4 \bmod q$, we get a LWE encryption of μ_3 which clears the most significant of μ_2. This can be verified by checking the two possible values for MSB of μ_2, 0 and 1. Since $\mu_3 < q/2$, the evaluation of the negacyclic function f_2 over ct_3 outputs a LWE ciphertext ct_4 of message $\mu_4 = \mu_3$, since $f_2(x)$ is the identity function over $x \in [0, q/2)$. Then subtracting ct_4 from the original ciphertext ct_1 results in an encryption of μ_5 which equals to μ_1 except the $\log(q/\alpha)$ least significant bits are 0.

For simplicity of presentation, we omit the additions with constants in the procedure and the maximal error generated in it is bounded by 2β, which

is smaller than $\alpha/2$ based on the assumption that $\beta \leq \alpha/4$, guaranteeing the validity of those ciphertexts.

- **Modulus Switching and Iteration**. Denote the $\log(Q/q)$ MSB of μ_5 as μ_5' and $ct_5 = (b_5, \mathbf{a}_5)$ with secret key $\mathbf{s}$. Since

$$b - \langle \mathbf{a}, \mathbf{s} \rangle = \alpha \cdot \mu + e_5 = q \cdot \mu_5' + e_5,$$

the ciphertext ct_5 can be viewed as a valid encryption of μ_5', as $e_5 < \alpha/2 < q/2$. Using Modulus Switching in Lemma 4.4.1, we can switch ct_5 to a smaller modulus $Q/(q/\alpha)$, obtaining an encryption of the message μ_5' with a scaling factor of $\frac{Q/(q/\alpha)}{Q/q} = \alpha$. After $\lceil \frac{\log Q - \log q}{\log(q/\alpha)} \rceil$ iterations, the modulus Q will reduce to at most q, enabling the direct computation of the message's sign using PBS.

4

Lemma 4.4.1. *[6] For any $\mathbf{s} \in \mathbb{Z}_q^n$, $\mu \in \mathbb{Z}_t$ and ciphertext $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s},Q}(m)$ with subgaussian error of parameter σ, there exists an efficient modulus switching algorithm, which outputs a $\mathsf{LWE}_{\mathbf{s},q}(m)$ ciphertext with subgaussian error of parameter $\sqrt{(q\sigma/Q)^2 + 2\pi(\|\mathbf{s}\|^2 + 1)}$.*

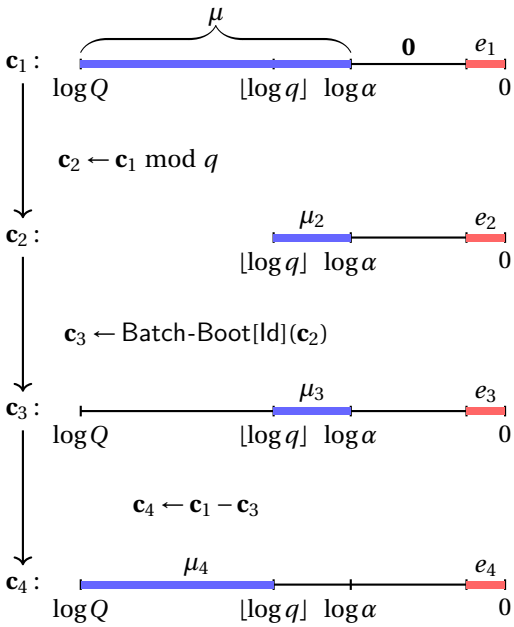


Figure 4.3: Workflow of Batch-HomFloor algorithm

Batch PBS Technique. We propose a batch homomorphic floor function in Fig 4.3 that only requires one invocation of bootstrapping per chunk, compared to the best performance of [17] requiring two bootstrapping as shown in Fig 4.2, and moreover

enables us to deal with multiple ciphertexts at the same time, while [17] only support one ciphertext at a time by comparison. This benefits from the property that our batch bootstrapping procedure can evaluate arbitrary functions even limited to negacyclic ones.

- Batch-HomFloor. As shown in Fig 4.3, denote the input as r LWE ciphertexts $\mathbf{c}_1 := \{(b_i, \mathbf{a}_i) \in \mathbb{Z}_Q^{n+1}\}_{i\in[r]}$ which encrypts a message μ_i with modulus Q respectively. We make the same assumption as in [17] that $e < \beta \leq \alpha/4$ where e is the error for input LWE ciphertexts and β is the fresh noise generated after the batch bootstrapping. The following operation $\mathbf{c}_1 \bmod q$ computes a ciphertext set $\mathbf{c}_2$ that encrypts values μ_i modulo q. We then evaluate the identity function Id over $\mathbf{c}_2$,

$$\mathsf{Id}(x) = x \text{ for } x \in \mathbb{Z}_q, \tag{4.11}$$

  resulting in a packed ciphertext $\mathbf{c}_3$ encrypting the last $\lfloor \log q \rfloor$ least significant bits of $(\mu_1, \cdots, \mu_r)$. Then the ciphertext $\mathbf{c}_3$ is unpacked into r LWE ciphertexts by UnPack and subtracted from the input LWE ciphertexts; thus we get the aimed r LWE ciphertexts. We provide our Batch-HomFloor procedure in Algorithm 2 and its analysis in Lemma 4.4.2.

- **Modulus Switching and Iteration**. We iteratively use the Batch-HomFloor algorithm as a subroutine to clear the LSBs of messages and switch the ciphertext modulus to a smaller one until smaller than q. Then the sign function can be evaluated directly by Batch-Sign in Section 4.3.2. The full procedure is shown in 3 Algorithm with analysis in Lemma 4.4.3.

For simplicity of presentation, given a message μ, we define $fl(\mu)$ the floored message of μ as:

$$fl(\mu) = \left\lfloor \frac{\alpha}{q} \cdot \mu \right\rceil \cdot \frac{q}{\alpha}, \tag{4.12}$$

This operation scales the message μ by the factor $\frac{\alpha}{q}$, rounds the result to the nearest integer, and then scales it back by multiplying by $\frac{q}{\alpha}$. The outcome is a version of μ where the $\log(q) - \log(\alpha)$ least significant bits are set to zero.

Lemma 4.4.2. *Algorithm 2 outputs LWE ciphertexts of messages* $\{fl(\mu_1), \cdots, fl(\mu_r)\}$ *with the error bounded by a Gaussian with parameter* 2β*, under the same assumption as [17] that the error in each LWE ciphertext* $e_i < \beta < \alpha/4$.

Proof. Recall that a LWE can be decrypted correctly only if its error is smaller than $Q/(2q) = \alpha/2$ for message space $\mathbb{Z}_q$ and ciphertext modulus $\mathbb{Z}_Q$. For each i, denote $(b_i, \mathbf{a}_i) = (b_i, \langle \mathbf{a}_i, s\rangle + \alpha\mu_i + e_i)$ for some error $|e_i| < \beta$. Let $\mu_i = ms(\mu_i) + ls(\mu_i)$ for which $ls(\mu_i)$ is $\lfloor \log(q) \rfloor - \log(\alpha)$ LSBs of μ_i and $ms(\mu_i) = 2^{\lfloor \log(q) \rfloor - \log(\alpha)} \cdot \mu'$ for some μ' is the remaining MSBs of μ_i. Then we have $\alpha\mu_i + e_i = \alpha \cdot ls(\mu_i) + e_i \bmod q$, and therefore, $(c_i, \mathbf{d}_i)$ in line 1 is valid ciphertext of $ls(\mu_i)$ with error e_i since $|e_i| < \beta < \alpha/4$ by our assumption.

Algorithm 2: Batch-HomFloor($\mathbf{c}$)

Input: Identity function $\mathsf{Id}: \mathbb{Z}_q \to \mathbb{Z}_q$,
r LWE ciphertexts $\mathbf{c} = \{(b_i, \mathbf{a}_i) \in \mathbb{Z}_Q^{n+1}\}_{i\in[r]}$ of messages μ_i, resp.
Bootstrapping and evaluation keys for Batch-Boot.

Output: LWE ciphertexts of message $\{fl(\mu_i)\}_{i\in[r]}$, as defined in Eq. (4.12).

1 $(c_i, \mathbf{d}_i) \leftarrow (b_i, \mathbf{a}_i) \bmod 2^{\lfloor \log(q) \rfloor}$ for $i \in [r]$
2 $(c_i, \mathbf{d}_i) \leftarrow \mathsf{Batch\text{-}Boot}^i_{unpk}[\mathsf{Id}]\{(c_i, \mathbf{d}_i)\}$
3 $(e_i, \mathbf{f}_i) \leftarrow (b_i, \mathbf{a}_i) - (c_i, \mathbf{d}_i)$ for $i \in [r]$
4 **return** $(e_i, \mathbf{f}_i)$

In line 2, evaluating the identity function over $\{(c_i, \mathbf{d}_i)\}$ gives a packed RLWE encryption (c, d) of message vectors $\{ls(\mu_1), ls(\mu_2), \cdots, ls(\mu_r)\}$ with modular Q and noise β. For each $i \in [r]$, we then unpack (c, d) to get LWE encryption of μ_i under the original secret key s. The whole process is $\mathsf{Batch\text{-}Boot}^i_{unpk}$ in line 2. We know each result $(c_i, \mathbf{d}_i)$ has the same noise level as (c, d), i.e., subguassian with parameter β.

Subtracting $(c_i, \mathbf{d}_i)$ from $(a_i, \mathbf{b}_i)$ in line 3, we finally get a LWE encryption of $ms(\mu_i)$ with error bounded by

$$e_\beta + \beta < 2\beta < \alpha/2,$$

by our assumption, which is a valid ciphertext. □

Algorithm 3: Batch-HomSign$_Q$($\mathbf{c}$)

Input: r LWE ciphertexts $\mathbf{c} = (\mathbf{c}_i \in \mathbb{Z}_Q^{n+1})_{i\in[r]} = \big((b_i, \mathbf{a}_i)\big)$ of messages μ_i, respectively.

Output: A packed RLWE ciphertext of message vector $(\mathsf{Sign}(\mu_i))$

1 **if** $Q > q$ **then**
2 $\quad (c_i, \mathbf{d}_i) \leftarrow \mathsf{Batch\text{-}HomFloor}(Q, \mathbf{c})$
3 $\quad (b_i, \mathbf{a}_i) \leftarrow \lceil \frac{\alpha}{2^{\lfloor \log q \rfloor}} \cdot (c_i, \mathbf{d}_i) \rfloor$
4 $\quad Q \leftarrow \alpha Q / 2^{\lfloor \log q \rfloor}$
5 $\quad \mathbf{c} \leftarrow \big((b_i, \mathbf{a}_i)\big)$
6 **end**
7 $(b_i, \mathbf{a}_i) \leftarrow (q/Q) \cdot (b_i, \mathbf{a}_i)$
8 $(c, d) \leftarrow \mathsf{Batch\text{-}HomSign}((b_i, \mathbf{a}_i)) \pmod Q$
9 **return** (c, d)

Lemma 4.4.3. *For LWE ciphertexts* $\{\mathbf{c}_i\}$ *with modulus* Q, Batch-HomSign$_Q$ *outputs a packed RLWE ciphertext of message vector* $\{\mathsf{sign}(\mu_i)\}$ *after* $\lfloor \frac{\log Q}{\log(q/\alpha)} \rfloor + 1$ *calls to* Batch-Boot.

Proof. By Lemma 4.4.2, Batch-HomFloor in line 2 computes an encryption of message vector $(fl(\mu_1), \cdots, fl(\mu_r))$ with error bounded by 2β. Lines 3-4 switch the

ciphertext modulus from Q to $\alpha Q/q$ and returns LWE encryptions of the same message vector $(fl(\mu_1),\cdots,fl(\mu_r)) \in \mathbb{Z}^r_{Q/q}$, with the error bounded by

$$(\alpha/q)2\beta + (\beta+1)/2 < \beta < \alpha/4,$$

where the first inequality holds since $\beta > 2$ and $q > 4\alpha$.

We can repeat this procedure until $Q < q$ and switch the module back to q in line 7. Then the sign can be batched evaluated directly by Batch-HomSign in line 8 to return a packed RLWE encryption of message vector $\{\mathsf{Sign}(\mu_1),\cdots,\mathsf{Sign}(\mu_r)\}$. □

4.4.2. DECOMPOSITION AND RECONSTRUCTION

In this section, we extend the idea of decomposition and removal in Sect. 4.4.1 to the evaluation of arbitrary linear functions over Q/α. The goal is the following. Given LWE ciphertexts $(c_i, \mathbf{d}_i)$ encrypting message $\mu_i \in \mathbb{Z}_{Q/\alpha}$ and an arbitrary linear function $f : \mathbb{Z}_{Q/\alpha} \to \mathbb{Z}_{Q/\alpha}$, we should compute a RLWE ciphertext encrypting $\sum_i f(\mu_i)\cdot \mathsf{v}_i$.

Note that for each message $\mu_i \in Q/\alpha$, the following digit decomposition of μ_i holds:

$$\mu_i = (-1)^{\mathsf{MSB}(\mu)} \cdot \sum_j \mu_{ij} \cdot (2^{\lfloor \log q \rfloor / \log \alpha})^j. \tag{4.13}$$

Our high-level idea is to decompose messages $\{\mu_i\}_{i\in[r]}$ into digits $\{\mu_{ij}\}$ which are less than q and then batch evaluate $\{\mu_{ij}\}_{i\in[r]}$ with a new function $\tilde{f}_j(x) := f\big(x \cdot (2^{\lfloor \log q \rfloor / \log \alpha})^j\big)$, followed by the reconstruction step that adds the evaluations of $\tilde{f}_j$ together. The full algorithm is shown in Alg. 4.

Even though we have already shown in Section 4.4.1 how to evaluate the MSB over large precisions, here we assume the MSB of each message μ_i is 0 for two reasons: (1) one single bit is negligible compared to a large precision, (2) the evaluation of each digit returns a packed RLWE ciphertext. In order to reconstruct the evaluation over input messages, we should multiply the evaluation of MSB with the evaluation of other digits. Then a scheme switching technique in [23] should be adopted to transfer RLWE ciphertexts into equivalent RGSW ones and then perform the external product between RGSW and RLWE ciphertexts, since there does not exist direct multiplication between two RLWE ciphertexts. It obviously requires relatively expensive operations. Then the correctness of Alg. 4 follows from

$$\begin{aligned}\sum_j \tilde{f}_j(\mu_{ij}) &= \sum_j f\big(\mu_{ij} \cdot (2^{\lfloor \log q \rfloor / \log \alpha})^j\big) \\ &= f\big(\sum_j \mu_{ij} \cdot (2^{\lfloor \log q \rfloor / \log \alpha})^j\big) \\ &= f(\mu_i)\end{aligned}$$

where the second equality holds because f is linear and the homomorphic extraction of each digit μ_{ij} is the same to the second step in Fig. 4.2 and Fig. 4.3.

Lemma 4.4.4. *For LWE ciphertexts* $(\mathbf{c}_i)$ *with modulus* Q, $\mathsf{Batch\text{-}HomSign}_Q$ *outputs a packed RLWE ciphertext of message vector* (μ_i) *after* $\lfloor \frac{\log Q}{\log(q/\alpha)} \rfloor + 2$ *calls to* Batch-Boot.

Algorithm 4: $\mathsf{Batch\text{-}Boot}_Q[f](\mathbf{c})$

Input: Linear function $f:\mathbb{Z}_q\to\mathbb{Z}_q$,
r LWE ciphertexts $\mathbf{c}=(\mathbf{c}_i\in\mathbb{Z}_Q^{n+1})_{i\in[r]}=\big((b_i,\mathbf{a}_i)\big)$ of messages μ_i, respectively,
Boostrapping and evaluation keys for Batch-Boot.
Evaluation functions $\tilde{f}_j:\mu\to f(\mu\cdot(2^{\lfloor\log q\rfloor/a})^j)$

Output: A packed RLWE ciphertext of message vector $\{f(\mu_i)\}$

1 $j\leftarrow 0,(c,d)\leftarrow(0,0)$
2 **if** $Q>q$ **then**
3 $\quad(c_i,\mathbf{d}_i)\leftarrow(b_i,\mathbf{a}_i) \bmod 2^{\lfloor\log(q)\rfloor}$ for $i\in[r]$
4 $\quad(c,d)\leftarrow(c,d)+\mathsf{Batch\text{-}Boot}[\tilde{f}_j]((c_i,\mathbf{d}_i))$
5 $\quad(e_i,\mathbf{f}_i)\leftarrow\mathsf{Batch\text{-}HomFloor}(Q,\mathbf{c})$
6 $\quad(b_i,\mathbf{a}_i)\leftarrow\lceil\frac{\alpha}{2^{\lfloor\log q\rfloor}}\cdot(e_i,\mathbf{f}_i)\rfloor$
7 $\quad Q\leftarrow\alpha Q/2^{\lfloor\log q\rfloor}$
8 $\quad\mathbf{c}\leftarrow\{(b_i,\mathbf{a}_i)\}$
9 $\quad j\leftarrow j+1$
10 **end**
11 $\mathbf{c}\leftarrow(q/Q)\cdot\mathbf{c}$
12 $(l,m)\leftarrow\mathsf{Batch\text{-}Boot}[\tilde{f}_j](\mathbf{c})\pmod Q$
13 $(c,d)\leftarrow(c,d)+(l,m)$
14 **return** (c,d)

Proof. We use the same notation as Eq. (4.13). At j-th iteration, line 3 by Lemma 4.4.2 computes LWE encryptions of the message $(\mu_{1j},\mu_{2j},\cdots,\mu_{rj})$ with noise bounded by a sub-gaussian with parameter β. Line 4 first calls the Batch-Boot to compute an encryption of $(\tilde{f}_j(\mu_{1j}),\tilde{f}_j(\mu_{2j}),\cdots,\tilde{f}_j(\mu_{rj}))$, which are accumulated to (c,d), an encryption of $\sum_{k=0}^{j}\mu_{ik}\cdot(2^{\lfloor\log q\rfloor/\log a})^k$. Similar to Alg. 3, lines 5-7 return ciphertexts of messages that are equal to $(\mu_1,\cdots,\mu_r)$ except clearing the j-th $\lfloor\log q\rfloor-\log(\alpha)$ bits.

After the if loop, the length of remain message is smaller than q and switched to q in line 11. The most significant bits are evaluated in line 12, which are then added to (c,d), resulting in the evaluation of f over all digits.

Regarding the error, the error of (c,d) in line 14 is accumulated by $N_1=\lfloor\frac{\log Q}{\log(q/\alpha)}\rfloor+1$ callings to Batch-Boot in the if loop and 1 calling to Batch-Boot in line 12, which therefore results in subgaussian error with parameter $(N_1+1)\cdot\beta$ in the final output in line 13. That is

$$(\lfloor\frac{\log Q}{\log(q/\alpha)}\rfloor+2)\cdot\beta, \tag{4.14}$$

Then the output is a valid ciphertext only if the error in Eq. 4.14 is assumed to be smaller than $\alpha/2$. □

4.5. CONCLUSION

In this paper, we propose a novel batch bootstrapping technique for FHE, enabling the homomorphic evaluation of arbitrary functions over n ciphertexts within a polynomial modulus, using $\widetilde{O}(n^{0.75})$ FHE multiplications. As an application,

we demonstrate how to evaluate common activation functions in CNNs, further extending the capability to batch processing. An interesting avenue for future work is to design a novel bootstrapping method that achieves the same goal with an optimal amortized cost of $\widetilde{O}(1)$ FHE multiplications.

REFERENCES

[1] C. Gentry. "Fully homomorphic encryption using ideal lattices". In: *ACM STOC 2009*. Ed. by M. Mitzenmacher. ACM, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[2] Z. Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP". In: *CRYPTO 2012*. Ed. by R. Safavi-Naini and R. Canetti. Vol. 7417. LNCS. Springer, 2012, pp. 868–886. DOI: 10.1007/978-3-642-32009-5_50.

[3] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *ACM Trans. Comput. Theory* 6.3 (2014), 13:1–13:36. DOI: 10.1145/2633600.

[4] Z. Brakerski, C. Gentry, and S. Halevi. "Packed Ciphertexts in LWE-Based Homomorphic Encryption". In: *PKC 2013*. Ed. by K. Kurosawa and G. Hanaoka. Vol. 7778. LNCS. Springer, 2013, pp. 1–13. DOI: 10.1007/978-3-642-36362-7_1.

[5] J. Fan and F. Vercauteren. "Somewhat Practical Fully Homomorphic Encryption". In: *IACR Cryptol. ePrint Arch.* (2012), p. 144. URL: http://eprint.iacr.org/2012/144.

[6] L. Ducas and D. Micciancio. "FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second". In: *EUROCRYPT 2015, Part I*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 617–640. DOI: 10.1007/978-3-662-46800-5_24.

[7] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. "Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE". In: *ASIACRYPT 2017, Part I*. Ed. by T. Takagi and T. Peyrin. Vol. 10624. LNCS. Springer, 2017, pp. 377–408. DOI: 10.1007/978-3-319-70694-8_14.

[8] C. Gentry, A. Sahai, and B. Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5.

[9] D. Micciancio and Y. Polyakov. "Bootstrapping in FHEW-like Cryptosystems". In: *WAHC 2021*. WAHC@ACM, 2021, pp. 17–28. DOI: 10.1145/3474366.3486924.

[10] I. Chillotti, D. Ligier, J. Orfila, and S. Tap. "Improved Programmable Bootstrapping with Larger Precision and Efficient Arithmetic Circuits for TFHE". In: *ASIACRYPT 2021, Part III*. Ed. by M. Tibouchi and H. Wang. Vol. 13092. LNCS. Springer, 2021, pp. 670–699. DOI: 10.1007/978-3-030-92078-4_23.

[11] I. Chillotti, M. Joye, and P. Paillier. “Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks”. In: *CSCML 2021*. Ed. by S. Dolev, O. Margalit, B. Pinkas, and A. A. Schwarzmann. Vol. 12716. LNCS. Springer, 2021, pp. 1–19. DOI: 10.1007/978-3-030-78086-9_1.

[12] D. Micciancio and J. Sorrell. “Ring Packing and Amortized FHEW Bootstrapping”. In: *ICALP 2018*. Ed. by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 100:1–100:14. DOI: 10.4230/LIPICS.ICALP.2018.100.

[13] A. Guimarães, H. V. L. Pereira, and B. V. Leeuwen. “Amortized Bootstrapping Revisited: Simpler, Asymptotically-Faster, Implemented”. In: *ASIACRYPT 2023, Part VI*. Ed. by J. Guo and R. Steinfeld. Vol. 14443. LNCS. Springer, 2023, pp. 3–35. DOI: 10.1007/978-981-99-8736-8_1.

[14] F. Liu and H. Wang. “Batch Bootstrapping I: - A New Framework for SIMD Bootstrapping in Polynomial Modulus”. In: *EUROCRYPT 2023, Part III*. Ed. by C. Hazay and M. Stam. Vol. 14006. LNCS. Springer, 2023, pp. 321–352. DOI: 10.1007/978-3-031-30620-4_11.

[15] Z. Liu and Y. Wang. “Amortized Functional Bootstrapping in less than 7ms, with $\tilde{O}(1)$ polynomial multiplications”. In: *ASIACRYPT 2023, Part III*. Ed. by J. Guo and R. Steinfeld. Vol. 14443. LNCS. Springer, 2023, pp. 353–384. DOI: 10.1007/978-981-99-8736-8_1.

[16] F. Liu and H. Wang. “Batch Bootstrapping II:” in: *EUROCRYPT 2023, Part III*. Ed. by C. Hazay and M. Stam. Vol. 14006. LNCS. Springer, 2023, pp. 353–384. DOI: 10.1007/978-3-031-30620-4_12.

[17] Z. Liu, D. Micciancio, and Y. Polyakov. “Large-Precision Homomorphic Sign Evaluation Using FHEW/TFHE Bootstrapping”. In: *ASIACRYPT 2022, Part II*. Ed. by S. Agrawal and D. Lin. Vol. 13792. LNCS. Springer, 2022, pp. 130–160. DOI: 10.1007/978-3-031-22966-4_5.

[18] A. C. David. *Galois Theory*. Hoboken, NJ: John Wiley & Sons, 2004. ISBN: 978-0-521-46713-1.

[19] V. Lyubashevsky, C. Peikert, and O. Regev. “A Toolkit for Ring-LWE Cryptography”. In: *EUROCRYPT 2013*. Ed. by T. Johansson and P. Q. Nguyen. Vol. 7881. LNCS. Springer, 2013, pp. 35–54. DOI: 10.1007/978-3-642-38348-9_3.

[20] K. Conrad. “Cyclotomic Extensions”. In: (2013). URL: https://kconrad.math.uconn.edu/blurbs/galoistheory/cyclotomic.pdf.

[21] J. Alperin-Sheriff and C. Peikert. “Faster Bootstrapping with Polynomial Error”. In: *CRYPTO 2014, Part I*. Ed. by J. A. Garay and R. Gennaro. Vol. 8616. LNCS. Springer, 2014, pp. 297–314. DOI: 10.1007/978-3-662-44371-2_17.

[22] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. “Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds”. In: *ASIACRYPT 2016, Part I*. Ed. by J. H. Cheon and T. Takagi. Vol. 10031. LNCS. 2016, pp. 3–33. DOI: 10.1007/978-3-662-53887-6_1.

[23] G. D. Micheli, D. Kim, D. Micciancio, and A. Suhl. “Faster Amortized FHEW Bootstrapping Using Ring Automorphisms”. In: *PKC 2024, Part IV*. Ed. by Q. Tang and V. Teague. Vol. 14604. LNCS. Springer, 2024, pp. 322–353. DOI: 10.1007/978-3-031-57728-4_11.

5

VOLUME AND ACCESS PATTERN LEAKAGE-ABUSE SE ATTACK

Searchable Encryption schemes provide secure search over encrypted databases while allowing admitted information leakages. Generally, the leakages can be categorized into `access` *and* `volume pattern`*. In most existing SE schemes, these leakages are caused by practical designs but are considered an acceptable price to achieve high search efficiency. Recent attacks have shown that such leakages could be easily exploited to retrieve the underlying keywords for search queries. Under the umbrella of attacking SE, we design a new Volume and Access Pattern Leakage-Abuse Attack (VAL-Attack) that improves the matching technique of LEAP (CCS '21) and exploits both the* `access and volume patterns`*. Our proposed attack only leverages leaked documents and the keywords present in those documents as auxiliary knowledge and can effectively retrieve document and keyword matches from leaked data. Furthermore, the recovery performs without false positives. We further compare VAL-Attack with two recent well-defined attacks on several real-world datasets to highlight the effectiveness of our attack and present the performance under popular countermeasures.*

This chapter is based on the paper "VAL: Volume and Access Pattern Leakage-Abuse Attack with Leaked Documents" by Lambregts, S., **Chen, H.**, Ning, J. and Liang, K. in ESORICS (1) 2022: 653-676

5.1. INTRODUCTION

In practice, to protect data security and user privacy (e.g., under GDPR), data owners may choose to encrypt their data before outsourcing to a third-party cloud service provider. Encrypting the data enhances privacy and gives the owners the feeling that their data is stored safely. However, this encryption relatively restricts the searching ability. Song et al. [1] proposed a Searchable Encryption (SE) scheme to preserve the search functionality over outsourced and encrypted data. In the scheme, the keywords of files are encrypted, and when a client wants to query a keyword, it encrypts the keyword as a token and sends it to the server. The server then searches the files with the token corresponding to the query, and afterwards, it returns the matching files. Since the seminal SE scheme, many research works have been presented in the literature, with symmetrical [2–5] and asymmetrical encryption [6–9]. Nowadays, SE schemes have been deployed in many real-world applications such as ShadowCrypt [10] and Mimesis Aegis [11].

Leakage. In an SE scheme, an operational interaction is usually defined as a client sending a query to the server and the server responding to the query with the matching files. Nevertheless, this interaction could be eavesdropped on by an attacker. The messages could be intercepted because they are sent over an unprotected channel, or the attacker is the cloud service provider itself, who stores and accesses all the search requests and responses. The attacker may choose to match the query with a keyword such that he can comprehend what information is present on the server. The query and response here are what we may call *leakage*. In this work, we consider two main types of *leakage patterns*: the `access pattern`, the response from the server to a query, and the `search pattern`, which is the frequency a query is sent to the server. Besides these types, we also consider the `volume pattern` as leakage. This pattern is seen as the size of the stored documents on the server. The leakage patterns can be divided into four levels, by Cash et al. [12]. In this work, we consider our leakage level to be L2, which equals the fully-revealed occurrence pattern, together with the volume pattern to create a new attack on the SE scheme. Note that a formal definition of the leakages is given in Section 5.3.1.

Attacks on SE. There exist various attacks on SE that work and perform differently. Most of these attacks take the leaked files as auxiliary knowledge. Islam et al. [13] presented the foundation for several attacks on SE schemes. They stated that, with sufficient auxiliary knowledge, one could create a co-occurrence matrix for both the leakage and the knowledge so that it can easily map queries to the keywords based on the lowest distance. Cash et al. [12] later proposed an attack where the query can be matched to a particular keyword based on the total occurrence in the leaked files. These attacks with knowledge about some documents are known as *passive attacks with pre-knowledge*. Blackstone et al. [14] developed a $\text{Subgraph}_{\text{VL}}$ attack that provides a relatively high query recovery rate even with a small subset of the leaked documents. The attack matches keywords based on unique document volumes as if it is the response pattern. Ning et al. [15]

later designed the LEAP attack. LEAP combines the existing techniques, such as co-occurrence and the unique number of occurrences, to match the leaked files to server files and the known keywords to queries based on unique occurrences in the matched files. It makes good use of the unique count from the Count attack [12], a co-occurrence matrix from the IKK attack [13] (although LEAP inverts it to a document co-occurrence matrix) and finally, unique patterns to match keywords and files. Note that we give related work and general comparison in Section 5.6.

Limitations. The works in [12–15] explain their leakage-abusing methods, but they only abuse a single leakage pattern, while multiple are leaked in SE schemes. Besides the leakage patterns, the state-of-the-art LEAP attack abuses the `access pattern` but does not exploit its matching techniques to the full extent. In addition to extending their attack, a combination of leakage can be used to match more documents and queries.

We aim to address the issue of matching keywords by exploiting both the `access pattern` and `volume pattern`. The following question arises naturally:

Could we match queries and documents in a passive attack by exploiting the volume and access patterns to capture a high recovery rate against popular defences?

Contributions. We answer the above research question by designing an attack that matches leaked files and keywords. Our attack expands the matching techniques from the LEAP attack [15] and exploits the `volume pattern` to match more documents. The attack improves the LEAP attack by fully exploring the leakage information and combining the uniqueness of document volume to match more files. These matches can then be used to extract keyword matches. All the matches found are correct, as we argue that false positives are not valuable in real-world attacks.

- Besides exploiting the `access pattern`, we also abuse `volume pattern` leakage. We match documents based on a unique combination of volume and number of keywords with both leakage patterns. We can match almost all leaked documents to server documents using this approach.
- We match keywords using their occurrence pattern in matched files.
- Besides matching keywords in matched files, we use all leaked documents for unique keyword occurrence, expanding the keyword matching technique from the LEAP attack. We do this to get the maximum amount of keyword matches from the unique occurrence pattern.
- We run our attack against three different datasets to test the performance, where we see that the results are outstanding as we match almost all leaked documents and a considerable amount of leaked keywords. Finally, we compare our attack to the existing state-of-the-art LEAP and $\text{Subgraph}_{\text{VL}}$ attacks.

Our attack performs great in revealing files and underlying keywords. In particular, it surpasses the LEAP attack, revealing significantly more leaked files and keywords. VAL-Attack recovers almost 98% of the known files and above 93% of the keyword matches available to the attacker once the leakage percentage reaches 5%. When 10% of the Enron database is leaked, which is 3,010 files with 4,962 keywords, we match 2,950 files and 4,909 queries, respectively, corresponding to 98% and 99%. VAL-Attack can still compromise encrypted information, e.g., over 90% recovery (with 10% leakage) under volume hiding in Enron and Lucene, even under several popular countermeasures. We note that our proposed attack is vulnerable to a combination of padding and volume hiding.

5.2. PRELIMINARIES

5.2.1. SEARCHABLE ENCRYPTION

In a general SE scheme, a user encrypts her data and uploads the encrypted data to a server. After uploading the data, the user can send a query containing an encrypted keyword to the server, and the server will then respond with the corresponding data. We assume the server is honest-but-curious, meaning that it will follow the protocol but will try to retrieve as much information as possible.

The scheme. At a high level, an SE scheme consists of three polynomial-time algorithms: ENC, QUERYGEN and SEARCH [5, 16–19]. Definition 5.2.1 shows the scheme in more detail. The client runs the algorithm ENC and encrypts the plaintext documents and the corresponding keywords before uploading them to the server. ENC outputs an encrypted database EDB, which is sent to the server. QUERYGEN, run by the user, requires a keyword and outputs a query token that can be sent to the server. The function SEARCH is a deterministic algorithm that is executed by the server. A query q is sent to the server; the server takes the encrypted database EDB and returns the corresponding identifiers of the files $EDB(q)$. After it has retrieved the file identifiers, the user has to do another interaction with the server to retrieve the actual files.

Definition 5.2.1. *A searchable encryption scheme includes three algorithms {Enc, QueryGen, Search} that operate as follows:*

- $Enc(K,F)$: *the encryption algorithm takes a master key K and a document set $F = \{F_1,...,F_n\}$ as input and outputs the encrypted database $EDB := \{Enc_k(F_1),...,Enc_K(F_n)\}$;*
- $QueryGen(w)$: *the query generation algorithm takes a keyword w as input and outputs a query token q.*
- $Search(q,EDB)$: *the search algorithm takes a query q and the encrypted database EDB as input and outputs a subset of the encrypted database EDB, whose plaintext contains the keyword corresponding to the query q.*

Leakage. A query and the server response are considered the `access pattern`. The documents passed over the channel have their volume; this information is considered the `volume pattern`. In Section 5.3.1, we will explain the leakage in more detail.

5.2.2. NOTATION

In the VAL-Attack, we have m' keywords (w) and m queries (q), and n' leaked-documents and n server documents, denoted as d_i and ed_i, respectively; for a single document, similarly for w_i and q_i. Note w_i may not be the underlying keyword for query q_i, equal for d_i and ed_i. The notations are given in Table 5.1.

Table 5.1: Notation Summary

F	Plaintext document set, $F = \{d_1, ..., d_n\}$	F'	Leaked document set, $F' = \{d_1, ..., d_{n'}\}$
E	Server document set, $E = \{ed_1, ..., ed_n\}$	W	Keyword universe, $W = \{w_1, ..., w_m\}$
W'	Leaked keyword set, $W' = \{w_1, ..., w_{m'}\}$	Q	Query set, $Q = \{q_1, ..., q_m\}$
A	$m' \times n'$ matrix of leaked documents	B	$m \times n$ matrix of server documents
M'	$n' \times n'$ co-occurrence matrix of F'	M	$n \times n$ co-occurrence matrix of E
v_i	Volume (bit size) of document i	$\lvert d_i \rvert$	Number of keywords in document i
C	Set of matched documents	R	Set of matched queries

5.3. MODELS

In an ideal situation, there is no information leaked from the encrypted database, the queries sent, or the database setup. Unfortunately, such a scheme is not practical in real life as it costs substantial performance overheads [20]. The attacker and the leakage are two concerns in SE schemes, and we will discuss them both in the following sections, as they can vary in different aspects.

5.3.1. LEAKAGE MODEL

Leakage is what we define as information that is (unintentionally) shared with the outer world. In our model, the attacker can intercept everything sent from and to the server. The attacker can intercept a query that a user sends to the server and the response from the server. It then knows which document identifiers correspond to which query. This *query → document identifier* response is what we call the `access pattern`. The leakage is defined as [14]:

Definition 5.3.1 (`access pattern`)**.** *The function access pattern (AP)* $= (AP_{k,t})_{k,t\in\mathbb{N}}$ *:* $F(k) \times W^t(k) \to [2^{[n]}]^t$*, such that* $AP_{k,t}(D, w_1, ..., w_t) = D(w_1), ..., D(w_t)$.

As discussed earlier, we assume the leakage level is L2 [12], where the attacker does not know the frequency or the position of the queried keywords in the document response.

The `volume pattern` is leakage that tells the size of the document. It is relevant to all response leaking encryption schemes [3, 5, 16, 21–23] and ORAM-based SE schemes [24]. The leakage is defined formally as follows [14]:

Definition 5.3.2 (`volume pattern`)**.** *The function volume pattern (Vol)* $= (Vol_{k,t})_{k,t\in\mathbb{N}} : F(k) \times W^t(k) \to \mathbb{N}^t$, *such that* $Vol_{k,t}(D, w_1, ..., w_n) = ((|d|_w)_{d\in D(w_1)}, ..., (|d|_w)_{d\in D(w_n)})$, *where* $|\cdot|_w$ *represents the volume in bytes.*

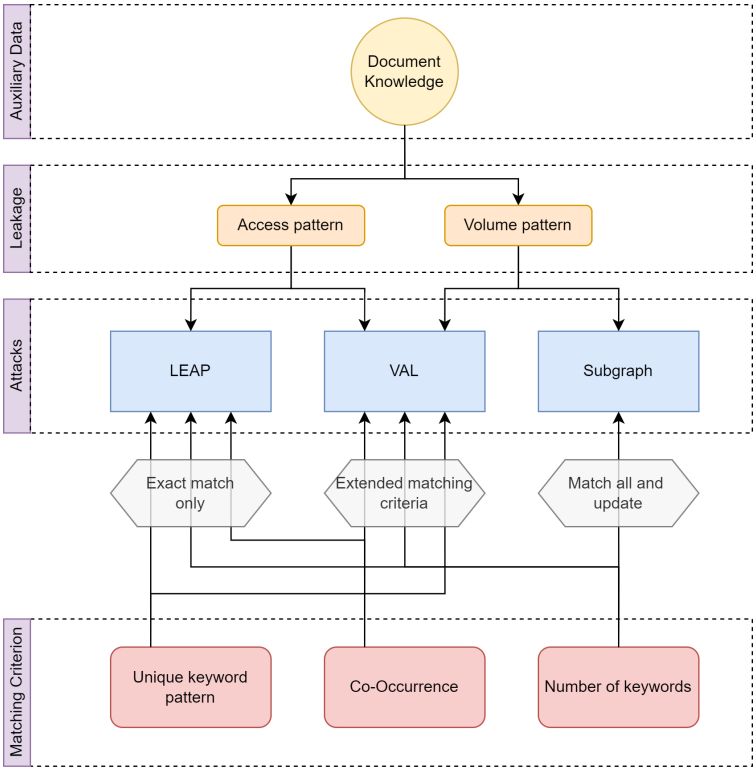


Figure 5.1: Technical Framework of Existing Attacks

5.3.2. ATTACK MODEL

The attacker in SE schemes can be a malicious server that stores encrypted data. Since the server is honest-but-curious [14], it will follow the encryption protocol but wants to learn as much as possible. Therefore, the attacker is passive but still eager to learn about the content present on the server. Our attacker has access to some

leaked plaintext documents, keeps track of the `access and volume pattern` and tries to reveal the underlying server data. Fig. 5.1 shows a visualization of our attack model. We assume that the attacker has access to all the queries and responses used in the SE scheme. This number of queries is realistic because if one waits long enough, all the queries and results will eventually be sent over the user-server channel. The technical framework delineates the LEAP, $\text{Subgraph}_{\text{VL}}$ and our designed attack.

The attacker in our model has access to some unencrypted files stored on the server. This access can be feasible because of a security breach at the setup phase of the scheme, where the adversary can access the revealed files. Another scenario is if a user wants to transfer all of his e-mails from his unencrypted mail storage to an SE storage server. The server can now access all the original mail files, but new documents will come as new e-mails arrive. Therefore, the adversary has partial knowledge about the encrypted data present on the server. The attacker has no access to any existing query to keyword matches and only knows the keywords present in the leaked files. With this information, the attacker wants to match as many encrypted document identifiers to leaked documents and queries to keywords such that he can understand what content is stored on the server.

The passive attacker is less potent than an active attacker, who can upload documents, with chosen keywords, to the server to match queries to keywords [25]. Furthermore, the attacker has no access to the encryption or decryption oracle. Because the attacker relies on the `access and volume pattern` countermeasures that hide these patterns will reduce the attack performance.

5.4. THE PROPOSED ATTACK

5.4.1. MAIN IDEA

At a high level, our attack is built from the LEAP attack [15] by elevating the keyword matching metric to increase the number of keyword matches. Furthermore, each document is labelled with its document volume and number of keywords, and VAL-attack matches using the uniqueness of this label, improving the recovery rate. We first extend the matching technique from LEAP. The approach does not consist of only checking within the matched documents but also keeping track of the occurrence in the unmatched files. This method results in more recovered keywords for the improvement of LEAP that provides a way to match rows that do not uniquely occur in the matched files. We expand the attack by exploiting the `volume pattern` since the document size is also leaked from response leaking encryption schemes, as described in Section 5.3.1. We can extend the comprehensive attack by matching documents based on the `volume pattern`. Our new attack fully explores the leakage information and matches almost all leaked documents. We increase the keyword matches with the maximal file matches to provide excellent performance.

5.4.2. LEAKED KNOWLEDGE

The server stores all the documents in the scheme. There are a total of n plaintext files denoted as the set $F = \{d_1, ..., d_n\}$, with in total m keywords, denoted as the set $W = \{w_1, ..., w_m\}$. We assume the attacker can access:

- The total number of leaked files (i.e. plaintext files) is n' with in total m' keywords. Suppose $F' = \{d_1, ..., d_{n'}\}$ is the set of documents known to the attacker and $W' = \{w_1, ..., w_{m'}\}$ is the corresponding set of keywords that are contained in F'. Note that $n' \leq n$ and $m' \leq m$.
- The set of encrypted files, denoted as, $E = \{ed_1, ..., ed_n\}$ and corresponding query tokens, $Q = \{q_1, ..., q_m\}$ with underlying keyword set W.
- The volume of each server observed document or leaked file is denoted as v_x for document d_x or server document ed_x. The number of keywords or tokens is represented as the size of the document $|d_x|$ or $|ed_x|$ for the same documents, respectively.

The attacker can construct an $m' \times n'$ binary matrix A, representing the leaked documents and their corresponding keywords. $A[d_x][w_y] = 1$ iff. keyword w_y occurs in document d_x. The dot product of A is denoted as the symmetric $n' \times n'$ matrix M', whose entry is the number of keywords that are contained in both document d_x and document d_y. We give an example of the matrices with known documents in Fig. 5.2.

After observing the server's files and query tokens, the attacker can construct an $m \times n$ binary matrix B, representing the encrypted files and related query tokens. $B[ed_x][q_y] = 1$ iff. query q_y retrieved document ed_x. The dot product of B is denoted as the symmetric $n \times n$ matrix M, whose entry is the number of query tokens that retrieve files ed_x and ed_y from the server. We give an example of the matrices with observed encrypted documents in Fig. 5.3.

$$A = \begin{array}{c} \\ w_1 \\ w_2 \\ \vdots \\ w_{m'} \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & \cdots & d_{n'} \end{array} \\ \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{pmatrix} \end{array} \qquad M' = \begin{array}{c} \\ d_1 \\ d_2 \\ \vdots \\ d_{n'} \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & \cdots & d_{n'} \end{array} \\ \begin{pmatrix} 5 & 2 & \cdots & 3 \\ 2 & 6 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 3 & 0 & \cdots & 10 \end{pmatrix} \end{array}$$

Figure 5.2: Matrix A and M' Example

$$B = \begin{array}{c} \\ q_1 \\ q_2 \\ \vdots \\ q_m \end{array} \begin{array}{c} \begin{array}{cccc} ed_1 & ed_2 & \cdots & ed_n \end{array} \\ \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix} \end{array} \qquad M = \begin{array}{c} \\ ed_1 \\ ed_2 \\ \vdots \\ ed_n \end{array} \begin{array}{c} \begin{array}{cccc} ed_1 & ed_2 & \cdots & ed_n \end{array} \\ \begin{pmatrix} 4 & 3 & \cdots & 1 \\ 3 & 9 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \cdots & 9 \end{pmatrix} \end{array}$$

Figure 5.3: Matrix B and M Example

5.4.3. OUR DESIGN

The basis of the attack is to recursively find row and column mappings between the two created matrices, A and B, where a row mapping represents the underlying keyword of a query sent to the server, and a column mapping indicates the match between a server document identifier and a leaked plaintext file. Note that each

$$B_c \quad ed_3 \; ed_2 \; \cdots \; ed_t \qquad A_c \quad d_1 \; d_2 \; \cdots \; d_t$$
$$\begin{matrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_m \end{matrix} \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 0 \end{pmatrix} \qquad \begin{matrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{matrix} \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}$$

Figure 5.4: Matrix A_c and B_c Example

$$B_r \quad ed_1 \; ed_2 \; \cdots \; ed_n \qquad A_r \quad d_1 \; d_2 \; \cdots \; d_{n'}$$
$$\begin{matrix} q_3 \\ q_5 \\ q_2 \\ \vdots \\ q_t \end{matrix} \begin{pmatrix} 0 & 0 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix} \qquad \begin{matrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_t \end{matrix} \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}$$

Figure 5.5: Matrix A_r and B_r Example

$$B_r \quad ed_1 \; ed_2 \; \cdots \; ed_n \qquad A_r \quad d_1 \; d_2 \; \cdots \; d_{n'}$$
$$\begin{matrix} q_3 \\ q_5 \\ q_2 \\ \vdots \\ q_t \end{matrix} \begin{pmatrix} 0 & 0 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix} \qquad \begin{matrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_t \end{matrix} \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}$$

Figure 5.6: An Example of Extended Matrix A

leaked document is still present on the server, meaning that $n' \leq n$ and there is a matching column in B for each column in A. Similarly to the rows, each known keyword corresponds to a query, so $m' \leq m$ as we could know all the keywords, but we do not know for sure. In theory, there is a correct row mapping for each row in A to a row in B. The goal of the VAL-Attack is to find as many correct mappings as possible.

We divide the process of finding as many matches as possible into several steps. The first step is to prepare the matrices for the rest of the process. The algorithm then maps columns based on unique column-sum, as they used in the Count attack [12], but instead of using it on keywords, we try to match documents here. Another step is matching documents based on unique volume and the number of keywords or tokens. As this combination can be a unique pattern, we can match many documents in this step. The matrices M and M' are used to match documents based on co-occurrence. Eventually, we can pair keywords on unique occurrences in the matched documents when several documents are matched. This technique is used in the Count attack [12], but we 'simulate' our own 100% knowledge here. With the matched keywords, we can find more documents, as these will give unique rows in matrices A and B that can be matched. We will introduce these functions in detail in the following paragraphs.

Initialization. First, we initialize the algorithm by creating two empty dictionaries, to which we eventually add the correct matches. We create one dictionary for documents and the other for the matched keywords, C (for column) and R (for row). Next, as we want to find unique rows in the matrices A and B, we must extend matrix A. It could be possible that not all underlying keywords are known

beforehand, in which case $n' < n$, and we have to extend matrix A to find equal columns. Therefore we extend matrix A to an $m \times n'$ matrix that has the first m' rows equal to the original matrix A and the following $m - m'$ rows of all 0s. See Fig. 5.6 for an example. The set $\{w_{m'+1}, ..., w_m\}$ represents the keywords that do not appear in the leaked document set F'.

Number of keywords. Now that the number of rows in A and B are equal, we can find unique column-sums to match documents. This unique sum indicates that a document has a unique number of keywords and can thus be matched based on this unique factor. Similar to the technique in the Count attack [12], we sum the columns, here representing the keywords in A and B. The unique columns in B can be matched to columns in A, as they have to be unique in A as well. If a column$_j$-sum of B is unique and column$_{j'}$-sum of A exists, we can match documents ed_j and $d_{j'}$ because they have the same unique number of keywords.

Volume and keyword pattern. The next step is matching documents based on volume and keyword pattern. If there is a server document ed_j with a unique combination of volume v_j and number of tokens $|ed_j|$ and there is a document $d_{j'}$ with the same combination, we can match document ed_j to $d_{j'}$. However, if multiple server documents have the same pattern, we need to check for unique columns with the already matched keywords between these files. Initially, we will have no matched keywords, but we will rerun this step later in the process. Fig. 5.7 shows a concrete example, and Algorithm 5 describes our method.

Co-occurrence. When having some matched documents, we can use the co-occurrence matrices M and M' to find other document matches. For an unmatched server document ed_x, we can try an unmatched leaked document d_y. If $M_{x,k}$ and $M'_{y,k'}$ are equal for each matched document pair $(ed_k, d_{k'})$ and no other document $d_{y'}$ has the same results, then we have a new document match between ed_x and d_y. The algorithm for this step is shown in Algorithm 6.

Keyword matching. We match keywords using the matched documents. To this end, we create matrices B_c and A_c by taking the columns of matched documents from matrices B and A. Note that these columns will be rearranged to the order of the matched documents, such that column B_{c_j} is equal to column $A_{c_{j'}}$ for document match $(ed_j, d_{j'})$. Matrices B_c and A_c are shaped $m \times t$ and $m' \times t$, respectively, for t matched documents. We give the algorithm for this segment in Algorithm 7 and a simple example in Fig. 5.4.

A row in the matrices indicates in which documents a query or keyword appears. If a row$_i$ in B_c is unique, row$_i$ is also unique in B, similar to A_c and A. Hence, for row$_i$ in B_c, that is unique, and if there is an equal row$_j$ in A_c, we can conclude that the underlying keyword of q_i is w_j.

Nevertheless, if row$_i$ is not unique in B_c, we can still try to match the keyword to a query. A keyword can occur more often in the unmatched documents than their query candidates; thus, they will not be valid candidates. We create a list B_x with for each similar row$_i$ in B_c the sum of row$_i$ in B; similar for list A_x, with row$_i$ in A_c and

(a) Multiple documents with the same pattern of volume and number of keywords/tokens.

Leaked files	⋯	d_4	d_6	d_8	⋯	$d_{n'}$
Volume	⋯	120	120	120	⋯	120
#Keywords	⋯	15	15	15	⋯	18

Server files	⋯	ed_6	ed_9	ed_{10}	⋯	ed_n
Volume	⋯	120	120	120	⋯	150
#Tokens	⋯	20	15	15	⋯	15

(b) With the already matched keywords, create unique columns to match documents. Here d_6 and ed_8 can be matched, as well as d_9 and ed_{15}.

$$A_{CR}\begin{array}{c} \\ w_2 \\ w_3 \\ w_5 \\ \vdots \\ w_t \end{array}\begin{array}{c} \begin{array}{cccc} d_4 & d_6 & d_8 & d_9 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \qquad B_{CR}\begin{array}{c} \\ q_1 \\ q_3 \\ q_{15} \\ \vdots \\ q_t \end{array}\begin{array}{c} \begin{array}{cccc} ed_8 & ed_9 & ed_{10} & ed_{15} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 5.7: Document matching on volume and number of keywords. Given multiple candidates, match on a unique column with the already matched keywords.

the sum of row$_i$ in A. Next, if the highest value of A_x, which is A_{x_j}, is higher than the second-highest value of B_x, we can conclude that keyword w_j corresponds to the highest value of B_x, i.e. B_{x_j}, which means that w_j matches with q_j. We put an example in Fig. 5.8.

Keyword order in documents. We aim to find more documents based on unique columns given the query and keyword mappings. First, we create matrices B_r and A_r with the rows from the matched keywords in R. B_r and A_r are submatrices of B and A, respectively, with rearranged row order. B_r and A_r are shaped $t \times n$ and $t \times n'$, respectively, for t matched keywords. Note that we show an example in Fig. 5.5. If any column$_j$ of B_r is unique and there exists an equal column$_{j'}$ in A_r, we know that ed_j is a match with $d_{j'}$.

The next step is to set the rows of the matched keywords to 0 in B and A. Then,

Algorithm 5: matchByVolume(R, A, B)

Input: R: set of matched rows,
A: $m \times n'$ matrix representing document features,
B: $m \times n$ matrix representing encrypted document features.
Output: C': a mapping of matched documents between A and B.

Function `matchByVolume`(R, A, B)**:**
  Initialize $C' \leftarrow \{\}$
  Extract patterns $\leftarrow \{(v_j, |ed_j|) \mid v_j$ is volume, $|ed_j|$ is #tokens of $ed_j\}$
  for $p \in$ *patterns* **do**
    enc_docs $\leftarrow [ed_j \mid$ pattern $p]$
    if |enc_docs| $= 1$ **then**
      $ed_j \leftarrow$ enc_docs[0]
      $C'[ed_j] \leftarrow d_{j'}$ with pattern p
    end
    else if $|R| > 0$ **then**
      docs $\leftarrow [d_{j'} \mid$ pattern $p]$
      $B_{CR} \leftarrow$ enc_docs columns and R rows of B
      $A_{CR} \leftarrow$ docs columns and R rows of A
      for *column*$_j \in B_{CR}$ *that is unique* **do**
        $C'[ed_j] \leftarrow d_{j'}$ with column$_j \in A_{CR}$
      end
    end
  end
  return C'

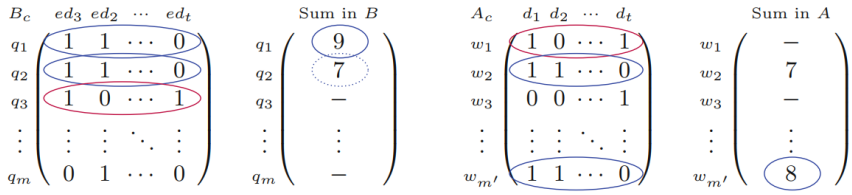


Figure 5.8: Example of matching keywords in matched documents. Query q_3 has a unique row and therefore matches with keyword w_1. Queries q_1, q_2 and keywords w_2, $w_{m'}$ have the same row. However, keyword $w_{m'}$ occurs more often in A than w_2 and query q_2 in B. Therefore q_1 matches with $w_{m'}$.

similar to before, we use the technique from the Count attack [12]; we sum the updated columns in A and B and try to match the unique columns in B to columns in A. If a column$_j$-sum of B is unique and an equal column$_{j'}$-sum in A exists, we can match document ed_j and $d_{j'}$.

The complete algorithm of our VAL-attack is in Algorithm 8.

Algorithm 6: coOccurrence(C, M, M', A, B)

Input: C: current set of matches,
M: $n \times n$ co-occurrence matrix for encrypted documents,
M': $n' \times n'$ co-occurrence matrix for plain documents,
A: $m \times n'$ matrix representing document features,
B: $m \times n$ matrix representing encrypted document features.
Output: C: updated set of matches.

```
while C is increasing do
    foreach d_{j'} ∉ C do
        sum_{j'} ← column_{j'}-sum of A
        candidates ← {ed_j ∉ C | column_j-sum of B = sum_{j'}}
        foreach ed_j ∈ candidates do
            foreach (ed_k, d_{k'}) ∈ C do
                if M_{j,k} ≠ M'_{j',k'} then
                    candidates ← candidates \{ed_j}
                end
            end
        end
        if |candidates| = 1 then
            ed_j ← candidates[0]
            C[ed_j] ← d_{j'}
        end
    end
end
return C
```

5.4.4. COUNTERMEASURE DISCUSSIONS

Some countermeasures have been proposed to mitigate leakage-abuse attacks [12, 13, 26, 27]. The main approaches are padding and obfuscation. Below, we have some discussions on the countermeasures.

The IKK attack [13] and the Count attack [12] discussed a padding countermeasure, where they proposed a technique to add fake document identifiers to a query response. These false positives could then later be removed by the user. This technique is also called *Hiding the Access Pattern* [28].

The LEAP attack [15] crucially relies on the number of keywords per document, and if the scheme adds fake query tokens to documents on the server, they will not be able to match with their known documents. However, they also proposed a technique that describes a modified attack that is better resistant to padding. This technique, which is also used in the Count attack [12], makes use of a window to match keywords. But this will give false positives and thus reduce the performance of the attack.

The Subgraph$_{\text{VL}}$ attack [14] depends on the volume of each document. Volume-hiding techniques from Kamara et al. [29] reduce the attack's performance, but it is not clear if they completely mitigate the attack.

Algorithm 7: matchKeywords(C, A, B)

Input: C: set of columns for matching,
A: $m \times n'$ matrix representing document features,
B: $m \times n$ matrix representing encrypted document features.
Output: R: a mapping of matched keywords between A and B.

Function `matchKeywords`(C, A, B)**:**
  Initialize $R \leftarrow \{\}$
  $B_c \leftarrow C$ columns of B
  $A_c \leftarrow C$ columns of A
  for $row_i \in B_c$ **do**
    if row_i *is unique in* B_c **then**
      if $row_{i'} \in A_c = row_i$ **then**
        $R[q_i] \leftarrow w_{i'}$
      end
    end
    else
      `// Match based on occurrence in (server) files`
    end
    docs $\leftarrow \{i' \in A_c \mid A_c[i'] = \text{row}_i\}$
    e_docs $\leftarrow \{j \in B_c \mid B_c[j] = \text{row}_i\}$
    $B_x \leftarrow$ sum of rows in B[e_docs], sorted descending
    $A_x \leftarrow$ sum of rows in A[docs], sorted descending
    if $B_x[1] < A_x[0] < B_x[0]$ **then**
      $i_x \leftarrow$ index of $B_x[0] \in$ e_docs
      $j_x \leftarrow$ index of $A_x[0] \in$ docs
      $R[q_{i_x}] \leftarrow w_{j_x}$
    end
  end
return R

A padding technique that will make all documents of the same size, i.e. adding padding characters, will reduce the uniqueness in matching based on the volume of a document. If the padding technique can be extended such that false positives are added to the `access pattern`, we have no unique factor in matching documents based on the number of keywords per file. Therefore, a combination of the two may decrease the performance of the VAL-Attack.

Algorithm 8: Attack(A, B, M', M)

Input: A: $m' \times n'$ matrix, B: $m \times n$ matrix,
M': $n' \times n'$ co-occurrence matrix for plain documents,
M: $n \times n$ co-occurrence matrix for encrypted documents.
Output: R: matched rows, C: matched columns.

$C \leftarrow \{\}, R \leftarrow \{\}$// Initialization
$A \leftarrow A$ where rows are extended with zeros to size $m \times n'$
$\text{vector}_A, \text{vector}_B \leftarrow [\,]$// Initialize column sums for matching
for $j \in [n]$ **do**
  $\text{vector}_B[j] \leftarrow$ sum of column B_j
end
for $j' \in [n']$ **do**
  $\text{vector}_A[j'] \leftarrow$ sum of column $A_{j'}$
end
foreach $vector_{B_j} \in vector_B$ *that is unique* **do**
  if $vector_{A_{j'}} = vector_{B_j}$ **then**
    $C[ed_j] \leftarrow d_{j'}$
  end
end
$C \leftarrow C \cup \text{matchByVolume}(R, A, B)$// Match documents with unique volume
$C \leftarrow C \cup \text{coOccurrence}(C, M, M', A, B)$// Match documents based on co-occurrence
while R *or* C *is increasing* **do**
  $R \leftarrow R \cup \text{matchKeywords}(C, A, B)$// Match keywords in matched documents
  $B_r \leftarrow R$ rows of B
  $A_r \leftarrow R$ rows of A
  foreach $column_j \in B_r$ *that is unique* **do**
    if $column_{j'} \in A_r = column_j$ **then**
      $C[ed_j] \leftarrow d_{j'}$
    end
  end
  $C \leftarrow C \cup \text{matchByVolume}(R, A, B)$
  Set row $B_j \leftarrow 0$ if $q_j \in R$// Reset matched rows in B
  Set row $A_{j'} \leftarrow 0$ if $k_{j'} \in R$// Reset matched rows in A
  for $j \in [n]$ *where* $ed_j \notin C$ **do**
    $\text{vector}_B[j] \leftarrow$ sum of column B_j
  end
  for $j' \in [n']$ *where* $d_{j'} \notin C$ **do**
    $\text{vector}_A[j'] \leftarrow$ sum of column $A_{j'}$
  end
  foreach $vector_{B_j} \in vector_B$ *that is unique and* $ed_j \notin C$ **do**
    if $vector_{A_{j'}} = vector_{B_j}$ *and* $d_{j'} \notin C$ **then**
      $C[ed_j] \leftarrow d_{j'}$
    end
  end
  $C \leftarrow C \cup \text{coOccurrence}(C, M, M', A, B)$
end
return R, C

5.5. EVALUATION

We set up the experiments to run the proposed attack to evaluate the performance. Furthermore, we compare the file and query recovery of the VAL-Attack with the results from the LEAP [15] and Subgraph$_{VL}$ attack [14]. We notice that the LEAP attack is not resistant to the test countermeasures, and Blackstone et al. [14] argue for their Subgraph$_{VL}$ attack that it is not clear whether volume-hiding constructions may mitigate the attack altogether. From this perspective, we only discuss the performance of VAL-Attack against countermeasures in Section 5.5.3. It would be an interesting problem to test the countermeasures on the LEAP and Subgraph$_{VL}$ attacks, but that is orthogonal to the focus of this work.

5.5.1. EXPERIMENTAL SETUP

We used the Enron dataset [30] to run our comparison experiments. We leveraged the *_sent_mail* folder from each of the 150 users from this dataset, resulting in 30,109 e-mails from the Enron corporation. The second dataset we used is the Lucene mailing list [31]; we specifically chose the "java-user" mailing list from the Lucene project for 2002-2011. This dataset contains 50,667 documents. Finally, we did the tests on a collection of Wikipedia articles. We extracted plaintext documents from Wikipedia in April 2022 using a simple wiki dump[1] and used the tool from David Shapiro [32] to extract plaintext data, resulting in 204,737 files. The proposed attack requires matrices of size $n \times n$; therefore, we limited the number of Wikipedia files to 50,000. We used Python 3.9 to implement the experiments and run them on machines with different computing powers to improve running speed.

To properly leverage those data from the datasets for the experiments, we first extracted the information of the Enron and Lucene e-mail content. The title's keywords, the names of the recipients or other information present in the e-mail header were not used for queries. NLTK corpus [33] in Python is used to get a list of English vocabulary and stopwords. We removed the stopwords with that tool and stemmed the remaining words using Porter Stemmer [34]. We further selected the most frequent keywords to build the keyword set for each document. For each dataset, we extracted 5,000 words as the keyword set W. Within the Lucene e-mails, we removed the unsubscribe signature because it appears in every e-mail.

The server files (n) and keywords (m) are all files from the dataset and 5,000 keywords, respectively. The leakage percentage determines the number of files (m') known to the user. The attacker only knows the keywords (n') leaked with these known documents. The server files and queries construct a matrix B of size $m \times n$; while the matrix A of size $m' \times n'$ is constructed with the leaked files. We took the dot product for both matrices and created the matrices M and M', respectively. Note that the source code to simulate the attack and obtain our results is available here: https://github.com/StevenL98/VAL-Attack.

Because our attack does not create false positives, the accuracy of the retrieved files and keywords is always 100%. Therefore, we calculated the percentage of files

[1] https://dumps.wikimedia.org/simplewiki/20220401/simplewiki-20220401-pages-meta-current.xml.bz2

and keywords retrieved from the total leaked files and keywords. Each experiment is run 20 times to calculate an average over the simulations. We chosen 0.1%, 0.5%, 1%, 5%, 10%, 30% as leakage percentages. The lower percentages are chosen to compare with the results from the LEAP attack [15], and the maximum of 30% is chosen because of the stagnation in query recovery.

5.5.2. EXPERIMENTAL RESULTS

The results tested with the different datasets are given in Fig. 5.9a and Fig. 5.9b, which show the number and percentage of files and keywords recovered by our attack. The solid line is the average recovery in those plots, and the shades are the error rate over the 20 runs.

We can see that the VAL-attack recovers almost 98% of the known files and above 93% of the keywords available to the attacker once the leakage percentage reaches 5%. These percentages are based on the leaked documents. When 10% of the Enron database is leaked, which is 3,010 files with 4,962 keywords, we can match 2,950 files and 4,909 queries, corresponding to 98% and 99%, respectively. The Lucene dataset is more extensive than Enron, and therefore we have more files available for each leakage percentage. One may see that we can recover around 99% of the leaked files and a rising number of queries, starting from 40% of the available keyword set. The Wikipedia dataset does not consist of e-mails but rather lengthy article texts. We reveal fewer files than the e-mail datasets, but we recover just below 90% of the leaked files, and from 1% leakage, we recover more available keywords than the other datasets. This difference is probably because of the number of keywords per file since the most frequent keywords are chosen.

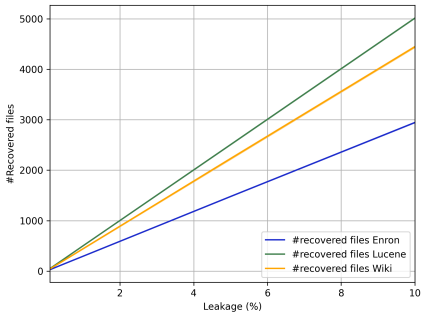
With the technique we proposed, one can match leaked documents to server documents for almost all leaked documents. Next, the algorithm will compute the underlying keywords to the queries. It is up to the attacker to allow false positives and improve the number of (possible) correctly matched keywords, but we decided not to include it.

Comparison. We compare the performance of VAL-Attack to two attacks with the Enron dataset. One is the LEAP attack [15] (which is our cornerstone), while the other is the $\text{Subgraph}_{\text{VL}}$ attack [14] (as they use the volume pattern as leakage). We divide the comparison into two parts: the first is for recovering files, and the second is for queries recovery.

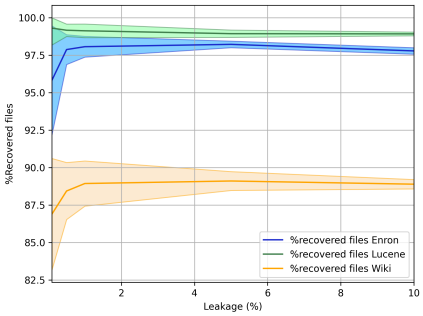
As shown in Fig. 5.10, we recover more files than the LEAP attack, and the gap in files recovered expands as the leakage percentage increases, see Fig. 5.10a.i. The difference in the percentage of files recovered is stable, as VAL-Attack recovers about eight percentage points more files than the LEAP attack, see Fig. 5.10a.ii. The comparison outcome for recovered queries can be seen in Fig. 5.10b. We can see that the recovered queries do not show a significant difference with the LEAP attack as that attack performs outstandingly in query recovery. The most significant difference is around 5% leakage, where VAL-Attack retrieves around 100 queries more than the LEAP attack, which could influence a real-world application. Compared to the $\text{Subgraph}_{\text{VL}}$, we see in Fig. 5.10b.ii that the combination of the `access pattern` and the `volume pattern` is a considerable improvement; we reveal about

(a) Exact number and relative percentage of recovered files

(a.i) #Files recovered

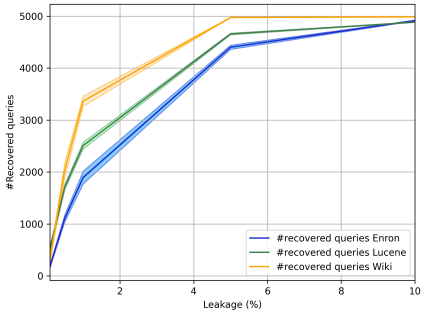


(a.ii) %Files recovered



(b) Exact number and relative percentage of recovered queries

(b.i) #Queries recovered



(b.ii) %Queries recovered

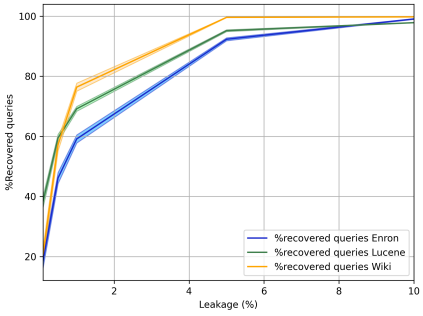


Figure 5.9: Results for VAL-Attack, with the actual number and the percentage of recovered files and queries for different leakage percentages.

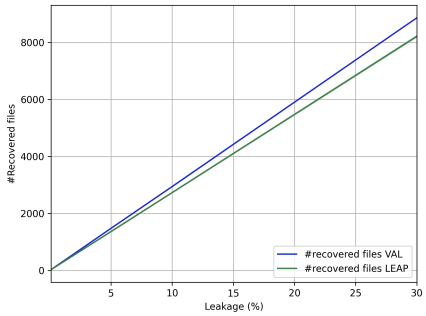
60 percentage points more of the available queries.

5.5.3. COUNTERMEASURE PERFORMANCE

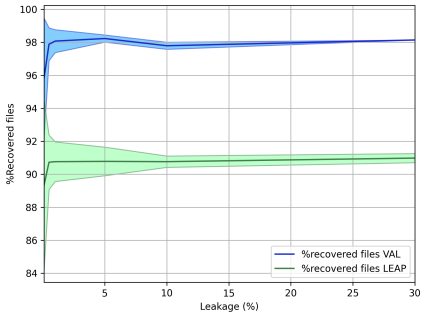
As discussed in Section 5.4.4, there are several options for countermeasures against attacks on SE schemes. Moreover, since our attack exploits both the `access and volume pattern`, countermeasures must mitigate both leakage patterns. The former can be mitigated by padding the server result, while the latter may be handled using volume-hiding techniques. However, these approaches may come with impractical side effects. Padding the server response requires more work on the client-side to filter out the false positives. This padding can cause storage and reading problems because the user has to wait for the program to filter out the correct results. The volume-hiding technique [29] may easily yield significant storage overhead and could

(a) Comparison with LEAP [15] based on the number and percentages of files recovered

(a.i) #Files recovered

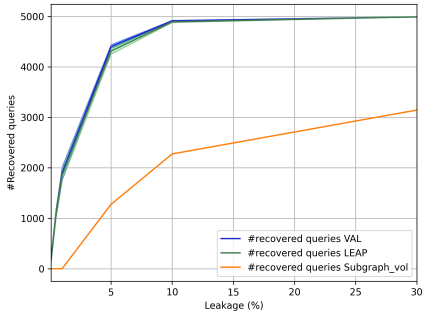


(a.ii) %Files recovered



(b) Comparison with LEAP [15] and Subgraph$_{VL}$ [14] based on the number and percentages of queries recovered

(b.i) #Queries recovered



(b.ii) %Queries recovered

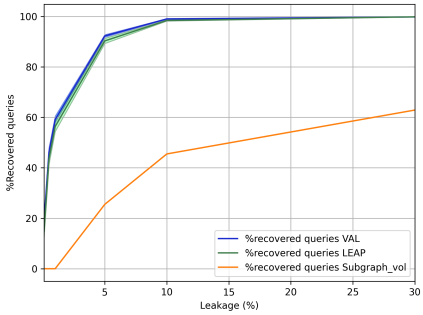


Figure 5.10: Comparison of VAL-Attack

therefore not be practical in reality. Luckily, Patel et al. [35] illustrated how to reduce this side effect whilst mitigating the attack.

It is possible to mitigate our attack theoretically by using a combination of padding and volume hiding. We tested the VAL-attack's performance with padding, volume hiding and further a combination, but we did not examine by obfuscation due to hardware limitations.

We padded the server data using the technique described by Cash et al. [12]. Each query returned a multiplication of 500 server files, so if the original query returned 600 files, the server now returned 1,000. Padding is done by adding documents to the server response that to done contain the underlying keyword. These documents can then later be filtered by the client, but will obfuscate the client's observation. We took the naïve approach from Kamara et al. [29] for volume hiding, where we padded each document to the same volume. By adding empty bytes to a document, it will grow in size. If done properly, all files will eventually have the same size that

can not be distinguished from the actual size.

We ran the countermeasure experiments on the Enron and the Lucene dataset. We did not perform the test on the Wikipedia dataset, but we can predict that the countermeasures may affect the attack performance. We predict that a single countermeasure will not entirely reduce the attack effectiveness, but a combination may do.

Because of the exploitation of the two leakage patterns, we see in Table 5.2 that our attack can still recover files and underlying keywords against only a single countermeasure. Under a combination of padding and volume hiding, our attack cannot reveal any leaked file or keyword.

Table 5.2: Performance of VAL-Attack with countermeasures

Dataset		Enron						Lucene					
Countermeasure		Padding		Volume hiding		Padding & Vol. Hiding		Padding		Volume hiding		Padding & Vol. Hiding	
Files	0.1%	25	(83.7%)	27	(89.5%)	0	(0%)	45	(88.9%)	10	(28.4%)	0	(0%)
	0.5%	103	(68.4%)	137	(90.7%)	0	(0%)	191	(75.3%)	95	(37.4%)	0	(0%)
	1%	208	(69.0%)	274	(90.9%)	0	(0%)	381	(75.3%)	147	(28.9%)	0	(0%)
	5%	1,114	(74.0%)	1,365	(90.7%)	0	(0%)	2332	(92.0%)	2452	(96.8%)	0	(0%)
	10%	1,910	(63,4%)	2,736	(90.9%)	0	(0%)	4,073	(80.4%)	4,891	(96.5%)	0	(0%)
	30%	5,358	(59.0%)	8,219	(91.0%)	0	(0%)	10,343	(68.0%)	-[a]		0	(0%)
Queries	0.1%	94	(10.4%)	172	(14.8%)	0	(0%)	377	(27.7%)	153	(10.6%)	0	(0%)
	0.5%	433	(18.1%)	1,059	(43.3%)	0	(0%)	724	(25.3%)	663	(22.8%)	0	(0%)
	1%	414	(12.8%)	1,836	(56.3%)	0	(0%)	556	(15.3%)	748	(20.5%)	0	(0%)
	5%	53	(1.1%)	4,290	(89.9%)	0	(0%)	87	(1.8%)	4,659	(95.2%)	0	(0%)
	10%	11	(0.2%)	4,890	(98.4%)	0	(0%)	33	(0.7%)	4,872	(97.6%)	0	(0%)
	30%	1	(0.0%)	4,993	(99.9%)	0	(0%)	10	(0.2%)	-[a]		0	(0%)

[a] Did not run due to hardware limitations

Table 5.2 is read as follows: The number below the countermeasure is the exact number of retrieved files or queries, with the relative percentage between brackets. So for 0.1% leakage under the padding countermeasure, we revealed, on average, 25 files, which was 83.7% of the leaked files. Each experiment ran 20 times. Due to runtime and hardware limitations, we did not run the experiment with 30% leakage on the Lucene dataset. However, since we have the results for 10% leakage and the results for the Enron dataset, we can predict the outcome for 30%. Similar to the Enron dataset, the recovered data in Lucene increases as the leakage percentage grows. Therefore, we predict that 30% leakage results in the Lucene dataset is a bit higher than the 10% leakage.

5.5.4. DISCUSSION ON EXPERIMENTS

We chose specific parameters in the experiments and only compared our attack with two popular attacks [14, 15]. We give more discussions below.

Parameters. We used 5,000 high selectivity keywords, i.e. keywords that occur the most in the dataset. This number is chosen because a practical SE application will probably not have just a few search terms in a real-world scenario. Other attacks [12–14] have experimented with only 150 query tokens and 500 keywords, and we argue that this may not be realistic. Our attack is able to recover almost all

underlying keywords for an experiment with 500 keywords because the number of files is still equal, but a slight variation in keyword occurrence.

We cut the number of Wikipedia files to 50,000. We did this to better present the comparison with the Enron and Lucene datasets. The attack may also take longer to run when all Wikipedia files are considered. The results will also differ as the number of files leaked increases similarly. The percentage of files recovered will probably be the same because of keyword distribution among the files.

If we ran the experiments with a higher leakage percentage, the attack would eventually recover more files, as more are available, but we would not recover more keywords. As with 30% leakage, we see that we have recovered all 5,000 keywords.

Our attack performs without false positives. And we did so because they would not improve the performance, and an attacker cannot better understand the data if he cannot rely on it. If we allowed the attack to return false positives, we would have 5,000 matches for underlying keywords, of which not all are correct. The attack performance will not change since we will only measure the correct matches, which we already did.

Attack comparison. In Fig. 5.10a, we only compared our attack with the LEAP attack rather than the $\text{Subgraph}_{\text{VL}}$ attack. We did so because the latter does not reveal encrypted files and thus cannot be compared. If we choose to compare the attack to ours, we would have to rebuild their attack using their strategy, which is out of the scope of this work.

We used the Enron dataset to compare the VAL-Attack to the LEAP and the $\text{Subgraph}_{\text{VL}}$. In their work [14, 15], they used the Enron dataset to show their performance. If we used the Lucene or Wikipedia dataset instead to present the comparison, we would have no foundation in the literature to support our claim. A comparison of all the datasets would still show that our attack surpasses the attacks since, in theory, we exploit more.

We discussed other attacks, like the IKK and the Count attack, but we did not compare their performance with ours. While these attacks exploit the same leakage, we could still consider them. However, since LEAP is considered the most state-of-the-art attack and it has already been compared with the other attacks in [15], we thus only have to compare the LEAP attack here. Accordingly, a comparison with all attacks would not affect the results and conclusion of this paper.

5.6. RELATED WORK

The Count attack [12] uses the number of files returned for the query as their matching technique; The SubgraphVL [14] matches keywords based on unique document volumes as if it is the response pattern, and the LEAP attack [15] uses techniques from previous attacks to match leaked documents and keywords with high accuracy. Besides the attacks that exploit similar leakage to our proposed attack, we may also review those attacks that do not. An attack that leverages similar documents as auxiliary knowledge, called Shadow Nemesis, was proposed by Pouliot et al. [36]. They created a weighted graph matching problem in the attack and solved it using Path or Umeyama. Damie et al. [37] presented the Score attack,

requiring similar documents, and they matched based on the frequency of keywords in the server and auxiliary documents. Both attacks use co-occurrence matrices to reveal underlying keywords. The Search attack by Liu et al. [38] matches based on the search pattern, i.e. the frequency pattern of queries sent to the server. Table 5.3 briefly compares the attacks based on leakage, auxiliary knowledge, false positives and exploiting techniques. The reviewed attacks described above are not mainly relevant to our proposed attack; thus, we did not put them in the comparison in Section 5.

Table 5.3: Comparison of Different Attacks. A: Access Pattern, S: Search Pattern, V: Volume Pattern.

Attack	**Leakage**	**Auxiliary Data**	**False Positive**
IKK [13]	A	Docs, Queries	✓
Shadow Nemesis [36]	A	Similar	✓
Score [37]	A	Similar, Queries	✓
Search14 [38]	S	Search Freq.	✓
ZKP [25]	A	All Keywords	×
Count [12]	A	Docs	✓
Subgraph$_{VL}$ [14]	V	Docs	✓
LEAP [15]	A	Docs	×
VAL-Attack	A, V	Docs	×

5.7. CONCLUSION

We proposed the VAL-attack to improve the matching technique from the LEAP attack, leveraging the leakage from the `access pattern` and the `volume pattern` which is a combination that has not been exploited before. We showed that our attack provides excellent performance, and we compared it to the LEAP attack and the subgraph$_{VL}$ attack. The number of matched files is with more remarkable improvement than the number of queries recovered compared to the LEAP attack. The attack recovers around 98% of the leaked documents and above 90% for query recovery with very low leakage. Since the proposed attack uses both the document size and the response per query, it requires strong (and combined) countermeasures and thus, is more harmful than existing attacks.

REFERENCES

[1] D. X. Song, D. A. Wagner, and A. Perrig. “Practical Techniques for Searches on Encrypted Data”. In: *IEEE S&P 2000*. IEEE, 2000, pp. 44–55. DOI: 10.1109/SECPRI.2000.848445.

[2] R. Bost, B. Minaud, and O. Ohrimenko. “Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives”. In: *ACM CCS 2017*. Ed. by B. Thuraisingham, D. Evans, T. Malkin, and D. Xu. ACM, 2017, pp. 1465–1482. DOI: 10.1145/3133956.3133980.

[3] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. “Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation”. In: *NDSS 2014*. The Internet Society, 2014. DOI: 10.14722/ndss.2014.23264.

[4] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. “Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, 2013, pp. 353–373. DOI: 10.1007/978-3-642-40041-4_20.

[5] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. “Searchable symmetric encryption: improved definitions and efficient constructions”. In: *ACM CCS 2006*. Ed. by A. Juels, R. N. Wright, and S. D. C. di Vimercati. ACM, 2006, pp. 79–88. DOI: 10.1145/1180405.1180417.

[6] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. “Public Key Encryption with Keyword Search”. In: *EUROCRYPT 2004*. Ed. by C. Cachin and J. Camenisch. Vol. 3027. LNCS. Springer, 2004, pp. 506–522. DOI: 10.1007/978-3-540-24676-3_30.

[7] R. Zhang and H. Imai. “Combining Public Key Encryption with Keyword Search and Public Key Encryption”. In: *IEICE Trans. Inf. Syst.* 92-D.5 (2009), pp. 888–896. DOI: 10.1587/transinf.E92.D.888.

[8] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. “Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions”. In: *CRYPTO 2005*. Ed. by V. Shoup. Vol. 3621. LNCS. Springer, 2005, pp. 205–222. DOI: 10.1007/11535218_13.

[9] Q. Zheng, S. Xu, and G. Ateniese. “VABKS: Verifiable attribute-based keyword search over outsourced encrypted data”. In: *IEEE INFOCOM 2014*. IEEE, 2014, pp. 522–530. DOI: 10.1109/INFOCOM.2014.6847976.

[10] W. He, D. Akhawe, S. Jain, E. Shi, and D. X. Song. “ShadowCrypt: Encrypted Web Applications for Everyone”. In: *ACM CCS 2014*. Ed. by G. Ahn, M. Yung, and N. Li. ACM, 2014, pp. 1028–1039. DOI: 10.1145/2660267.2660326.

[11] B. Lau, S. P. Chung, C. Song, Y. Jang, W. Lee, and A. Boldyreva. “Mimesis Aegis: A Mimicry Privacy Shield-A System’s Approach to Data Privacy on Public Cloud”. In: *USENIX 2014*. Ed. by K. Fu and J. Jung. USENIX Association, 2014, pp. 33–48. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/lau.

[12] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. “Leakage-Abuse Attacks Against Searchable Encryption”. In: *ACM CCS 2015*. Ed. by I. Ray, N. Li, and C. Kruegel. ACM, 2015, pp. 668–679. DOI: 10.1145/2810103.2813700.

[13] M. S. Islam, M. Kuzu, and M. Kantarcioglu. “Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation”. In: *NDSS 2012*. The Internet Society, 2012. URL: https://www.ndss-symposium.org/ndss2012/access-pattern-disclosure-searchable-encryption-ramification-attack-and-mitigation.

[14] L. Blackstone, S. Kamara, and T. Moataz. “Revisiting Leakage Abuse Attacks”. In: *NDSS 2020*. The Internet Society, 2020. DOI: 10.14722/ndss.2020.23103.

[15] J. Ning, X. Huang, G. S. Poh, J. Yuan, Y. Li, J. Weng, and R. H. Deng. “LEAP: Leakage-Abuse Attack on Efficiently Deployable, Efficiently Searchable Encryption with Partially Known Dataset”. In: *ACM CCS 2021*. Ed. by Y. Kim, J. Kim, G. Vigna, and E. Shi. ACM, 2021, pp. 2307–2320. DOI: 10.1145/3460120.3484540.

[16] S. Kamara, C. Papamanthou, and T. Roeder. “Dynamic searchable symmetric encryption”. In: *ACM CCS 2012*. Ed. by T. Yu, G. Danezis, and V. D. Gligor. ACM, 2012, pp. 965–976. DOI: 10.1145/2382196.2382298.

[17] B. Minaud and M. Reichle. *Dynamic Local Searchable Symmetric Encryption*. arXiv preprint. 2021. arXiv: 2201.05006. URL: https://arxiv.org/abs/2201.05006.

[18] J. Li, X. Niu, and J. S. Sun. “A Practical Searchable Symmetric Encryption Scheme for Smart Grid Data”. In: *IEEE ICC 2019*. IEEE, 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761599.

[19] I. Demertzis and C. Papamanthou. “Fast Searchable Encryption With Tunable Locality”. In: *ACM SIGMOD 2017*. Ed. by S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu. ACM, 2017, pp. 1053–1067. DOI: 10.1145/3035918.3064057.

[20] Z. Gui, K. G. Paterson, and S. Patranabis. *Rethinking Searchable Symmetric Encryption*. Cryptology ePrint Archive, Paper 2021/879. 2021. URL: https://eprint.iacr.org/2021/879.

[21] R. Bost. “$\sum o\varphi o\varsigma$: Forward Secure Searchable Encryption”. In: *ACM CCS 2016*. Ed. by E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi. ACM, 2016, pp. 1143–1154. DOI: 10.1145/2976749.2978303.

[22] M. Chase and S. Kamara. “Structured Encryption and Controlled Disclosure”. In: *ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. LNCS. Springer, 2010, pp. 577–594. DOI: 10.1007/978-3-642-17373-8_33.

[23] S. Kamara and C. Papamanthou. “Parallel and Dynamic Searchable Symmetric Encryption”. In: *FC 2013*. Ed. by A. Sadeghi. Vol. 7859. LNCS. Springer, 2013, pp. 258–274. DOI: 10.1007/978-3-642-39884-1_22.

[24] Q. Ma, J. Zhang, Y. Peng, W. Zhang, and D. Qiao. “SE-ORAM: A Storage-Efficient Oblivious RAM for Privacy-Preserving Access to Cloud Storage”. In: *IEEE CSCloud 2016*. Ed. by M. Qiu, L. Tao, and J. Niu. IEEE Computer Society, 2016, pp. 20–25. DOI: 10.1109/CSCloud.2016.24.

[25] Y. Zhang, J. Katz, and C. Papamanthou. “All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption”. In: *USENIX 2016*. Ed. by T. Holz and S. Savage. USENIX Association, 2016, pp. 707–720. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/zhang.

[26] G. Chen, T. Lai, M. K. Reiter, and Y. Zhang. “Differentially Private Access Patterns for Searchable Symmetric Encryption”. In: *IEEE INFOCOM 2018*. IEEE, 2018, pp. 810–818. DOI: 10.1109/INFOCOM.2018.8486381.

[27] Z. Shang, S. Oya, A. Peter, and F. Kerschbaum. “Obfuscated Access and Search Patterns in Searchable Encryption”. In: *NDSS 2021*. The Internet Society, 2021. URL: https://www.ndss-symposium.org/ndss-paper/obfuscated-access-and-search-patterns-in-searchable-encryption/.

[28] Y. Kortekaas. *Access Pattern Hiding Aggregation over Encrypted Databases*. Oct. 2020. URL: http://essay.utwente.nl/83874/.

[29] S. Kamara and T. Moataz. “Computationally Volume-Hiding Structured Encryption”. In: *EUROCRYPT 2019, Part II*. Ed. by Y. Ishai and V. Rijmen. Vol. 11477. LNCS. Springer, 2019, pp. 183–213. DOI: 10.1007/978-3-030-17656-3_7.

[30] C. William W. Cohen MLD. *Enron Email Datasets*. 2015. URL: https://www.cs.cmu.edu/~enron/.

[31] Apache. *Mail Archieves of Lucene*. 1999. URL: https://mail-archives.apache.org/mod_mbox/#lucene.

[32] D. Shapiro. *Convert Wikipedia database dumps into plaintext files*. 2021. URL: https://github.com/daveshap/PlainTextWikipedia.

[33] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O’Reilly Media, Inc, 2009.

[34] M. F. Porter. “An algorithm for suffix stripping”. In: *Program* 40 (1980), pp. 211–218.

[35] S. Patel, G. Persiano, K. Yeo, and M. Yung. “Mitigating Leakage in Secure Cloud-Hosted Data Structures: Volume-Hiding for Multi-Maps via Hashing”. In: *ACM SIGSAC 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM, 2019, pp. 79–93. DOI: 10.1145/3319535.3354213.

[36] D. Pouliot and C. V. Wright. “The Shadow Nemesis: Inference Attacks on Efficiently Deployable, Efficiently Searchable Encryption”. In: *ACM SIGSAC 2016*. Ed. by E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi. ACM, 2016, pp. 1341–1352. DOI: 10.1145/2976749.2978401.

[37] M. Damie, F. Hahn, and A. Peter. “A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents”. In: *USENIX 2021*. Ed. by M. Bailey and R. Greenstadt. USENIX Association, 2021, pp. 143–160. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/damie.

[38] C. Liu, L. Zhu, M. Wang, and Y. Tan. “Search pattern leakage in searchable encryption: Attacks and new construction”. In: *Information Sciences* 265 (2014), pp. 176–188. ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.11.021.

6

FILE-INJECTION ATTACKS ON SE, BASED ON BINOMIAL STRUCTURES

One distinguishable feature of file-inject attacks on searchable encryption schemes is the 100% query recovery rate, i.e., confirming the corresponding keyword for each query. The main efficiency consideration of file-injection attacks is the number of injected files. In the work of Zhang et al. (USENIX 2016), $|\log_2|K||$ *injected files are required, each of which contains* $|K|/2$ *keywords for the keyword set* K*. Based on the construction of the uniform* (s,n)*-set, Wang et al. need fewer injected files when considering the threshold countermeasure. In this work, we propose a new attack that further reduces the number of injected files where Wang et al. needs up to 38% more injections to achieve the same results. The attack is based on an increment* (s,n)*-set, which is also defined in this paper.*

This chapter is based on the paper "File-Injection Attacks on Searchable Encryption, Based on Binomial Structures" by Langhout, T., **Chen, H.**, and Liang K. in ESORICS (3) 2024: 424-443

6.1. INTRODUCTION

Ensuring exclusive data access remains a paramount concern, often necessitating external cloud servers due to limited user storage capacity. To enable efficient data searches, these servers must implement search-over-plaintext methods for speed and efficacy.

Song *et al.* [1] were pioneers in proposing a cryptographic scheme tailored to address the challenge of searching encrypted data, particularly enabling controlled and concealed keyword searches. This general searchable encryption (SE) framework entails the storage of an index and database on the server. Each keyword within a file undergoes independent encryption, alongside the encryption of the file as a whole. Retrieval of files containing specific keywords involves the user generating a token by encrypting the desired keyword, which is then matched against all encrypted keywords stored on the server. Upon a match, the entire encrypted file is returned to the user for decryption. Since the introduction of this foundational scheme, numerous researchers have proposed diverse variants of SE schemes [2–8]. These schemes offer varying levels of file and keyword privacy, with the ORAM scheme emerging as the most secure, effectively concealing access pattern leakage [8]. However, schemes with minimal leakage patterns tend to be computationally intensive and impractical. Alternatively, other proposed schemes, while computationally less burdensome, permit a marginally higher degree of leakage. Cash *et al.* [9] categorized these schemes into distinct leakage levels: L1, L2, L3, and L4, each revealing different degrees of information about keyword occurrences. Subsequent studies have demonstrated the potential exploitation of even minimal leakage to extract significant information from databases, emphasizing the critical role of prior knowledge in facilitating successful attacks [10–13]. Recovery of keywords involves retrieving the keyword associated with the queried token, representing an encrypted keyword of a file.

Attacks on SE schemes may manifest as either passive or active. Passive attacks entail the observation of leakage patterns to construct keyword-query matches [11, 14–16]. These attacks refrain from interfering with protocols and leverage preexisting knowledge to execute their strategies. Passive attacks typically target weaker schemes exhibiting higher leakage levels (L2-L4) and often necessitate external or prior knowledge for execution. Conversely, active attacks involve servers injecting files into a user's database to glean insights. Injection attacks leverage either file access patterns or volume patterns [9, 12, 17–20]. This paper will be based on file-injections with the use of file access pattern leakage. Active attacks, typically assuming L1 leakage or less, necessitate minimal prior knowledge, contrasting with the requirements of passive attacks. Successful recovery of keywords in active attacks is consistently achieved with 100% accuracy, with the performance metric being the number of injections required for a successful attack.

Cash *et al.* [9] were among the first to introduce an active attack wherein the server sends files to the client, subsequently encrypted and stored by the client. These attacks typically assume L2-L3 leakage, akin to passive attacks. Attackers construct files of their choosing and transmit them to users, compelling the application of the scheme to the received file, thereby enabling the observation of ciphertext by

the server. Zhang *et al.* [17] categorized such attacks as file-injection attacks and introduced the Binary-attack, premised on L1 leakage and injecting half of the keyword universe per injection, akin to binary search methodologies. Note that there have been other types of attacks on SE systems, e.g., [21–24], and also interesting secure solutions, such as [25, 26]. We refer interested readers to these research works.

Countermeasures such as thresholds and padding are implemented to impede the success of attacks. Thresholds impose limits on the number of keywords a file can contain, while padding obscures actual results by introducing additional files alongside queried files. Wang *et al.* [18] proposed an alternative approach to injection attacks based on finite set theory, offering superior performance compared to previous methods. This approach, known as the FST-attack, necessitates fewer injections than the Binary-attack under certain conditions, leveraging so-called (s,n)-sets to enhance attack efficacy. Despite these advancements, the FST-attack's reliance on complete (s,n)-sets for all identified keywords represents a notable limitation.

Organization of the paper. The organization of the rest of the paper goes as follows. In Chapter 6.2, the description of the SSE scheme, file-injection, thresholds, and the latest state-of-the-art full file-injection attack on SSE schemes are given. In Chapter 6.3, our Binomial-attack is explained in detail, together with how it can easily be applied under any threshold and dataset size, and the performances of our attack compared to previous file-injection attacks are visualised. In Chapter 6.4, we show the consequences of padding on our attack and compare these with the consequences on the FST-attack. In Chapter 6.5, a mitigation is proposed to perform better under a scheme that uses padding, with minimal trade-off. In Chapter 6.6 and 6.7 the results and untouched topics of the paper are debated. Finally, the paper is summarized in Chapter 6.8.

6.2. PRELIMINARIES

6.2.1. SEARCHABLE ENCRYPTION

A *searchable encryption* (SE) scheme has three algorithms: encryption, search, and update (only for dynamic) algorithms.

The encryption algorithm takes as input a set of files $F = \{F_1, \cdots, F_n\}$ and a secret key from the data owner, and outputs the encrypted files. These ciphertexts are then stored on the cloud server. The search algorithm takes a secret key and a keyword k as input, and outputs a query(token) t, which allows the cloud server to search the files that contain the corresponding keyword k among the encrypted files. The data owner can then decrypt the returned documents from the server and identify all related files to the keyword k. The update algorithm only applies to the dynamic SE schemes, which outputs updated files, given a secret key and a set of files.

6.2.2. FILE INJECTION ATTACK

One of the goals of the attacker is called query recovery. The attack attempts to recover the underlying keywords to queries, which threatens query privacy and file privacy. We focus on file-injection attacks in this paper.

Instead of passive attacks, the attacker in file-injection attacks is active by sending to the data owner some proper documents, which are then encrypted by the latter and also stored on the cloud according to the SE schemes. As an example, one can inject files to a user by sending designed emails in the email system. The attacker then observes the returned files, especially its own injected files, corresponding to the queries through the search algorithm. According to the returned (previously injected) files, the attacker can achieve the goal of query recovery.

The first file-injection attack is proposed by Cash *et al.*[9] and further improved by Zhang *et al.*[17]. We show an example of the binary-search attack in Table 6.1 that injects $\log_2(|K|)$ files and achieves a 100% query recovery, where $|K|$ is the size of the keyword set. In this example, if returned files corresponding to a query t are F_1 and F_3, then we know its underlying keyword is k_2. Analogously, other keywords can also be matched according to different combinations of returned files.

Files	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
F_1	1	1	1	1	0	0	0	0
F_2	1	0	1	0	1	0	1	0
F_3	1	1	0	0	1	1	0	0

Table 6.1: An example of the binary-attack file with a keyword universe of 8. Keywords are assigned into files: F_1, F_2, F_3, where 1 denotes the presence of the corresponding keyword and 0 indicates its absence.

In this work, our file-injection attack is based on the same assumption in [9, 17] that the attacker knows the file access pattern (i.e., knowing the returned files according to queries) and also can identify the files on the cloud corresponding to its injected files. One distinguishable feature of file-inject attacks is the 100% query recovery rate, so we evaluate the efficiency of such attacks from the number of injected files.

Wang *et al.*[18] further improved the work [17] to deal with the countermeasures of a threshold of a maximal number of keywords in each file.

6.2.3. FST-ATTACK

In this section, we review the definition of a uniform (s,n)-set and how the FST-attack works [18], based on the uniform (s,n)-set. The method to construct a uniform (s,n)-set of a finite set is presented by Liu and Cao [27].

Definition 6.2.1 (Uniform (s,n)-set [18])**.** *Let a set $A=\{d_1,d_2,\cdots,d_m\}$ and the subsets $A_1,A_2,\cdots,A_n \subset A$ be called a uniform (s,n)-set of A ($m \geq \binom{n}{s-1}$) if the following three conditions are satisfied:*

- $|A_1| = |A_2| = \cdots = |A_n|$;
- *For any* s *subsets* $A_{i_1}, \cdots, A_{i_s} \in \{A_1, \cdots, A_n\}$, *there is* $\bigcup_{j=1}^{s} A_{i_j} = A$;
- *For any* $s-1$ *subsets* $A_{i_1}, \cdots, A_{i_{s-1}} \in \{A_1, \cdots, A_n\}$ *there is* $\bigcup_{j=1}^{s-1} A_{i_j} = A \backslash \{d_i\}$.

Where n *denotes the number of injected files,* $|A_i|$ *the size of each injected file, and* m *the keyword universe.*

A uniform (s,n)-set for a finite set with size m has the following properties, when we choose $m = \binom{n}{s-1}$.

Lemma 6.2.2 ([18])**.** *Let* $(A_1, A_2, \cdots, A_n)$ *be a uniform* (s,n)*-set, then we have*

- *Size. The size of each file* A_i *is* $|A_i| = \binom{n-1}{s-1}$ *for* $1 \le i \le n$.
- *Intersection. Let* $r = n - s + 1$, *then the size of the intersection of arbitrary* r *files is only 1:* $|\cap_{j=1}^{r} A_{i_j}| = 1$.

Based on the uniform (s,n)-set, Wang *et al.* [18] present a file-injection attack to SE. We assume the keyword set $\mathbb{K} = \{k_1, k_2, \cdots, k_m\}$.

Their basic attack is to first construct a uniform (s,n)-set $\{A_1, A_2, \cdots, A_n\}$ based on the technique presented by Liu and Cao [27] for the keyword set $\mathbb{K}$ such that $\binom{n}{s-1} \ge m$ it means the maximal number of keywords in the uniform (s,n)-set is greater than the keyword size m), and then generate a file set of size n: $\{D_1, D_2, \cdots, D_n\}$, where the file D_i contains the same keyword in the A_i for $1 \le i \le n$. Those files are then injected into the SE scheme, and the attack recovers the keyword corresponding to a token by the returned $n - s + 1$ files. The correctness of the basic attack is guaranteed by Lemma 6.2.2, i.e., there only exists one keyword in the intersection of $n - s + 1$ files.

When the threshold countermeasure is taken into consideration, that is the number of keywords in each file should be smaller than a threshold T, they proposed an advanced file-injection attack, aiming at obtaining a minimum n, the number of files that should be injected. Towards this goal, they choose the minimum n such that

$$\begin{cases} \binom{n-1}{s-1} \le T \\ \binom{n}{s-1} \ge m. \end{cases} \tag{6.1}$$

Moreover, they present look-up tables to determine the optimal s and n corresponding to the threshold T and the number of keywords in different intervals.

As an example of recovering 23 keywords $\{k_1, k_2, \cdots, k_{23}\}$ with threshold 7, we solve the Eq. (6.1) for $T = 7$ and $m = 23$ and then get a minimum $n = 8$ and $s = n - 1 = 7$. The corresponding injected files according to a uniform $(7,8)$-set are shown in Table

6.2. Note that some parts in files $\{D_1, \cdots, D_8\}$ are left as blank since the number of keywords in these files has reached 23. Each keyword can be matched by $n-s+1=2$ returned files. For example, if files D_1 and D_2 are returned after a query to a token t, we know the corresponding keyword to t is k_1.

	(7,8)-**set**						
Files	Col_1	Col_2	Col_3	Col_4	Col_5	Col_6	Col_7
F_1	k_{22}	k_{23}					
F_2	k_{16}	k_{17}	k_{18}	k_{19}	k_{20}	k_{21}	
F_3	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}	k_{21}	
F_4	k_7	k_8	k_9	k_{10}	k_{15}	k_{20}	
F_5	k_4	k_5	k_6	k_{10}	k_{14}	k_{19}	
F_6	k_2	k_3	k_6	k_9	k_{13}	k_{18}	
F_7	k_1	k_3	k_5	k_8	k_{12}	k_{17}	k_{23}
F_8	k_1	k_2	k_4	k_7	k_{11}	k_{16}	k_{22}

Table 6.2: An example of recovering 23 keywords with threshold T=7 by the uniform (7,8)-set.

This example also explains our major motivation: a single uniform (s,n)-set of the keyword set may not maximize the ability of a file injection attack, or in other words, the number of injected files n is not optimal. The reason is the number of keywords in injected files may be far from reaching the threshold.

6.3. A NEW FILE-INJECTION ATTACK

In this chapter, we present our new file-injection attack on searchable encryption schemes. It is based on our new definition of a subset family of a finite set, called *increment* $[r,n]$*-set*. Our main technique is to construct an increment $[r,n]$-set of the keyword set. Compared to the uniform (s,n)-set used in [18], the increment $[r,n]$-set enables us to put more keywords in the injected files, thus significantly reducing the number of injected files.

6.3.1. INCREMENT $[r,n]$-SET

The main idea of the increment $[r,n]$-set is to optimize the available space in the injected files, which are defined as follows.

Definition 6.3.1 (Increment $[r,n]$-set)**.** *Let A be a set, then the subsets $A_1, A_2, \cdots, A_n \subset A$ are called an increment $[r,n]$-set of A if the following conditions are satisfied:*

- $|A_1| = |A_2| = \cdots = |A_{n-r+1}|$;

- *Elements in $A_1, A_2, \cdots, A_n$ are separated into r blocks such that the i-th $(1 \le i \le r)$ block of $A_1, A_2, \cdots, A_n$ forms a uniform $(n-i+1, n)$-set of the union set of the i-th block of $A_1, A_2, \cdots, A_n$.*

An (s, n)-set is here a single block and the increment $[r, n]$-set consists out of multiple (s, n)-sets (blocks), where the r is increased per block. Recall that for a uniform (s, n)-set, a keyword can be uniquely recovered by $r = n - s + 1$ returned files. Therefore, the keywords in the i-th block of an increment $[r, n]$-set are determined by $n - (n - i + 1) + 1 = i$ files, since the i-th block is a uniform $(n - i + 1, n)$-set by definition. That is, the keywords in the 1st block can be represented by 1 file, the keywords in the 2nd block by 2 files, and so on. This is what we call an *increment*. We refer to Table 6.3 for a visual example.

We denote the i-th block of A_j by A_j^i for $1 \le j \le n$, and then we get the following corollary which follows from Lemma 6.2.2.

Corollary 6.3.2. *If $(A_1, A_2, \cdots, A_n)$ is an increment $[r, n]$-set of A, then we have*

$$|A_j^i| = \begin{pmatrix} n-1 \\ n-i \end{pmatrix},$$

for $1 \le i \le r, 1 \le j \le n$.

Therefore, we know that the size of A_j is $|A_j| = \sum_{i=1}^{s} \binom{n-1}{n-i}$ for $1 \le j \le n$. The main idea of our basic file-injection attack is to construct an increment $[r, n]$-set, instead of completely independent (s, n)-sets spread over different chunks of files, as FST does. We are aiming at reducing the total number of injected files n to as few files as possible. Keywords are recovered according to the different combinations of returned files (details are present in Section [6.3.2]).

We give an example of an increment $[r, n]$-set of the keyword set $\{k_1, k_2, \cdots, k_{23}\}$ with threshold seven for a comparison to the example in Table 6.2. We compute r and the minimum n such that

$$\begin{cases} \sum_{i=1}^{r} \binom{n-1}{n-i} \le 7 \\ \sum_{i=1}^{r} \binom{n}{n-i} \ge 23, \end{cases} \tag{6.2}$$

and then we get $r = 3$ and $n = 6$. The increment $[3, 6]$-set of the aimed keyword set is shown in Table 6.3. Compared to Example 6.2.3, it reduces the number of injected files from 8 to 6! Every space in these files is filled with keywords, while still controlling the total number of keywords within the threshold.

6.3.2. CONSTRUCTION OF INCREMENT $[r, n]$-SET

In this section, we present a way to construction of increment $[r, n]$-set of a finite set, which uses the method of constructing uniform (s, n)-set as a subroutine (we also provide a new construction of uniform (s, n)-set in the full version [28]).

	(6,6)-set	(5,6)-set					(4,6)-set
Files	Col_1	Col_2	Col_3	Col_4	Col_5	Col_6	Col_7
F_1	k_1	k_7	k_{10}	k_{13}	k_{15}	k_{19}	k_{22}
F_2	k_2	k_7	k_{11}	k_{14}	k_{16}	k_{20}	k_{22}
F_3	k_3	k_8	k_{11}	k_{13}	k_{17}	k_{21}	k_{22}
F_4	k_4	k_8	k_{12}	k_{14}	k_{18}	k_{19}	k_{23}
F_5	k_5	k_9	k_{12}	k_{15}	k_{17}	k_{20}	k_{23}
F_6	k_6	k_9	k_{10}	k_{16}	k_{18}	k_{21}	k_{23}

Table 6.3: An example of recovering 23 keywords with threshold T=7 by an increment [3,6]-set, which is divided into 3 blocks. Keywords in the 1st, 2nd, and 3rd block can be recovered by 1, 2, and 3 returned files, respectively.

Given as input the size of the keyword m and threshold of the number of keywords in a file T, we aim to construct an increment $[r,n]$-set of the keyword set with the minimum n such that (1) the size of each file should not be greater than the threshold T, and (2) the maximal number of keywords that those files can recover is at least m. To maximize the recovery ability under condition (1), our overall idea is to construct r uniform $(n-i+1,n)$-sets for $1 \le i \le r$ and return the first T columns as the aimed set. Then by Lemma 6.2.2, we know the first $r-1$ blocks take $\sum_{i=1}^{r-1}\binom{n-1}{n-i}$ columns and can recover $\sum_{i=1}^{r-1}\binom{n}{n-i}$ keywords in total. The last block takes the rest $T-\sum_{i=1}^{r-1}\binom{n-1}{n-i}$ columns and allows to recover $\lfloor n/r \cdot [T-\sum_{i=1}^{r-1}\binom{n-1}{n-i}]\rfloor$ keywords. Then the condition (2) is equal to

$$\sum_{i=1}^{r-1}\binom{n}{n-i} + \left\lfloor \frac{n}{r} \cdot \left[T - \sum_{i=1}^{r-1}\binom{n-1}{n-i} \right] \right\rfloor \ge m. \tag{6.3}$$

We proceed in the discussion of r starting from 1 to T. For each r, we record all the possible n to the Inequality 6.3, with the minimum one as the optimal solution. For simplicity of exposition, we denote $NK(r,n)$ as the left part of the above inequality. The whole process of constructing an increment $[r,n]$-set is present in Algorithm 9.

Going back to Example 6.2.3, we compute the Inequality 6.3 to get $candidate = [(2,7),(3,6),(4,7)]$. Then we know the optimal increment $[r,n]$-set is $r=3$, and $n=6$.

6.3.3. BINOMIAL-ATTACK

Given the keyword universe $\mathbb{K} = \{k_1, k_2, \cdots, k_m\}$ and the threshold T as the maximal number of keywords in each file, we present our file injection attack in Algorithm 10, which is based on the increment $[r,n]$-set of the $\mathbb{K}$.

Now that the structure of the attack is understood, we can proceed to calculate the required number of injections to achieve the desired number of identifiable keywords. There are multiple formulas to calculate the required number of injections.

Algorithm 9: Construction of Increment $[r, n]$-Set

Input: Number of keywords m, threshold T.
Output: An increment $[r, n]$-set of the keyword set $\{k_1, k_2, \cdots, k_m\}$.

Initialize an empty candidate set: $candidate \leftarrow [\,]$
for $r = 1$ **to** T **do**
  Solve n from $NK(r, n) \geq T$ and denote the minimum n as n_0
  Append (r, n_0) to $candidate$
end
Find (r, n_0) with the minimum n_0 and corresponding r from $candidate$
for $i = 1$ **to** r **do**
  Construct a uniform $(n_0 - i + 1, n_0)$-set of keywords with index from $\sum_{j=1}^{i-1} \binom{n_0-1}{n_0-j}$ to $\sum_{j=1}^{i} \binom{n_0-1}{n_0-j}$ using the technique proposed in [28]
end
return the first T columns of the created files

Algorithm 10: Binomial-Attack

Input: Keyword set $\mathbb{K} = \{k_1, k_2, \cdots, k_m\}$, threshold T, a query token t.
Output: Keyword corresponding to the token t.

Generate an increment $[r, n]$-set $A_1, A_2, \cdots, A_n$ of $\mathbb{K}$ with threshold T using Algorithm 9
for $j = 1$ **to** n **do**
  Create a file D_j containing the same keywords as A_j
end
Inject files $\{F_1, F_2, \cdots, F_n\}$ into the SE scheme
return the corresponding keyword to t based on the returned i files $(1 \leq i \leq r)$

The appropriate formula to utilize depends on at which $(n-r+1, n)$-set the threshold will limit the attack from injecting more combinations.

Deciding the Number of Injections.

The attack always starts with $r = 1$ and progresses incrementally from there on forth. At some point within an $(n-r+1, n)$-set, the threshold will limit the number of keywords it can inject. Refer to Table 6.3 for a visual representation.

By Eq. 6.3 we know the number of keywords an $(n-r+1, n)$-set can utilize for a certain threshold, under a specific r. When the threshold is reached in the $(n-1, n)$-set, where $r = 2$ the equation can be written in terms of n like the following:

$$F_2(K, T) = \frac{2K}{T+1} \tag{6.4}$$

Similarly to the $(n-2, n)$-set, where $r = 3$, the equation becomes:

$$F_3(K, T) = \frac{-(3+2T) + \sqrt{(3+2T)^2 + 24K}}{2} \tag{6.5}$$

The formulas $F_4(K, T)$ and beyond are only of relevance when the threshold is a significant portion of the number of keywords that need to be injected.

Deciding the Injection Formulas.

The next step involves determining the appropriate utilization of each formula for different scenarios.

To determine the appropriate value for r in the increment $[r,n]$-set, we must assess whether the threshold allows for additional keywords in the files following a uniform $(n-r+1,n)$-set. This evaluation must be conducted for each r, commencing at $r=2$. By Lemma 6.2.2 we know the first two blocks utilize a total of n columns. Therefore if $T>n$, the $(n-2,n)$-set can also be used. However, the value of n remains unknown at this stage. To address this uncertainty, we substitute $n=T$ into F_2. This yields the threshold at which both the $(n-r+1,n)$-sets in the increment $[2,n]$-set become uniform and precisely meet the threshold. If the keyword universe exceeds this value, the attack will require more than T injections. Conversely, if the keyword universe falls below this value, fewer than T injections are required. Consequently, there will be residual space in the injected files for (at least) the $(n-2,n)$-set. The minimum value of K to only be able to build up to an Increment $[2,n]$-set is outlined as follows:

$$Min_{F_2}(T)=\frac{1}{2}T^2+\frac{1}{2}T \tag{6.6}$$

F_2 should be applied when the outcome of $Min_{F_2}(T)\leq K$. Alternatively, the formula of Min_{F_3} determines whether F_3 or F_4 should be utilized. Following the same procedures as before, we get:

$$Min_{F_3}(T)=\frac{2+T}{3}\cdot\frac{1+\sqrt{8T-7}}{2}+\frac{T-1}{3} \tag{6.7}$$

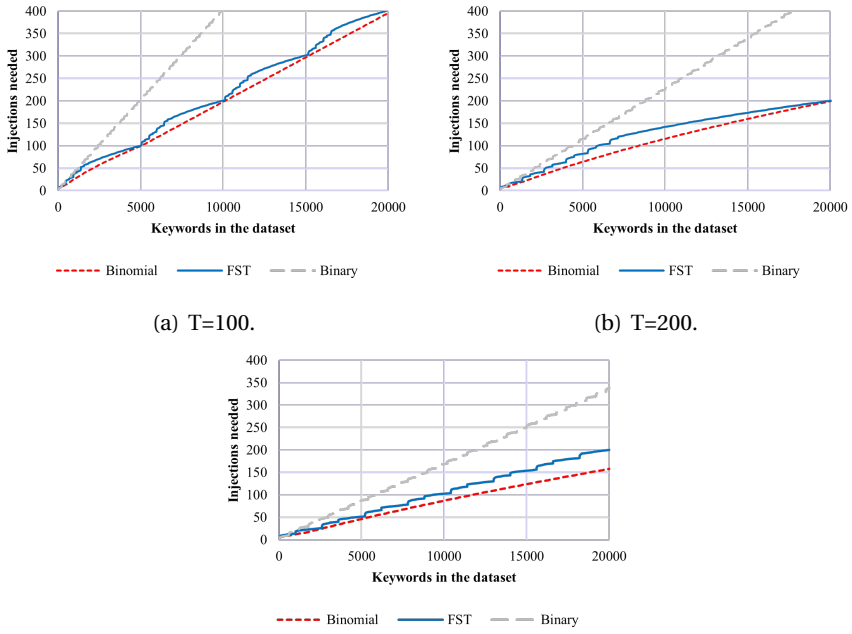
These formulas already hold an improvement over FST, since FST had a lookup table with overlapping values and no clear points to choose from. Using our previous example 6.2.3, we see $Min_{F_2}(7)>23$ and $Min_{F_3}(7)<23$. This means we need to use F_3, which results in $F_3(23,7)=6$ files.

6.3.4. Performance under Different Thresholds

The results consistently demonstrate the superiority of the Binomial-attack over the FST-attack across various thresholds and dataset sizes.

The Binomial-attack consistently outperforms the FST-attack with at least one injected file, regardless of the dataset size. With a threshold of 200, the most substantial disparity occurs in datasets ranging from 7 200 to 7 400 keywords, where the FST-attack requires 33 more injections compared to the Binomial-attack. This represents a 38% increase in injections needed by the FST-attack for equivalent results. In previous studies, the Enron dataset [29] served as a benchmark for attack performance. When applied to the Enron dataset, the FST-attack requires 83 files to cover the entire keyword universe, whereas the Binomial-attack accomplishes this with only 65 files. Thus, in this real dataset scenario, the FST-attack necessitates 28% more injections than the Binomial-attack.

To provide a comprehensive overview of these differences, Fig. 6.1 illustrates the comparative performance of the Binary-, FST-, and Binomial-attack across various thresholds, with datasets ranging up to 20 000 keywords.



(a) T=100.

(b) T=200.

(c) T=300.

Figure 6.1: Performance of different injection attacks under different thresholds.

6.4. File-injection Attacks on SE Schemes with Keyword Padding

Keyword padding serves as a countermeasure within the SE scheme aimed at obscuring query results by returning more files than necessary. In addition to the files containing the queried keyword, the scheme also includes random files from the dataset in its response. In this chapter, we delve into the consequences of padding and compare the implications between the FST- and Binomial-attack methodologies. Previous studies, such as the FST- and Binary-attack, explored this topic assuming a file dataset of 30 109 files and a keyword universe of 5 050 keywords. The scheme adopts a threshold of 200, and on average, a query yields matches on 560 files, with an additional 60% of random files included (336 files). Section 6.4.1 delves into the quantitative effects of padding, while Section 6.4.2 presents a visual exploration of these effects.

6.4.1. Calculating the Effects

To assess the impact, three key steps are necessary. Firstly, we must determine the average number of additional injected files returned as a consequence of their selection for padding. Subsequently, we can proceed to determine the average number of keyword combinations we can generate. These combinations represent distinct file arrangements utilized for the unique identification of a single keyword, collectively referred to as the candidate set for a query. Finally, the last step entails re-executing the attack on the candidate set to pinpoint the specific keyword utilized.

Injected Files from Padding.

To calculate the average number of injected files chosen during padding, we can utilize the hypergeometric distribution function. Our population size is $30109 - 560 = 29549$, since the matched files for the query can not be chosen for the padding. The number of successes will be $F3(5050, 200) = 64.8 \approx 65$ files, minus the average injected file response, leaving $65 - 3 = 62$ successes. The sample size is 336. We can calculate the probabilities for all possible numbers of successes in the sample and then multiply each probability by the corresponding number of successes. The results are then summed to determine the average number of injected files (p) chosen in the padding:

$$p = \sum_{n=1}^{62} \frac{\binom{62}{n}\binom{29549-62}{336-n}}{\binom{29549}{336}} \tag{6.8}$$

Average Candidate Set Size.

The average candidate set size is determined by three key factors associated with each $(n-r+1, n)$-set used to identify keywords. The first factor considers the number of possible combinations within the given $(n-r+1, n)$-set when $r+p$ injected files are returned. The second factor accounts for the ratio of combinations utilized in that $(n-r+1, n)$-set compared to its total possible combinations. The third factor represents the ratio of identifiable keywords in the $(n-r+1, n)$-set to the total number of identifiable keywords. Multiplying these three factors together yields the average candidate set size per $(n-r+1, n)$-set. Summing the results across all $(n-r+1, n)$-sets provides the overall average candidate set size:

$$\sum_{r=1}^{R} \binom{r+p}{r} \cdot \frac{|K_r|^2}{\binom{n}{r} \cdot |K_R|} \tag{6.9}$$

Number of Extra Injections Needed.

A straightforward method to determine the number of extra injections required is to analyze on a per-query basis. By considering the average candidate set size per query, we can execute our attack specifically for that particular candidate set to recover the searched keyword. While this approach is not optimal, it suffices for comparison purposes with the FST-attack.

6.4.2. VISUALISING THE EFFECTS

This section will demonstrate the effects of padding on both FST and the Binomial-attack. While the Binomial-attack may not always appear significantly better based solely on the average candidate set size per query, it's important to consider that FST is generally less efficient, requiring more injections to cover the same candidate set. Here, we present the results for a scheme with a threshold of 200 in Fig. 6.2. Results for different thresholds are available in Figures 6.4, 6.5, 6.6, 6.7, 6.8.

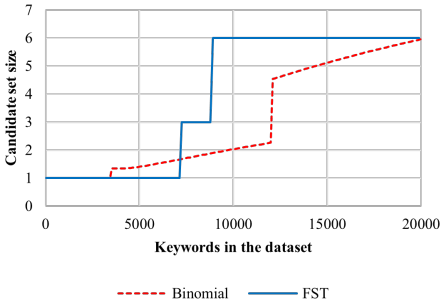


Figure 6.2: Candidate set size per query, T=200.

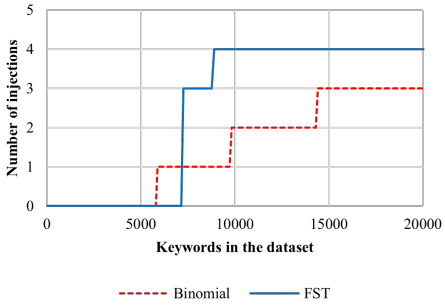


Figure 6.3: Extra injection size per query, T=200.

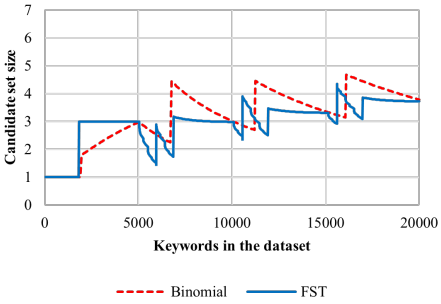
TARGETING THE WHOLE DATASET.

In Fig. 6.2, we see the average sizes of candidate sets for different dataset sizes. The corresponding number of extra injections required for the candidate sets is illustrated in Fig. 6.3. While there is a small dataset size range where the Binomial-attack requires one more injection than the FST-attack, FST generally performs worse for all other dataset sizes.

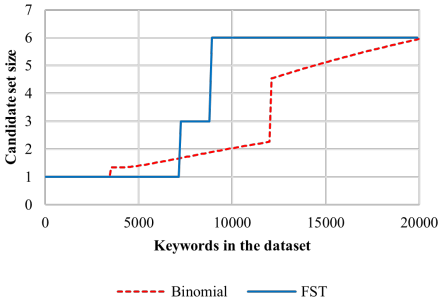
TARGETING A SUBSET OF THE DATASET.

When targeting a subset of the keyword universe, fewer injections are required to cover the target set, benefiting both attacks. However, not every query relates to a keyword in the target set. When combined with padding, this may not pose an issue if we assume a consistent average number of injected files in the padding. For instance, if two injected files are returned and the average padding injection is also two, it suggests a search for a keyword not in the target set. However, if a return of two injected files could also indicate a search for a keyword occurring once or twice, all searches become candidate sets. While these candidate sets may not contain actual keywords from the target set, distinguishing beforehand is impossible. The only option is to re-perform the attack on the candidate set.

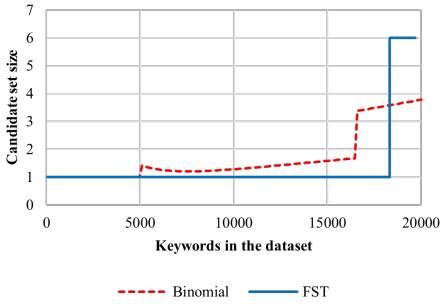
In this scenario, our attack performs notably worse. This is because the Binomial-attack initiates with an (n, n)-set. FST does not follow this approach,



(a) T=100.

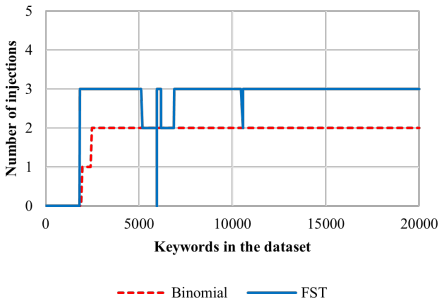


(b) T=200.

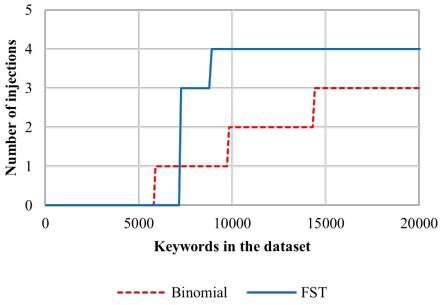


(c) T=300.

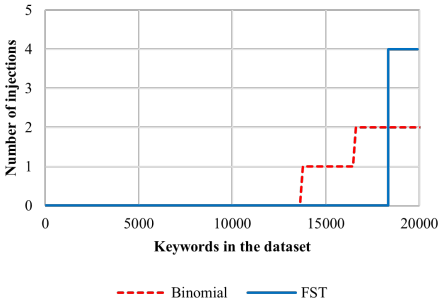
Figure 6.4: Candidate set sizes per query when padding is applied, under different thresholds, where the target set is the full keyword universe, for the standard Binomial-attack.



(a) T=100.

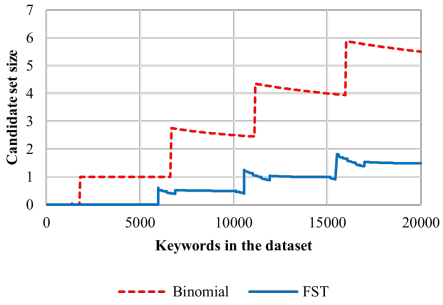


(b) T=200.

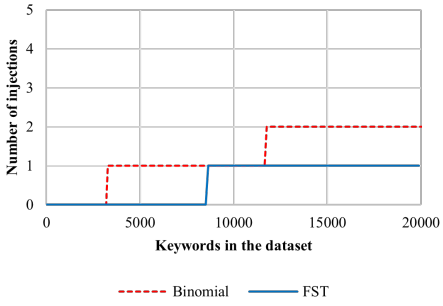


(c) T=300.

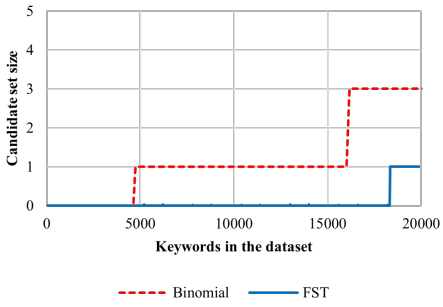
Figure 6.5: Extra injection sizes per query when padding is applied, under different thresholds, where the target set is the full keyword universe, for the standard Binomial-attack.



(a) T=100.

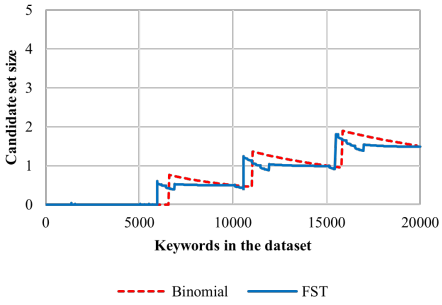


(b) T=200.

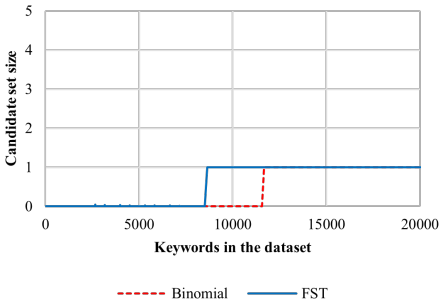


(c) T=300.

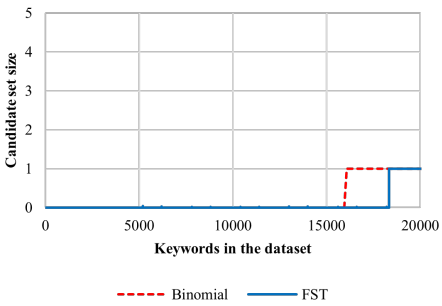
Figure 6.6: Candidate set sizes per query that is not in the target set when padding is applied, under different thresholds, where the target set is a subset of the full keyword universe, for the standard Binomial-attack.



(a) T=100.

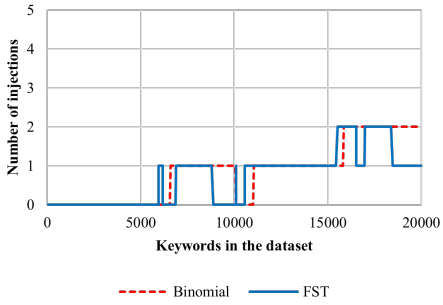


(b) T=200.

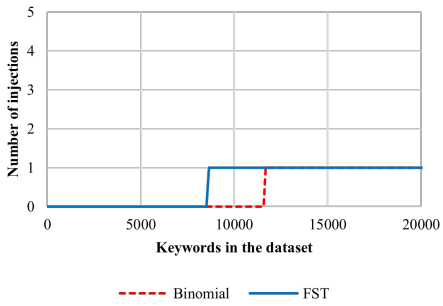


(c) T=300.

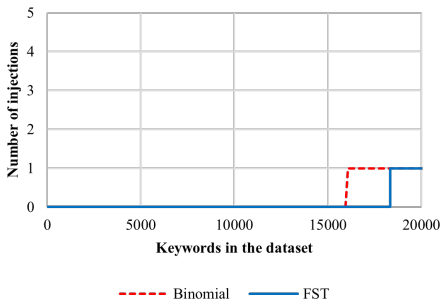
Figure 6.7: Candidate set sizes per query that is not in the target set when padding is applied, under different thresholds, where the target set is a subset of the full keyword universe, for the adopted Binomial-attack.



(a) T=100.



(b) T=200.



(c) T=300.

Figure 6.8: Extra injection sizes per query that is not in the target set when padding is applied, under different thresholds, where the target set is a subset of the full keyword universe, for the adopted Binomial-attack.

resulting in fewer potential combinations when all preceding $(n-r+1,n)$-sets are included in the candidate set. Figure 6.9 illustrates the number of extra injections required when searching for a keyword that is not in the target set.

6.5. ADOPTED BINOMIAL-ATTACK

When the target set is a subset of the dataset, searches for keywords outside the target set result in additional candidate sets. To mitigate the size of these extra candidate sets, adjustments to the attack methodology are necessary. This chapter outlines the modifications required to minimize candidate size while maintaining effectiveness. Despite the trade-off, the attack consistently requires fewer initial injections than FST.

6.5.1. REMOVING THE (n,n)-SET

In the Binomial-attack, the lowest value for r is always one. While this minimizes the space occupied in injected files, it also leads to greater overlap with keywords spread across multiple injected files. Conversely, higher values of r in the $(n-r+1,n)$-sets for all keywords result in smaller candidate sets per query. To reduce the size of candidate sets, keywords should not be identified with only one injected file, meaning the attack starts from $(n-1,n)$ instead of (n,n). This frees up space that can be allocated to a different $(n-r+1,n)$-set.

6.5.2. RESULTS AFTER THE MITIGATION

The number of identifiable keywords decreases by either $\frac{n}{2}$ or $\frac{2n}{3}$, depending on which $(n-r+1,n)$-set the attack terminates due to the threshold. Refer to Table 6.4 for a visual representation of this transformation.

In Fig. 6.10, the difference in extra injections required between the FST- and adopted Binomial-attack is illustrated. FST consistently requires an equal or greater number of injections to recover candidate sets.

	(5,6)-**set**					(4,6)-**set**	
Files	Col_1	Col_2	Col_3	Col_4	Col_5	Col_6	Col_7
F_1	k_1	k_4	k_7	k_9	k_{13}	k_{16}	k_{19}
F_2	k_1	k_5	k_8	k_{10}	k_{14}	k_{16}	k_{18}
F_3	k_2	k_5	k_7	k_{11}	k_{15}	k_{16}	k_{18}
F_4	k_2	k_6	k_8	k_{12}	k_{13}	k_{17}	k_{18}
F_5	k_3	k_6	k_9	k_{11}	k_{14}	k_{17}	k_{19}
F_6	k_3	k_4	k_{10}	k_{12}	k_{15}	k_{17}	k_{19}

Table 6.4: Distribution of an Increment [3,6]-set, without (6,6)-set, T=7.

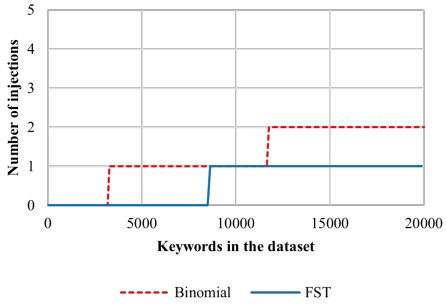


Figure 6.9: Extra injection sizes per query that is not in the target set, T=200.

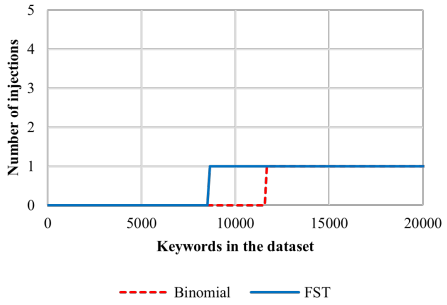


Figure 6.10: Extra injection sizes per query that is not in the target set, T=200, for the adopted attack.

6.6. DISCUSSION

In addition to padding, there exist other countermeasures aimed at increasing the difficulty of attacks. One such countermeasure involves the creation of clusters of keywords, as described in [11]. When a search query is initiated for one of the keywords within a cluster, all files containing keywords from the same cluster are returned. This approach not only obscures the specific keyword being searched for, but also introduces ambiguity regarding the association of injected files with specific keywords. Due to the potential for multiple combinations of keywords within the returned files, the attacker may be compelled to employ higher $(n-r+1,n)$-sets, necessitating a greater number of injected files. It is important to note, that this countermeasure assumes a static keyword universe and may require modification to accommodate dynamic searchable encryption scenarios.

Despite its theoretical appeal, searchable encryption has yet to achieve widespread adoption in practical applications and can vary significantly in its configurations, including the implementation of countermeasures. Consequently, predicting the exact characteristics of a searchable encryption scheme in practice remains challenging. Nevertheless, there is value in speculating on the potential implications of different settings and attempting to assess the scheme's security under various conditions, even if these scenarios remain largely theoretical at present. This makes it harder to determine how big the safety issues of the schemes are.

6.7. FUTURE WORK

The additional injections required to neutralize candidate sets are primarily utilized to compare the attack against FST. However, the method itself is far from optimal. As presented in this paper and the FST paper, each keyword necessitates multiple additional injections. This approach may result in a greater number of injections than initially required for the attack. A more efficient strategy involves combining candidate sets and reusing earlier injections, thereby reducing the overall number

of additional injections required. However, the optimal method for achieving this remains to be determined.

This attack is an active attack that makes no use of leakage apart from the returned injected files. In contrast, other attacks combine active and passive methods [19]. If Binary- or FST-attack methods are employed, they could be enhanced by incorporating the Binomial-attack. Revisiting these attacks may reveal potential improvements. We also note that further exploration into fields such as coding theory and combinatorics using our increment $[r,n]$-set could yield relevant connections and contributions and vice-versa.

6.8. CONCLUSION

The Binomial-attack represents a significant advancement over existing active attack methods. It maximizes the storage of keywords within a limited number of injected files by employing an Increment $[r,n]$-set to identify keywords. This approach iterates through all possible combinations of an $(n-r+1,n)$-set starting from $r=1$, progressing with $r=r+1$ until no additional space is available in the files. The adopted Binomial-attack starts at $r=2$ to decrease the candidate set size for a query when the SE scheme uses padding as a countermeasure.

Our findings demonstrate that, regardless of the presence or absence of a threshold, the Binomial-attack consistently outperforms both the Binary- and FST-attack methods. However, when padding is introduced, there are specific threshold and dataset size combinations where FST requires fewer additional injections on average. It remains uncertain whether this advantage would persist with the implementation of a more efficient keyword recovery method.

REFERENCES

[1] D. X. Song, D. A. Wagner, and A. Perrig. “Practical Techniques for Searches on Encrypted Data”. In: *IEEE S&P 2000*. IEEE, 2000, pp. 44–55. DOI: 10.1109/SECPRI.2000.848445.

[2] R. Bost, B. Minaud, and O. Ohrimenko. “Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives”. In: *ACM CCS 2017*. Ed. by B. Thuraisingham, D. Evans, T. Malkin, and D. Xu. ACM, 2017, pp. 1465–1482. DOI: 10.1145/3133956.3133980.

[3] D. Cash and S. Tessaro. “The Locality of Searchable Symmetric Encryption”. In: *EUROCRYPT 2014*. Ed. by P. Q. Nguyen and E. Oswald. Vol. 8441. LNCS. Springer, 2014, pp. 351–368. DOI: 10.1007/978-3-642-55220-5_20.

[4] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili. “New Constructions for Forward and Backward Private Symmetric Searchable Encryption”. In: *CCS 2018*. Ed. by D. Lie, M. Mannan, M. Backes, and X. Wang. ACM, 2018, pp. 1038–1055. DOI: 10.1145/3243734.3243833.

[5] S. Kamara and T. Moataz. “Boolean Searchable Symmetric Encryption with Worst-Case Sub-linear Complexity”. In: *EUROCRYPT 2017, Part III*. Ed. by J. Coron and J. B. Nielsen. Vol. 10212. LNCS. 2017, pp. 94–124. DOI: 10.1007/978-3-319-56617-7_4.

[6] S. Patel, G. Persiano, and K. Yeo. *Symmetric Searchable Encryption with Sharing and Unsharing*. Ed. by J. López, J. Zhou, and M. Soriano. 2018. DOI: 10.1007/978-3-319-98989-1_11.

[7] S. Sun, X. Yuan, J. K. Liu, R. Steinfeld, A. Sakzad, V. Vo, and S. Nepal. “Practical Backward-Secure Searchable Encryption from Symmetric Puncturable Encryption”. In: *CCS 2018*. Ed. by D. Lie, M. Mannan, M. Backes, and X. Wang. ACM, 2018, pp. 763–780. DOI: 10.1145/3243734.3243782.

[8] M. Naveed. “The Fallacy of Composition of Oblivious RAM and Searchable Encryption”. In: *IACR Cryptol. ePrint Arch.* 2015 (2015), p. 668. URL: https://api.semanticscholar.org/CorpusID:11042885.

[9] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. “Leakage-Abuse Attacks Against Searchable Encryption”. In: *ACM CCS 2015*. Ed. by I. Ray, N. Li, and C. Kruegel. ACM, 2015, pp. 668–679. DOI: 10.1145/2810103.2813700.

[10] M. S. Islam, M. Kuzu, and M. Kantarcioglu. “Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation”. In: *NDSS 2012*. The Internet Society, 2012. URL: https://www.ndss-symposium.org/ndss2012/access-pattern-disclosure-searchable-encryption-ramification-attack-and-mitigation.

[11] C. Liu, L. Zhu, M. Wang, and Y. Tan. “Search pattern leakage in searchable encryption: Attacks and new construction”. In: *Information Sciences* 265 (2014), pp. 176–188. ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.11.021.

[12] L. Blackstone, S. Kamara, and T. Moataz. “Revisiting Leakage Abuse Attacks”. In: *NDSS 2020*. The Internet Society, 2020. DOI: 10.14722/ndss.2020.23103.

[13] J. Ning, X. Huang, G. S. Poh, J. Yuan, Y. Li, J. Weng, and R. H. Deng. “LEAP: Leakage-Abuse Attack on Efficiently Deployable, Efficiently Searchable Encryption with Partially Known Dataset”. In: *ACM CCS 2021*. Ed. by Y. Kim, J. Kim, G. Vigna, and E. Shi. ACM, 2021, pp. 2307–2320. DOI: 10.1145/3460120.3484540.

[14] D. Pouliot and C. V. Wright. “The Shadow Nemesis: Inference Attacks on Efficiently Deployable, Efficiently Searchable Encryption”. In: *ACM SIGSAC 2016*. Ed. by E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi. ACM, 2016, pp. 1341–1352. DOI: 10.1145/2976749.2978401.

[15] S. Oya and F. Kerschbaum. “Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption”. In: *USENIX 2021*. Ed. by M. D. Bailey and R. Greenstadt. USENIX Association, 2021, pp. 127–142.

[16] M. Damie, F. Hahn, and A. Peter. “A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents”. In: *USENIX 2021*. Ed. by M. Bailey and R. Greenstadt. USENIX Association, 2021, pp. 143–160. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/damie.

[17] Y. Zhang, J. Katz, and C. Papamanthou. “All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption”. In: *USENIX 2016*. Ed. by T. Holz and S. Savage. USENIX Association, 2016, pp. 707–720. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/zhang.

[18] G. Wang, Z. Cao, and X. Dong. “Improved File-injection Attacks on Searchable Encryption Using Finite Set Theory”. In: *Comput. J.* 64.8 (2021), pp. 1264–1276. DOI: 10.1093/COMJNL/BXAA161.

[19] X. Zhang, W. Wang, P. Xu, L. T. Yang, and K. Liang. “High Recovery with Fewer Injections: Practical Binary Volumetric Injection Attacks against Dynamic Searchable Encryption”. In: *USENIX Security 2023*. Ed. by J. A. Calandrino and C. Troncoso. USENIX Association, 2023, pp. 5953–5970.

[20] R. Poddar, S. Wang, J. Lu, and R. A. Popa. “Practical Volume-Based Attacks on Encrypted Databases”. In: *IEEE European Symposium on Security and Privacy, EuroS&P 2020*. IEEE, 2020, pp. 354–369. DOI: 10.1109/EUROSP48549.2020.00030.

[21] H. Nie, W. Wang, P. Xu, X. Zhang, L. T. Yang, and K. Liang. “Query Recovery from Easy to Hard: Jigsaw Attack against SSE”. In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by D. Balzarotti and W. Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/nie.

[22] M. Zhang, Z. Shi, H. Chen, and K. Liang. “Inject Less, Recover More: Unlocking the Potential of Document Recovery in Injection Attacks Against SSE”. In: *IACR Cryptol. ePrint Arch.* (2024), p. 515. URL: https://eprint.iacr.org/2024/515.

[23] B. Ho, H. Chen, Z. Shi, and K. Liang. “Similar Data is Powerful: Enhancing Inference Attacks on SSE with Volume Leakages”. In: *IACR Cryptol. ePrint Arch.* (2024), p. 516. URL: https://eprint.iacr.org/2024/516.

[24] J. Ning, J. Xu, K. Liang, F. Zhang, and E. Chang. “Passive Attacks Against Searchable Encryption”. In: *IEEE Trans. Inf. Forensics Secur.* 14.3 (2019), pp. 789–802. DOI: 10.1109/TIFS.2018.2866321. URL: https://doi.org/10.1109/TIFS.2018.2866321.

[25] T. Chen, P. Xu, S. Picek, B. Luo, W. Susilo, H. Jin, and K. Liang. “The Power of Bamboo: On the Post-Compromise Security for Searchable Symmetric Encryption”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/the-power-of-bamboo-on-the-post-compromise-security-for-searchable-symmetric-encryption/.

[26] D. Liu, W. Wang, P. Xu, L. T. Yang, B. Luo, and K. Liang. “d-DSE: Distinct Dynamic Searchable Encryption Resisting Volume Leakage in Encrypted Databases”. In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by D. Balzarotti and W. Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/liu-dongli.

[27] R. Liu and Z. F. Cao. “Two new methods of distributive management of cryptographic key”. In: J. Commun., 8, 1987, pp. 10–14.

[28] T. Langhout, H. Chen, and K. Liang. “File-Injection Attacks on Searchable Encryption, Bases on Binomial Structures”. In: *IACR Cryptol. ePrint Arch.* (2024). URL: https://eprint.iacr.org/2024/1000.

[29] C. William W. Cohen MLD. *Enron Email Datasets*. 2015. URL: https://www.cs.cmu.edu/~enron/.

7

DISCUSSION

With the rapid development of the big data industry, the volume of data generated every day is explosively increasing, raising the challenge of big data storage and usage. A perfect way to deal with this challenge is by outsourcing the huge data storage and complicated computation tasks to a cloud, such as Google, Amazon, and Microsoft. This eliminates the expensive cost of purchasing hardware and software; however, it increases concerns about data privacy, considering the curiosity of cloud servers and potential attacks from external sources. To alleviate these concerns, users should encrypt the data with a secret key before uploading it to the cloud.

The security of cryptographic systems ensures the privacy of data stored on the cloud. However, practical challenges arise when working with encrypted cloud data, such as key management for long-term storage, computations over encrypted data, the usage of encrypted data, encrypted cloud migration, and ensuring quantum security. This thesis investigates three main research topics: updatable encryption, fully homomorphic encryption and searchable encryption, to address the security of long-term stored data, computing over encrypted data, and the search functionality on encrypted data, respectively. Additionally, we propose constructions for quantum-secure UE and FHE schemes.

In this chapter, we present our findings, discuss their limitations, and propose potential future work. We will address the challenges presented in Section 1.2 and the research questions listed in Section 1.3. Our research is organized into three topics: Updatable Encryption, Fully Homomorphic Encryption, and Searchable Encryption.

7.1. UPDATABLE ENCRYPTION

In this section, we explain the first two research questions related to updatable encryptions and our contributions in Chapters 2 and 3. Firstly, we investigate all the existing work on security notions for UE schemes and provide a clear relationship among those notions. Secondly, we introduce a stronger notion of adaptive security for ciphertext-dependent UE schemes and propose a quantum-secure UE construction with the desired security.

CIPHERTEXT-INDEPENDENT UPDATABLE ENCRYPTION

In Chapter 2, we address the following question:

Q1: *What are the relations among all existing security notions for updatable encryption schemes?*

In each security game, the adversary has access to various oracles, allowing it to corrupt information about epoch keys, update tokens, and ciphertexts from the challenger. In the confidentiality game, the adversary submits a challenge message and a challenge ciphertext based on its available information. It then receives a ciphertext from the challenger and must determine whether it is an encryption of the challenge message or an update of the challenge ciphertext. The adversary can continue querying oracles before providing its final guess.

In the integrity game, the adversary's objective is to forge a valid ciphertext. In both security games, some combinations of oracles may lead to a trivial win for the adversary. Therefore, the challenger will check if those trivial win conditions are triggered during the game using a bookkeeping technique developed in [1]. We present the confidentiality game in Fig. 7.1.

To analyze the relationship between any two of the eight variants of each security notion, we construct a reduction $\mathscr{B}$. The reduction runs the security experiment of one variant while simulating responses to the adversary $\mathscr{A}$'s queries in the security experiment of the other variant and forwards $\mathscr{A}$'s guess to its own challenger. If $\mathscr{A}$ does not trigger any trivial win conditions, neither will $\mathscr{B}$, ensuring that $\mathscr{B}$'s advantage is at least as large as $\mathscr{A}$'s. Therefore, the relationship between the two variants depends on the relation of trivial win conditions in each update direction setting.

Our result clarifies the relations of UE in different update directions and shows a surprising finding: UE schemes with no-directional key updates are actually equivalent to those with backward-leak uni-directional key updates. At first glance, one may think that UE schemes with no-directional key updates should be strictly stronger than those with any of the other three key update directions, just as it was proved in [2] that no-directional key updates leak less information about keys, tokens, and ciphertexts than bi-directional and forward-leak uni-directional key updates. Based on our result, when analyzing the security notions, we can treat UE schemes with no-directional key updates as those with backward-leak uni-directional key updates.

LIMITATIONS AND FUTURE WORK

We theoretically provide detailed comparisons among different security notions, but the efficient construction of the strongest no-directional c-i UE remains an open problem. The difficulty arises from the requirement that update tokens should not reveal any information about either the old key or the new key, yet they must allow for updating all ciphertexts under the old key. Currently, only two c-i UE schemes with no-directional key updates have been proposed. One is presented by Slamanig [3], which is based on the SXDH assumption, thus requiring expensive exponential

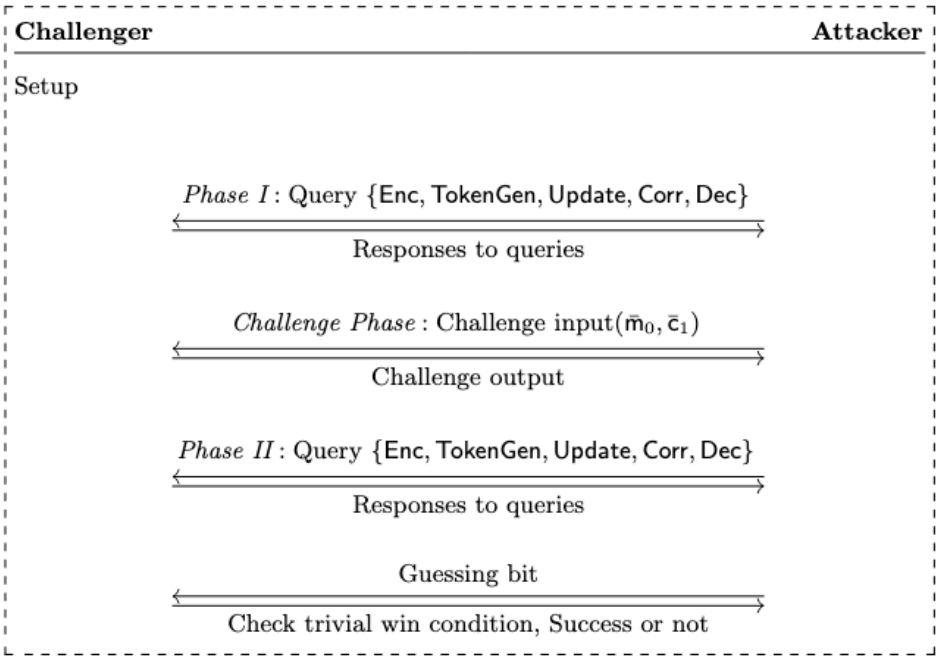


Figure 7.1: The confidentiality game for UE schemes. During the query phase, the adversary is provided with various oracles to infer secret information about the scheme. The goal of the adversary is to distinguish between the fresh encryption and the updated ciphertext. Trivial win conditions are checked at the end of the game.

operations. The other is introduced by Nishimaki [4], which relies on the existence of indistinguishability obfuscation but remains purely theoretical.

Another promising direction for future work is to extend the analysis techniques for security notions developed in this thesis to other cryptographic primitives, such as proxy re-encryption, identity-based encryption (IBE), and attribute-based encryption (ABE). In the security definitions of these schemes, they share a similar challenge game, as shown in Fig. 7.1. The adversary is provided with oracles related to the syntax of each scheme and aims to distinguish between two ciphertexts (confidentiality game) or provide a valid ciphertext (integrity game). Moreover, these schemes all have a similar syntax that transforms ciphertexts from one key to another, or one node to another. Therefore, the techniques and insights developed here could inspire stronger and more comprehensive security notions for proxy re-encryption, IBE, and ABE.

CIPHERTEXT-DEPENDENT UPDATABLE ENCRYPTION

In Chapter 3, we address the following question:

Q2: *How to build a more secure and more efficient ciphertext-dependent updatable encryption scheme?*

Existing works of c-d UE are not sufficient in analyzing the maximal ability of the adversary. It is reflected in two aspects: the decryption ability of the adversary is not considered in the confidentiality notions, resulting in a CPA-like confidentiality, not CCA security; meanwhile, the adversary is only allowed to select some epoch keys to corrupt in the start of the security game, rather than corrupting private information at any time during the game, which thus leads to only selective security, not adaptive security. Moreover, the quantum security of UE schemes is not considered in prior works on c-d UE.

In our work, we overcome the above limitations by proposing a lattice-based c-d UE scheme. To achieve this, we first build a new CCA-1 PKE scheme inspired by [5], which is based on the lattice trapdoor techniques and serves as the underlying encryption scheme for our UE scheme. The PKE ciphertext contains an invertible matrix as the header part and a LWE sample as the payload part, with the former of which the plaintext can be decrypted correctly from the latter. Its security is based on the hardness of LWE problems. We state that the PKE scheme cannot achieve CCA-2 security as other LWE encryption schemes, as the decryption of a challenge ciphertext with extra small noise, which is also a valid ciphertext, reveals the information of the challenge plaintext.

For the UE construction, our goal is to transform our PKE ciphertext from one key to another key. We propose a new update methodology by generating a so-called key-switching matrix. Directly multiplying such matrix with the old ciphertext would result in a new PKE ciphertext under the new key. This matrix can only be generated for the data owner who holds the old key, which is the trapdoor for some blocks of the ciphertext and is used to solve an LWE problem in the matrix generation. However, the key-switching matrix alone is not sufficient to achieve our confidentiality notion for UE, since the ciphertext unlinkability cannot be satisfied. We showed two improvements to tackle this: first, we improve a new random invertible matrix as the header of the new ciphertext, instead of using the same header as the old ciphertext; second, we improve the randomness used for encrypting plaintext in the ciphertext payload by adding a fresh encryption of message 0. We provide a detailed reduction proof of the security for our UE scheme.

We furthermore provide a packing technique that allows for the simultaneous encryption and update of multiple messages in a single ciphertext. By doing so, we alleviate the cost of downloading the ciphertext header required in the token generation process, cause only one ciphertext header needs to be downloaded during this process. Compared to the basic UE scheme, we provide a new encoding algorithm that encodes multiple messages into a polynomial ring. We show that messages can be decrypted degree by degree and we also generalize the token generation procedure in the polynomial ring setting.

LIMITATIONS AND FUTURE WORK.

Our PKE and UE schemes suffer from the same limitation as other LWE encryption schemes in that they cannot achieve CCA-2 security and integrity. This results from the fact that adding any small error to a ciphertext still results in a valid ciphertext, which obviously breaks the integrity. Additionally, querying the decryption oracle on the challenge ciphertext with an additional small error helps the adversary recover the underlying message, thus winning the CCA-2 game.

For PKE, this limitation can be efficiently addressed by using the Naor-Yung (NY) transform [6] or the Fujisaki-Okamoto (FO) transform [7], which is capable of turning a CPA-secure scheme into a CCA-2 secure one. This is also how Kyber works, which is one of the finalists in the NIST post-quantum cryptography project [8]. Kyber first designs a CPA-secure PKE, which is then used to apply the FO-transform to obtain a CCA-secure KEM. However, it is not straightforward to apply the NY/FO-transform to turn a CPA-secure UE scheme into a CCA-2 secure one. The difficulty lies in the proper design of the update algorithm such that it can be suitably adapted to the transform.

We conclude the research on UE with a challenging work. The first FHE scheme introduced by Gentry [9] and all its subsequent works require a "circular security" assumption, namely that it is safe to encrypt old secret keys with new keys. Such an idea has inspired the UE construction with backward directional key updates in [4]. In turn, we suggest an open problem that if no-directional updatable encryption, which is able to update ciphertext without revealing old and new keys, can be used to construct FHE that does not rely on the assumption.

7.2. FULLY HOMOMORPHIC ENCRYPTION

In this section, we discuss our contributions in Chapter 4 to address the following question:

Q3: *Can we build a batch programmable bootstrapping over large message space, within a polynomial modulus?*

Recent works aim to improve the efficiency of FHE through batch bootstrapping, which combines the advantage of low amortized cost from second-generation FHE and low noise growth from third-generation FHE. However, such schemes also inherit the limitations of third-generation FHE schemes, namely that they support only very small precision (limited to 4-5 bits). On the other hand, another advantage of third-generation FHE, called the programmable property, is missing in state-of-the-art works. Programmable bootstrapping (PBS) allows us to evaluate a univariate function simultaneously while refreshing the ciphertexts. It is therefore desirable to develop methods for large-precision batch bootstrapping with the programmable property. Moreover, it is also interesting to explore whether we can overcome the long-standing limitation of programmable bootstrapping, namely, the requirement that the function to be evaluated must be negacyclic.

In Chapter 4, we make two contributions to *batch programmable bootstrapping*

(batch PBS). Firstly, we propose a basic batch PBS over a more general message space $\mathbb{Z}_t$ for $t > 2$, which incurs only polynomial noise growth while maintaining the amortized cost as in state-of-the-art schemes. This method allows the evaluation of a univariate function over *multiple ciphertexts* simultaneously while refreshing the noise. The overall idea is to design a new look-up table suitable for our batch bootstrapping mechanism. This table is blindly rotated during noise refreshing, and the aimed value from the encrypted rotated table will be efficiently extracted in the final step. To overcome the negacyclicity requirement of PBS, we observe that this limitation arises from the polynomial setting. We solve this by working with ciphertexts in general cyclotomic rings, instead of the cyclotomic rings of powers of two used in PBS [10, 11]. This results in one more bit of precision than that in PBS, but it is still not practical for direct application. As an application, we demonstrate accurate evaluation of activation functions commonly used in Convolutional Neural Networks, including Sign, ReLU, and max in a batch setting.

Secondly, we optimize the basic batch PBS by introducing two homomorphic decomposition algorithms, which allow us to homomorphically decompose a large plaintext into several ciphertexts encrypting smaller chunks of the input plaintext. For each chunk, we can evaluate arbitrary functions using the basic batch PBS as a black box, leading to further improvements in precision. The first decomposition algorithm works for evaluating functions like Sign, where the value of the function is determined by only a few most significant bits of the message. The second algorithm is more suitable for functions whose value is not determined by only a few bits. These two decomposition algorithms further extend our batch programmable bootstrapping to support larger message spaces.

7

LIMITATIONS AND FUTURE WORK.

In practice, a client can delegate a complex computational task to a cloud service, which performs FHE operations on the encrypted data. After the computation, the client receives the encrypted result, decrypts it using their secret key, and obtains the value of the target function evaluated on their private data.

This scenario is promising but has two limitations in practice. Firstly, how can the client be convinced that the computation is correct, or in other words, how can we ensure the integrity of FHE? As summarized in [12], there are three techniques to construct verifiable FHE: Message Authentication Codes, Zero-Knowledge Proofs, and Trusted Execution Environments. However, these techniques are only useful if the verification process is lightweight for cloud clients, considering they have limited computational resources. Therefore, the construction of efficient *verifiable FHE* is an interesting direction. Secondly, this scenario only allows secure computing over private data from a single client. In practice, it is often necessary to process encrypted data from various sources, as the data can be encrypted by different clients with different keys. In such cases, we should use a generalized notion of HE, called Multi-key Homomorphic Encryption (MKHE). Several MKHE schemes have been proposed in the literature, such as multi-key TFHE [13, 14], multi-key BFV [15], and multi-key NTRU [16]. The difficulty lies in designing a hybrid product between a single-key ciphertext and a multi-key ciphertext. The single-key ciphertext serves

as the bootstrapping or key-switching key of a single party. The construction of more efficient multi-key homomorphic encryption, in terms of computational costs, ciphertext length, and bootstrapping key size, is a promising direction for future work.

The research in this thesis investigates two topics separately: updatable encryption and fully homomorphic encryption, to address two different application scenarios: long-term storage and computing over encrypted data. It is natural to consider combining these two schemes to provide broader applications. At first glance, key switching techniques in FHE could be applied to update ciphertexts, and FHE itself allows secure computation. However, a deeper investigation into the security and efficiency of this idea is necessary, which we leave for future work.

7.3. SEARCHABLE ENCRYPTION

In this section, we explain the research questions 4 and 5 related to searchable encryption and our contributions in Chapters 5 and 6. Firstly, our passive attack expands the matching techniques of the state-of-the-art LEAP attack [17] and further leverages the volume pattern to match a greater number of documents. Secondly, we propose a new active attack on SE to tackle the threshold countermeasures.

PASSIVE SE ATTACKS.

In Chapter 5, the following question is addressed.

Q4: *Can we design a passive SE attack by fully exploiting both the volume and access patterns to capture a high recovery rate?*

At a high level, our attack builds on the LEAP attack [17] by enhancing the keyword matching metric to increase the number of keyword matches. Each document is labeled with its document volume and the number of keywords it contains. Our attack leverages the uniqueness of this label for matching, significantly improving the recovery rate.

In detail, we first extend LEAP's matching technique by not only checking within the matched documents but also tracking occurrences in unmatched files. This approach recovers more keywords by addressing rows that do not uniquely occur in the matched files, thereby improving LEAP's performance. Additionally, we expand the attack by exploiting the volume pattern, as document sizes are leaked in response-leaking encryption schemes. By incorporating the volume pattern into our matching strategy, our attack becomes more comprehensive, enabling the matching of nearly all leaked documents.

This enhanced approach maximizes both document matches and keyword matches, fully utilizing the available leakage information to deliver excellent performance. Experimental results show that our new attack recovers approximately eight percentage points more files compared to the LEAP attack.

ACTIVE SE ATTACKS.

In Chapter 6, the following question is addressed.

Q5: *Can we design an active SE attack that is more practical when considering the threshold countermeasure?*

Countermeasures like thresholds and padding are employed to mitigate the success of attacks. Thresholds set a limit on the number of keywords a file can contain, while padding masks actual results by adding extra files alongside the queried files. Wang et al. [18] introduced a new approach to injection attacks, based on finite set theory, which outperforms earlier methods. Known as the FST-attack, this approach requires fewer injections than the Binary-attack under specific conditions by utilizing a so-called uniform (s,n)-sets to improve attack effectiveness, where integers s,n are parameters to balance the threshold and efficiency. However, a key limitation of the FST-attack is it is less practical under a large volume of keywords in the threshold setting.

We introduce a new file-injection attack on searchable encryption schemes, leveraging our novel definition of a subset family of a finite set, referred to as an increment $[r,n]$-set. The core technique involves constructing an increment $[r,n]$-set from the keyword set. Unlike the uniform (s,n)-set used in [18], the increment $[r,n]$-set allows for the inclusion of more keywords in the injected files, significantly reducing the required number of file injections. A high-level explanation of the difference between a uniform (s,n)-set and an increment $[r,n]$-set is that the latter consists of a sequence of (s_i,n)-sets, where s_i is an increasing integer from a predefined set $\{s_i\}$. Visually, the former attack can be imagined as filling a bottle with a fixed number of balls of uniform size, while our approach starts with the largest ball and gradually reduces its radius until it fits within the threshold. The experiment shows our attack reduces the number of injected files by up to 38% compared to the work by Wang et al. [18].

LIMITATIONS AND FUTURE WORK.

We present an efficient passive and active attack against searchable encryption, but the efficient construction of SE schemes that resist the proposed attacks remains an open problem. Although the padding technique, which ensures all documents are of the same size by adding padding characters, can mitigate both our passive and active attacks, it increases the workload on the client side, as it requires filtering out false positives. Additionally, padding can lead to storage and performance issues, as the user must wait for the program to identify and return the correct results.

Another promising area for future research in searchable encryption is the efficient construction of asymmetric searchable encryption (ASE). Note that in this thesis, we primarily focused on symmetric SE, where the query is generated using the same key that encrypts the document. However, a more practical scenario in applications is asymmetric SE (ASE), where the data owner encrypts the document using a public key, and the query is generated using the corresponding secret key. Constructing ASE with post-quantum security is an interesting research topic.

7.4. CONCLUSION

Big data is generated daily from various sources and devices. To utilize big data effectively, two key actions are essential: storage and analysis. Over the past decades, cloud services have gained increasing popularity by enabling users to outsource these complex tasks, allowing them to focus solely on leveraging the data. However, this relies on trust in the cloud server. One way to securely use a cloud server without relying on trust in the cloud is through encryption.

In this thesis, we explored cloud-related cryptographic schemes and addressed several open problems concerning security, efficiency, and functionality. Our work covers three research topics: updatable encryption, fully homomorphic encryption, and searchable encryption. First, we analyzed the existing security notions for ciphertext-dependent UE schemes and provided a clear relationship among all of these notions. Second, we overcame the limitations of prior work on ciphertext-dependent UE schemes by addressing adaptive security, CCA-1 security, and post-quantum security. We also introduced a packing technique that allows for the simultaneous encryption and update of multiple messages, alleviating the downloading cost in the token generation process.

In order to improve the efficiency and applicability of FHE, we then explored how to propose a batch bootstrapping technique that combines the advantages of different generations of FHE in terms of noise growth and computation cost. We first proposed a basic batch programmable bootstrapping technique, which allows the evaluation of an arbitrary univariate function over multiple ciphertexts simultaneously while refreshing the noise. We further optimized the basic batch programmable bootstrapping by introducing two homomorphic decomposition algorithms to support larger precision.

Finally, we investigated both passive and active attacks on searchable encryption schemes. We first proposed an efficient passive attack that recovers around 98% of the leaked documents and over 90% of queries with very low leakage. Since the proposed attack utilizes both the document size and the response per query, it requires stronger countermeasures than existing attacks. Additionally, we presented a file-injection attack based on the binomial search structure, making a significant advancement over existing active attack methods. This approach maximizes the storage of keywords within a limited number of injected files by using a new notion in finite set theory to identify keywords.

Our work focuses on the secure use of cloud services, but it can be applied to related cryptographic primitives and protocols, such as identity-based encryption, attribute-based encryption, proxy re-encryption, privacy-preserving machine learning, and multi-party computation.

REFERENCES

[1] A. Lehmann and B. Tackmann. “Updatable encryption with post-compromise security”. In: *EUROCRYPT 2018, Part III*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10822. LNCS. Springer. Heidelberg, 2018, pp. 685–716. DOI: 10.1007/978-3-319-78372-7_22.

[2] Y. Jiang. “The direction of updatable encryption does not matter much”. In: *ASIACRYPT 2020, Part III*. Ed. by S. Moriai and H. Wang. Vol. 12493. LNCS. Springer. Heidelberg, 2020, pp. 529–558. DOI: 10.1007/978-3-030-64840-4_18.

[3] D. Slamanig and C. Striecks. *Puncture 'Em All: Updatable Encryption with No-Directional Key Updates and Expiring Ciphertexts*. Cryptology ePrint Archive, Paper 2021/268. https://eprint.iacr.org/2021/268. 2021.

[4] R. Nishimaki. “The Direction of Updatable Encryption Does Matter”. In: *PKC 2022*. Ed. by G. Hanaoka, J. Shikata, and Y. Watanabe. Vol. 13178. LNCS. Cham: Springer, 2022, pp. 194–224. ISBN: 978-3-030-97131-1. DOI: 10.1007/978-3-030-97131-1_7.

[5] D. Micciancio and C. Peikert. “Trapdoors for lattices: Simpler, tighter, faster, smaller”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.

[6] M. Naor and M. Yung. “Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks”. In: *STOC 1990*. Ed. by H. Ortiz. Baltimore, Maryland, USA: ACM, 1990, pp. 427–437. DOI: 10.1145/100216.100273.

[7] E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: *J. Cryptol.* 26.1 (2013), pp. 80–101. DOI: 10.1007/S00145-011-9114-1. URL: https://doi.org/10.1007/s00145-011-9114-1.

[8] NIST. *NIST Releases First 3 Finalized Post-Quantum Encryption Standards*. Accessed: 2024-11-30. 2024. URL: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards.

[9] C. Gentry. “Fully homomorphic encryption using ideal lattices”. In: *ACM STOC 2009*. Ed. by M. Mitzenmacher. ACM, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[10] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. “Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE”. In: *ASIACRYPT 2017, Part I*. Ed. by T. Takagi and T. Peyrin. Vol. 10624. LNCS. Springer, 2017, pp. 377–408. DOI: 10.1007/978-3-319-70694-8_14.

[11] L. Ducas and D. Micciancio. “FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second”. In: *EUROCRYPT 2015, Part I*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 617–640. DOI: 10.1007/978-3-662-46800-5_24.

[12] A. Viand, C. Knabenhans, and A. Hithnawi. “Verifiable Fully Homomorphic Encryption”. In: *arXiv preprint arXiv:2301.07041* (2023). URL: https://arxiv.org/abs/2301.07041.

[13] H. Chen, I. Chillotti, and Y. Song. “Multi-Key Homomorphic Encryption from TFHE”. In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*. Ed. by S. D. Galbraith and S. Moriai. Vol. 11922. LNCS. Springer, 2019, pp. 446–472. DOI: 10.1007/978-3-030-34621-8_16.

[14] Y. Akin, J. Klemsa, and M. Önen. “A Practical TFHE-Based Multi-Key Homomorphic Encryption with Linear Complexity and Low Noise Growth”. In: *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*. Ed. by G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis. Vol. 14344. LNCS. Springer, 2023, pp. 3–23. DOI: 10.1007/978-3-031-50594-2_1.

[15] H. Chen, W. Dai, M. Kim, and Y. Song. “Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM, 2019, pp. 395–412. DOI: 10.1145/3319535.3363207.

[16] B. Xiang, J. Zhang, K. Wang, Y. Deng, and D. Feng. *NTRU-based Bootstrapping for MK-FHEs without using Overstretched Parameters*. Cryptology ePrint Archive, Paper 2024/1898. 2024. URL: https://eprint.iacr.org/2024/1898.

[17] J. Ning, X. Huang, G. S. Poh, J. Yuan, Y. Li, J. Weng, and R. H. Deng. “LEAP: Leakage-Abuse Attack on Efficiently Deployable, Efficiently Searchable Encryption with Partially Known Dataset”. In: *ACM CCS 2021*. Ed. by Y. Kim, J. Kim, G. Vigna, and E. Shi. ACM, 2021, pp. 2307–2320. DOI: 10.1145/3460120.3484540.

[18] G. Wang, Z. Cao, and X. Dong. “Improved File-injection Attacks on Searchable Encryption Using Finite Set Theory”. In: *Comput. J.* 64.8 (2021), pp. 1264–1276. DOI: 10.1093/COMJNL/BXAA161.

ACKNOWLEDGEMENTS

Throughout my Ph.D. journey, I have been fortunate to receive tremendous support, encouragement, and understanding from colleagues, friends, and family. You are the ones who made this journey enjoyable and helped me persevere through the difficult times. I sincerely thank all of you for being by my side every step of the way.

Foremost, my deepest gratitude goes to my daily supervisor, Dr. Kaitai Liang, and my promoter, Prof. Inald Lagendijk. Kaitai, I'm truly thankful for your constant support, patience, and encouragement throughout this journey. Our weekly meetings significantly influenced my research mindset—you've shown me how to approach problems independently and critically, which I hold in high regard. I also deeply value the professional growth I've gained through your guidance—whether by introducing me to others at conferences or encouraging me to pursue topics that spark my curiosity. To my promoter, Prof. Inald Lagendijk, I find great inspiration in your professionalism and greatly appreciate the valuable suggestions, ideas, and thoughtful discussions you've shared with me during this time. I'll carry your advice forward as I work toward my long-term goals—and I promise to let you know when they come true.

Furthermore, I sincerely thank the committee members—Prof. Liqun Chen, Prof. Colin Boyd, Prof. Fernando Kuipers, Dr. Jos H. Weber, and Prof. George Smaragdakis—for taking the time to review my thesis, providing insightful and constructive comments, and participating in my defense ceremony.

Next, I would like to thank all the wonderful colleagues I had the pleasure of working with during my Ph.D. Dear George, I'm grateful for the thoughtful suggestions and support you provided as I developed my academic career. I will never forget the light from your office shining in the dark. I sincerely hope that one day, I can become a researcher like you. Dear Mauro, thank you for supporting my academic visit to Padova and giving me the valuable opportunity to present my work among master's students. I was always touched to receive your emails, even late at night.

Dear Jelle, it has been a privilege to meet you and to share so many research interests. I'm especially thankful for your kindness in introducing your friends in our field to me during RWC in Sofia. I would also like to express my sincere thanks to Lilika. I learned a great deal from you while co-supervising master's students—it was a truly valuable experience. Dear Alexios, I appreciate your kindness in sharing your experience with paper submissions—it encouraged me greatly. Dear Stjepan, thank you for your kind support during the summer school. Dear Sicco, thanks for welcoming me so warmly to Thursday drink activities. Dear Sandra, I truly appreciate the way you always provide clear and detailed information to help me solve problems. Dear Zeki, thanks for the opportunity to be the teaching assistant for your course—it was a rewarding experience. Dear Jorrit, thank you for thoughtfully organizing group activities—they helped strengthen our Ph.D. group's connection. Your smile always brings light and warmth to our days. Dear

Stefanos, thanks for making our office a more supportive and pleasant place to work. I'm especially thankful to all the other members of the CYS family—Harm, Daniël, Clinton, Florine, Marina, Azade, Marwan, Yuqian, Maarten, Tjitske, Murtuza, Jesús and Misha. Our exchanges of ideas have been remarkably thought-provoking—your perspectives expanded my thinking and enriched my academic development. Thanks so much for your support.

This work would not have been possible without the collaborations with several esteemed researchers and outstanding master's students, from whom I have learned immensely. Dear Yao Jiang Galteland, I still vividly remember the feedback you gave me on my first article. You generously shared your research skills, writing advice, and more—your support left a lasting impact on me as a beginner and continues to guide me today. After our joint work, you also shared your job-hunting experience and offered practical suggestions, which I am sincerely grateful for. I also thank the excellent master's students I worked with: Steven, Björn, Hakan, Jeroen, Tjard, Manning, Lesley, Nicolas, Ken, Richard, Viraj and Peijie. Your curiosity, dedication, and fresh perspectives not only made our collaborations productive and enjoyable but also helped me grow as a mentor and researcher. I'm grateful to Dr. Chaoyun Li and Chunlei Li, whose mentorship and guidance helped me avoid many detours and saved me from unnecessary struggles.

Now, I also would like to express my gratitude to my dear friends. Dear Rui, from the moment I arrived in Delft until now, through daily life's ups and downs, I've always felt your support. I truly appreciate it from the bottom of my heart. Dear Tianyu, thank you for organizing so many wonderful gatherings for our friends, for generously sharing your room with us, and for always taking care of things—like cleaning the dishes after we left. Dear Huimin, it is my greatest honor to be your paranymph. I am deeply grateful for the help you've given to my family, and it means a great deal to me during those difficult times. I wish you the best in Darmstadt. Dear Dazhuang, thank you for your excellent ideas and insightful suggestions that helped me tackle complex issues over the past two years. Dear Yanqi, thank you for the unforgettable moments we shared while preparing for TMB. Dear Jing and Jinke, thank you for all the joy we shared during our outings, gatherings, and gaming sessions. Those moments brought so much happiness and are memories I'll always treasure. Dear Zeshun, thank you for the fantastic dinner event and for giving me a ride while we explored places together in the Netherlands and Germany—it was full of fun and great memories. Dear Shihui and Hao, I am truly grateful to both of you for introducing me to such enriching concepts in cryptography. Your guidance has greatly deepened my interest in the subject.

Dear Qian, Anton and Lelie, your presence made this unfamiliar land feel like home. Crossing paths with you has been one of the gentlest gifts this journey has given me. I wish you all the best in everything ahead. Dear Jiang Bei, Qin Qin and Andy, I will never forget what it means to have someone I can always rely on. Your support has been a great source of comfort and strength throughout my journey. Dear Dingyang, Sijia and Jiahe, thank you for the lovely time we shared with the little ones. I wish them all the best as they grow up strong and happy. To all my basketball friends, Liu Zhengxuan, Li Bo, Huang Ruopeng, Shan Xu, and Xu Zhiyuan, it is my great pleasure to become friends with you. In the end, I'm grateful for meeting you all at Eurocrypt 2025: Liu Xiang, Zhang Zhou, Wang Yunhao, Feng Yansong, and Ding Xiaohui and for the joyful time we spent

in Madrid. May your talents continue to flourish in all that you pursue.

Lastly, I wish to convey my deepest appreciation to my wife, Xiaodan. Without you, I could not have completed this dissertation. Thank you for all the contributions you made to our family, especially during the times when I couldn't be there. Your steady encouragement and heartfelt devotion have given me the strength to pursue my dreams—and the wings to fly even higher. This dissertation is dedicated to you. To Yi, I will do my best to be a good role model for you. I remain forever grateful to my loving family and my parents for their constant, unwavering support.

To the maple tree by the library—thank you for witnessing my seasons, and for listening in stillness when I needed it most.

Thank you all for walking this journey with me. In the words of Queen Elizabeth II: *We are just passing through. Our purpose here is to observe, to learn, to grow, to love... and then we return home.*

Curriculum Vitæ

Huanhuan Chen was born in Nanyang, Henan Province, China. He obtained his bachelor's degree in Mathematics and Applied Mathematics from Heilongjiang University, Harbin, China, in 2015. He then earned his master's degree in Pure Mathematics from Nankai University, China, in 2018.

Since June 2021, he has been pursuing a Ph.D. in the Cyber Security research group at the Faculty of Electrical Engineering, Mathematics and Computer Science at Delft University of Technology, the Netherlands. His Ph.D. research is supervised by Dr. Kaitai Liang and Prof. dr.ir. R.L. Lagendijk. His research interests include post-quantum (lattice-based) cryptography, applied cryptography, and fully homomorphic encryption.

List of Publications

Conference and Workshop

8. Wang, R., Wang, X., **Chen, H.**, Decouchant, J., Picek, S., Laoutaris, N. and Liang, K. MUD-GUARD: Taming Malicious Majorities in Federated Learning using Privacy-Preserving Byzantine-Robust Clustering. In *ACM SIGMETRICS* (2025), ACM, pp. 1–41.

7. Zhang, M., Shi, Z., **Chen, H.** and Liang, K. Inject Less, Recover More: Unlocking the Potential of Document Recovery in Injection Attacks Against SSE. In *CSF* (2024), IEEE, pp. 311–323.

6. Langhout, T.J., **Chen, H.** and Liang, K. File-Injection Attacks on Searchable Encryption, Based on Binomial Structures. In *ESORICS* (2024), vol. 14984 of *Lecture Notes in Computer Science*, Springer, pp. 424–443.

5. Ho, B., **Chen, H.**, Shi, Z. and Liang, K. Similar Data is Powerful: Enhancing Inference Attacks on SSE with Volume Leakages. In *ESORICS* (2024), vol. 14985 of *Lecture Notes in Computer Science*, Springer, pp. 105–126.

4. **Chen, H.**, Galteland, Y.J., and Liang, K. CCA-1 Secure Updatable Encryption with Adaptive Security. In *ASIACRYPT* (2023), vol. 14442 of *Lecture Notes in Computer Science*, Springer, pp. 374–406

3. Lambregts, S., **Chen, H.**, Ning, J. and Liang, K. VAL: Volume and Access Pattern Leakage-Abuse Attack with Leaked Documents. In *ESORICS* (2022), vol. 13554 of *Lecture Notes in Computer Science*, Springer, pp. 653–676.

2. **Chen, H.**, Fu, S., and Liang, K. No-Directional and Backward-Leak Uni-Directional Updatable Encryption Are Equivalent. In *ESORICS* (2022), vol. 13554 of *Lecture Notes in Computer Science*, Springer, pp. 387–407.

1. Tjiam, K., Wang, R., **Chen, H.** and Liang, K. Your smart contracts are not secure: investigating arbitrageurs and oracle manipulators in Ethereum. In *Proceedings of the 3rd Workshop on Cyber-Security Arms Race* (2021), ACM, pp. 25–35.

Preprint

1. **Chen, H.**, CONTI, M., Giannetsos, T. and Liang, K. Batch Programmable Bootstrapping, within A Polynomial Modulus. In submission.


TUDelft
Delft
University of
Technology