Peekaboo, I See Your Queries
Passive Attacks Against DSSE Via Intermittent Observations

Nie, Hao; Wang, Wei; Xu, Peng; Chen, Wei; Yang, Laurence T.; Conti, Mauro; Liang, Kaitai

DOI
10.1145/3719027.3765075

Publication date
2025

Document Version
Final published version

Published in
CCS 2025 - Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security

Citation (APA)
Nie, H., Wang, W., Xu, P., Chen, W., Yang, L. T., Conti, M., & Liang, K. (2025). Peekaboo, I See Your Queries: Passive Attacks Against DSSE Via Intermittent Observations. In *CCS 2025 - Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2429-2443). ACM. https://doi.org/10.1145/3719027.3765075

RESEARCH-ARTICLE

Peekaboo, I See Your Queries: Passive Attacks Against DSSE Via Intermittent Observations

HAO NIE, Huazhong University of Science and Technology, Wuhan, Hubei, China

WEI WANG, Huazhong University of Science and Technology, Wuhan, Hubei, China

PENG XU, Huazhong University of Science and Technology, Wuhan, Hubei, China

WEI CHEN, Huazhong University of Science and Technology, Wuhan, Hubei, China

LAURENCE T YANG, St. Francis Xavier University, Antigonish, NS, Canada

MAURO CONTI, Örebro University, Orebro, Orebro, Sweden

View all

Open Access Support provided by:

Huazhong University of Science and Technology

University of Turku

Örebro University

St. Francis Xavier University

Published: 19 November 2025

Citation in BibTeX format

CCS '25: ACM SIGSAC Conference on Computer and Communications Security
October 13 - 17, 2025
Taipei, Taiwan

Conference Sponsors:
SIGSAC

CCS '25: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (November 2025)
https://doi.org/10.1145/3719027.3765075
ISBN: 9798400715259

Peekaboo, I See Your Queries: Passive Attacks Against DSSE Via Intermittent Observations

Hao Nie
Huazhong University of Science and Technology
Wuhan, Hubei, China
nie@hust.edu.cn

Wei Wang*
Huazhong University of Science and Technology
Wuhan, Hubei, China
viviawangwei@hust.edu.cn

Peng Xu
Huazhong University of Science and Technology
Wuhan, Hubei, China
xupeng@hust.edu.cn

Wei Chen
Huazhong University of Science and Technology
Wuhan, Hubei, China
hust_cw@hust.edu.cn

Laurence T. Yang
St. Francis Xavier University
Antigonish, Nova Scotia, Canada
ltyang@gmail.com

Mauro Conti
University of Padua
Padua, Veneto, Italy
Örebro University
Örebro, Örebro County, Sweden
mauro.conti@unipd.it

Kaitai Liang
TU Delft
Delft, South Holland, Netherlands
University of Turku
Turku, Southwest Finland, Finland
kaitai.liang@tudelft.nl

Abstract

Dynamic Searchable Symmetric Encryption (DSSE) allows secure searches over a dynamic encrypted database but suffers from inherent information leakage. Existing passive attacks against DSSE rely on persistent leakage monitoring to infer leakage patterns, whereas this work targets intermittent observation - a more practical threat model. We propose Peekaboo - a new universal attack framework - and the core design relies on inferring the search pattern and further combining it with auxiliary knowledge and other leakage. We instantiate Peekaboo over the SOTA attacks, Sap (USENIX' 21) and Jigsaw (USENIX' 24), to derive their "+" variants (Sap+ and Jigsaw+). Extensive experiments demonstrate that our design achieves >0.9 adjusted rand index for search pattern recovery and ∽90% query accuracy vs. FMA's ∽30% (CCS' 23). Peekaboo's accuracy scales with observation rounds and the number of observed queries but also it resists SOTA countermeasures, with >40% accuracy against file size padding and >80% against obfuscation.

CCS Concepts

• **Security and privacy → Management and querying of encrypted data**; **Cryptanalysis and other attacks**.

Keywords

Dynamic Searchable Symmetric Encryption, Leakage Abuse Attacks, Intermittent-observation Attacker

ACM Reference Format:
Hao Nie, Wei Wang, Peng Xu, Wei Chen, Laurence T. Yang, Mauro Conti, and Kaitai Liang. 2025. Peekaboo, I See Your Queries: Passive Attacks Against DSSE Via Intermittent Observations. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3765075

1 Introduction

Searchable Symmetric Encryption (SSE) [6, 7, 10, 11, 13, 17, 23, 27, 39–41, 46] allows a client to search an encrypted database on a server without disclosing searched keywords and the files, while its variant, dynamic SSE (DSSE) [6, 7, 10, 13, 17, 23, 27, 40, 41, 46], additionally enables secure updates to the database. Existing DSSE leaks certain information, even with forward and backward privacy (FP/BP) [6, 7, 10, 13, 17]. Passive attackers[1] [5, 8, 12, 19, 21, 25, 29–32, 35, 45] (Table 1) exploit this leakage by observations and prior knowledge (known/similar data) to recover search queries.

These attacks [19, 37, 44] against DSSE typically rely on a "strong" assumption that the attacker can perform continuous observations as a so-called *persistent attacker*. Such attackers monitor leakage over time and track changes in search queries, enabling them to reconstruct a comprehensive leakage profile across the entire query history. For example, the leakage from any two consecutive queries for the same keyword is consistent, as these queries are issued

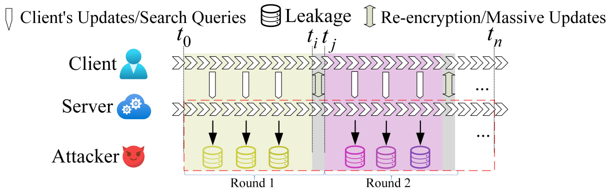
---

[*] Corresponding author.

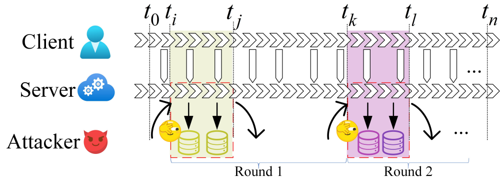
[1] Passive attackers differ from active attackers [5, 34, 47, 48], who intentionally inject dummy files into the database, which is orthogonal to this work.

CCS '25, Taipei

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3765075



(a) Intermittent Observation, Case I



(b) Intermittent Observation, Case II

Figure 1: Intermittent observations: Case I, the attacker cannot match the queries (and their leakage) between rounds due to database re-encryption or massive updates on most of database files; Case II, the attacker only observes multiple short-term periods and skips the leakage between t_j and t_k.

and observed in close succession, and the underlying database is unlikely to update significantly within such a short interval.

In real-world scenarios, the assumption of persistent observation may not hold, as attackers are often limited to *intermittent observations* due to practical constraints:

- Case I: Database re-encryption following file updates disrupts the attacker's ability to maintain continuous observations [2].
- Case II: The attacker's observation must be intermittent, as sustained monitoring increases the risk of detection and potential interruption. Modern attackers, such as Trickbot [1] and certain APT attackers [3], eavesdropping on the server or the communication channel, often operate in sleep mode to evade detection, thereby acquiring only intermittent observations.

Note that we provide concrete examples in the full version[28].

To adapt to the landscape, we consider more practical attackers, namely *intermittent-observation attackers (IOAs)*. Like [14, 33, 42], the IOAs, repeating multiple rounds of observing and then staying offline, can only capture the leakage in multiple short-term periods. The attackers miss certain leakage of search queries and all related updates during the offline periods. As in Figure 1a, an IOA observes until re-encryption or massive updates happen at time $[t_i, t_j]$ changing many files, such that the leakage observed by the attacker before and after can be entirely different and mismatched. In Figure 1b, an IOA observes in periods $[t_i, t_j]$ and $[t_k, t_l]$. Due to missing leakage between t_j and t_k, it cannot match the encrypted files from the former period to those in the latter. Please refer to detailed examples in the full version[28].

Re-encryption/massive updates in Figure 1a and updates of files during the offline period in Figure 1b can significantly impact the leakage patterns of the queries. The volume pattern (VP, indicating the number of files in response) and the access pattern (AP, indicating the identities of the encrypted files in the response) of search queries can vary entirely between two different rounds[3]. When the attacker re-observes the query for the same keyword in the later period, it loses the search pattern (SP, indicating whether two search queries are for the same keyword) between the two queries due to the re-encryptions. As a result, the observed leakage is "isolated" from each other, and further observing for more periods cannot provide extra advantages for the attacker. Compared to a persistent attacker, we argue that an IOA is a "weaker-in-leakage-but-stronger-in-attack" variant — one that operates with less leakage, without access to "a full picture" of leakage profile, yet achieves comparable attack performance. We note that some works [4, 5, 22] also introduce similar "intermittent" concepts, but they obtain the copy of encrypted database instead of query leakage; thus, they cannot recover queries [5, 22].

Challenges: intermittent observations against DSSE. In most passive attacks [8, 12, 21, 25, 29–32, 35], the SP is essential. The attacker must first categorize all search queries by their underlying keywords to identify unique entries. Some attacks [25, 29, 31, 32] also require the search frequency to match the queries with the keywords, while the SP is crucial to deduce the search frequency. In the context of intermittent observations, SP is not available to the attacker, and deducing it from other leakage is challenging since the observation is intermittent. To obtain the "full picture" of SP, the attacker must acquire the SP within a single observation period

Table 1: Comparison: passive attacks against (D)SSE[1]

Target	Type	Attack	Leakage	Doc	Freq	P or I[2]
SSE	Known-Data	Subgraph^{ID} [5]	AP	-	-	-
		Count [8]	AP,VP	-	-	-
		LEAP [30]	AP	-	-	-
	Similar-Data	GraphM [35]	AP	●	○	-
		SAP [31]	SP,VP	●	●	-
		RSA [12]	AP	●	○	-
		IHOP [32]	SP,AP	●	●	-
		Jigsaw [29]	VP,SP,AP	●	◐	-
DSSE	Similar-Data	FMA [44]	FVP/SP	○	●	P
		Sap+	FVP/AP	●	●	I
		Jigsaw+	FVP/AP	●	◐	I

[1] "AP" is the access pattern, "VP" is the volume pattern, "FVP" is the file volume pattern, and "SP" is the search pattern. The attacks targeting SSE all require the "SP" to get unique queries. We omit the "SP" for these attacks and only tag them with "SP" when they use the "SP" to obtain the frequency of search queries. "Doc" denotes whether the attacker needs similar documents, and "Freq" for similar query frequency. The known-data attacks require a part of the plaintexts of the database rather than similar prior knowledge. The "●" indicates heavy dependency on specific prior knowledge, the "◐" suggests moderate dependency, and the "○" denotes no prior knowledge is required.

[2] "P" and "I" are the persistent and intermittent-observation attackers, respectively. We note that the encrypted database remains "unchanged" during observation in the context of SSE (instead of DSSE), so the concepts of persistent and intermittent-observation attackers do not apply.

---

[2] It is possible that the observed leakage is completely shuffled if a DSSE imports a key-update technology [10] to re-encrypt the database.

[3] The volumes of queries naturally change due to additions and deletions, while the AP of the queries for the same keyword is *refreshed*, as required by the forward and backward security [44].

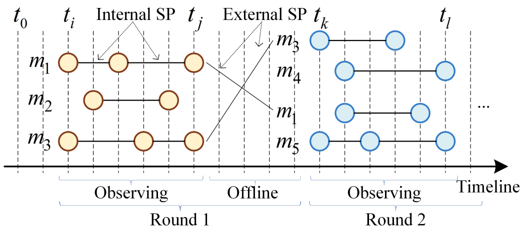


Figure 2: Internal and external SP of search queries. The m_i is the i-th keyword. The circles are the search queries observed by the attacker. In each row of an observation round, the queries are for the same keyword m_i in the front.

(*internal SP*) and between different observation periods (*external SP*), see Figure 2.

Previous attacks [32, 37, 38, 44] proposed to infer the SP could recover the internal SP. In [37], Salmani et al. develop an attack that utilizes the AP to infer the SP. The attacker must match the AP precisely between queries, and further, any updates on keywords (e.g., adding or deleting) from files can disable such a matching. [32] and [38] propose a method that clusters queries with obfuscated AP using k-means, which could also be used to recover the internal SP. Xu et al. [44] introduce the FMA attack, where the attacker takes the file volume pattern (FVP, indicating the size of each file in the response) to compute the similarity of queries so as to group similar queries to obtain the SP. Inferring the external SP is more difficult, as the leakage is periodically hidden from the attacker. In this sense, the attacker cannot distinguish whether two encrypted files from different rounds are for the same file (as the inferred identities and the size of files change due to the updates). This easily disables Salmani et al.'s attack [37] and the k-means based clustering [32, 38]. Furthermore, if multiple updates occur between any two rounds and these updates result in changes to the file size, the similarity of queries calculated in the FVP is significantly distorted, thereby impacting the FMA's performance. These attacks cannot perform effectively with intermittent observation.

Besides the SP, current attacks [5, 8, 12, 18, 21, 29, 32, 35] use other leakages. The attacker cannot leverage them from multiple observation rounds. From the AP, the inferred identities of the queries for the keyword differ across rounds, meaning the information is "terminated" between rounds. A similar issue occurs in the VP, as the volume of a keyword can vary significantly in different rounds.

Inspirations. Though leakage and updates are not accessible to the IOA during the offline period, the semantic relationships among keywords remain consistent within the database. For example, the word "searchable" is likely to appear frequently in a file containing the term "encryption", both before and after database updates. Based on this, we can construct a co-occurrence graph for the database's keywords, incorporating the associated search queries. In each round, we group the search queries for the same keyword based on their similarity in leakage patterns, thereby inferring the internal SP. Next, we construct the co-occurrence graph for these query groups. Finally, we match these graphs and merge corresponding groups from different rounds, inferring the external SP.

Contributions. We investigate the IOA targeting DSSE, marking the first type of attacks with intermittent observation. To capture this new attack, we propose Peekaboo, with our core designs focusing on search pattern inference (P1) and query recovery (P2). P1 infers both internal and external SP by leveraging either AP or FVP leakage; while P2 takes the output of P1, along with either AP or FVP, and the auxiliary similar knowledge to recover all queries.

1) We define the IOA, which has not been investigated in prior work, and formalize the leakage under the context. The proposed attacker is a practical variant compared to the persistent attacker.

2) We propose Peekaboo, which first infers the internal SP by grouping search queries. In each round, when a new search query comes in, Peekaboo calculates the similarity between the unknown query and the last query in each group using either AP or FVP. The query then joins the group with the largest similarity that exceeds a defined threshold. Next, Peekaboo uses the internal SP to calculate the co-occurrence matrix of the groups in each round. Between any two rounds, we match the groups from one round to another by solving a quadratic assignment problem based on the co-occurrence matrix. We iterate the process over rounds to infer all of the external SP. We then process the leakage and the auxiliary knowledge based on the recovered SP and provide an interface for query recovery. Existing similar-data attacks (with minor adaptations) can use this interface to recover queries through intermittent observations. We use Sap [31] and Jigsaw [29] to instantiate Sap+ and Jigsaw+.

3) We conduct extensive experiments to demonstrate the attack performance of Peekaboo. P1 demonstrates high accuracy in inferring SP, providing an adjusted rand index above 0.9 in most cases; and P2 achieves strong performance in query recovery. For instance, Peekaboo with Jigsaw+ achieves approximately 90% and 50% accuracy with AP and FVP, respectively. In comparison, the accuracy of FMA [44] is about 30% for both AP and FVP. Furthermore, Peekaboo with Jigsaw+ maintains $>$ 40% accuracy under file size padding and $>$ 80% under obfuscation in most cases.

2 The Framework of Peekaboo

We use upper-case boldface to represent matrices, lower-case boldface for vectors, upper-case italics for collections, and lower-case italics for individual values. We use superscripts to denote the identity of the round or time slot. For example, Q^x represents the queries in the x-th round, and $\mathbf{f}^i$ denotes the frequency in the i-th time slot. We use $\{\cdot\}$ to denote a set or a list of elements and $[\cdot]$ for a vector of numbers. $[a]$ is used as a shorthand for $[1, 2, ..., a]$. $|\cdot|$ represents the size of a set or a list and $\#d$ represents the size of a file d. $\mathbf{v}[i]$ refers to the i-th value of a vector and $\mathbf{M}[i][j]$ is the value in the i-th row and j-th column of $\mathbf{M}$. We also use $L[i]$ to refer to the i-th element of a list, with $L[-1]$ specifically denoting the last element. $|a|$ is for the absolute value of a and $||\mathbf{v}||$ represents the Euclidean norm of a vector $\mathbf{v}$. Table 2 summarizes the frequently used notations.

2.1 Scenarios

A client maintains a database and uses a DSSE scheme to outsource it to a server. The DSSE generally includes three protocols: *Setup*, *Update*, and *Query*. Though the plaintexts of the database and search queries are not directly leaked, the attacker can observe patterns

Table 2: Summary of notations.

Notation	Description
η	Total number of rounds.
$\boldsymbol{\sigma}$	Number of online time slots in each round.
$\boldsymbol{\varsigma}$	Number of offline time slots in each round.
τ	$\tau = \sum_{i=1}^{\eta} \boldsymbol{\sigma}[i]$ is the total number of time slots of observation during the η rounds.
Q^x	$Q^x = \{q_1^x, ..., q_{l_x}^x\}$ is the observed query sequence of the x-th round.
AP^x	$AP^x = \{DB^x(q_1^x), ...\}$ is the access pattern leakage of the x-th round.
FVP^x	$FVP^x = \{\{\#d_i \mid i \in DB^x(q_1^x)\}, ...\}$ is the file volume pattern leakage of the x-th round.
F^x	The collection of all observed distinctive files (or distinctive file sizes with the FVP leakage) of the x-th round.
Gs^x	$Gs^x = \{G_1, ...\}$ is a partition of the Q^x of the x-th round.
$\mathbf{ID}^x$	The index matrix of all the queries of groups of the x-th round (size $\lvert Gs^x \rvert \times \lvert F \rvert$).
$\mathbf{IDH}^x$	The index matrix of the first queries of groups of the x-th round (size $\lvert Gs^x \rvert \times \lvert F \rvert$).
$\mathbf{IDT}^x$	The index matrix of the last queries of groups of the x-th round (size $\lvert Gs^x \rvert \times \lvert F \rvert$).
$\mathbf{F}_r$	The matrix of the client's search frequency of τ time slots.
$\mathbf{F}_s$	The matrix of the search frequency of the keywords known by the attacker.
$\mathbf{V}_r$	The matrix of the max volume of the queries in each group of τ time slots.
$\mathbf{V}_s$	The matrix of the volume of keywords of τ time slots known by the attacker.
$maxlevel$	The match of the InferESP between two rounds i and j has an interval less than $maxlevel$, i.e. $\lvert i - j \rvert \le maxlevel$.
p_g	The ratio of the removed matches in Match.

and recover search queries. Existing DSSE schemes provide forward/backward privacy (FP/BP) [6, 7] and the BP includes Type I, II, and III, each offering progressively weaker security [7]. We provide details of DSSE and related attacks in the full version[28].

We consider IOAs against DSSE with only passive and intermittent observation of leakage. They can observe the leakage from queries over multiple periods while being offline between observations, either intentionally or due to constraints, as illustrated in Figure 1. We refer to each online-then-offline cycle as a round and employ multiple rounds to complete an attack.

An IOA only needs to 1) eavesdrop on the server, or 2) eavesdrop on the communication channel, with the same purpose of query recovery. This assumption is weaker than previous attacks assuming the server is the attacker and can observe continuously.

The IOA can gain VP and FVP (without AP and SP) against DSSE with Type I BP-security with ORAM-like techniques [44]; for other types of DSSE, the attacker can additionally obtain AP and internal SP[4]. If the file is re-encrypted before transmission, the IOA eavesdropping on channel cannot acquire AP and SP. We assume the external SP is always concealed as re-encryptions happen during offline periods. We provide a description of these leakages in Section 2.2. The server itself may function as a persistent attacker, observing leakage continuously, which differentiates it from an IOA (see the full version[28]).

2.2 Peekaboo Attacker

Definition 1 (($\eta, \boldsymbol{\sigma}, \boldsymbol{\varsigma}$) intermittent observation). *A $(\eta, \boldsymbol{\sigma}, \boldsymbol{\varsigma})$ intermittent observation includes η rounds of observation. The i-th round begins with online observation, obtaining the leakage of search queries Q^i for $\boldsymbol{\sigma}[i]$ time slots. Each online period follows with an offline period of $\boldsymbol{\varsigma}[i]$ time slots without observation.*

The attacker with the above observation ability is denoted as an intermittent-observation attacker (IOA). The attacker can observe a collection of queries $Q = \{Q^1, ..., Q^\eta\}$, where $Q^x = \{q_1^x, ..., q_{l_x}^x\}$ is the search queries observed in the x-th round. For the search queries observed from the η rounds, we define the leakage patterns, AP, FVP, VP, and SP.

Definition 2 (Leakage patterns). *The leakage patterns, AP, FVP, VP, and SP, of $Q = \{Q^1, ..., Q^\eta\}$ are:*

- *The **AP** of Q is the family of functions $AP : EDB \times Q \longrightarrow AP$, where EDB is the encrypted database. The $AP = \{AP^1, ..., AP^\eta\}$, where $AP^x = \{DB^x(q_1^x), ..., DB^x(q_{l_x}^x)\}$ contains all the file identities of the response of each query in Q^x.*
- *The **FVP** of Q is the family of functions $FVP : EDB \times Q \longrightarrow FVP$. The $FVP = \{FVP^1, ..., FVP^\eta\}$, where $FVP^x = \{\{\#d_i | i \in DB^x(q_1^x)\}, ..., \{\#d_i | i \in DB^x(q_{l_x}^x)\}\}$ contains all the file sizes of the response of the queries in Q^x.*
- *The **VP** of Q is the family of functions $VP : EDB \times Q \longrightarrow VP$. The $VP = \{VP^1, ..., VP^\eta\}$, where $VP^x = \{|DB^x(q_1^x)|, ..., |DB^x(q_{l_x}^x)|\}$.*
- *The **SP** of Q is the family of functions $SP : EDB \times Q \longrightarrow SP$. The $SP = \{G_1, ..., G_n\}$ is a partition of the set $Q^1 \cup ... \cup Q^\eta$, where the search queries in the G_i are for the same keyword, and the search queries of G_i and G_j $(i \neq j)$ are for two different keywords.*

We separate the SP of Q into the internal SP (ISP) and the external SP (ESP). The ISP indicates whether two search queries from the same round are for the same keyword, while the ESP focuses on the different rounds, i.e., whether two queries from the different rounds are associated with the same keyword.

Definition 3 (Internal and external SP). *The SP of Q consists of two parts:*

- *The **Internal SP** of the i-th round is the family of functions $ISP : EDB \times Q^i \longrightarrow ISP^i$. The ISP^i is a partition $ISP^i = \{G_1^i, ..., G_n^i\}$ of Q^i, where each group G_j^i contains the queries for the same keyword, and different groups contain queries for different keywords.*
- *The **External SP** between ISP^i and ISP^j is the family of functions $ESP : EDB \times (ISP^i, ISP^j) \longrightarrow \mathbf{ESP}^{i,j}$. The $\mathbf{ESP}^{i,j}$ is a binary matrix with the size $|ISP^i| \times |ISP^j|$, where $\mathbf{ESP}^{i,j}[x][y] = 1$ indicates the search queries of the x-th group of ISP^i and the y-th group of ISP^j are for the same keyword, otherwise, $\mathbf{ESP}^{i,j}[x][y] = 0$.*

---

[4] For certain work [37], the attacker also cannot track the SP from the search tokens.

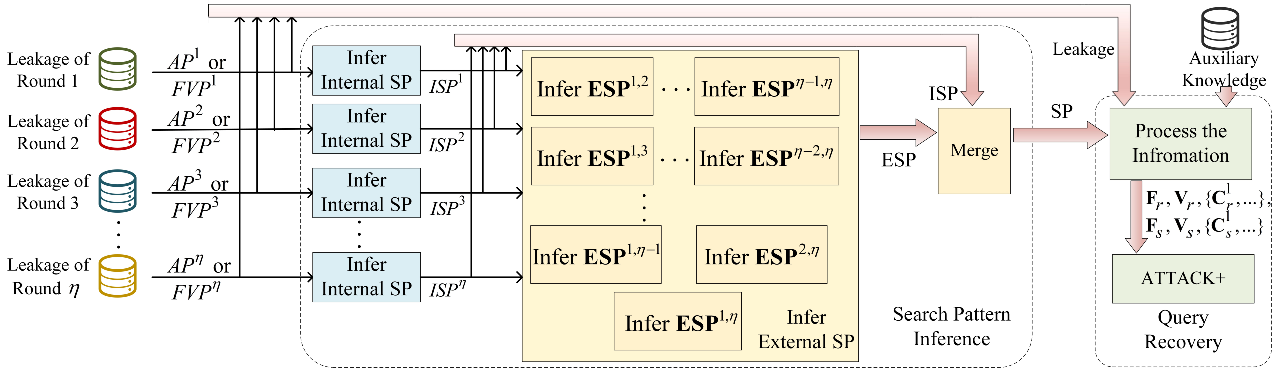


Figure 3: Overview: the Peekaboo attack.

The shuffling of database between two successive rounds. Between two successive rounds x and $x+1$, the database is shuffled and the files have also been updated during the offline interval between them. For the AP, suppose in the x-th round, the attacker observes that the i-th encrypted file is returned to a query; in round $x+1$, the client queries the same keyword again, and the same file is responded. But, due to updates, the attacker cannot recognize the file as the same one. As a result, it re-maps this file in $(x+1)$-th round as the j-th encrypted file. We here denote the AP leakage as $AP^x = \{h^x(DB(q_1^x)), ..., h^x(DB(q_{l_x}^x))\}$ and $AP^{x+1} = \{h^{x+1}(DB(q_1^{x+1})), ..., h^{x+1}(DB(q_{l_{x+1}}^{x+1}))\}$, where h^x and h^{x+1} are distinct mappings of file identifies. Similarly, with FVP, updates that alter file sizes prevent the attacker from matching files across rounds. Moreover, additions and deletions on files make the patterns from two rounds more distorted, making it more difficult to correlate patterns across different observation periods.

Differences between the persistent attacker and the IOA. We note that a persistent attacker can execute $(1, \boldsymbol{\sigma}, \varsigma)$ intermittent observation, which entails observing a single round for a long duration. Since the database undergoes continuous updates, $DB^x(q_i)$ differs from $DB^x(q_j)$ even if q_i and q_j target the same keyword k. Despite these variations, the persistent attacker can track the updates over time to identify consistent patterns. Successive queries for the same keyword typically produce similar leakage so that they are easy to match [44].

In contrast, when $\eta > 1$, an attacker with $(\eta, \boldsymbol{\sigma}, \varsigma)$ intermittent observation faces greater difficulty, due to fragmented and incomplete leakage. As previously mentioned, the response to the same keyword can vary significantly in two rounds, i.e., DB^x and DB^{x+1} for the same keyword k could be entirely different.

In summary, the persistent attacker requires a continuous and full observation for query recovery (i.e., a sufficiently large $\boldsymbol{\sigma}[1]$) but runs a high risk of being detected. On the other hand, the IOA, with intermittent observation, is more practical but gains limited access to leakage. Our ultimate goal is to find a way to effectively merge partial leakage from different rounds to enable practical and accurate query recovery under intermittent observation.

The Peekaboo attacker is an IOA. We assume the Peekaboo attacker knows either AP or FVP, as discussed in Section 2.1. The attacker can trivially infer the VP. We also say the attacker has no prior knowledge of SP. In existing similar-data attacks, the attacker requires auxiliary knowledge, such as the search frequency of keywords and a similar auxiliary dataset, to recover the search queries. We also allow such auxiliary knowledge for the Peekaboo attacker. Formally, we define the Peekaboo attacker as follows.

Definition 4 (Peekaboo Attacker). *The Peekaboo attacker is a $(\eta, \boldsymbol{\sigma}, \varsigma)$ intermittent observation attacker with the AP or FVP of $Q = \{Q^1, ..., Q^\eta\}$ as its observed leakage. The Peekaboo attacker also possesses a search frequency $\mathbf{F}_s$ of a keyword set W_s and an auxiliary dataset $DB_s = \{DB_s^1, ..., DB_s^\tau\}$, where $\tau = \sum_{i \in [\eta]} \boldsymbol{\sigma}[i]$, and DB_s^i is the auxiliary dataset at the i-th time slot.*

Attacker's Target. The attacker utilizes the observed leakage and attempts to infer an $\widetilde{SP}$ from the FVP or AP, making the $\widetilde{SP}$ as similar to the SP as possible. The attacker then utilizes the inferred $\widetilde{SP}$ along with other leakage and the auxiliary knowledge to recover the underlying keywords of search queries.

2.3 Overview of Peekaboo

Our proposed attack consists of search pattern inference (P1) and query recovery (P2). Note that we illustrate Peekaboo in Figure 3.

Search pattern inference. In P1, Peekaboo first infers the internal SP. In each round i, the attacker utilizes the AP^i or the FVP^i to group the search queries for the same keyword. For a new search query, it calculates the similarity of the new search query with the last search query in each group by the AP^i or the FVP^i. As we assume the observing time of a round lasts shortly, the changes of the database are limited. Thus, the similarity relying on the AP^i or FVP^i remains accurate. The search query then joins the group with the largest similarity that exceeds a threshold; otherwise, the search query forms a new group.

The attacker then merges the groups from each round to infer the external SP. Since the database may change significantly between two rounds, the similarity between queries from different rounds could be heavily distorted by noise. Thus, the method of inferring the internal SP is not applicable to the external SP. To infer the external SP, the attacker calculates the co-occurrence matrix of the groups in each round, relying on the internal SP. The problem of inferring the external SP is then converted to quadratic assignment problems to match the co-occurrence matrix of different rounds.

We iterate the matching between groups of all rounds to infer all of the external SP. If the co-occurrence relation remains across rounds, updates or even re-encryption of the database [10] cannot influence the results. Finally, the first part of Peekaboo merges the matched groups and produces the SP.

Query recovery. In P2, the attacker utilizes the merged groups of P1, the observed leakage, and the auxiliary knowledge to recover search queries. For each merged group outputted by P1, the attacker counts the queries in the group in each time slot to get the search frequency. It also records the maximum volume of the queries in each time slot in the same group. The attacker realigns the FVP or AP leakage of groups in every round according to the groups. Based on the FVP or AP leakage, the attacker calculates the co-occurrence matrix of the groups in every round.

Existing *similar-data* attacks can utilize the above information (with minor adaptations) to recover the keyword for each merged group. We show how to instantiate Peekaboo for two recent SOTA attacks, Sap [31] and Jigsaw [29], as Sap+ and Jigsaw+. Rather than assuming a static database, Sap+ and Jigsaw+ target the DSSE. We note that the Sap attack does not rely on co-occurrence information, and the Jigsaw attack has a higher accuracy (similarly in SAP+ and Jigsaw+). Therefore, one may implement either of them depending on the attack scenario. Other *similar-data* attacks can also be instantiated similarly. With modifications, it is also possible for one to implement the instantiations of Peekaboo on previous *known-data* attacks. We provide discussions in the full version[28].

3 Peekaboo: Search Pattern Inference

In most passive attacks [8, 12, 18, 21, 25, 29–32, 35], the attacker relies on the SP (indicating whether two search queries are for the same keyword) to recover search queries. As shown in Section 2.1, the SP remains concealed in DSSE with Type I BP-security and is not available to the IOAs in the communication channel. Under intermittent observation, the external SP always remains concealed from the attacker. We propose the first part (P1) of Peekaboo, inferring the SP from the FVP (indicating the size of each file in the response) or AP (indicating the identities of the encrypted files in the response), with intermittent observations. In Sections 3.1 and 3.2, we demonstrate how queries are matched within a single round to infer the internal SP and across multiple rounds to infer the external SP.

3.1 Inference of Internal SP

We here infer the internal SP by the FVP or AP leakage. To decide whether two queries, the i-th and j-th query in Q^x, are for the same keyword, Xu et al. [44] use the FVP. Specifically, they calculate the response similarity between two queries:

$$rsp(q_i^x, q_j^x) = |(FVP_i^x \Cap FVP_j^x)|/|(FVP_i^x \Cup FVP_j^x)|. \tag{1}$$

Given two collections $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_n\}$, they define the intersection $Z = X \Cap Y$ as the collection of elements, including duplication, appearing in both X and Y. For instance, given $X = \{1, 1, 2, 2\}$ and $Y = \{1, 2, 2\}$, then their intersection is $Z = X \Cap Y = \{1, 2, 2\}$. Likewise, their union is defined as $W = X \Cup Y = \{1, 1, 2, 2\}$. Then, they calculate qeq as follows, where $qeq(q_i^x, q_j^x) = 1$ represents queries Q_i^x and Q_j^x are for the same keyword and $qeq(q_i^x, q_j^x) = 0$ otherwise.

$$qeq(q_i^x, q_j^x) = \begin{cases} 1 & , rsp(q_i^x, q_j^x) \geq \delta \\ 0 & , rsp(q_i^x, q_j^x) < \delta \end{cases} \tag{2}$$

Algorithm 1 Inferring the internal SP.

```
1:  Procedure InferISP($Q^x$)
2:  $Gs^x \leftarrow \emptyset$;
3:  for all $q \in Q^x$ do
4:      $Cand \leftarrow \emptyset$;
5:      $Q_{end} \leftarrow \{G[-1] | G \in Gs^x\}$;
        ▷ $G[-1]$ is the last element of $G$;
6:      for all $q_{end} \in Q_{end}$ do
7:          Add $(rsp(q, q_{end}), q_{end})$ to $Cand$ if $qeq(q, q_{end})$ is 1;
8:      if $Cand$ is empty then
9:          Add $G = \{q\}$ to $Gs^x$;
10:     else
11:         Sort $Cand$ in descending order according to $Cand.rsp$;
12:         Extract $(rsp, q_{end}) \leftarrow Cand[1]$ and add $q$ to the end of $G$,
            where $q_{end} \in G$ and $G \in Gs^x$;
13: return $Gs^x$;
14: End Procedure
```

We use the same rsp and qeq in the FVP scenario. For the AP scenario, we define the rsp as:

$$rsp(q_i^x, q_j^x) = |(AP_i^x \cap AP_j^x)|/|(AP_i^x \cup AP_j^x)|. \tag{3}$$

Based on this, we describe the inference of internal SP in Algorithm 1. We group the queries for the same keywords, and Gs^x contains all the groups. When a new query q is observed, the attacker extracts the last query q_{end} in each group and checks if the two queries are under the same keyword (line 6-7). If the query q matches other queries, the attacker includes it to the end of the group where the last query q_{end} of this group has the largest $rsp(q, q_{end})$ (line 11-12). Otherwise, the attacker generates a new group containing only q and puts the new group to Gs^x (line 8-9).

We use the last query q_{end} in each group when calculating the $rsp(q, q_{end})$ to minimize the influence of potential updates by the client. Note that if queries are from different rounds, the updates between two rounds could be too substantial, and using the same method for them may result in incorrect matches.

3.2 Inference of External SP and Merge

We recall that the IOA cannot distinguish whether two encrypted files from two rounds are under the same file with the AP or FVP leakage. However, the co-occurrence of queries (i.e., the probability of two queries appearing in the same file) remains across different rounds. For instance, the words "searchable" and "encryption" may frequently show up together in the database both before and after updates. We use co-occurrence to match queries across rounds.

First, we use the output of Algorithm 1 to get index matrices in one round. In the x-th round, the attack lists all the distinct file identities as F^x (for the FVP scenario, F^x contains the distinct file sizes). For each $G \in Gs^x$ of the output of Algorithm 1, we can get a

Algorithm 2 Inferring the external SP.

```
1:  Procedure InferESP($\{Gs^1, ...\}$, $\{\mathbf{IDH}^1, ...\}$, $\{\mathbf{IDT}^1, ...\}$)
2:  Initialize $M$ as a set that contains all groups of $\{Gs^1, ...\}$;
3:  for $i = 1$ to $\min(maxlevel, \eta - 1)$ do
4:      for $j = 1$ to $\eta - i$ do
5:          $M_{new} \leftarrow$ Match($Gs^j, \mathbf{IDT}^j, Gs^{j+i}, \mathbf{IDH}^{j+i}$);
6:          for $(G_1, G_2)$ in $M_{new}$ do
7:              $GMerge_1 \leftarrow$ GetMergedGroup($G_1, M$);
8:              $GMerge_2 \leftarrow$ GetMergedGroup($G_2, M$);
9:              if $GMerge_1$ and $GMerge_2$ does not contain queries from
                the same round then
10:                 $M.add(GMerge_1 \cup GMerge_2)$;
11:                 $M.del(GMerge_1)$;
12:                 $M.del(GMerge_2)$;
13: return $M$
14: End Procedure
15:
16: Procedure Match($Groups_1, \mathbf{ID}_1, Groups_2, \mathbf{ID}_2$)
17: $\mathbf{C}_1 \leftarrow \mathbf{ID}_1\mathbf{ID}_1^\top/|\mathbf{ID}_1[0]|, \mathbf{C}_2 \leftarrow \mathbf{ID}_2\mathbf{ID}_2^\top/|\mathbf{ID}_2[0]|$;
18: $P \leftarrow QuadraticAssignment(\mathbf{C}_1, \mathbf{C}_2)$;
    ▷ Get a match between $Groups_1$ and $Groups_2$ with a quadratic
    assignment algorithm;
19: Remove some incorrectly matched groups from $\mathbf{P}$ with a ratio
    of $p_g$ according to $\mathbf{C}_1$ and $\mathbf{C}_2$;
20: return Pairs of groups that are matched in $P$;
21: End Procedure
22:
23: Procedure GetMergedGroup($G, M$)
24: for $Group \in M$ do
25:     if $G \subseteq Group$ then
26:         return $Group$;
27: return $\emptyset$;
28: End Procedure
```

binary vector of size $|F^x|$ indicating whether the response of the first query in G contains the files in F^x, 1 for yes otherwise 0 (for the FVP scenario, the vector can indicate whether the response of the first query in G contains the file with the size in F^x). Then, we construct the index matrix $\mathbf{IDH^x}$ of the first query of each group, where each row of $\mathbf{IDH^x}$ corresponds to a G in Gs^x and the row is the binary vector above. Similarly, we construct the $\mathbf{IDT^x}$ of the last query of each group. We also build the matrix $\mathbf{ID^x}$ for later query recovery, where, in each row i, if the response of one of the queries in the group includes the j-th file in F^x, $\mathbf{ID^x}[i][j] = 1$.

Inferring the external SP (Algorithm 2) takes the groups $\{Gs^1, ..., Gs^\eta\}$ from the first module and the $\{\mathbf{IDH}^1, ..., \mathbf{IDH}^\eta\}$ and $\{\mathbf{IDT}^1, ..., \mathbf{IDT}^\eta\}$ as inputs and outputs the merged groups M. We first initialize the M as a set containing all groups of Gs^1,..., and Gs^η. We match groups of any two rounds by iteration (line 3-4). To save running time, we set a *maxlevel* that limits matching between two rounds when there are many rounds between them. When matching groups from round x and y ($x < y$), we extract the $\mathbf{IDT}^x$ and $\mathbf{IDH}^y$ and call the Match (line 5). We note that the files are refreshed between two index matrices from two different rounds. The Match first constructs the co-occurrence matrix $\mathbf{C}_1 = \mathbf{IDT}^x(\mathbf{IDT}^x)^\top/|F^x|$ and $\mathbf{C}_2 = \mathbf{IDH}^y(\mathbf{IDH}^y)^\top/|F^y|$ of Gs^x and Gs^y, respectively. With $\mathbf{C}_1$ and $\mathbf{C}_2$, we can estimate the mapping $\mathbf{P}$ by

$$\mathbf{P} = \underset{\mathbf{P} \in \mathcal{P}}{\operatorname{argmax}} \Pr(\mathbf{C}_1 | \mathbf{C}_2, \mathbf{P}), \tag{4}$$

where $\mathbf{P}$ is a matrix and $\mathbf{P}[i][j] = 1$ means the i-th group of Gs^x is matched to j-th group of Gs^y, otherwise $\mathbf{P}[i][j] = 0$. In [32], Oya et al. formalize query recovery using the co-occurrence matrix as a Quadratic Assignment Problem (QAP). They aim to match the co-occurrence matrices of keywords and queries, and propose an iterative heuristic attack called IHOP. Similarly, the problem of finding $\mathbf{P}$ is a QAP and can be solved by existing algorithms [21, 32, 35]. With some enhancements, we apply IHOP to determine such a mapping. Since co-occurrence persists across rounds, database updates do not affect the results. As some queries could not appear in both rounds x and y and still participate in the matching, we remove certain incorrectly matched groups from $\mathbf{P}$ by a ratio of p_g, according to $\mathbf{C}_1$ and $\mathbf{C}_2$. The details are in the full version[28].

For any pair of groups (G_1, G_2) in new matches, we call the GetMergedGroup to get the former merged group $GMerge_1$ of G_1 from M, where $GMerge_1$ is an element of M that contains G_1. We also obtain the former merged group $GMerge_2$ of G_2 from M. If both $GMerge_1$ and $GMerge_2$ contain the queries from the same round, indicating a conflict with a previous matching, the current matching will be disregarded. For example, in rounds 1, 2, and 3, there is group A in round 1, groups B and C in round 2, and group D in round 3. Group A is matched with B, and C is matched with D already. If group A now matches D, this conflicts with the previous match. Otherwise, we merge the two groups $GMerge_1$ and $GMerge_2$ in M (line 10-12). The output M contains all the groups, with each group comprising the queries from all considered rounds corresponding to the same keyword.

4 Peekaboo: Query Recovery

We propose the second part (P2) of Peekaboo to use the output of the P1 along with the leakage and the auxiliary knowledge to recover queries, see Figure 3. The P2 processes the leakage with the M to generate the frequency, volume, and co-occurrence information of groups. In this part, the attacker can easily call previous similar-data attacks with adaptations to recover the keyword of each group. We propose the instantiations, Sap+ and Jigsaw+, based on the SOTA attacks, Sap [31] and Jigsaw [29].

4.1 Attacker's Knowledge

Attackers' knowledge derived from auxiliary information. From a similar dataset D_s, the attacker obtains a similar keyword universe W_s, so that it can construct the frequency of each keyword as $\mathbf{F}_s = [\mathbf{f}_s^1, ..., \mathbf{f}_s^\tau]$ from public search frequency [29, 31, 32, 44], where $\mathbf{f}_s^i$ is a vector of length $|W_s|$ indicating the query frequency of each keyword in W_s in time slot i. Similarly, the attacker builds the volume of each keyword as $\mathbf{V}_s = [\mathbf{v}_s^1, ..., \mathbf{v}_s^\tau]$, where $\mathbf{v}_s^i[j]$ is the max number of files in D_s, including the j-th keyword in W_s during the time slot i. The attacker can further construct the index matrix as $\{\mathbf{ID}_s^1, ..., \mathbf{ID}_s^\eta\}$ based on W_s and D_s during the observation of each round.

Algorithm 3 Query recovery.

```
1:  Procedure QueryRec($M$, $\{\mathbf{ID}^1, ...\}$, $\{\mathbf{ID}_s^1, ...\}$, $\mathbf{F}_s$,$\mathbf{V}_s$)
2:  Initialize $\mathbf{F}_r$ as a $\tau \times |M|$ matrix with all zeros;
3:  Initialize $\mathbf{V}_r$ as a $\tau \times |M|$ matrix with all zeros;
4:  for all $k \in [\eta]$ do
5:      Initialize $\mathbf{ID}_r^k$ as a $|M| \times |\mathbf{ID}^k[0]|$ matrix with all zeros;
6:  for all $i \in [|M|]$ do
7:      $Group \leftarrow M.pop()$;
        ▷ Group contains search queries from all rounds that are matched as the queries for the same keyword;
8:      Set the $\tau$ elements in the $i$-th column of $\mathbf{F}_r$ as the numbers of queries of Group in $\tau$ time slots;
9:      Set the $\tau$ elements in the $i$-th column of $\mathbf{V}_r$ as the numbers of queries of Group in $\tau$ time slots;
10:     for all $k \in [\eta]$ do
11:         Find $j$ so that the Group contains the $j$-th group of $Gs^k$;
12:         Set the $i$-th row of $\mathbf{ID}_r^k$ as the $j$-th row of $\mathbf{ID}^k$;
13: Normalize each row of $\mathbf{F}_r$ by dividing the sum of that row;
14: for all $k \in [\eta]$ do
15:     $\mathbf{C}_r^k \leftarrow \mathbf{ID}_r^k(\mathbf{ID}_r^k)^\top / |\mathbf{ID}_r^k[0]|$;
16:     $\mathbf{C}_s^k \leftarrow \mathbf{ID}_s^k(\mathbf{ID}_s^k)^\top / |\mathbf{ID}_s^k[0]|$;
17: Call Attack+($\mathbf{F}_r, \mathbf{V}_r, \{\mathbf{C}_r^1, ...\}, \mathbf{F}_s, \mathbf{V}_s, \{\mathbf{C}_s^1, ...\}$) to recover the keyword of each group of $M$;
18: End Procedure
```

Attackers' knowledge from P1 of Peekaboo. The attacker uses the merged groups M, the index matrix $\{\mathbf{ID}^1, ..., \mathbf{ID}^\eta\}$ corresponding to the groups $\{Gs^1, ..., Gs^\eta\}$, and the VP of each search query.

4.2 Query Recovery with SP

Recall that in previous similar-data attacks against SSE [12, 25, 29, 31, 32, 35], the attacker uses the frequency, volume, and co-occurrence information to recover search queries. We process the same knowledge for search queries and keywords, but in a dynamic setting. The details are in Algorithm 3.

We first initialize the $\mathbf{F}_r$ and $\mathbf{V}_r$ as matrices with size $\tau \times |M|$ to record the search frequency and volume of each group in M in total τ time slots. We set the index matrices of the groups of M in each round as $[\mathbf{ID}_r^1, ..., \mathbf{ID}_r^\eta]$, where the matrices are initially zeros (line 2-5). For the i-th merged *Group* in M, we count the number of queries in *Group* of each time slot k and record the number in $\mathbf{F}_r[k][i]$ (line 8). We further mark down the max volume (the number of files in the response) of the queries in *Group* in each time slot k in $\mathbf{V}_r[k][i]$ (line 9). We realign the rows in $\mathbf{ID}^k$ as $\mathbf{ID}_r^k$ according to the order of merged groups in M. If the client issues search queries for a keyword in some rounds but not in the i-th round, the $\mathbf{ID}^i$ has no records for that keyword. We naturally keep that row in $\mathbf{ID}_r^i$ with zeros. Based on the index matrix of the keywords and the merged groups in M, we calculate the co-occurrence matrix. For each k of total η rounds, we compute $\mathbf{C}_r^k \leftarrow \mathbf{ID}_r^k(\mathbf{ID}_r^k)^\top/|\mathbf{ID}_r^k[0]|$ and $\mathbf{C}_s^k \leftarrow \mathbf{ID}_s^k(\mathbf{ID}_s^k)^\top/|\mathbf{ID}_s^k[0]|$ (line 14-16).

Peekaboo can instantiate Attack+ based on prior similar-data attacks, using the above information, i.e., the frequency, volume, and the co-occurrence of groups and keywords. We propose two instantiations in Section4.3 and Section 4.4 and provide a generic idea of instantiations over other attacks in Section 4.5.

About η and τ. We use the co-occurrence matrix for η rounds, along with the volume and frequency information for τ time slots. The volume and frequency vary across slots, and the attacker can observe these changes to acquire additional information. While the attacker could use the co-occurrence matrix for τ time slots, each time slot only involves a subset of the keywords, resulting in index and the co-occurrence matrices with a high proportion of zeros.

4.3 Instantiation: Sap+

We provide a brief review of Sap in the full version[28] and refer the reader to [31] for more details. The Sap attack assumes a static volume of search queries and keywords, using the volume and frequency information to solve a maximum likelihood problem and map search queries to keywords. In a DSSE, the database undergoes updates, altering keyword volume. Thus the original Sap attack cannot be directly applied to the dynamic volumes. In Sap+, we refine the maximum likelihood problem to

$$\mathbf{P} = \underset{\mathbf{P} \in \mathcal{P}}{\operatorname{argmax}} \Pr(\boldsymbol{\rho}, \mathbf{F}_r, \mathbf{V}_r, \mathbf{n}_D | \mathbf{F}_s, \mathbf{V}_s, \mathbf{P}), \tag{5}$$

where the $\mathbf{V}_r$ records the volume of groups in each time slot instead of the unchanged volume $\mathbf{v}_r$ in Sap and $\mathbf{n}_D$ is the vector of the number of total encrypted files in each time slot. Accordingly, we modify the cost matrix in Sap to

$$\begin{aligned} \mathbf{C}_v[i][j] = -\sum_{k=1}^{\tau} (&\mathbf{n}_D[k] \cdot \mathbf{v}_r^k[j] \cdot \log \mathbf{v}_s^k[i] + \\ &\mathbf{n}_D[k](1 - \mathbf{v}_r^k[j]) \cdot \log(1 - \mathbf{v}_s^k[i])) \end{aligned} \tag{6}$$

which summarizes the costs of different time slots. Like Sap, Sap+ also uses the Hungarian algorithm [24] to find a mapping $\mathbf{P}$ that indicates which keyword matches each group in M.

4.4 Instantiation: Jigsaw+

We here adapt the Jigsaw attack [29] to the dynamic scenario, where search frequency, volume, and co-occurrence information change over time. Note we review Jigsaw in the full version[28].

Similar to Sap, Jigsaw uses a static volume of search queries and keywords. It also uses static frequency information, i.e., the total search frequency of all time slots and a static co-occurrence relation. Thus, we should apply minor adaptation for the instantiation. Specifically, we revise the utilization of the total search frequency of queries in Jigsaw to the search frequency of groups in each time slot as $\mathbf{F}_r = \{\mathbf{f}_r^1, ..., \mathbf{f}_r^\tau\}$. Also, we use the volume of groups in each time slot as $\mathbf{V}_r = \{\mathbf{v}_r^1, ..., \mathbf{v}_r^\tau\}$. In Jigsaw+, we replace the static co-occurrence in all observed queries with the dynamic co-occurrence of groups as $\{\mathbf{C}_r^1, ..., \mathbf{C}_r^\eta\}$. At last, the auxiliary information $(\mathbf{f}_s, \mathbf{v}_s, \mathbf{C}_s)$ of Jigsaw is extended to the dynamic version $(\{\mathbf{f}_s^1, ..., \mathbf{f}_s^\tau\}, \{\mathbf{v}_s^1, ..., \mathbf{v}_s^\tau\}, \{\mathbf{C}_s^1, ..., \mathbf{C}_s^\eta\})$.

To utilize the volume and frequency information of multiple time slots, we modify the differential distance of the i-th group and the

j-th group in M as

$$d_i = \min_{j<|M| \wedge j \neq i} \sum_{k=1}^{\tau} \alpha \cdot |\mathbf{v}_r^k[i] - \mathbf{v}_r^k[j]| + (1-\alpha)|\mathbf{f}_r^k[i] - \mathbf{f}_r^k[j]|, \tag{7}$$

which summarizes the distance of all time slots. Based on this, Jigsaw+ can locate *BaseRec* distinctive groups instead of distinctive search queries. Similarly, the distance $s(i,j)$ between the i-th group of M and the j-th keyword in W_s is

$$s(i,j) = \sum_{k=1}^{\tau} \alpha \cdot |\mathbf{v}_r^k[i] - \mathbf{v}_s^k[j]| + (1-\alpha)|\mathbf{f}_r^k[i] - \mathbf{f}_s^k[j]|. \tag{8}$$

The subsequent confirmation uses the co-occurrence matrix of observations. Instead of using the static co-occurrence matrix, Jigsaw+ extracts the co-occurrence matrix of different rounds as $\{\mathbf{C}_r^{1\prime}, ..., \mathbf{C}_r^{\eta\prime}\}$ of the matched groups and $\{\mathbf{C}_s^{1\prime}, ..., \mathbf{C}_s^{\eta\prime}\}$ of their corresponding keywords based on $\{\mathbf{C}_r^1, ..., \mathbf{C}_r^\eta\}$ and $\{\mathbf{C}_s^1, ..., \mathbf{C}_s^\eta\}$. Then we define the *revconf* of the i-th recovered distinctive group as $revconf = \sum_{k=1}^{\eta} ||\mathbf{C}_r^{k\prime}[i] - \mathbf{C}_s^{k\prime}[i]||$.

Then, Jigsaw+ extracts the $\{\mathbf{C}_{rs}^1, ..., \mathbf{C}_{rs}^\eta\}$ and $\{\mathbf{C}_{ss}^1, ..., \mathbf{C}_{ss}^\eta\}$, as in Jigsaw. We set the *score* between the i-th group of the left unmatched groups and the j-th unmatched keyword j as $score = -\ln(\beta \sum_{k=1}^{\eta} ||\mathbf{C}_{rs}^k[i] - \mathbf{C}_{ss}^k[j]|| + (1-\beta)s(i,j))$.

Finally, Jigsaw+ calculates the *certainty* as the difference between the largest and the second largest *score*, then recovers the top *RefSpeed* groups with the highest *certainty*. Jigsaw+ treats the recovered groups as known matches and repeats the process until all groups are matched to keywords.

4.5 Instantiations of Other Attacks

Similar-data attacks. Peekaboo generates the dynamic search frequency, volume, and co-occurrence matrix of merged groups in M and keywords in W_s. A straightforward way to make instantiations over other similar-data attacks is to average the generated dynamic information to recover search queries. Another approach is to iterate the calculation in the attacks over multiple rounds and further aggregate the results as in Sap+ and Jigsaw+. For example, in Jigsaw+, we can compute the differential distances for each round and then summarize them (Equation 7) to replace the differential distance used in Jigsaw. We also provide an instantiation based on IHOP [32] in Appendix A, i.e. IHOP+. Any future similar-data attacks relying on frequency, volume, or co-occurrence information can adopt this method as an instantiation of Peekaboo.

Known-data attacks. We discuss the possibility of instantiating Peekaboo over previous known-data attacks (see the full version[28]).

5 Evaluation

We introduce settings and evaluation metrics, illustrate the performance of Peekaboo, and finally compare the query recovery results with the benchmarks. Our code is available in https://github.com/hustcpss/Peekaboo.

5.1 Setup

Dataset. We use the Enron and Lucene datasets in the experiments. The Enron email dataset [43] has 30,109 emails between 2000-2002, while the Lucene email dataset [15] includes 66,491 emails between 2001-2020, where all these emails are tagged with timestamps indicating the time they were sent. We also conduct experiments under another type of dataset of our attacks, i.e., Wikipedia[16] (results are in the full version[28]). We use a total of 5,525 keywords extracted from the datasets. Among these, we download the daily search trends of 3,000 keywords from PageViews [26], covering from July 2019 to July 2024. The keyword count is consistent with previous works [12, 31, 32, 35]. This approach is practical and also supported by Zipf's law [29, 49], which suggests that frequently used words are limited in number. Practical databases, such as those from [2], often utilize fewer than 3,000 keywords. We randomly divide the email dataset for each day into two halves, with one half considered as the client's dataset and the other half as a similar dataset known by the attacker. In practice, the attacker can obtain such a similar dataset. For example, an insider with access to a comparable email database or an industry competitor with a structurally similar database may exploit such resources to facilitate attacks. We assume the attacker is given half of the dataset that is a common experimental setting in prior work [12, 29, 31, 32, 35].

Client. We use the Enron and Lucene datasets to simulate real-world client behaviors in DSSE. We highlight that simulating SSE operations is straightforward: without any updates, there is no distinction between a persistent attacker and an IOA. The client adds the emails to the encrypted dataset daily according to the timestamps. For example, on the i-th day of the experiment, the client stores the emails with timestamps from the i-th day after the first day of the year 2000 for the Enron dataset. After the updates, the client deletes the outdated emails. For Enron, we assume the client can delete emails one year prior, while for Lucene, the client deletes the emails three years prior, due to the lower volume of emails per year in Lucene. We also assume the client randomly deletes emails from all the stored ones daily to simulate deletion behavior in practice. We set the number of randomly deleted emails to be ten percent of the number of newly added emails. For each day, the client can issue multiple queries for the extracted keywords according to the search frequency from PageViews.

Attacker. The attacker updates the dataset and deletes outdated emails to minimize the distributional differences between the attacker's and the client's datasets. We state that Peekaboo can also apply to the case that there is no update, i.e., the attacker only obtains a static dataset (see the full version[28]). The attacker does not imitate the client's random deletion behavior, as it is unlikely that the attacker would predict which files the client will delete. Thus, its deletion cannot contribute to the similarity of the datasets. We also assume the attacker can access the extracted keywords and obtain the corresponding search frequency, i.e., the true query frequencies of PageViews in our experiments. We note that if the instantiated Attack does not require query frequency, the same holds for Attack+. We provide the results of Jigsaw+ without query frequency in the full version[28] accordingly. It thus can generate the $\mathbf{F}_s$, $\mathbf{V}_s$, and $\{\mathbf{ID}_s^1, ..., \mathbf{ID}_s^\eta\}$ of the extracted keywords based on the similar dataset and search frequency. We specifically restrict the attacker to intermittent observations, where it attacks over η rounds. In the i-th round, the attacker observes the leakage for $\sigma[i]$ days, acquiring the search query sequence Q^i and the corresponding leakage, either AP^i or FVP^i. After that, it goes offline. We repeat the above strategy for subsequent rounds.

Parameter selection. Peekaboo introduces three hyperparameters, i.e., δ, $maxlevel$, and p_g. We provide the experimental results and analyses of these hyperparameters in Appendix B. Similar to FMA[44], δ is the threshold that reflects the changes in the responded files between successive queries of the same keyword within one round and is recommended to be larger than 0.5 [44]. Results (Figure 13a, Appendix B) show that it is best to choose a value between 0.6 and 0.95. $maxlevel$ is used to reduce the runtime cost. A smaller $maxlevel$ can result in shorter runtime with slightly decreased accuracy. From Figure 13b, we see that Jigsaw+ reaches near-maximum accuracy when $maxlevel$ is around 5. p_g is used to discard the mistakes in matching. Experiments in Figures 13c and 13d support selecting a value between 0.05 and 0.2. If more mistakes are expected (e.g., using only FVP or against countermeasures), p_g should be set slightly higher. Unless otherwise specified, we set $\delta = 0.95$, $maxlevel = 5$, and $p_g = 0.05$. Other parameters of Attack+, such as α and β in Jigsaw+, are introduced by the original works of the corresponding Attack, and we follow the recommendations provided in the corresponding references.

Evaluation metrics. We use the *adjusted rand index* (ARI) [20] for the evaluation of SP inference, which is based on the rand index (RI) [36]. The RI is a measure of the similarity between two partitions for a set S containing n elements. Concretely, given two partitions $X = \{X_1, ..., X_m\}$ and $Y = \{Y_1, ..., Y_k\}$, the RI can be calculated as $RI = (a + b)/\binom{n}{2}$, where a is the number of times a pair of elements belongs to the same partitions across X and Y, and b is the number of times a pair of elements belongs to the different partitions across X and Y. The ARI is the "adjusted-for-chance" version of RI, which is $ARI = (RI - ExpectedRI)/(MaxRI - ExpectedRI)$, where $ExpectedRI$ is the expected value of RI, and $MaxRI$ is the value of RI in the most ideal partition scenario (always equal to 1). This value lies between −1 and 1. The closer the ARI approaches 1, the more similar the two partitions are, and vice versa. For the evaluation of the SP inference, we group all the queries of the same keyword as partition X and treat the output of the P1 as Y.

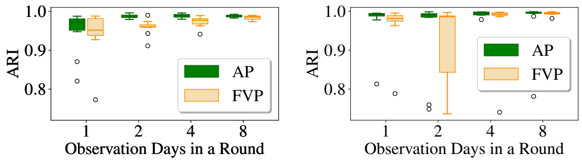
We use the *accuracy* to evaluate the performance of the query recovery attacks. The accuracy is calculated as the proportion of correctly recovered search queries to the total observed queries.

Ethical concerns. All datasets used in our experiments are publicly available and commonly used in previous studies. We provided the discussion of ethical considerations in Appendix C.

5.2 Evaluation of SP Inferring

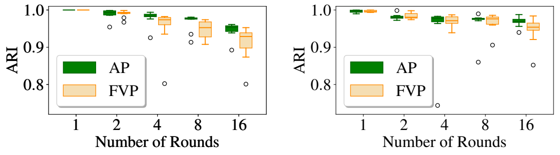
We use 500 keywords with the biggest volume in each dataset as the keyword universe. The client issues 1,000 queries daily.

The impact of observing days. We assume the attack lasts for 5 rounds, with 20 days per round. At the beginning of each round, we set that the attacker observes 1, 2, 4, and 8 days, respectively, for the rest of the 20 days, the attacker stays offline. As shown in Figure 4, the ARI of Peekaboo reaches above 0.9 in most cases, meaning most of the queries are accurately clustered w.r.t. the underlying keywords. As the number of days in a round increases, the attacker gains more information in the round to match two groups across rounds, leading to a gradual increase in the ARI. The ARI based on AP is higher than that based on FVP. Recall that FVP reveals the size of each file. But the attacker still cannot distinguish between two files of the same size, which influences the matching process.



(a) Enron **(b) Lucene**

Figure 4: The ARI of the search pattern inferring of Peekaboo with different observation times in each round.



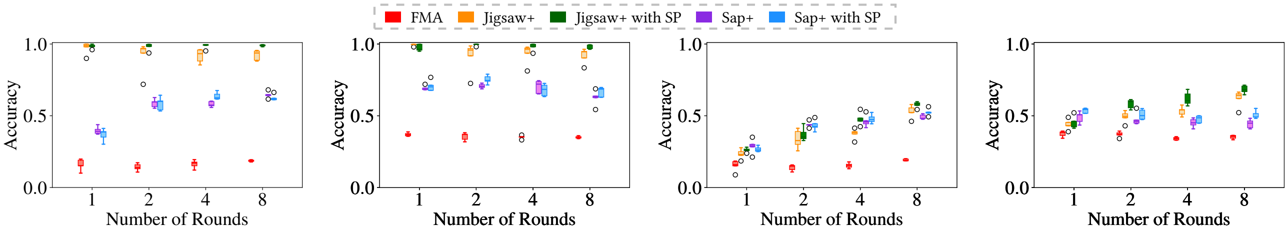
(a) Enron **(b) Lucene**

Figure 5: The ARI of the search pattern inferring of Peekaboo with different numbers of rounds.

The impact of round number. We set the round number to 1, 2, 4, 8, and 16, and in each round, the attacker observes for 2 days and goes offline for 18 days. The results are in Figure 5. We see that with an increasing number of rounds, the ARI tends to decrease over time. This is so because the errors in earlier matches affect the inference of the external SP in later rounds, causing an accumulation of errors. Although the ARI decreases with the increasing number of rounds, it still maintains practical performance, obtaining > 0.9 even when the round number reaches 16. As in Section 5.3, the decreasing ARI has minimal impact on query recovery as the round number increases, since the attacker can gain more information about the queries with many extra rounds of observations.

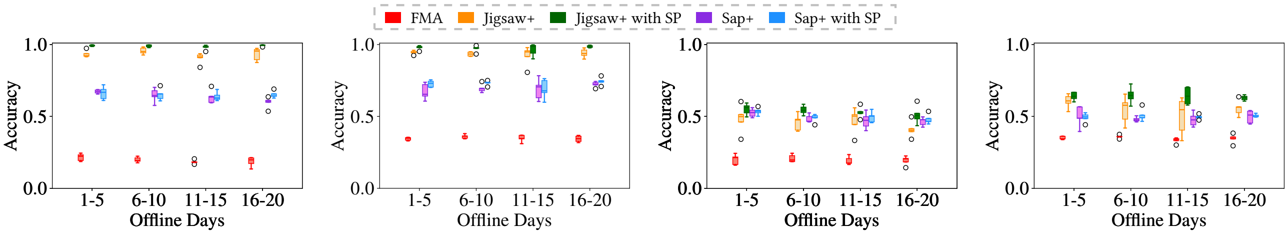
5.3 Evaluation of Query Recovery

We provide the comparison among Peekaboo with Jigsaw+ (denoted as Jigsaw+) and Peekaboo with Sap+ (denoted as Sap+). Note that since IHOP and Jigsaw exploit the same information and achieve comparable accuracy [29], we expect IHOP+ to match Jigsaw+ in performance. This is confirmed by the comparison of IHOP+ and Jigsaw+ in Appendix A. We also present the results of FMA [44]. Recall that FMA first recovers the SP of the observed queries using the Equations 1 and 2 and then calculates the frequency of queries. It treats the keywords with similar frequency of a query as candidates and narrows them down across different time slots. If there is only one candidate keyword left, FMA recovers the query. Besides the FVP leakage, we also test the FMA with AP leakage by replacing Equation 1 of FMA with Equation 3. To better understand the inferred SP, we set two benchmark attacks, "Jigsaw+ with SP"



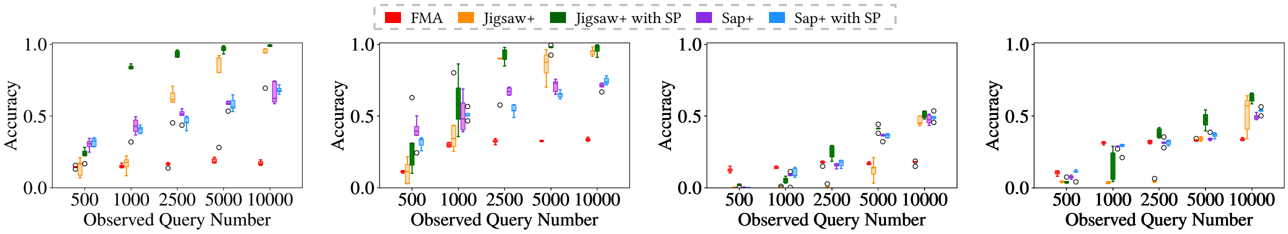
(a) Enron, AP **(b) Lucene, AP** **(c) Enron, FVP** **(d) Lucene, FVP**

Figure 6: The accuracy results of Jigsaw+, Sap+, FMA, Jigsaw+ with SP, and Sap+ with SP in Enron and Lucene with different numbers of rounds with the AP or FVP leakage.



(a) Enron, AP **(b) Lucene, AP** **(c) Enron, FVP** **(d) Lucene, FVP**

Figure 7: The accuracy results of Jigsaw+, Sap+, FMA, Jigsaw+ with SP, and Sap+ with SP in Enron and Lucene with different offline days in each round with the AP or FVP leakage.



(a) Enron, AP **(b) Lucene, AP** **(c) Enron, FVP** **(d) Lucene, FVP**

Figure 8: The accuracy results of Jigsaw+, Sap+, FMA, Jigsaw+ with SP, and Sap+ with SP in Enron and Lucene with different numbers of observed queries in each round with the AP or FVP leakage.

and "Sap+ with SP", where we assume the attacker knows the SP (both the internal and external) instead of inferring the SP, and then utilizes P2 of Peekaboo to recover the queries.

For Jigsaw+ and Jigsaw+ with SP, we set the α to 0.5 and β to 0.9. For the AP experiments, we set the *BaseRec* to 25 and *ConfRec* to 10, while for the FVP experiments, *BaseRec* is set to 15 and *ConfRec* to 5. For Sap+ and Sap+ with SP, α is set to 0.5. For FMA, δ is 0.95. We evaluate the attacks using either AP or FVP leakage.

The impact of round number. We set the round number to 1, 2, 4, and 8, with each round consisting of 10 days. In each round, the attacker observes 1 day and goes offline for 9 days. During the observation, the attacker monitors 10,000 queries each day. Other parameters remain the same as those in Section 5.2. The results are in Figure 6. In general, Jigsaw+ and Sap+ outperform FMA. With AP, the accuracy of Jigsaw+ and Sap+ is approximately 90% and 60%, respectively, while the accuracy of FMA is about 35%. With FVP, the gap between FMA and Jigsaw+ is roughly 30%. We see that the accuracy of Jigsaw+ and Sap+ increases as the round number increases. Observing multiple rounds provides the attacker with more information to recover queries, yielding higher accuracy for Jigsaw+ and Sap+. The accuracy of FMA changes little, and we believe this is because FMA does not consider updates between two observations, so the results with multiple rounds are similar to those with just one round. We also notice that although the accuracy of Jigsaw+ and Sap+ is lower than their with-SP versions, the gap is small, indicating that the search pattern inferred by Peekaboo is sufficient for query recovery in both Jigsaw+ and Sap+.

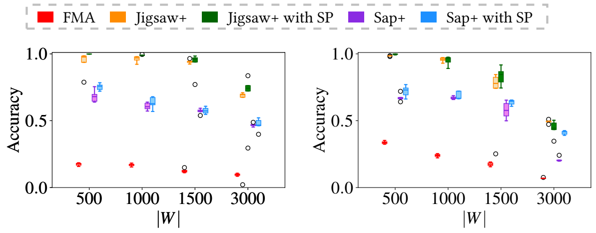
The impact of offline days. We also investigate attack performance with varying numbers of offline days in each round. We set the offline days by randomly sampling from 1-5, 6-10, 11-15, and 16-20, respectively. The attacker observes 1 day in each round and repeats this for 5 rounds. Other parameters remain unchanged.

The results are in Figure 7. The accuracy of Jigsaw+ and Sap+ does not decrease as the number of offline days increases, showing the inferred external SP is minimally affected by the offline days. For FMA, since it does not consider SP matching for different rounds, its performance remains with different offline days.

The impact of observed search queries. We also test the situations while the attacker observes a different number of search queries. We set the observed queries each day to 500, 1,000, 2,500, 5,000, and 10,000. The round number is set to 5, and the observing and offline days are set to 1 and 19, respectively. We keep other parameters the same as in the previous experiments. The results are presented in Figure 8. Increasing the observed queries, we can obtain better accuracy for Jigsaw+, Sap+, and their with-SP versions. With fewer observed search queries, the groups from different observations may vary significantly in terms of the underlying keywords, which negatively impacts the matching of groups in P1 of Peekaboo. Meanwhile, the attacker has less information to recover queries, resulting in relatively lower accuracy. But for FMA, as it uses the FVP or AP to group search queries and relies solely on frequency to recover them, the information available to the attacker is limited. As a result, the accuracy changes only slightly as the observed query number increases.

The impact of keyword universe size. We test the attacks with different keyword universe sizes (denoted as $|W|$), using 500, 1000, 1500, and 3000 keywords, with the largest volume in each dataset. The attacker observes 20,000 queries per day. For Jigsaw+ and Jigsaw+ with SP, we set the β to 0.7 as the larger keyword universe contains more low-volume keywords and their co-occurrence information is greatly noised due to the low volume. The accuracy is captured in Figure 9. The performance of all the attacks declines as the keyword universe size increases. In the Enron, the accuracy of Jigsaw+ and Sap+ drops from $> 95\%$ to 70% and from 70% to 45%, respectively, as $|W|$ jumps from 500 to 3000. FMA has $< 25\%$ accuracy even with only 500 keywords in Enron. The decline in accuracy when $|W| = 3000$ is expected, as a larger keyword universe increases the number of candidate keywords for each search query, introducing more low-volume keywords that are more sensitive to noise given by database updates.

Runtime and memory cost. The time complexity of Algorithm 1 is $O(|Q^x| \cdot |Gs^x| \cdot \log |Gs^x|)$, while Algorithm 2 requires $O(\eta^2 \cdot T_{\text{MATCH}})$. T_{MATCH} is the runtime of the MATCH function, which depends on the *QuadraticAssignment* it invokes. The runtime of Algorithm 3 mainly depends on the ATTACK+. We also present the runtime results in Table 3, where we execute INFERISP in parallel using 5 threads. Peekaboo completes in a few minutes, while FMA requires several hours. As the keywords universe size increases, Peekaboo's runtime also increases, as it must perform more computations on the co-occurrence matrix, which grows in size with $|W|$. As for FMA, however, its performance improves instead. This occurs because, as $|W|$ increases, the volume of new keywords and the average response size for queries decrease, making the calculation of similarities between queries less complex. We further state that our attack has modest memory requirements. Under the same settings as in Figures 6a, 6b, and the test of Wikipedia, Jigsaw+ and Sap+ consume at most 6.8, 6.6 and 15.2 GB of memory when evaluated on Enron, Lucene, and Wikipedia, respectively.



(a) Enron, AP **(b) Lucene, AP**

Figure 9: The accuracy of Jigsaw+, Sap+, FMA, Jigsaw+ with SP, and the Sap+ with SP in Enron and Lucene with different keyword universe sizes $|W|$ with the AP leakage.

Table 3: The runtime results of algorithms with different keyword universe sizes $|W|$.

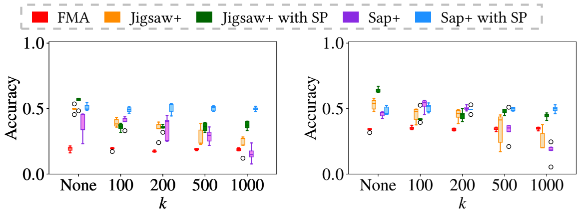
Algorithm	Runtime (s) with different $\|W\|$			
	500	1000	1500	3000
Jigsaw+	284.89	355.93	817.37	3089.82
Sap+	285.42	312.16	592.33	1304.54
FMA	17490.01	10169.8	7063.96	4006.73

6 Against Countermeasures

We evaluate the attacks in Section 5.3 under the padding of file size and the obfuscation of AP.

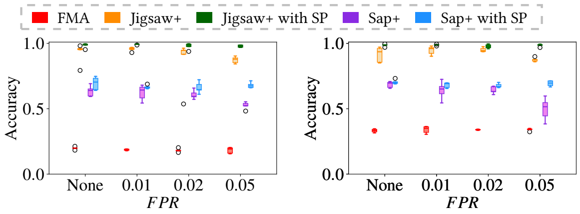
Padding. We assume the client pads each file size to a multiple of k, with k set to 100, 200, 500, and 1,000. We use this to add noise to the FVP and affect the results of both the SP inference and query recovery. For instance, when $k = 500$, there are only 57 and 125 distinct file sizes in Enron and Lucene, respectively; when $k = 1000$, these numbers decrease to 34 and 80, respectively. As the FVP is noised, the subsequent calculation of the co-occurrence is also affected by the noise. Thus, we use a larger p_g in P1 of Peekaboo to remove more matches between groups. We also set a smaller β in Jigsaw+ to balance the noise in the co-occurrence matrix. We set the p_g to 0.15 and the β to 0.7. As previous works [29, 32, 38] have shown that adaptations can improve attack accuracy against countermeasures, we apply the same approach. Specifically, the attacker adopts a similar adaption as in [29], applying the same padding with identical parameters as the client to its own dataset in order to minimize the difference between the two datasets. All attacks in our experiments incorporate this adaptation. The attacks are tested over 5 rounds, with 1 day of observation (10,000 search queries per day) and 9 days of offline in each round. Other parameters are consistent with those in Section 5.3. As padding does not interfere with the AP, we examine the attacks using only the FVP leakage.

We present the results in Figure 10. The accuracy of Sap+ with SP (about 50%) remains stable, as the attacker has access to the SP, and Sap+ does not rely on information related to the size of the files. In contrast, Jigsaw+ with SP loses 20% performance, as it depends on the co-occurrence information of groups, which is affected by the noise introduced by the padding of file sizes. For Jigsaw+ and Sap+, the inference of the SP in Peekaboo is influenced by the padding, resulting in a drop in accuracy from approximately



(a) Enron, FVP **(b) Lucene, FVP**

Figure 10: The accuracy of Jigsaw+, Sap+, FMA, Jigsaw+ with SP, and the Sap+ with SP in Enron and Lucene against the padding of the file size with the FVP leakage.



(a) Enron, AP **(b) Lucene, AP**

Figure 11: The accuracy of Jigsaw+, Sap+, FMA, Jigsaw+ with SP, and the Sap+ with SP in Enron and Lucene against obfuscation with the AP leakage.

50% to about 20%. The performance of FMA remains largely stable, at about 20% in Enron and 35% in Lucene. We say that FMA (and P1 of Peekaboo) calculates the similarity between two queries using the intersection and union of file sizes, including duplicates. Both the file sizes and the number of files in the leakage contribute to this calculation and the number of files is not affected. Thus, the padding only delivers minor influence on the similarity calculation.

Obfuscation of the AP. We adopt a similar countermeasure as in [9] and assume that during the attacker's offline, the client downloads the whole dataset and rebuild the index to simulate dynamic updates under obfuscation. While reconstructing the index, the client deletes the index of files for each keyword with a probability of TPR and adds files not present in the keyword's response with a probability of FPR. We set the TPR to 0.999 and FPR to 0.01, 0.02, and 0.05. The attacker employs a strategy similar to [29] which applies the same obfuscation to the attacker's dataset using identical parameters as the client, in order to minimize the difference between the two datasets. All attacks in our experiments employ the same adaptation. The observation settings are identical as those in padding and other parameters are consistent with those in Section 5.3.

In Figure 11, the results are largely consistent with those from the experiments in the previous section. Jigsaw+ and Jigsaw+ with SP lead with about 90% accuracy, followed by Sap+ and Sap+ with SP at around 60%. FMA achieves only 20% accuracy in Enron and about 35% in Lucene. The accuracy of Jigsaw+ with SP and Sap+ with SP remains relatively stable across varying FPR values. Sap+ with SP relies on frequency and volume. The frequency remains unaffected by obfuscation because it assumes the attacker knows the SP. Though the volume is affected by obfuscation, the attacker's adoption of the same obfuscation method balances out the added volume across keywords, preserving accuracy. For Jigsaw+ with SP, the strong performance against obfuscation mirrors that of Jigsaw [29], maintaining high accuracy. The accuracy of Jigsaw+ and Sap+ also remains stable when $FPR \leq 0.02$ but drops by approximately 15% when FPR increases to 0.05.

Discussions about Countermeasures. To counter Peekaboo, a practical approach is to limit the number of search queries the attacker can observe per round by quickly detecting and blocking the attacker's access. As we assume the attacker eavesdrops on the server or the communication channel, the server can implement stricter intrusion detection systems to limit the duration and number of rounds the attacker can observe. This reduces the amount of information available to the attacker, making it harder to perform effective query recovery. For example, in Figure 8, when the number of observed queries is limited, the attack accuracy drops. Similarly, as illustrated in Figure 6c and 6d, further restricting the round number of observations can also reduce the accuracy when dealing with the FVP leakage. However, if the attacker gains access to the AP, even a single observation round can produce high accuracy.

The padding proves effective in mitigating the FVP leakage. But with the AP leakage, Peekaboo remains robust even against dynamic obfuscation. We believe that Peekaboo can pose severe threats to other DSSE schemes that reveal AP and include padded dummy files in the responses. A possible approach to counter Peekaboo is to implement "stronger" parameters for padding or obfuscation. However, developing an efficient dynamic padding or obfuscation remains an open challenge. Alternatively, technologies like ORAM or PIR could prevent AP and FVP leakage, providing a possible defense against Peekaboo. But they typically involve significant communication or computation overhead.

7 Conclusion

In this work, we consider an intermittent-observation attacker who has only intermittent observation ability against DSSE. We formalize the leakage and propose a new attack called Peekaboo. In Peekaboo, the attacker first infers the SP with the AP or FVP leakage and then combines it with auxiliary knowledge and the leakage to recover search queries. Peekaboo is a generic interface for similar-data attacks. We propose Jigsaw+ and Sap+ as instantiations. We conduct extensive experiments to confirm that Peekaboo can achieve a well-inferred SP, and Peekaboo with Jigsaw+ and Sap+ respectively provide 90% and 60% accuracy with AP, and about 50% and 45% with FVP. Our design also demonstrates practical efficiency. Even against countermeasures, Peekaboo still maintains its threats.

8 Acknowledgments

We sincerely thank all reviewers and the shepherd for their insightful and constructive feedback toward improving the paper. This work was supported by the National Key Research and Development Program of China under Grant No. 2022YFB4501500 and the National Natural Science Foundation of China under Grant No. 62372201 and No. 62272186.

References

[1] 2018. Latest Trickbot Variant has New Tricks Up Its Sleeve. https://www.cyberbit.com/endpoint-security/latest-trickbot-variant-has-new-tricks-up-its-sleeve/

[2] 2023. WHO Model List of Essential Medicines. https://www.who.int/publications/i/item/WHO-MHP-HPS-EML-2023.02

[3] N Ambika. 2020. Improved Methodology to Detect Advanced Persistent Threat Attacks. In *Quantum Cryptography and the Future of Cyber Security.*

[4] Ghous Amjad, Seny Kamara, and Tarik Moataz. 2019. Breach-Resistant Structured Encryption. *Proc. Priv. Enhancing Technol.* 2019, 1 (2019), 245–265. doi:10.2478/POPETS-2019-0014

[5] Laura Blackstone, Seny Kamara, and Tarik Moataz. 2020. Revisiting Leakage Abuse Attacks. In *27th Annual Network and Distributed System Security Symposium.* https://www.ndss-symposium.org/ndss-paper/revisiting-leakage-abuse-attacks/

[6] Raphael Bost. 2016. $\sum o\varphi o\varsigma$: Forward Secure Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* 1143–1154. doi:10.1145/2976749.2978303

[7] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. 2017. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.* 1465–1482. doi:10.1145/3133956.3133980

[8] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.* 668–679. doi:10.1145/2810103.2813700

[9] Guoxing Chen, Ten-Hwang Lai, Michael K. Reiter, and Yinqian Zhang. 2018. Differentially Private Access Patterns for Searchable Symmetric Encryption. In *2018 IEEE Conference on Computer Communications.* 810–818. doi:10.1109/INFOCOM.2018.8486381

[10] Tianyang Chen, Peng Xu, Stjepan Picek, Bo Luo, Willy Susilo, Hai Jin, and Kaitai Liang. 2023. The Power of Bamboo: On the Post-Compromise Security for Searchable Symmetric Encryption. In *30th Annual Network and Distributed System Security Symposium.* https://www.ndss-symposium.org/ndss-paper/the-power-of-bamboo-on-the-post-compromise-security-for-searchable-symmetric-encryption/

[11] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security.* 79–88. doi:10.1145/1180405.1180417

[12] Marc Damie, Florian Hahn, and Andreas Peter. 2021. A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents. In *30th USENIX Security Symposium.* 143–160. https://www.usenix.org/conference/usenixsecurity21/presentation/damie

[13] Haochen Dou, Zhenwu Dan, Peng Xu, Wei Wang, Shuning Xu, Tianyang Chen, and Hai Jin. 2024. Dynamic Searchable Symmetric Encryption With Strong Security and Robustness. *IEEE Trans. Inf. Forensics Secur.* 19 (2024), 2370–2384. doi:10.1109/TIFS.2024.3350330

[14] Yang Du, Daniel Genkin, and Paul Grubbs. 2022. Snapshot-Oblivious RAMs: Sub-logarithmic Efficiency for Short Transcripts. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference*, Vol. 13510. 152–181. doi:10.1007/978-3-031-15985-5_6

[15] Apache Foundation. 1999. Mail Archieves of Lucene. https://mailarchives.apache.org/mod_mbox/#lucene

[16] Wikipedia Foundation. 2020. Wikipedia databases. https://www.wikipedia.org.

[17] Javad Ghareh Chamani, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. 2018. New Constructions for Forward and Backward Private Symmetric Searchable Encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.* 1038–1055. doi:10.1145/3243734.3243833

[18] Zichen Gui, Kenneth G. Paterson, and Sikhar Patranabis. 2023. Rethinking Searchable Symmetric Encryption. In *44th IEEE Symposium on Security and Privacy.* 1401–1418. doi:10.1109/SP46215.2023.10179460

[19] Jacob Haltiwanger and Thang Hoang. 2024. Exploiting Update Leakage in Searchable Symmetric Encryption. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy.* 115–126. doi:10.1145/3626232.3653260

[20] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2 (1985), 193–218.

[21] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *19th Annual Network and Distributed System Security Symposium.* https://www.ndss-symposium.org/ndss2012/access-pattern-disclosure-searchable-encryption-ramification-attack-and-mitigation

[22] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Jamie DeMaria, Andrew Park, and Amos Treiber. 2024. MAPLE: MArkov Process Leakage attacks on Encrypted Search. *Proc. Priv. Enhancing Technol.* 2024, 1 (2024), 430–446. doi:10.56553/POPETS-2024-0025

[23] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security.* 965–976. doi:10.1145/2382196.2382298

[24] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[25] Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu-an Tan. 2014. Search pattern leakage in searchable encryption: Attacks and new construction. *Inf. Sci.* 265 (2014), 176–188. doi:10.1016/J.INS.2013.11.021

[26] Marcel Ruiz Forns MusikAnimal, Kaldari. 2015. Pageviews Toolforge. https://pageviews.toolforge.org/

[27] Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter. 2014. Dynamic Searchable Encryption via Blind Storage. In *2014 IEEE Symposium on Security and Privacy.* 639–654. doi:10.1109/SP.2014.47

[28] Hao Nie, Wei Wang, Peng Xu, Wei Chen, Laurence T. Yang, Mauro Conti, and Kaitai Liang. 2025. Peekaboo, I See Your Queries: Passive Attacks Against DSSE Via Intermittent Observations Full Version. https://doi.org/10.48550/arXiv.2509.03806

[29] Hao Nie, Wei Wang, Peng Xu, Xianglong Zhang, Laurence T. Yang, and Kaitai Liang. 2024. Query Recovery from Easy to Hard: Jigsaw Attack against SSE. In *33rd USENIX Security Symposium.* https://www.usenix.org/conference/usenixsecurity24/presentation/nie

[30] Jianting Ning, Xinyi Huang, Geong Sen Poh, Jiaming Yuan, Yingjiu Li, Jian Weng, and Robert H. Deng. 2021. LEAP: Leakage-Abuse Attack on Efficiently Deployable, Efficiently Searchable Encryption with Partially Known Dataset. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security.* 2307–2320. doi:10.1145/3460120.3484540

[31] Simon Oya and Florian Kerschbaum. 2021. Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption. In *30th USENIX Security Symposium.* 127–142. https://www.usenix.org/conference/usenixsecurity21/presentation/oya

[32] Simon Oya and Florian Kerschbaum. 2022. IHOP: Improved Statistical Query Recovery against Searchable Symmetric Encryption through Quadratic Optimization. In *31st USENIX Security Symposium.* 2407–2424. https://www.usenix.org/conference/usenixsecurity22/presentation/oya

[33] Giuseppe Persiano and Kevin Yeo. 2023. Limits of Breach-Resistant and Snapshot-Oblivious RAMs. In *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference*, Vol. 14084. 161–196. doi:10.1007/978-3-031-38551-3_6

[34] Rishabh Poddar, Stephanie Wang, Jianan Lu, and Raluca Ada Popa. 2020. Practical Volume-Based Attacks on Encrypted Databases. In *IEEE European Symposium on Security and Privacy.* 354–369. doi:10.1109/EUROSP48549.2020.00030

[35] David Pouliot and Charles V. Wright. 2016. The Shadow Nemesis: Inference Attacks on Efficiently Deployable, Efficiently Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* 1341–1352. doi:10.1145/2976749.2978401

[36] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[37] Khosro Salmani and Ken Barker. 2021. Don't fool yourself with Forward Privacy, Your queries STILL belong to us!. In *CODASPY '21: Eleventh ACM Conference on Data and Application Security and Privacy.* 131–142. doi:10.1145/3422337.3447838

[38] Zhiwei Shang, Simon Oya, Andreas Peter, and Florian Kerschbaum. 2021. Obfuscated Access and Search Patterns in Searchable Encryption. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021.* The Internet Society. https://www.ndss-symposium.org/ndss-paper/obfuscated-access-and-search-patterns-in-searchable-encryption/

[39] Dawn Xiaodong Song, David A. Wagner, and Adrian Perrig. 2000. Practical Techniques for Searches on Encrypted Data. In *2000 IEEE Symposium on Security and Privacy.* 44–55. doi:10.1109/SECPRI.2000.848445

[40] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. 2014. Practical Dynamic Searchable Encryption with Small Leakage. In *21st Annual Network and Distributed System Security Symposium.* https://www.ndss-symposium.org/ndss2014/practical-dynamic-searchable-encryption-small-leakage

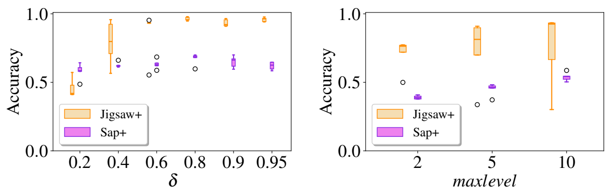
[41] Shi-Feng Sun, Xingliang Yuan, Joseph K. Liu, Ron Steinfeld, Amin Sakzad, Viet Vo, and Surya Nepal. 2018. Practical Backward-Secure Searchable Encryption from Symmetric Puncturable Encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.* 763–780. doi:10.1145/3243734.3243782

[42] Viet Vo, Xingliang Yuan, Shi-Feng Sun, Joseph K. Liu, Surya Nepal, and Cong Wang. 2023. ShieldDB: An Encrypted Document Database With Padding Countermeasures. *IEEE Trans. Knowl. Data Eng.* 35, 4 (2023), 4236–4252. doi:10.1109/TKDE.2021.3126607

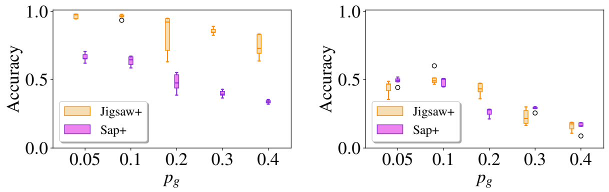
[43] CMU William W. Cohen, MLD. 2015. Enron Email Datasets. https://www.cs.cmu.edu/~./enron/

[44] Lei Xu, Leqian Zheng, Chengzhi Xu, Xingliang Yuan, and Cong Wang. 2023. Leakage-Abuse Attacks Against Forward and Backward Private Searchable Symmetric Encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security.* 3003–3017. doi:10.1145/3576915.3623085

[45] Lei Xu, Anxin Zhou, Huayi Duan, Cong Wang, Qian Wang, and Xiaohua Jia. 2024. Toward Full Accounting for Leakage Exploitation and Mitigation in Dynamic



(a) δ**, Enron, AP** **(b)** *maxlevel***, Enron, AP**



(c) p_g**, Enron, AP** **(d)** p_g**, Enron, FVP**

Figure 13: The accuracy of Jigsaw+ and Sap+ with different hyperparameters, i.e., δ**,** *maxlevel***, and** p_g**.**

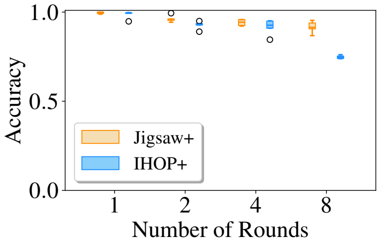


Figure 12: The accuracy of Jigsaw+ and IHOP+ in Enron with AP leakage.

Encrypted Databases. *IEEE Trans. Dependable Secur. Comput.* 21, 4 (2024), 1918–1934. doi:10.1109/TDSC.2023.3296189

[46] Peng Xu, Willy Susilo, Wei Wang, Tianyang Chen, Qianhong Wu, Kaitai Liang, and Hai Jin. 2022. ROSE: Robust Searchable Encryption With Forward and Backward Security. *IEEE Trans. Inf. Forensics Secur.* 17 (2022), 1115–1130. doi:10.1109/TIFS.2022.3155977

[47] Xianglong Zhang, Wei Wang, Peng Xu, Laurence T. Yang, and Kaitai Liang. 2023. High Recovery with Fewer Injections: Practical Binary Volumetric Injection Attacks against Dynamic Searchable Encryption. In *32nd USENIX Security Symposium.* 5953–5970. https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-xianglong

[48] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. 2016. All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption. In *25th USENIX Security Symposium.* 707–720. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/zhang

[49] George Kingsley Zipf. 2016. *Human behavior and the principle of least effort: An introduction to human ecology.* Ravenio books.

A Instantiation: IHOP+

IHOP includes two coefficients for the costs of matching: c for the quadratic terms and d for the linear terms. With the co-occurrence matrix information provided by Peekaboo, the c and d (in [32], Equation 5) are set to

$$\begin{aligned} c_{i,i',j,j'} = -\sum_{k=1}^{\eta} \mathbf{n}_D (\mathbf{C}_r^k[j][j'] \log(\mathbf{C}_s^k[i][i']) \\ -(1-\mathbf{C}_r^k[j][j']) \log(1-\mathbf{C}_s^k[i][i'])), \end{aligned} \tag{9}$$

$$\begin{aligned} d_{i,j} = -\sum_{k=1}^{\eta} \mathbf{n}_D (\mathbf{C}_r^k[j][j]) \log(\mathbf{C}_s^k[i][i]) \\ -(1-\mathbf{C}_r^k[j][j]) \log(1-\mathbf{C}_s^k[i][i])), \end{aligned} \tag{10}$$

which summarizes the costs in different rounds. With the frequency information provided by Peekaboo, the d (in [32], Equation 6) is set to

$$d_{i,j} = -\sum_{k=1}^{\tau} \boldsymbol{\rho}[k](\mathbf{f}_r^k[j] \log(\mathbf{f}_s^k[i]), \tag{11}$$

which represents the costs in each time slot. As both the co-occurrence and frequency information are available, IHOP+ summarizes the Equation 10 and 11 as in IHOP. Following the attack strategy of IHOP, IHOP+ can recover the keywords of search queries in each group of M.

In [29], it is reported that IHOP performs with accuracy similar to Jigsaw. We here compare IHOP+ with Jigsaw+ under the same settings as in Figure 6a, Section 5.3, and present the results in Figure 12. The accuracy of IHOP+ is comparable to that of Jigsaw+ and declines as the number of rounds increases.

B Parameter Selection and Impact

We provide the results of Jigsaw+ and Sap+ under different hyperparameter settings, i.e., δ, *maxlevel*, and p_g. If the response similarity in Equations 1 and 3 is no less than δ, then the two queries are considered candidates for querying the same keyword. The results in Figure 13a show that Jigsaw+ and Sap+ have the best accuracy of above 90% and 0.6%, respectively, when δ is between 0.6 and 0.95. *maxlevel* cancels the matching between two rounds if there are more than *maxlevel* rounds between them. A small *maxlevel* reduces runtime but may also lead to lower accuracy. We test Jigsaw+ and Sap+ with 16 rounds, and the results are shown in Figure 13b. The accuracy increases as *maxlevel* increases, and Jigsaw+ and Sap+ reach about 80% and 50% accuracy, respectively, when *maxlevel* is no less than 5. p_g is the ratio of removed matches in MATCH, in case some queries do not appear in both rounds of the input but still participate in matching, which causes errors. Other matching errors, such as those caused by noisy information, can also be filtered out by p_g. We evaluate p_g with AP and FVP, and the results show that p_g can be chosen between 0.05 and 0.2.

C Ethical Consideration

This work introduces a more restricted yet practical attacker based on intermittent observation, aiming to inspire new studies on enhancing the security of DSSE. It delivers a positive impact on the development and deployment of DSSE. All experiments use publicly available datasets widely used in prior studies and the experiments are conducted for research purpose only, in an isolated virtual environment. Queries are randomly generated using public trend datasets, and leakage is simulated (instead of "real" leakage). The experiments are not deployed to any real-world applications. All tested attacks, defenses, and related libraries are open-source. To support mitigation on attacks, we also discuss potential countermeasures in Section 6.