Digital Forgetting Using Key Decay

Darwish, Marwan Adnan; Zarras, Apostolis

DOI
10.1145/3555776.3577641

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC 2023

Digital Forgetting Using Key Decay

Marwan Adnan Darwish
Delft University of Technology
M.DarwishKhabbaz@tudelft.nl

Apostolis Zarras
University of Piraeus
Foundation for Research and Technology – Hellas
Zarras@ssl-unipi.gr

ABSTRACT

During the recent development of information technology and the prevalent breakthroughs of its services, more digital data tend to be readily stored online. Although the massive advantages, there is a pivotal necessity for curating digital data forgetting. Online content can pose perilous threats in terms of privacy and security that may hinder the right to be forgotten, encompassed by the GDPR act, since the released data can be archived and accessed retrospectively. Prior approaches focused on various access heuristics and elastic expiration times to make the data unreachable to some extent. However, there are still many pending issues related to the proposed studies, such as securing ephemeral key storage and co-ownership data deletion. In this paper, we attempt to tackle the problem of storing ephemeral keys during the estimated validity period. Hence, we devise a novel concept called key decay over time, which can achieve the ephemeral existence of the key. The decay idea entails the gradual, irreversible corruption of the key with time passing. In the current work, we combine the concept of gradual time elapsing and corruption into a single notion of the decay rate. Meanwhile, the irreversibility merit formed by randomness and various obfuscation strategies impedes retrospective attacks. Over time, the decay rate will give an estimated range for the key to be destroyed entirely. Finally, we implement and thoroughly assess a proof-of-concept regarding the key decay, including computational complexity and security analysis.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols; Privacy protection**;

KEYWORDS

Digital Forgetting; Retrospective Privacy; Key Decay

ACM Reference Format:
Marwan Adnan Darwish and Apostolis Zarras. 2023. Digital Forgetting Using Key Decay. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-March 31, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3555776.3577641

1 INTRODUCTION

Online data has been increasing rapidly, exponentially challenging our ability to manage and store it. IBM released a study that data growth witnessed a tremendous increment in 2020 to reach 40 trillion gigabytes (or 40 zettabytes), meaning every person on earth generates 1.7MB of data each second [11]. Meanwhile, according to Cisco, Internet consumers have reached 3.9 billion in 2018, and the number is expected to reach up to 5.6 billion in 2023 [5]. Nevertheless, the data owner may not need the online content to be accessible when its need ceases over time. On the other hand, the inadequate deletion practice of outsourced data jeopardizes plenty of security and privacy risks, including the possibility of data alteration or misuse by unauthorized parties and loss of confidentiality [12, 18].

The *General Data Protection Regulation* (GDPR) of the European Union enshrines the *Right to Erasure* or the *Right to be Forgotten* as it is widely known [15, 19]. This notion is gaining thrust and becoming extremely significant in protecting online privacy. Digital forgetting refers to the disappearance of data that has been uploaded to online storage platforms after it has fulfilled its purpose. In this sense, several underlying mechanisms have been developed to support digital data forgetting. For instance, cryptographic mechanisms have been proposed to encrypt the uploaded data in cipher form alongside the links. These links are the map for each data object to combine its key from ephemeral storage in order to be able to decrypt the online content within the validity period [21, 22]. On top of that, distributed architectures allow data owners to divide encryption/decryption keys in a distributed manner, using predetermined or flexible expiration periods, and bypassing single authority failures [4, 8, 14, 16, 25]. These approaches utilize various services, such as the *Domain Name System* (DNS), the *Distributed Hash Table* (DHT), or website pages, to store ephemeral keys (i.e., keys with a short lifetime).

Unfortunately, the previous approaches depend on ephemeral storage for their decryption keys. This is problematic due to insufficient deterrence of the archival process. Even with DNS entries, the storage for ephemeral keys is still vulnerable to being cached and used retroactively [20]. As long as the ephemeral keys exist, anyone with access and sufficient resources or the service provider can cache them beforehand. The adversary will be interested in reaching out to the data after its validity has ceased. The crucial characteristic to be offered by any infrastructure is to impede and prevent keys collection on a large scale during the lifetime of the online content. This is challenging due to the availability of the keys to the public on the ephemeral storage.

In this work, we focus on excluding the necessity of the key storage and preventing the archival process from a retrospective adversary acting periodically by caching the whole data on the available distributed infrastructures such as DNS. More specifically, we introduce the main idea of *key decay*. The decay concept is formed by three major ingredients: *time*, *corruption*, and *irreversibility*. Time implies a gradual sequence of events that follow one after another. Corruption entails changing from a state of soundness

SAC '23, March 27-March 31, 2023, Tallinn, Estonia
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9517-5/23/03.
https://doi.org/10.1145/3555776.3577641

or perfection into a completely unusable and unsound state. Irreversibility is closely related to the notion of time in physics. This must be added separately due to the simplicity of reversing any temporal sequence in computer science without adding extra means to ensure such irreversibility.

As such, encryption is based on a key powered by random numbers to obtain a unique output called a seed. The creation seed of the key depends on a value that gradually changes from an old state, resulting in an entirely new one. Finally, the key irreversibility concept consists of *randomization* and *obfuscation*: different elements are combined in randomization to obtain miscellaneous values from online sources, while obfuscation complicates the origin key by crystallizing several techniques.

In essence, our approach reinforces the online data deletion by ensuring the transience of the key without counting on any ephemeral storage. The creation seed (i.e., randomly generated value) for the key will be obtained from online data, such as YouTube views, Twitter reposts, and Facebook likes. These online values will be represented as x_i coordinate in the Lagrange-basis polynomial [7]. Gradually these online data values will modify, leading to utterly different values. This evolution of the values over time will coin the corruption concept. This corruption process for online values leads to a key forgetting in the end. For the key recovery, we use *Shamir Secret Sharing Schema* (SSSS) to combine the correct parts during the reconstruction phase. The more corruption, the less key recovery as time goes on. Eventually, when the key decays, the irreversibility attribute prevents retrospective attacks from exposing the origin of the key. Moreover, the decay rate will provide insights into the expiration time of the key (i.e., days, weeks, years) according to the ephemerality level of the online data. We implement a system incorporating SSSS with the decay notion as a proof of concept. Introducing the key decay concept ultimately prevents the retrospective impact of an attacker or cloud provider that aims to store private keys and use them afterward.

In summary, we make the following main contributions:

- We propose using key decay for digital forgetting and prove how this will provide security against retrospective attacks.
- We design a scheme that guarantees the key ephemerality without relying on ephemeral storage.
- We evaluate our key decay approach by implementing a prototype. Our results illustrate the decay rate in terms of evolution, stability, irreversibility, and computational complexity.

2 THREAT MODEL

Existing data deletion schemes [4, 8, 14, 16, 25] frequently adopt the following procedure: (*i*) Data owners outsource their data to service providers after encrypting the content. In addition, the owners provide the corresponding links necessary for compiling the decryption keys at the recipients' end. (*ii*) During the validity (i.e., a dedicated lifetime of the data), within a specific period stated by the owners (i.e., fixed or flexible), the recipients with valid access can combine the key bits to decrypt the online content. (*iii*) After the validity, no one should be able to access the encrypted data object except those who already have private keys. In principle, those schemes generally leverage ephemeral storage such as cache entries, including DHT, website ciphers, and the DNS cache to support digital oblivion.

The main objective of our approach is to impede the retrospective adversary and fulfill online privacy. As such, we assume an adversary who is only curious about accessing the online content after its expiration. We also assume a cloud provider, or any other party, who periodically caches the key in the ephemeral storage. Snapshotting all the shared content can defeat the concept of digital forgetting. Owners should be conscious that the provider storage and machines do not belong to them. Eventually, keys will be used to decrypt the content and expose privacy across the platform. In essence, data objects will be present even after the key is destructed. Once the dedicated expiration time is reached, acquiring the keys will help an adversary reveal the content again.

3 HIGH-LEVEL IDEA

This section discusses the research goals and the general view of our proposed scheme.

3.1 Security Goals

We must consider several security aspects to make a proposed solution as robust as possible. Therefore, in the following, we formalize our security goals.

- ***SG1 – Achieve Retrospective Privacy:*** An attacker must be incapable of accessing the data when the time is due.
- ***SG2 – Ensure the Ephemerality of the Key:*** Decryption keys should be destructed automatically by excluding the repository that stores them the whole time.
- ***SG3 – Data Confidentiality:*** Secure the data objects that will exist over the provider machine when the key decays after a certain period of time.
- ***SG4 – Anti-archiving:*** Hinder the service provider to cache or archive the ephemeral keys on a large scale during the content's lifetime.

SG1 pertains to retrospective privacy by successfully preventing access to a data object after its expiration. *SG2* and *SG3* guarantee that the key will be self-destructed over time. *SG4* prevents attackers or cloud providers from taking a snapshot of a sheer number of data objects with their keys. Caching all the sources in a way to generate a key for each uploaded object is an impossible way to reach. A large-scale attack is out of the question for the adversary.

3.2 Abstract Architecture

Our scheme aims to enable digital forgetting by encrypting content and then generating and reconstructing ephemeral decryption keys without needing temporary storage for the keys. In more detail, the sender utilizes the key generated by our scheme to encrypt specified content before uploading it. The scheme uses online values to create the key. The creation seed of the key counts on Lagrange basis polynomials to be utilized in both encryption and decryption phases [7]. After that, the creation seed will be utilized as an input for the AES algorithm to generate the private key [6]. Eventually, the recipient with valid access only will reconstruct the key similarly to obtain the original file. Figure 1 illustrates the high-level architecture of our proposed scheme.

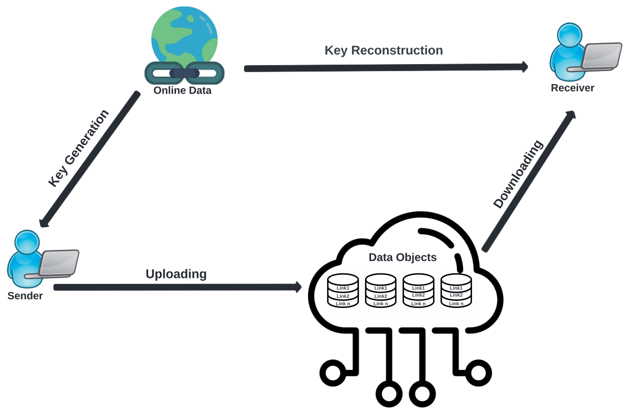

Figure 1: Proposed system model

4 SCHEME DESCRIPTION

This section delves into the details of the proposed system model. Our scheme-based key decay consists of four main steps: (i) *K*ey Generation Phase, (ii) *E*ncryption Phase, (iii) *R*econstruction Phase, and (iv) *D*ecay Phase. Table 1 provides a convenient summary of the notation used in this section.

4.1 Key Generation Phase

At this stage, data owners need a key to encrypt their content before uploading it. For each new key, our scheme performs the following:

Step 1. Our system depends on randomization to derive the key. The creation seed of the key will utilize various online sources (i.e., YouTube, Facebook, Twitter) to get certain types of values (i.e., views, likes, tweets). To this end, the scheme will produce a wide range of online values to be fitted within the Lagrange polynomial.

Step 2. After achieving the list of online values, they will be used to produce a random polynomial. The points are $P_i(x_i, y_i)$, where x_i represents the online values (i.e., views, likes, tweets), and y_i represents randomly generated values to fit the polynomial. In our case, Lagrange polynomial interpolation consists of two aspects: the **threshold** K, which means the minimum required number of points to produce the polynomial, and the **total number of points** N, which means the maximum number of points to produce the exact polynomial.

Step 3. From the online values mentioned above, the polynomial will take only K points (i.e., threshold) to be fitted. In principle, the Lagrange basis polynomial from K degree (i.e., minimum points) will be:

$$l_j(x) = \prod_{i\neq j, 0<i<k} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_k)}{(x_j - x_k)} \qquad (1)$$

By using this formula to generate the polynomial interpolation relying on the minimum points K, $F(x)$:

$$F(x) = \sum_{j=0}^{k} y_j l_j(x) \qquad (2)$$

Table 1: Summary of notation

Notation	Description
x, y	Coordinates of the polynomial
K	Threshold of the polynomial
N	Total number of the polynomial points
l	Lagrange-basis polynomial
$C(k, n)$	Key recovery combination
n	Total number of leading zeros

As a convention, the creation seed will be coefficient-free after getting the formula. The obtained polynomial represents this point's seed, $F(0)$.

Step 4. After the threshold (i.e., minimum points) and the creation seed are achieved, we use the same manner as in SSSS to generate the total number of shares (i.e., maximum points) to be represented within the exact polynomial interpolation. The reason behind generating more points is to increase the possibility of retrieving the polynomial of degree K. To do that, by using equation 2 and a new set of online types (i.e., views, likes, tweets), $P_n(x_1, x_2, \ldots.., x_n)$, to be replaced in the formula to get $P_n(y_1, y_2, \ldots.., y_n)$ as points/shares N belonging to the same polynomial. Consequently, the achieved polynomial will contain a total number of points, and any K out of N will be sufficient to produce the same polynomial.[1]

Step 5. Instead of using the actual values from the online sources, hashing function (i.e., SHA512) will be used to output a fixed-size hash. *Proof-of-Work* (PoW) is an additional mechanism to increase the computational complexity against brute force attacks [3]. For both x_i and y_i coordinates, the hash of a leading zero will be calculated from the binary value. The following equation refers to the number of leading zeros needed to get the target hash.

$$hash[:n] = SHA512(x_i, y_i) \qquad (3)$$

Step 6. On top of PoW, we also apply other obfuscation techniques (i.e., alternating sums, mod are explained in Section 5) to confuse the attacker's predictability and prevent retrospective brute-forcing.

Step 7. Eventually, the key will be derived after getting the seed from the Lagrange.

4.2 Encryption Phase

The sender will outsource the online content to the service provider at this juncture.

Step 1. Once the key is derived (i.e., from the creation seed), the data owner uses it to compile an *Encrypted Data Object* (EDO), which will later upload to the cloud. We use AES to encrypt the data; AES is a symmetric block cipher scheme. It uses keys of 128, 192, and 256 bits to convert these individual original blocks. After encrypting these blocks, it combines them to generate the ciphertext. In our scheme, we use a 256-bit key of the AES algorithm to encrypt the online content as it is more secure than smaller lengths.

[1]Both K and N values should be pre-selected by the user

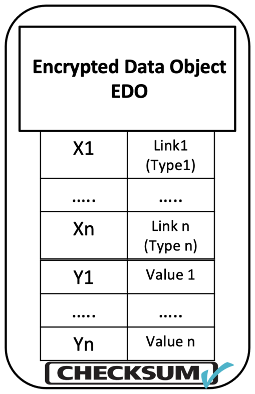

Figure 2: Data object structure

Step 2. Another prerequisite is the checksum, which will be calculated to detect any errors and verify the integrity of the original file later. We use the SHA512 hash to generate the checksum from the uploaded file.

Step 3. After encrypting the data, we get the EDO stored in the cloud provider. The EDO structure includes the encrypted data, the used points to generate the polynomial (x_i, y_i), and the checksum value from the original file. Figure 2 depicts how an EDO looks like.

4.3 Reconstruction Phase

During this phase, the recipient with valid access will reconstruct the key to decrypt the online content.

Step 1. The main task is to assemble the points for the Lagrange basis polynomial to reconstruct the seed. The data object (i.e., EDO) consists of two sections: the online values (i.e., link, type) to represent x_i and a list of values to represent y_i. For the actual polynomial construction, the minimum points (K) out of the total generated points (N) are sufficient for the correct reconstruction. The following formula considers the possible threshold combinations for the seed:

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{4}$$

Step 2. After reconstructing the seed from the polynomial formula $F(0)$, the key will be reconstructed successfully. Then, the receiver can decrypt the content into its original form again.

Step 3. Over time, some values of the online data (i.e., views, likes, tweets) may differ from the old ones, as the central idea of the proposed scheme is to decay the key. The checksum validation will ensure the file's integrity from the selected threshold out of the total number. If the checksum is not matched, the scheme will provide a new threshold combination $C(n,k)$ set to reconstruct the key until the match is fulfilled.

4.4 Decay Phase

The decay phase is an essential concept of our proposed scheme. The validity period to recover the key falls between the threshold and the total number of points (shares). In contrast, the key recovery will no more applicable if the threshold is broken (i.e., $K-1$), leading to a key decay. The following inequalities show both validity and decay intervals.

$$K \leqslant Validity \leqslant N \tag{5}$$

$$0 \leqslant Decay \leqslant K-1 \tag{6}$$

Over time, the online values will change, affecting the x_i coordinates used to construct the polynomial curve. This change will create new (i.e., corrupted) points that are different from the old ones. Our scheme will capture these changes and present a different K combination from N to achieve the seed and eventually the key. In case of corruption has affected more K combination set than the limit (see inequality 6), this will cause the key to decay.

In a nutshell, the proposed methodology is inspired by SSSS. Shamir's one starts by having the key and creating the total number of shares and threshold to divide it. However, our approach varies as online values will construct a Lagrange polynomial of degree K to obtain the seed. Then generating more points as the total number of points. Correspondingly, PoW will be incorporated to increase the complexity of large-scale attacks. Moreover, the randomization notion is fulfilled by relying on different online values. Key traces will vanish over time as the values will keep corrupting till the threshold K is broken, leading to the key decay. Even though the data object will be present, the key will be no more in hand, which accomplishes the main concept of promoting digital data forgetting.

5 EVALUATION

We have implemented a simulation prototype in Python to prove our proposed scheme's feasibility. Our prototype can dynamically provide values inbound of billions to simulate the online sources. Similarly, additional modules (Math, Random, Statistics, Itertools) simulate the Lagrange polynomial and the threshold combinations from the total number of shares. Cryptodome and Hashlib are used to encrypt and decrypt the content with the AES algorithm and PoW integration. The machine used for this evaluation is a MacBook (Processor: 2.3 GHz Dual-Core Intel Core i5, Memory: 8 GB).

Our experiments focused on four main aspects: randomization, decay rate, computational complexity, and security analysis. Finally, we demonstrate the limitations of the proposed scheme.

5.1 Randomization

As we mentioned (Section 4), the randomization notion is one of the crucial parts of our proposed scheme. In the real world, online sources provide various APIs (powered by pagination, different query order per request) to retrieve types (i.e., views, likes, tweets). To simulate that in our proposed system, we relied on different random generators inbound between million and billion to get the same values expected from the actual scenario. We even defined a scraper for limited sources (i.e., YouTube) to acquire the exact response with all the types included (i.e., views, duration, tags).

5.2 Decay Rate

The rate represents the total increment of the x_i value to decay over time. As mentioned, the creation seed will compose of online values, which will change constantly. Furthermore, three folds of

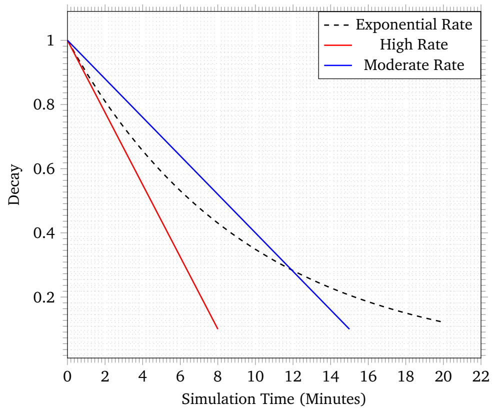

Figure 3: Decay rates

the decay were studied, (i) Evolution (i.e., the specific increment where the decay rate is due to forgetting the ephemeral keys), (ii) Stability (i.e., elapsed time taken for the key to transform from valid into decayed), and (iii) Irreversibility (i.e., the impossibility of reversing the operation to expose the origin of the key).

Evolution. We measured the decay rate with different types of evolution. By doing so, the incremental values will corrupt the key slowly till reaching complete decay. The rates studied vary between static rates (i.e., moderate and high (+100, +10000) respectively per minute) and exponential rate (i.e., e^{x_i} per minute). Figure 3 displays how different function influences decay in various fashions, starting from exponential to a more direct static approach. The main reason for this experiment is to simulate the positive relationship between corruption level and ratio/fixed increment over time. The following equation explains the decay factor for a key, where λ refers to the corruption rate:

$$Decay = 1 - \lambda \tag{7}$$

In our experiments, we decided to go with static rates as follows: (i) *Low Rate*, which means no change over time; (ii) *Moderate Rate*, which means an average increment over time; (iii) *High Rate*, which means a high increment over time. Table 2 shows the average increment over time (i.e., per minute in our case).

Table 2: Static Decay Rates

Decay Rate	Increment Per Minute
Low	0
Moderate	100
High	10000

However, the static decay rates lead us to reliable destruction of the key based on the parameters studied to initialize the rate, and forget the key over time. Furthermore, we focused on different combinations in order to measure the decay rate with the passage of time. Figure 4 shows the Lagrange polynomial before and after the decay rates (i.e., moderate and high rates) where the threshold K was surpassed, and the Lagrange polynomial has deviated from the original set of points.

Stability. We addressed two sides of stability as follows:

1. Without Using Means (Original): This entails using only one type (i.e., YouTube view) to represent the x_i coordinate of the $P_i(x_i, y_i)$. This one value will increase immediately after a limited period leading us to complete decay in no time. Table 3 shows the different combinations $C(k, n)$, and the elapsed time taken to corrupt the threshold using either one of the decay rates. The main reason for the selected combinations $C(k, n)$ is to cover various samples that will not add stability to the scheme by relying on one value. Also, the threshold was selected according to equation 4 to reach the highest possibility (i.e., high combinations acquired) of recovering the key.

Table 3: Decay Rate Combinations

No. Combinations	Random Decay Rate (min.)
$C_1(3, 5)$	4
$C_2(5, 10)$	6
$C_3(10, 20)$	11
$C_4(15, 30)$	16
$C_5(20, 40)$	22
$C_6(25, 50)$	26

2. Using Means: From a different viewpoint, using means entail taking a sheer number of values to represent the x_i coordinate of the $P_i(x_i, y_i)$. The mean will result in more stability of the acquired seed before being corrupted. Table 4 shows the elapsed time taken to corrupt a single x_i value using a different number of samples.

Table 4: Mean Samples

No. Samples	Moderate Rate (min.)	High Rate (min.)
10	8	3
50	45	16
75	69	25
100	80	35
150	110	40

Example. Let us assume we defined (K = 15 and N = 50), respectively. For a moderate decay rate of corruption, each x_i coordinate takes the mean output of 100 online values to represent its coordinate. It took 80 minutes to corrupt one x_i value. According to our experiments, the threshold is 15. It is applicable to mutate up to 35 points as a maximum to recover the same seed again. The total time to corrupt the threshold and retrieve a different value is roughly 2800 minutes. For a high decay rate of corruption, it took only 35 minutes to corrupt one x_i value. As a result, a total of 1200 minutes is enough to surpass the threshold leading to a key decay. For more experiments, Figure 5 compares the stability time with/without using means in terms of different threshold samples (i.e., [5, 30]) over time, locating the same total number N.

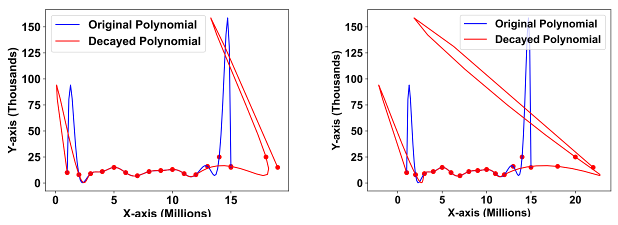

Figure 4: Moderate & High decay rates using means for 100 samples of $(15, 50)$ Lagrange-basis polynomial

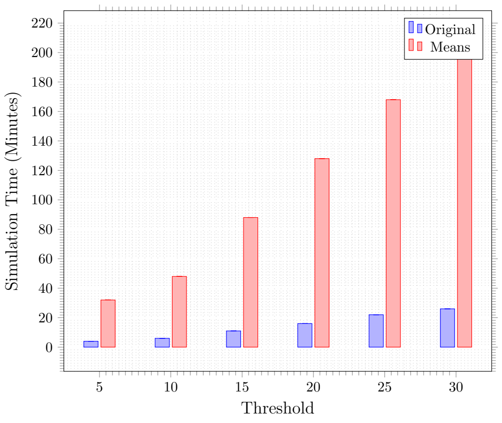

Figure 5: Stability comparison with/without means usage

Irreversibility. We highlighted two sides of irreversibility as follows:

1. Using Alternating Sums: The use of means that always determine the cumulative average (between old and new values) will be expected at some point. Thus, using an alternating sum will confuse the attackers by switching between positive and negative signs of the mean outcome to get a less predictable polynomial; the following formula indicates that; where k represents the threshold and i represents the increment number of alternating the sign:

$$Alter(sum) = \sum_{i=0}^{k} (-1)^i \widehat{\mu_i} \tag{8}$$

Figure 6 shows the alternating sums achieved from a polynomial after calculating the output with a negative sign.

2. Using Mod: From a different angle, the attacker could determine the order of the polynomial by linking the points to get a predictable path. This will reduce the unknown possibilities of the shares since all points lie on a smooth curve. By using the following formula to fill the exposed values (i.e., $K - 1$) from the threshold, respectively; where K represents the threshold, a represents the coefficients, and the seed represents $F(0)$:

$$F(x) = seed + a_1 x + a_2 x^2 + ... + a_{K-1} x^{K-1} \tag{9}$$

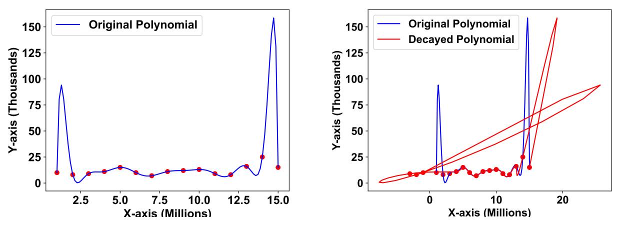

Figure 6: Decay rate using alternating sums for 100 samples of $(15, 50)$ Lagrange-basis polynomial

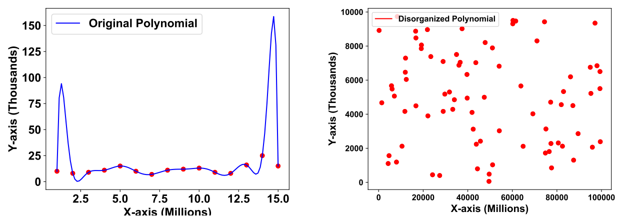

Figure 7: Difference between $(x, F(X))$, and $(x, F(x)\ mod\ p)$

After calculating the formulas (i.e., replacing and subtracting the exposed values), the attacker will obtain a fixed number range that will be used to guess instead of the infinite quantity of natural numbers. The attacker will narrow the prediction to expose the seed $F(0)$. To tackle this issue, finite field arithmetic will be used $p \in \mathbb{P} : p > N, seed$. The polynomial will be calculated as $(x, F(x)\ mod\ p)$ instead of $(x, F(x))$. Figure 7 represents the disorganized and disjointed polynomial using mod compared to the smooth one.

In light of the results, the low rate keeps the content for a very long period as there is no change. The threshold and the total number of points can be deduced from the previous experiments to estimate the expiration times for the online data forgetting for both moderate and high rates. The relationship between decay and recovery rate is inverse: the less threshold with more points, the more applicability for key recovery contrary to the decay rate. Eventually, implementation with/without means alone is inadequate for the real-world scenario to fulfill the irreversibility. Thus, we propose the obfuscation strategies (i.e., alternating sums and mod) to construct disorganized and unpredictable Lagrange polynomials.

5.3 Computational Complexity

The complexity of the proposed scheme means the amount of time and resources needed to achieve the target. PoW is an additional requirement to increase the computational complexity of the hash and make it time-consuming to obtain. Leading zeros is requested to find the optimal nonce (i.e., increment value) that matches the target hash with specific zeros [3]. Table 5 shows the different leading zeros implemented in our proposed framework to acquire only a single x_i value.

Table 5: Complexity Measurement

Number of Iterations	Time Elapsed (sec.)	Leading Zeros
27	0.000103	$Hash[:1] = 0$
601	0.00119	$Hash[:2] = 00$
21674	0.0206	$Hash[:3] = 000$
22748	1.10501	$Hash[:4] = 0000$
148454	600	$Hash[:5] = 00000$
890724	2881	$Hash[:6] = 000000$

The study shows the total iterations and time elapsed to get only one hash for a specific x_i value. The number of leading zeros determines the difficulty. Acquiring proper nonce to compute the hash is time-consuming; instead, the confirming part is relatively easy.

5.4 Security Analysis

We analyze the proposed approach regarding different security goals and threats.

Retrospective privacy. All physical data control is handed to the service provider when information is published on the Internet; data owners do not own the server storage. Also, the service provider or anyone can access and store all the keys within the ephemeral storage. However, our scheme provides a key decay notion without counting on ephemeral storage. Under those circumstances, the attacker will be incapable of performing a retrospective attack. The data will be secure, and retrospective privacy will be completely attained.

Brute force attack. An attack on all possible x_i coordinates (online values) involves guessing the old value according to the data creation time and trying to brute force the remaining values retrospectively. PoW was integrated within the scheme to prevent such attacks on a large scale and provide more durability for seed generation. In our prototype, we investigated the elapsed time taken to exploit a single value x_i (see Table 5).

Predictability. An attack to predict the incremental values that lie on the smooth polynomial and expose the seed. To address this problem, the results section introduced alternating sums with mod usage and assessed them thoroughly. The outcome distinguishes the difference by using these concepts to make the polynomial less predictable and more disorganized (i.e., disjointed) to be used in the real-world scenario (see Section 5.2).

Curious-but-non-interfering attacker. This attacker aims to snapshot the public data indiscriminately before its expiry for malicious use. The snapshot process will happen periodically (i.e., daily, weekly) to extract sensitive information. Our scheme relies on two central notions: randomization and the exclusion of ephemeral storage. Thus, this will hinder the curious attackers from caching and archiving the data periodically to keep them for future use.

5.5 Limitations

As with any other work, ours has its limitations, which are demonstrated below.

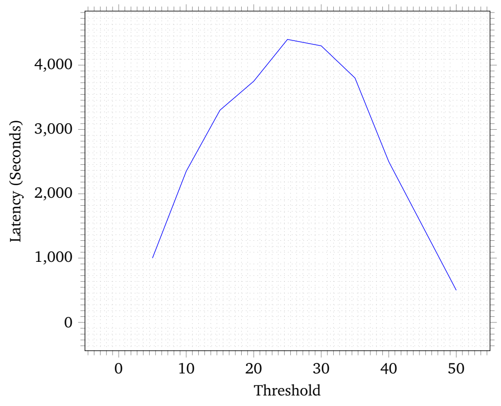

Figure 8: Latency for different thresholds

Latency. As we mentioned in the scheme description, the checksum will be validated during the reconstruction phase to get the correct polynomial seed $F(0)$. However, according to the combination calculation $\binom{N}{K}$, choosing a rate for the threshold close to half of the total number will give many possibilities for key recovery (see equation 4). This will add latency over the system to match the correct checksum by performing many attempts. Figure 8 shows the total latency added to the scheme by performing different samples of thresholds between $[5, 50]$.

Attacker with high computational power. The most recent models of GPUs, which have capabilities for integer computations, and specialized hardware like FPGAs, can be used by well-funded attackers to increase performance. According to certain studies, commercially available hardware can speed up public-key cryptosystems like the RSA scheme by about four times, which can then be applied to these challenges [10]. This will affect the power of the PoW technique by reducing the computational complexity introduced by our scheme.

6 RELATED WORK

This section briefly highlights the various current state-of-the-art studies that look into digital forgetting and provide an online data revocation between the data owners and the service providers. Thus, researchers have presented diverse solutions to compensate for the existing vulnerabilities and provide secure approaches to tackle this issue.

In this sense, the most common solution to cope with this vulnerability is to define the data's expiration time. Several techniques are proposed to help deal with this security issue, including flexible and predefined expiration periods (e.g., Ephemerizer [14], Vanish [8], Neuralyzer [25], EphPub [4], and Timed Revocation [16]. Another study [2] carried out the scheme using time-lock cryptographic puzzles to support digital forgetting. The essence of the proposed study is to limit the deletion or collection of the data during the lifetime of the content by adding complex overhead to the large-scale attack. The time-lock puzzle will provide proof of work to anyone that wants to access the data. Another approach [17] integrates an automated protocol for handling data revocation by examining contractual agreements between cloud providers and data owners. A protocol for this specification of revocation conditions was implemented using smart contracts on a local Ethereum blockchain. Yang et al. [24] have used a novel authentication data structure, namely, the number-rank-based Merkle hash tree (NR-MHT). The prime core of this research is to provide a provable data deletion scheme based on efficient data integrity auditing and dynamic data insertion. Xiong et al. [23] proposed a novel key derivation encryption algorithm that will be used to apply a secure deletion in the Internet of things devices. The proposal is based on flash memory's hierarchical structure by partially merging the block's erasure and deleting the key. The deletion will be from both the cipher form and components that belong to the key when the data validity is ceased. On the other hand, Ginart et al. [9] proposed deletion efficiency for large-scale learning systems and possibly deletion-efficient unsupervised clustering algorithms. They identified potential algorithmic principles that may facilitate deletion efficiency for different learning systems and paradigms. A lethe mechanism was introduced by Minaei et al. [13] to provide post-deletion privacy by withdrawing intermittently Twitter posts. The attacker will be confused between the temporary and permanent contents after a long time of the content deletion. The core of this mechanism is to provide a data owner's deniability against their online posts. This technique will prevent adversaries from archiving posts from the Twitter platform. Moreover, the Forgits data structure strengthens the online deletion by gradually dropping the lowest bits or pixels (least significant bits) from old to the most significant bits from new data [1]. This structure provides infinite retrievals by forgetting the older stored data. However, these approaches provide a digital forgetting-based third authority involvement to forget the desired data; they cannot eliminate the central need for third-party storage or actions. To the best of our knowledge, a key decay notion is a novel approach that has yet to be studied. In this work, we showed that it might be used to prevent adversaries from performing retrospective attacks on tailor-made data to be forgotten without the need for ephemeral key storage.

7 CONCLUSION

In this study, we proposed a novel scheme of *key decay* in the realm of digital forgetting. Because of the ubiquitousness and popularity of digital data, solutions that enable transientness and ephemerality are in high demand. Retrospective privacy is extremely critical since adversaries are keen to exploit the data after its expiration. Our approach introduced a decay vision that implies a change from soundness to complete corruption without counting on the key's storage throughout the period. Furthermore, using different resources to construct the key besides integrating the obfuscation strategies will impede the attackers from storing or taking a snapshot of the keys during the content's validity. Finally, we implement and thoroughly assess a prototype with promising results regarding decay rate, complexity, and irreversibility.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 883275 (HEIR).

REFERENCES

[1] Azza Abouzied and Jay Chen. 2015. Harnessing data loss with forgetful data structures. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*. ACM, USA, 168–173.

[2] Ghous Amjad, Muhammad Shujaat Mirza, and Christina Pöpper. 2018. Forgetting with puzzles: using cryptographic puzzles to support digital forgetting. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. ACM, USA, 342–353.

[3] The Chain Bulletin. 2019. The Chain Bulletin. https://chainbulletin.com/proof-of-work-explained-in-simple-terms.

[4] Claude Castelluccia, Emiliano De Cristofaro, Aurélien Francillon, and Mohamed-Ali Kaafar. 2011. Ephpub: Toward robust ephemeral publishing. In *2011 19th IEEE International Conference on Network Protocols*. IEEE, Canada, 165–175.

[5] Cisco. 2020. Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[6] Joan Daemen. 2020. *The Design of Rijndael:The Advanced Encryption Standard (AES) (Information Security and Cryptography)*. Springer; 2nd ed. 2020 edition, USA.

[7] Biswajit Das and Dhritikesh Chakrabarty. 2016. Lagrange's interpolation formula: representation of numerical data by a polynomial curve. *International Journal of Mathematics Trend and Technology* 34, 2 (2016), 23–31.

[8] Roxana Geambasu, Tadayoshi Kohno, Amit A Levy, and Henry M Levy. 2009. Vanish: Increasing Data Privacy with Self-Destructing Data.. In *USENIX security symposium*, Vol. 316. USENIX, USA, 10–5555.

[9] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making ai forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems* 32, 1 (2019).

[10] Owen Harrison and John Waldron. 2009. Efficient acceleration of asymmetric cryptography on graphics hardware. In *International conference on cryptology in africa*. Springer, Berlin, 350–367.

[11] IBM. 2020. Netezza and IBM Cloud Pak for Data: A knockout combo for tough data. https://www.ibm.com/blogs/journey-to-ai/2020/06/netezza-and-ibm-cloud-pak-a-knockout-combo-for-tough-data/.

[12] Viktor Mayer-Schönberger. 2011. *Delete: The virtue of forgetting in the digital age.* Princeton University Press, USA.

[13] Mohsen Minaei, Mainack Mondal, Patrick Loiseau, Krishna Gummadi, and Aniket Kate. 2019. Lethe: Conceal content deletion from persistent observers. *Proceedings on Privacy Enhancing Technologies* 2019, 1 (2019), 206–226.

[14] Radia Perlman. 2005. The ephemerizer: Making data disappear.

[15] Eugenia Politou, Efthimios Alepis, and Constantinos Patsakis. 2018. Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions. *Journal of cybersecurity* 4, 1 (2018), tyy001.

[16] Sirke Reimann and Markus Dürmuth. 2012. Timed revocation of user data: long expiration times from existing infrastructure. In *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. ACM, USA, 65–74.

[17] Theodor Schnitzler, Markus Dürmuth, and Christina Pöpper. 2019. Towards contractual agreements for revocation of online data. In *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, Lisbon, 374–387.

[18] Theodor Schnitzler, Shujaat Mirza, Markus Dürmuth, and Christina Pöpper. 2021. Sok: Managing longitudinal privacy of publicly shared personal online data. *Proceedings on Privacy Enhancing Technologies* 2021, 1 (2021), 229–249.

[19] Esther Shein. 2013. Ephemeral data. *Commun. ACM* 56, 9 (2013), 20–22.

[20] Verisign. 2020. Domain Names Industry Brief. https://www.verisign.com/en_US/domain-names/dnib/index.xhtml.

[21] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. 2018. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review* 34, 2 (2018), 304–313.

[22] Gregor Wegberg, Hubert Ritzdorf, and Srdjan Capkun. 2017. *Multi-User Secure Deletion on Agnostic Cloud Storage.* Technical Report. ETH Zurich.

[23] Jinbo Xiong, Lei Chen, Md Zakirul Alam Bhuiyan, Chunjie Cao, Minshen Wang, Entao Luo, and Ximeng Liu. 2020. A secure data deletion scheme for IoT devices through key derivation encryption and data analysis. *Future Generation Computer Systems* 111 (2020), 741–753.

[24] Changsong Yang, Yueling Liu, Feng Zhao, and Shubin Zhang. 2022. Provable data deletion from efficient data integrity auditing and insertion in cloud storage. *Computer Standards & Interfaces* 82 (2022), 103629.

[25] Apostolis Zarras, Katharina Kohls, Markus Dürmuth, and Christina Pöpper. 2016. Neuralyzer: Flexible expiration times for the revocation of online data. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy.* ACM, USA, 14–25.