There Is Always a Way Out! Destruction-Resistant Key Management: Formal Definition and Practical Instantiation

Yuan Zhang
University of Electronic Science and Technology of China
Chengdu, China
zhangyuan@uestc.edu.cn

Yaqing Song
University of Electronic Science and Technology of China
Chengdu, China
YaqingS@163.com

Shiyu Li
University of Electronic Science and Technology of China
Chengdu, China
Shai_Li@yeah.net

Weijia Li
University of Electronic Science and Technology of China
Chengdu, China
tokio_0@163.com

Zeqi Lai
Tsinghua University
Beijing, China
zeqilai@tsinghua.edu.cn

Qiang Tang
The University of Sydney
Sydney, Australia
qiang.tang@sydney.edu.au

ABSTRACT

A central advantage of deploying cryptosystems is that the security of large high-sensitive data sets can be reduced to the security of a very small key, i.e., a master key. The most popular way to manage the master key is to use a $(t, n)-$threshold secret sharing scheme: a user splits her/his key into n shares, distributes them among n key servers, and can recover the key with the aid of any t of them. However, it is vulnerable to device destruction: if all key servers and user's devices break down, the key will be permanently lost. We propose a Destruction-Resistant Key Management scheme, dubbed DRKM, which ensures the key availability even if destruction occurs. In DRKM, a user utilizes her/his n^* personal identification factors (PIFs) to derive a cryptographic key but can retrieve the key using any t^* of the n^* PIFs. As most PIFs can be retrieved by the user *per se* without requiring *stateful* devices, destruction resistance is achieved. With the integration of a $(t, n)-$threshold secret sharing scheme, DRKM also provides *portable* key access for the user (with the aid of any t of n key servers) before destruction occurs. DRKM can be utilized to construct a destruction-resistant cryptosystem (DRC) in tandem with any backup system. We formally prove the security of DRKM, implement a DRKM prototype, and conduct a comprehensive performance evaluation to demonstrate its high efficiency. We further utilize Cramer's Rule to reduce the required buffer to retrieve a key from 25 MB to 40 KB (for 256-bit security).

KEYWORDS

Destruction resistance, key management

1 INTRODUCTION

Secure and efficient key management schemes are cornerstones of any cryptosystem, which should satisfy the desired requirements of availability (users can always correctly recover their keys) and portability (users can access their keys from multiple devices). To this end, a user always stores her/his cryptographic keys in a repository [1–5]. Generally, the repository can be either instantiated by deploying a local device [3, 4] or subscribing to key access services from a dedicated service provider [5]. Such a repository always refers to a key server in literature [6, 7]. This paradigm has been widely utilized in commercial systems, e.g., Microsoft Azure [1] and Google Cloud Platform [2].

Despite the advantage of deploying a key server, a critical issue—vulnerability of the system against device destruction—arises naturally: as uncontrollable and unpredictable threats towards the key server always exist in reality, the user has to bear the risk that her/his keys would be permanently lost if the key server is destroyed. We stress that device destruction is not just a theoretical concern, and recent incidents have shown that it would happen with various manifestations which typically consist of hardware destruction and software unavailability [8–10]. Notably, a private key stored in a hard drive was permanently unavailable to its owner due to a hard drive breakdown, which directly caused that 7500 bitcoins (which are worth more than \$280 million today) could never be used by the owner [8]. In addition, after the key server storing users' cryptographic keys is hacked by ransomware attacks, all cryptographic services have been paralysed [9, 10]. As such, remaining availability in case of device destruction has become a primary requirement for key management schemes.

The most popular method to manage cryptographic keys is the threshold secret sharing scheme [11] (as well as its variants [12]), where n key servers are deployed, and each of them maintains a share of the key such that the key can be recovered with any t shares. Such a key management scheme provides a strong guarantee in terms of security and reliability: even if an adversary compromises $t - 1$ key servers, he cannot get any information about the key; the destruction of any $n - t$ key servers cannot hamper the key recovery. Due to the theoretically desirable properties and practical natures, threshold secret sharing schemes still serve as a key component for lots of high-sensitive systems (e.g., vault systems [13]) in the current age, even though the pioneering work was proposed by Shamir [11] more than 40 years ago. However, the fundamental issue of remaining at least t key servers available under any circumstance still exists. In reality, misfortunes causing simultaneous destruction of all key servers could still happen in any system, no

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–18

https://doi.org/XXXXXXX.XXXXXXX

matter what high degree of reliable measures would be taken. For instance, Amazon Web Services (AWS) suffered a major outage [14] due to misoperations, where all servers in the Amazon Simple Storage Service (S3) subsystems broke down, and many popular websites, e.g., Netflix and Slack, were affected [15]. Severe natural disasters also would directly destroy local servers and make them permanently unavailable to their users. A notable example is that the eruption of the Tonga volcano [16] in 2022 destroyed critical information infrastructure almost all over the country.

A natural way to mitigate this problem is to employ additional servers providing backup services: if some key servers break down, backup servers can continue to handle users' requests[1]. Nevertheless, this remedy cannot be applied for key management, since multiple backups of keys increase the danger of security breaches. This motivates us to consider the following question:

Motivation question 1

Can we have a key management scheme that ensures key availability even if all repositories (including users' devices and key servers) are destroyed?

The key observation behind our work is that destruction-resistant key availability can only be achieved by a key generation mechanism that enables the user to recover the key as needed without requiring any *stateful*[2] device. With the observation, we introduce two new concepts: reconstructable secret and un-reconstructable secret, depending on whether a stateful device is necessary for recovering the secret. We then propose a practical key derivation mechanism to generate reconstructable secrets, where the key idea is to generate the reconstructable secret using users' personal identification factors (PIFs)[3]. Specifically, we categorize PIFs into three types: device-dependent ones, device-independent ones, and storage-independent ones, where processing device-dependent/device-independent PIFs (for cryptography purposes), such as biometric characteristics [19–22], requires a stateful/stateless device, and processing storage-independent PIFs, e.g., passwords, does not even require some additional storage. A systematic analysis is provided in Section 2. We also notice that reconstructable secrets can be directly derived from device/storage-independent PIFs. With reconstructable secrets, it seems that a destruction-resistant key management scheme can be trivially constructed: a user constructs a master key from multiple reconstructable secrets and further utilizes it to derive other cryptographic keys. However, the above scheme is also confronted with the following issues.

Regarding functionality, the key recovery depends on a strong assumption that the user needs to keep all device-independent PIFs available under any circumstance. As a counterexample, if the user utilized a fingerprint to generate the master key, when large-scale disasters occur, the user's finger may be injured, and consequently the user cannot recover the master key until the finger heals.

Regarding convenience, the portability is also lost, since "device-independent" is not equivalent to "portable" (even if a PIF is device-independent, the user may not retrieve it anytime and anywhere). For instance, if a user has a camera capable of collecting irises, she/he can derive a secret from the iris. Subsequently, the user can only recover the secret when she/he equips such a specific-purpose device (that may not be the same as the previous one but has the same functionalities). Such a secret fails to achieve portability, and migrating it may cause new issues in terms of security and efficiency.

The above limitations further motivate us to consider the following question:

Motivation question 2

Can we have a destruction-resistant key management scheme that enables key recovery from a subset of original PIFs while achieving portability before destruction occurs?

We stress that the conventional threshold secret sharing [11] and its distributed variants [12] fail to achieve the key recovery from a subset of original PIFs, since they essentially share the same paradigm: first determine the secret and then split it into multiple shares; any threshold number of shares can reconstruct the secret. However, in destruction-resistant key management introduced before, the "shares" (i.e., the reconstructable secrets) are pre-determined by PIFs, and the master key is derived from them. To achieve the key recovery from a subset of original PIFs, a threshold key derivation mechanism should be designed, such that the master key can be "derived" from all pre-determined "shares" but can be recovered with only the threshold number of them. (By comparison, the "shares" are determined by PIFs rather than the master key as in conventional threshold secret sharing. The detailed comparison is provided in Appendix B.)

To achieve portability, the user can derive n different reconstructable secrets from different PIFs, employ n key servers, and let each key server maintain one secret. By doing so, the user can access the master key with the aid of key servers in a portable way[4] before the destruction occurs. Whereas, such an approach is vulnerable to trawling attacks [23]. Specifically, a PIF is not only used for generating the master key in one system but also used in other systems for other cryptographic purposes, e.g., secure authentication. In the above approach, the key servers can compromise enough information about the user's PIFs from the reconstructable secrets, which enables adversarial key servers to impersonate the user to access other services where the same PIFs are used for authentication.

To the best of our knowledge, we still lack a destruction-resistant key management scheme that enables key recovery from a subset of original PIFs after destruction occurs while achieving portability before destruction occurs.

1.1 Our contributions

In this paper, we propose a Destruction-Resistant Key Management scheme, dubbed DRKM, which goes one step beyond existing schemes [11, 12]. Specifically, our contributions are summarized as follows.

[1] To resist the destruction caused by natural disasters, the backup servers can be deployed around the world. However, this approach may violate data protection regulations in several countries [17, 18] and would be expensive to deploy in practice.

[2] *Stateful* means that the device stores some secret information related to the user.

[3] We utilize the terminology of "PIFs" here to distinguish from authentication factors (AFs). In digital systems, a PIF is the factor that uniquely identifies a user while an AF is considered as a special PIF that can be utilized to construct secure and usable authentication schemes. In other words, some PIFs cannot serve as AFs, e.g., DNA is a PIF but cannot be used to construct usable authentication schemes (due to its inconvenience and high costs).

[4] The user can authenticate herself/himself with portable PIFs.

- **Concepts of storage/device-independent PIFs**. We first propose three new concepts about personal identification factors (PIFs)—storage-independent PIFs, device-independent PIFs, and device-dependent PIFs—based on whether a PIF can be retrieved by the user *per se* without requiring any storage or a *stateful* device.
- **Concepts of (un-)reconstructable secrets.** We introduce two concepts about secrets—reconstructable secrets and un-reconstructable secrets. We point out that reconstructable secrets can be directly derived from storage/device-independent PIFs. We also present a series of methods to derive reconstructable secrets from device-dependent PIFs in tandem with storage/device-independent PIFs under certain conditions.
- **Construction for destruction-resistant key management.** We propose DRKM, a destruction-resistant and portable key management scheme. To achieve destruction resistance, DRKM utilizes a threshold key derivation mechanism to enable a user to derive a master key from n^* PIFs (which include storage/device-independent ones and might include device-dependent ones) during the setup phase and to recover the master key using any t^* of the n^* PIFs after destruction occurs. To achieve portability, DRKM adopts a multi-server-aided paradigm and utilizes a conventional (t, n)-threshold secret sharing scheme (t and n are independent of t^* and n^*) to distribute the master key among n key servers. As long as any t of n key servers are available, the user can access the master key in a portable way (i.e., she/he does not maintain any secret in local devices.). DRKM is compatible with existing backup systems and can be directly extended to a destruction-resistant cryptosystem (DRC) in tandem with any commercial cloud storage service, such as Google Drive [24], Dropbox [25].
- **Formal security proofs and prototype implementation**. We provide formal security definitions of DRKM and prove its security. Particularly, we prove that an adversary, who compromises $t-1$ key servers and t^*-1 reconstructable secrets, cannot get any information about the master key. We implement a DRKM prototype and conduct a comprehensive performance evaluation which shows that it would take about 120 ms to derive a master key from 10 popular PIFs and take less than 5 ms to recover the master key from any t^* secrets with $t^* = 12$ and $n^* = 20$. In addition, we utilize Cramer's Rule [26] to significantly reduce the required buffer to retrieve a key from 25 MB to 40 KB (for 256-bit security).

We demonstrate the viability of DRKM for two existing applications that can benefit from the desirable property of destruction resistance in Section 5.6. One of two applications extends to DRCs. We show how DRKM supports these applications without changing the current system architecture. Since destruction resistance is a fundamental requirement of any cryptosystem, we believe that DRKM has further useful applications.

1.2 Technical overview

The core of achieving destruction resistance is to be free from the reliance on stateful devices. In Section 2, we divide PIFs into three categories: storage-independent PIFs, device-independent PIFs, and device-dependent PIFs. A user can directly derive reconstructable secrets from storage/device-independent PIFs, and these secrets are independent of any stateful devices. We also present a series of methods to derive reconstructable secrets from device-dependent PIFs in conjunction with storage/device-independent PIFs.

With the above methods, a user first derives n^* reconstructable secrets from PIFs and then aggregates these secrets to obtain a master key. The challenge in designing DRKM is to achieve threshold retrieval for the aggregated master key, i.e., a master key is aggregated from n^* pre-determined secrets and can be retrieved from any t^* of them. To address the challenge, we utilize a threshold key derivation mechanism. Specifically, the user first constructs a n^*-degree polynomial $p(x)$ using the n^* secrets as its roots. In this polynomial, the constant term serves as the master secret, and the coefficients of $p(x)$ of degree n^*-1 down to t^* are published as the auxiliary information *aux*. With t^* secrets and *aux*, the degree of the polynomial $p(x)$ can be reduced from n^* to t^*-1. The user can compute the coefficients of $p(x)$ from degree t^*-1 down to 0 so as to obtain the master key.

We also integrate an aggregation-then-split mechanism into DRKM to achieve portable key access in normal times against trawling attacks. The master key derived from the n^* reconstructable secrets is further split into n shares using a conventional (t, n)-threshold secret sharing scheme, and each key server maintains a share. Adversarial key servers cannot compromise any information about the reconstructable secrets from the secret shares. This yields the final DRKM: before the destruction occurs, the user can recover the master key with the aid of key servers in a portable way; once the destruction occurs (i.e., the key servers and the user's devices are destroyed), the user can retrieve any t^* reconstructable secrets using all available PIFs at that time and recover the master key from them. In the extreme case where all devices, key servers, and even *aux* is unavailable, the user can also recover the master key from n^* original PIFs. A destruction-resistant cryptosystem (DRC) can be developed by directly integrating DRKM and a full-fledged backup system.

1.3 Comparison with concurrent work

A very recent work concurrent to DRKM (i.e., threshold multi-factor key derivation function, short for TMFKDF) proposed by Nair et al. [27] could be a partial solution to construct a destruction-resistant key management scheme: a user first randomly chooses a master key, splits it into multiple shares using a conventional threshold secret sharing scheme and encrypts each share under a PIF (using some key derivation functions).

Essentially, TMFKDF inherits the threshold property of the conventional threshold secret sharing scheme, where the master key is randomly chosen by the user rather than determined by PIFs. Therefore, some metadata, e.g., the encrypted shares, is inherently needed for key recovery, and dedicated storage for metadata is always required. By comparison, DRKM is completely orthogonal to the conventional sample-share-and-reconstruct idea as in TMFKDF. With DRKM, the user can also recover the master key from n^* reconstructable secrets even if any storage is unavailable.

In DRKM, it seems that some metadata is also required in some cases. For instance, if the user derives reconstructable secrets from biometric characteristics [19–22], some metadata, e.g., error correcting code [28, 29], is required to ensure the consistency of secrets in different derivations. However, we stress that such metadata in

DRKM is totally different from ciphertext in TMFKDF due to the following reasons.

The metadata in DRKM can be shared among different systems in which PIFs are utilized for other purposes, e.g., user authentication. As such, dedicated storage is not required, and the user can retrieve the metadata from other systems on demand. However, for TMFKDF, dedicated storage for ciphertext cannot be shared among other systems. Furthermore, it is promising to free from metadata in DRKM by utilzing new PIFs where deriving secrets from them does not require any storage. A detailed comparison is provided in Section 5.4.

Roadmap. The remainder of this paper is organized as follows. We introduce the concepts of storage/device independent PIFs in Section 2. We propose DRKM in Section 3 and give the formal security proof in Section 4. In Section 5, we detail the implementation and evaluate the performance of DRKM. Finally, we draw the conclusion and outlook for future research directions in Section 6.

2 PIFS AND RECONSTRUCTABLE SECRETS

2.1 Definitions of PIFs

We analyze popular PIFs and give a brief introduction to them in Appendix C, referring to “something the user knows”, “someone the user is”, and “something the user has” [30]. Intuitively, a PIF can uniquely identify a user, and thereby each PIF indicates a sole secret utilized to distinguish different PIFs. We observe that the utilization of some PIFs has to depend on stateful hardware devices that maintain the necessary state information, e.g., hardware tokens. Additionally, a succinct description of PIF is required, which includes the directions for use and necessary auxiliary information. The secret is private, and the description is public. We formally define a general PIF as follows.

Definition 1. *A PIF is a triple of arguments $(sta, \mu, desp)$, where sta is state information, μ is a unique secret, and desp is a description of the PIF, including the directions for use and necessary auxiliary information.*

We take a SIM card [31] as an example: sta represents the SIM card itself, μ is the secret key fused in it, and *desp* includes necessary auxiliary information (e.g., public parameters and authentication protocols used in the SIM card).

With the previous analyses, we can heuristically divide the above PIFs into three categories—storage-independent ones, device-independent ones, and device-dependent ones. In reality, passwords and PIN codes can be reconstructed from users’ memory and are inherently independent of any personal or public storage. Biometric characteristics can be reconstructed by specific-purpose devices (e.g., cameras used to collect irises) instead of *stateful* devices. For device-dependent PIFs (e.g., SIM cards, hardware tokens, and Intel SGX), once a stateful device is destroyed, the corresponding PIF cannot be recovered by the user per se.

Obliviously, storage-independent PIFs are also device-independent but device-independent ones need public information for reconstruction and are dependent on public storage. We capture the storage independence, device independence, and device dependence of PIFs by the following definitions, respectively.

Definition 2. *(Storage-independent PIF). A storage-independent PIF is stateless and can be represented by a triple of arguments $(\perp, \mu, desp)$, where $\perp$ represents that the generation and maintenance of μ do not rely on any public or personal storage.*

Definition 3. *(Device-independent PIF). A device-independent PIF is stateless and can be represented by a triple of arguments $(info, \mu, desp)$, where the generation and maintenance of μ is independent of devices and only rely on some public storage info.*

Definition 4. *(Device-dependent PIF). A device-dependent PIF is stateful and can be represented by a triple of arguments $(sta, \mu, desp)$, where the generation and maintenance of μ depend on a hardware device specified by sta.*

We observe that all biometric characteristics are device-independent, as they are determined by a user per se. However, they may not totally independent of storage, since some public information, e.g., error correcting code [28, 29], is needed when utilizing them. In cryptographic applications, how to make biometric characteristics free from dedicated storage is a fascinating open problem.

2.2 Reconstructable secrets

We first consider what it means to be “reconstructable”. Informally, a value is reconstructable if it is available without a specific stateful device and can be accessed anytime and anywhere. Reconstructable values generalize the notion of storage/device-independent PIFs, which is captured by Definition 5. For completeness, we also define un-reconstructable values in Definition 5.

Definition 5. *A two-valued probability distribution (σ, α) generated by an efficient probabilistic algorithm is reconstructable if it does not take state information sta as inputs, where α represents some (public) auxiliary knowledge about σ or its distribution. (σ, α) is un-reconstructable if the efficient probabilistic algorithm takes state information sta as inputs.*

We can trivially extend reconstructable values (σ, α) to reconstructable secrets by further requiring σ to be kept secretly. Similarly, a secret derived from an un-reconstructable value is un-reconstructable. In reality, we can utilize storage/device-independent PIFs to serve as reconstructable values to derive reconstructable secrets. For instance, fuzzy extractor algorithms [28, 29] can be utilized to extract a secret from a feature template of biometric characteristics. The secret is reconstructable, since fuzzy extraction algorithms ensure that the same secret can be extracted from similar but not identical feature templates. Directly (only) using passwords and PINs as reconstructable secrets cannot achieve a reasonable security guarantee due to their inherent limitations. To mitigate this problem, we can integrate multiple PIFs to derive a high min-entropy reconstructable secret. With the above analysis, we draw Theorem 1.

Theorem 1. *A value $s = F(\{PIF_1, PIF_2, \ldots, PIF_m\})$ is reconstructable if $PIF_1, PIF_2, \ldots, PIF_m$ are storage/device-independent, and F is some function.*

Proof. For $i = 1, \ldots, m, PIF_i = (\perp / \mathsf{info}_i, \mu_i, desp_i), s = \mathsf{F}(\{PIF_1, PIF_2, \ldots, PIF_m\}) = \mathsf{F}(\{\mu_i\}_{1 \le i \le m}, \{\mathsf{info}_i\}_{1 \le i \le m})$, where $(\{\mu_i\}_{1 \le i \le m}, \{\mathsf{info}_i\}_{1 \le i \le m})$ satisfies Definition 5, and $\{\mu_i\}_{1 \le i \le m}$ are secret information. Hence, s is a reconstructable secret. □

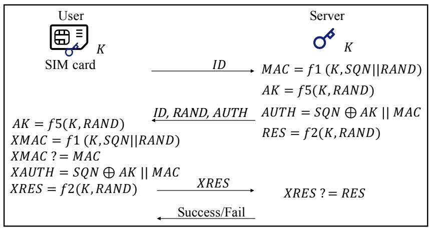


Figure 1: User authentication based on SIM cards.

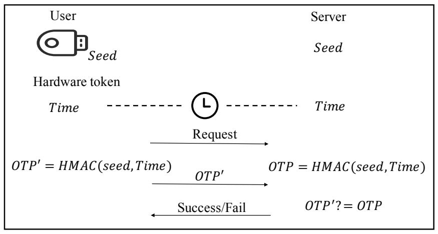


Figure 2: User authentication based on RSA SecurIDs.

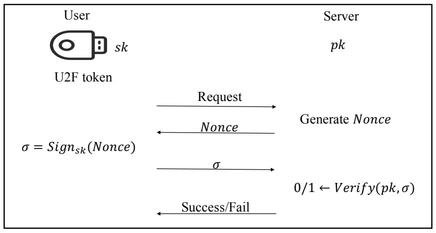


Figure 3: User authentication based on U2F tokens.

If all $PIF_1, PIF_2, \ldots, PIF_m$ are storage-independent, the secret $s = \mathsf{F}(\{PIF_1, PIF_2, \ldots, PIF_m\})$ does not rely on any public or personal storage, since s can be represented as $s = \mathsf{F}(\{\mu_i\}_{1\leq i\leq m}, \perp)$.

2.3 Conditionally reconstructable secrets

It seems impractical to derive a reconstructable secret from a device-dependent PIF, since a secret derived from a device-dependent PIF cannot be recovered by the user if the device is destroyed. However, we observe that a *conditional* reconstructable secret can be generated by a "hybrid model", i.e., we can derive a reconstructable secret from a special class of device-dependent PIFs in tandem with some storage/device-independent PIF(s) under a specific assumption.

For the device-dependent PIFs introduced in Appendix C, i.e., SIM cards, hardware tokens, and SGX, they essentially share the same paradigm, where a secret generated by the manufacturer is fused in the device. The user can only utilize the secret to compute authentication credentials but cannot extract it. We observe that if the underlying authentication scheme is based on the symmetric-key cryptographic primitives (e.g., MAC), the server will store the same secret that is fused in the device after registration. In this case, once the device is destroyed, the server (or the device's manufacturer) still stores the secret. As such, the secret can be recovered with the aid of the server or the manufacturer even if device destruction occurs.

In the following, we discuss how to derive a conditionally reconstructable secret from each of the device-dependent PIFs introduced in Appendix C.

SIM card. As shown in Figure 1, a user can utilize a SIM card to compute a MAC on a storage/device-independent PIF and set the MAC as the conditionally reconstructable secret derived from the SIM card. Since MAC is existentially unforgeable, and the PIF is secretly maintained by the user, the MAC-based secret is only known to the user. In addition, as the secret fused in the SIM card is also maintained by the cellular communication service provider, the user can recover the MAC-based secret with the aid of the provider.

Hardware token. For HMAC-based hardware tokens (as shown in Figure 2), the user can derive a conditionally reconstructable secret in the same way as that from the SIM card. However, ECDSA-based hardware tokens (as shown in Figure 3) cannot be utilized to derive a reconstructable secret, since the manufacturer does not maintain the secret stored in the user's hardware token. However, we notice that for some existing hardware tokens, this can be achieved by utilizing a well-known attack, i.e., the backdoor attack released by Snowden [32], where this approach actually does not need the assistance from the manufacturer [33].

Intel SGX. The Root Provisioning Key (PRK) fused in Intel SGX is shared by a user and Intel. Intel SGX architecture provides the *EGETKEY* instruction to derive a key from the RPK [34–36]. The user initializes an enclave for a storage/device-independent PIF and invokes *EGETKEY* for the enclave. The key output by the *EGETKEY* is the reconstructable secret derived from the Intel SGX. When reconstructing the secret, the user only needs to establish an enclave for the previous storage/device-independent PIF and gets the secret by calling the *EGETKEY* instruction.

Limitations of hybrid model. Regarding security, the user has to fully trust the servers or the manufacturers. Malicious servers and manufacturers may abuse the user's secret. Regarding reliability, the reconstructable secrets derived by the hybrid method are conditionally reconstructable. The servers or manufacturers have another mechanism to authenticate the user when the user's devices are destroyed, e.g., real-name systems. The reconstructability depends on the reliability of the servers or the manufacturers. Once the servers and manufacturers are destroyed, the user cannot recover the reconstructable secrets.

It is worth stressing that the above limitations are not contradictory to our primary motivation of resistance against device destruction due to the following reasons. First, the manufacturers and cellular communication service providers play an important role in constructing critical infrastructures, and thereby they would take lots of measures to ensure the reliability and security of their devices. Second, if the manufacturers and cellular communication service providers misbehave, it would cause a huge loss. As such, both the manufacturers and cellular communication service providers bear rigorous accountability from governments in reality. By comparison, the measures taken by application service providers and users to improve the reliability and security of key servers and devices are always weak, and the accountability is somewhat trivial.

3 THE PROPOSED DRKM

3.1 Notation

We utilize ℓ to denote the security parameter, and $|a|$ denotes the absolute value of a. $\vec{A}$ denotes a set $\{A_1, \ldots, A_n\}$. $[1, n]$ denotes the set $\{1, 2, \ldots, n\}$. $a \xleftarrow{\$} A$ denotes that a is uniformly chosen from A.

3.2 Definition of DRKM

DRKM consists of three entities: a user $\mathcal{U}$, a set of key servers $\vec{\mathcal{KS}} = \{\mathcal{KS}_1, \mathcal{KS}_2, \ldots, \mathcal{KS}_n\}$, and a backup system. There are four algorithms in DRKM, **Setup**, **Managing**, **Access**, and **Recovery**.

- $PP \leftarrow$ **Setup**(ℓ).

```
Indistinguishability:                                   $\mathsf{IND\text{-}Key}_{\mathcal{A}_1}(\ell)$
$\{S, s_1, \ldots, s_n, aux\} \leftarrow \textbf{Managing}(PIF_1, \ldots, PIF_{n^*})$
$S^\star \xleftarrow{\$} \{0,1\}^{|S|}$
$b \xleftarrow{\$} \{0,1\}$
If $b = 1$, $Key = S$; else, $Key = S^\star$
$b' \leftarrow \mathcal{A}_1(Key, s_{i_1}, \ldots, s_{i_{t-1}}, PIF_{i_1}, \ldots, PIF_{i_{t^*-1}}, aux)$
If $b' = b$, return 1
Else, return 0.

PIF privacy:                                            $\mathsf{PIF\text{-}Privacy}_{\mathcal{A}_2}(\ell)$
$\{S, s_1, \ldots, s_n, aux\} \leftarrow \textbf{Managing}(PIF_1, \ldots, PIF_{n^*})$
Select a subset $\overrightarrow{PIF}_1$ of $\{PIF_1, \ldots, PIF_{n^*}\}$, where $|\overrightarrow{PIF}_1| = t^* - 1$
Generate a PIF set $\overrightarrow{PIF}_0$, where $\overrightarrow{PIF}_0 \cap \{PIF_1, \ldots, PIF_{n^*}\} = \emptyset$ and $|\overrightarrow{PIF}_0| = t^* - 1$
$b \xleftarrow{\$} \{0,1\}$
$\overrightarrow{PIF} = \overrightarrow{PIF}_b$
$b' \leftarrow \mathcal{A}_2(\overrightarrow{PIF}, s_{i_1}, \ldots, s_{i_{t-1}}, aux)$
If $b' = b$, return 1; else, return 0.
```

Figure 4: The security experiments of DRKM.

On input the security parameter ℓ, this algorithm returns public parameters PP, where two thresholds, i.e., (t, n) and (t^*, n^*), are included. (t, n) is independent of (t^*, n^*). The larger t and t^* are, the stronger the security guarantee is but the higher $\mathcal{U}$'s costs to access and recover the keys are.

• $\{S, s_1, \ldots, s_n, aux\} \leftarrow \textbf{Managing}(PIF_1, \ldots, PIF_{n^*}, \mathsf{sta})$.

On input n^* PIFs $\{PIF_1, \ldots, PIF_{n^*}\}$ and (optional) public state information sta, this algorithm returns a master key S, n shares $\{s_1, \ldots, s_n\}$ of S, and auxiliary information aux. $\mathcal{U}$ generates a master key S using $\{PIF_1, \ldots, PIF_{n^*}\}$, sends the secret share s_i to $\mathcal{KS}_i$. $\mathcal{U}$ stores aux with a backup system.

• $S \leftarrow \textbf{Access}(s_{i_1}, \ldots, s_{i_t}, aux)$.

On input any t of n shares $\{s_{i_1}, \ldots, s_{i_t}\}$, this algorithm returns S. $\mathcal{U}$ gets $\{s_{i_1}, \ldots, s_{i_t}\}$ from t key servers and aux from the backup system and can access S.

• $S \leftarrow \textbf{Recovery}(PIF_{i_1}, \ldots, PIF_{i_{t^*}}, aux, \mathsf{sta})$.

On input t^* PIFs $\{PIF_{i_1}, \ldots, PIF_{i_{t^*}}\}$, aux, and (optional) sta, this algorithm returns the master key S. If the number of available key servers is less than t, $\mathcal{U}$ utilizes available PIFs $\{PIF_1, \ldots, PIF_{t^*}\}$ to recover S.

3.3 Functionality of DRKM

The primary functionality of DRKM is to ensure that users can recover their keys when both key servers and devices are unavailable. In the following, we refine three cases and discuss the functionality that should be satisfied case by case.

- *Normal times.* In normal times, both dedicated storage for sensitive information and general storage for non-sensitive information are available to users, i.e., any t key servers and the backup system are available. DRKM should achieve portability in normal times, i.e., enable users to access their master keys without maintaining any secret locally.
- *Partial destruction.* Partial destruction means that dedicated storage for sensitive information is destroyed but general storage for non-sensitive information is available to users. It indicates that all key servers and devices are destroyed but the backup system and any t^* of n^* PIFs are available. In this case, DRKM should enable users to recover their master keys.
- *Full destruction.* In the full destruction case, neither dedicated storage for sensitive information nor general storage for non-sensitive information is available to users, i.e., all key servers and devices are destroyed, and even the backup system is unavailable to users. Under this circumstance, DRKM should ensure the key recovery if all n^* PIFs are available.

3.4 Security of DRKM

The security goals of DRKM are as follows.

- Regardless of any information an adversary already has, he cannot extract any information about a user's master key S used for key derivation from the interaction messages between the user and other entities.
- An adversary who compromises key servers cannot obtain any information about PIFs that are utilized to derive the master key.

The security of DRKM is formally captured by Definition 6 and Definition 7, where the security experiments are provided in Figure 4. In Definition 6, we consider the adversary who can (1) corrupt $t^* - 1$ PIFs used to derive the master key; (2) compromise $t - 1$ key servers and the cloud server, but still cannot distinguish the master key from a uniformly-chosen key[5] with probability better than 1/2. In Definition 7, given $t - 1$ shares, an adversary cannot determine which PIFs are used to derive the master key corresponding to the $t - 1$ shares.

Definition 6. *(Indistinguishability). DRKM satisfies indistinguishability against any probabilistic polynomial-time (PPT) adversary $\mathcal{A}_1$ who compromises $t - 1$ key servers and $t^* - 1$ PIFs iff there is a negligible function negl such that*

$$\Pr[\mathsf{IND\text{-}Key}_{\mathcal{A}_1}(\ell) = 1] \le \frac{1}{2} + negl(\ell).$$

Definition 7. *(PIF privacy). DRKM protects the PIFs used to derive the master key against any PPT adversary $\mathcal{A}_2$ who compromises t shares if there exists a negligible function negl such that*

$$\Pr[\mathsf{PIF\text{-}Privacy}_{\mathcal{A}_2}(\ell) = 1] \le \frac{1}{2} + negl(\ell).$$

3.5 Construction of DRKM

We instantiate the backup system using a cloud server $\mathcal{CS}$. A user $\mathcal{U}$, a set of key servers $\vec{\mathcal{KS}} = \{\mathcal{KS}_1, \mathcal{KS}_2, \ldots, \mathcal{KS}_n\}$, and a cloud

[5] The uniformly-chosen key has the same length as the master key.

server $\mathcal{CS}$ are involved in DRKM. We assume that $\mathcal{U}$ has n^* reconstructable secrets which are derived from $\overrightarrow{PIF} = \{PIF_1, PIF_2, \ldots, PIF_{n^*}\}$[6].

Setup. With the security parameter ℓ, public parameters $PP = \{p, t^*, t, n^*, n, \mathsf{F}\}$ are determined, where p is a prime, t^* and t are two thresholds, n^* is the number of PIFs of $\mathcal{U}$, n is the number of the key servers, and F is secure key derivation function.

Managing. $\mathcal{U}$ utilizes her/his PIFs to manage $\vec{K}$ as follows.

- For $PIF_i \in \overrightarrow{PIF}$ and $i = 1, 2, \ldots, n^*$, $\mathcal{U}$ computes s_i^* as follows.
  - If PIF_i is storage/device-independent, compute $s_i^* = \mathsf{F}(PIF_i)$ by Theorem 1[7].
  - If PIF_i is device-dependent, $\mathcal{U}$ can compute a conditionally reconstructable secrets by utilizing the methods provided in Section 2.3.
- $\mathcal{U}$ generates a polynomial

$$p(x) = \prod_{i=1}^{n^*}(x - s_i^*),$$

  such that $p(s_i^*) = 0$ for $i = 1, 2, \ldots, n^*$.
- $\mathcal{U}$ outputs the coefficients of $p(x)$ of degree $n^* - 1$ down to t^* as the auxiliary information aux, i.e.,

$$aux = \{-\sum_{i=1}^{n^*} s_i^*, \sum_{\substack{A \subseteq [1,n^*] \\ |A|=2}} (\prod_{i \in A} s_i^*), -\sum_{\substack{A \subseteq [1,n^*] \\ |A|=3}} (\prod_{i \in A} s_i^*),$$
$$\ldots, (-1)^{n^*-t^*} \cdot \sum_{\substack{A \subseteq [1,n^*] \\ |A|=n^*-t^*}} (\prod_{i \in A} s_i^*)\}.$$

- $\mathcal{U}$ computes $S = \prod_{i=1}^{n^*} s_i^*$ as the master key.
- $\mathcal{U}$ uniformly chooses $a_1, \ldots, a_{t-1} \overset{\$}{\leftarrow} Z_p$ and generates a polynomial $f(x) = S + a_1 x + \cdots + a_{t-1} x^{t-1}$ over Z_p with degree at most $t - 1$.
- $\mathcal{U}$ computes $s_i = f(i)$ for $i = 1, 2, \ldots, n$.
- $\mathcal{U}$ establishes a secure channel with $\mathcal{KS}_i$ and sends s_i to $\mathcal{KS}_i$. $\mathcal{U}$ outsources aux to $\mathcal{CS}$.
- $\mathcal{KS}_i$ securely stores s_i.S

Access. $\mathcal{U}$ accesses S with the aid of $\vec{\mathcal{KS}}$ as follows.

- $\mathcal{U}$ establishes a secure channel with $\mathcal{KS}_i$, and $\mathcal{KS}_i$ sends s_i to $\mathcal{U}$ via the secure channel. This can be achieved by utilizing an authentication scheme based on $\overrightarrow{PIF}$.
- Upon receiving t secret shares (denoted by $\{s_{i_1}, \ldots, s_{i_t}\}$ for the sake of brevity), $\mathcal{U}$ computes $w_{i_l} = \prod_{\substack{1 \le j \le t \\ j \ne l}} \frac{i_j}{i_j - i_l}$ for $l = 1, \ldots, t$ and then computes $S = \sum_{l=1}^{t} w_{i_l} s_{i_l}$.

Recovery. When the number of available key servers is less than t, $\mathcal{U}$ recovers S with available PIFs as follows.

- $\mathcal{U}$ derives t^* secrets from available PIFs $\subseteq \overrightarrow{PIF}$. For the sake of brevity, we denote the secrets by $\{s_1^*, \ldots, s_{t^*}^*\}$.
- $\mathcal{U}$ generates a new polynomial

$$p_1(x) = x^{n^*} + \sum_{i=1}^{n^*-t^*} aux_i \cdot x^{n^*-i},$$

  where aux_i denotes the i-th element in aux.
- $\mathcal{U}$ solves Equation (1) to obtain $b_{t^*-1}, \cdots, b_0$.

$$\begin{cases} p_1(s_1^*) = & b_0 + b_1 s_1^* + \cdots + b_{t^*-1} {s_1^*}^{t^*-1} \\ \quad \cdots & \\ p_1(s_{t^*}^*) = & b_0 + b_1 s_{t^*}^* + \cdots + b_{t^*-1} {s_{t^*}^*}^{t^*-1} \end{cases} \tag{1}$$

- $\mathcal{U}$ computes $S = |b_0|$.

Note that Equation (1) can be represented by

$$\begin{pmatrix} 1 & s_1^* & \cdots & {s_1^*}^{t^*-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & s_{t^*}^* & \cdots & {s_{t^*}^*}^{t^*-1} \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ \cdots \\ b_{t^*-1} \end{pmatrix} = \begin{pmatrix} p_1(s_1^*) \\ \cdots \\ p_1(s_{t^*}^*) \end{pmatrix}.$$

To get b_0, $\mathcal{U}$ only needs to compute

$$\begin{pmatrix} b_0 \\ \cdots \\ b_{t^*-1} \end{pmatrix} = \begin{pmatrix} 1 & s_1^* & \cdots & {s_1^*}^{t^*-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & s_{t^*}^* & \cdots & {s_{t^*}^*}^{t^*-1} \end{pmatrix}^{-1} \cdot \begin{pmatrix} p_1(s_1^*) \\ \cdots \\ p_1(s_{t^*}^*) \end{pmatrix}.$$

3.6 Remark

In practice, directly solving Equation (1) may result in buffer overflows, which further causes key recovery failure, since s_i^* is too long (at least 256 bits). Essentially, we need to solve the following system of non-homogeneous linear equations:

$$\begin{pmatrix} 1 & s_1^* & \cdots & {s_1^*}^{t^*-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & s_{t^*}^* & \cdots & {s_{t^*}^*}^{t^*-1} \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ \cdots \\ b_{t^*-1} \end{pmatrix} = \begin{pmatrix} p_1(s_1^*) \\ \cdots \\ p_1(s_{t^*}^*) \end{pmatrix},$$

where $b_0, \ldots, b_{t^*-1}$ are unknown. We present an efficient algorithm to solve the equation as follows.

We set

$$A = \begin{pmatrix} 1 & s_1^* & \cdots & {s_1^*}^{t^*-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & s_{t^*}^* & \cdots & {s_{t^*}^*}^{t^*-1} \end{pmatrix},$$

and notice that A is a $t^* \times t^*$ Vandermonde matrix.

Consider the determinant of A as

$$det(A) = \prod_{1 \le j \le i \le t^*} (s_i^* - s_j^*).$$

When $s_1^*, \ldots, s_{t^*}^*$ are different, $det(A) \ne 0$. According to Cramer's Rule [26], iff $det(A) \ne 0$, the above system of non-homogeneous linear equations has a unique solution:

$$\left(\frac{det(A_1)}{det(A)}, \cdots, \frac{det(A_{t^*})}{det(A)}\right)^T, \tag{2}$$

where

$$A_j = \begin{pmatrix} 1 & s_1^* & \cdots & {s_1^*}^{j-1} & p_1(s_1^*) & {s_1^*}^{j+1} & \cdots & {s_1^*}^{t^*-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ 1 & s_{t^*}^* & \cdots & {s_t^*}^{j-1} & p_1(s_t^*) & {s_1^*}^{j+1} & \cdots & {s_{t^*}^*}^{t^*-1} \end{pmatrix}$$

for $j = 1, \ldots, t^*$.

[6] In reality, the number of PIFs can be different from that of reconstructable secrets. For the sake of simplicity, we assume that they are equal.

[7] DRKM supports to derive a secret from multiple PIFs. In the construction, we only describe how to derive a secret from a single storage/device-independent PIF for clarity.

Assume the algebraic complements of A are $\{A_{ij}\}$, where $1 \leq i, j \leq t^*$. Then we have

$$det(A_j) = \sum_{i=1}^{t^*} p_1(s_i^*) det(A_{ij}).$$

In the implementation, directly computing A^{-1} would take $\approx$ 25 MB buffer when we set $\ell = 256$ bits and $t^* = 5$, which always causes buffer overflow. By using Equation (2), we can efficiently solve Equation (1) without errors, where the required buffer is reduced to $\approx$ 40 KB.

3.7 Deploying DRKM

DRKM is compatible with existing cryptosystems. In reality, $\mathcal{U}$ has two methods to deploy DRKM as follows.

Integrating with encryption. For registered cryptosystems, $\mathcal{U}$ can utilize S to encrypt a set existing cryptographic keys $\vec{K}$ and outsource the ciphertexts to $\mathcal{CS}$. Subsequently, $\mathcal{U}$ can access S with the aid of key servers or with t^* PIFs and decrypt the ciphertexts to obtain $\vec{K}$. $\mathcal{U}$ can utilize private-key encryption or public-key encryption, and we discuss their pros and cons.

Private-key encryption. The computation costs of private-key encryption are much less than that of public-key one. In addition, $\mathcal{U}$ can decrypt $\vec{K}$ on demand when utilizing the CTR mode. However, when a user generates a new key, $\mathcal{U}$ has to recover the master key S and then encrypt the new key, which introduces extra costs.

Public-key encryption. Some public-key encryption algorithms, e.g., ElGamal encryption, support updating the ciphertexts without decryption when the public/private keys are updated (i.e., support for proxy re-encryption). Specifically, when $\mathcal{U}$ updates S, $\mathcal{U}$ does not need to first decrypt $\vec{C}$ with the previous private key and then encrypt $\vec{K}$ with the newly-generated public key. $\mathcal{U}$ can directly utilize the proxy re-encryption to update the ciphertexts of $\vec{K}$. What is more, When $\mathcal{U}$ has a newly-generated key, $\mathcal{U}$ can directly utilize the public key to encrypt the key without recovering S.

Integrating with KDFs. For a new cryptosystem, a more efficient method is to directly derive various new keys from S with mature KDFs [37, 38]. In this case, the user does not need to perform encryption or decryption operations.

Key rotation. Although an adversary who compromises less than t^* PIFs cannot get any information about the master key, the PIF leakage would reduce the security level of DRKM. Therefore, the master key should be updated after some PIFs are leaked. $\mathcal{U}$ can re-execute the algorithms in DRKM to rotate the master key using undisclosed PIFs. In this way, an adversary who compromises some previously used PIFs cannot get any information about the updated master key.

4 SECURITY ANALYSIS

4.1 Indistinguishability

The security of DRKM relies on the entropy of the master key derived from the PIFs. Assume that $\{PIF_1, \ldots, PIF_{n^*}\}$ are utilized to compute the master key, where the min-entropy of s_i^* derived from PIF_i is x_i and $x_1 \leq x_2 \leq \cdots \leq x_{n^*}$. We assume that an adversary $\mathcal{A}_1$ has corrupted $t-1$ key servers and obtained t^*-1 secrets $\{s_1^*, \ldots, s_{t^*-1}^*\}$. $\mathcal{A}_1$ cannot break the security of DRKM, i.e., DRKM satisfies Definition 6, which is captured by Theorem 2.

```
Indistinguishability:                              eIND-Key_{$\mathcal{A}_1$}($\ell$)
$\{s_1^*, \ldots, s_{n^*}^*\}$ are derived from $\{PIF_1, \cdots, PIF_{n^*}\}$ with F
Compute $S = \prod_{i=1}^{n^*} s_i^*$
Compute $aux$ (shown in the Managing algorithm)
$a_1, \ldots, a_{t-1} \leftarrow Z_p$
Generate $f(x) = S + a_1 x + \cdots + a_{t-1} x^{t-1}$
Compute $s_i = f(i)$ for $i = 1, 2, \ldots, n$
$S^\star \xleftarrow{\$} \{0,1\}^{|S|}$
$b \xleftarrow{\$} \{0,1\}$
If $b = 1$, $Key = S$; else, $Key = S^\star$
$b' \leftarrow \mathcal{A}_1(Key, s_{i_1}, \ldots, s_{i_{t-1}}, s_1^*, \ldots, s_{t^*-1}^*, aux)$
If $b' = b$, return 1; else, return 0
```

Figure 5: The eIND-Key$_{\mathcal{A}_1}(\ell)$ game.

Theorem 2. *Assuming F is a secure KDF, DRKM is secure against any PPT adversary $\mathcal{A}_1$ who compromises up to $t-1$ key servers and t^*-1 secrets $\{s_1^*, \ldots, s_{t^*-1}^*\}$, i.e., $\mathcal{A}_1$ cannot get any information about S from $t-1$ shares and t^*-1 secrets.*

Proof. To prove Theorem 2, we define the eIND-Key$_{\mathcal{A}_1}(\ell)$ game as shown in Figure 5. eIND-Key$_{\mathcal{A}_1}(\ell)$ outputs 1 iff $b = b'$, i.e., $\mathcal{A}_1$ can distinguish S from a random string. We prove the theorem as a series of games.

Game 1: this game is the same as the eIND-Key$_{\mathcal{A}_1}(\ell)$ game with the exception of one difference: $\mathcal{A}_1$ is given $t-1$ random numbers chosen from Z_p rather than $t-1$ shares. If there is a difference in $\mathcal{A}_1$ success probability between Game 1 and eIND-Key$_{\mathcal{A}_1}(\ell)$, it indicates that $\mathcal{A}_1$ can get extra information about the S from $t-1$ shares.

In DRKM, we utilize Shamir's threshold secret sharing scheme to share the master key S among the key servers. If the min-entropy of S is l-bit, the probability of $\mathcal{A}_1$ successfully guessing S is also 2^{-l} without any auxiliary information.

In DRKM, we have $S = f(0) = \sum_{i=1}^{t} w_i s_i$. Furthermore, we have $w_i s_t = S - \sum_{i=1}^{t-1} w_i s_i$. We notice that any value of S corresponds to a unique value of s_t. If $\mathcal{A}_1$ correctly guesses s_t, he can get the correct S. Next, we prove that $\{s_1, \ldots, s_{t-1}\}$ does not imply any information about s_t. These points $(1, s_1), \ldots, (t, s_t)$ correspond to a unique function $f(x) = S + a_1 x + \cdots + a_{t-1} x^{t-1}$. For the fixed points $(1, s_1), \ldots, (t-1, s_{t-1})$, any value of s_t may correspond to a unique function with degree $t-1$. Hence, $\mathcal{A}_1$ cannot get any information about s_t from $\{s_1, \ldots, s_{t-1}\}$.

Without $\{s_1, \ldots, s_{t-1}\}$, the probability of $\mathcal{A}_1$ successfully guessing s_t is 2^{-l}. Hence, we have

$$|\Pr[\text{eIND-Key}_{\mathcal{A}_1}(\ell) = 1] - \Pr[\text{Game } 1_{\mathcal{A}_1}(\ell) = 1]| \leq 2^{-l}.$$

Game 2: Game 2 is the same with Game 1, with one difference. $\mathcal{A}_1$ in Game 2 is given t^*-1 random secrets which have the same distribution with $s_{i_1}, \ldots, s_{i_{t-1}}^*, s_1^*$.

We suppose that the lower entropy of a PIF, the higher the risk of leakage. In Game 1, $\mathcal{A}_1$ has obtained $t^* - 1$ secrets $\{s_1^*, \ldots, s_{t^*-1}^*\}$. With the auxiliary information *aux*, $\mathcal{A}_1$ can construct a polynomial

$$p_1(x) = x^{n^*} + \sum_{i=1}^{n^*-t^*} aux_i \cdot x^{n^*-i},$$

where aux_i denotes the i-th element in *aux*.

With $t^* - 1$ sub-secrets $\{s_1^*, \ldots, s_{t^*-1}^*\}$, $\mathcal{A}_1$ can construct the following equation system

$$\begin{cases} p_1(s_1^*) = & b_{t^*-1} {s_1^*}^{t^*-1} + \cdots + b_1 s_1^* + b_0 \\ \quad \cdots & \\ p_1(s_{t^*-1}^*) = & b_{t^*-1} {s_{t^*-1}^*}^{t^*-1} + \cdots + b_1 s_{t^*-1}^* + b_0 \end{cases},$$

where $b_{t^*-1}, \cdots, b_0$ are the unknown to be solved. There exists countless valid $\{b_{t^*-1}, \cdots, b_0\}$, and $\mathcal{A}_1$ cannot determine which is the correct b_0. The straightforward way for $\mathcal{A}_1$ to determine b_0 is to construct another equation with $s_{t^*}^*$ as

$$p_1(s_{t^*}^*) = b_{t^*-1} {s_{t^*}^*}^{t^*-1} + \cdots + b_1 s_{t^*}^* + b_0.$$

The maximum probability that the adversary can obtain a new secret is $2^{-x_{t^*}}$ (where $\mathcal{A}_1$ directly guesses $s_{t^*}^*$). Therefore, given $t^* - 1$ secrets $\{s_1^*, \ldots, s_{t^*-1}^*\}$ and the auxiliary information *aux*, the probability of $\mathcal{A}_1$ getting the master key S is no more than $2^{-x_{t^*}}$. Hence, we have

$$|\Pr[\text{Game } 1_{\mathcal{A}_1}(\ell) = 1 - \Pr[\text{Game } 2_{\mathcal{A}_1}(\ell) = 1]| \le 2^{-x_{t^*}}.$$

Next, we analyze the probability of $\mathcal{A}_1$ winning Game 2. In our proof, we follow the security definition of KDF proposed in [39], as shown in Definition 8.

Definition 8. *A key derivation function (KDF) is secure with respect to an m-min-entropy source of key material* Σ *if no adversary* $\mathcal{A}$ *can distinguish the key generated from* Σ *and a random string of the same length with probability better* $1/2 + 2^{-m}$.

If F is a secure KDF, the min-entropy of S is $\sum_{i=1}^{t^*} x_i$-bit. Hence, we have

$$\Pr[\text{Game } 2_{\mathcal{A}_1}(\ell) = 1] \le \frac{1}{2} + 2^{-\sum_{i=1}^{t^*} x_i}.$$

Therefore, the probability of $\mathcal{A}_1$ winning eIND-Key$_{\mathcal{A}_1}(\ell)$ is no more than $\frac{1}{2} + 2^{-\sum_{i=1}^{t^*} x_i - 1} + 2^{-x_{t^*}}$. □

Remark. Theorem 2 implies that the master key S is a uniform l-bit cryptographic key, where $l \ge \sum_{i=1}^{t^*} x_i$. Theoretically, the min-entropy of the secrets derived from the biometric characteristic is determined by the user per se. However, in practice, the biometrics extraction algorithms would influence the entropy of the extracted secrets. The entropy of the secrets derived from the device-dependent PIFs is determined by the security parameter of the corresponding authentication schemes.

4.2 PIF privacy

We prove that DRKM satisfies Definition 7, i.e., an adversary $\mathcal{A}_2$ who compromises $t-1$ secret shares cannot get any information about the secrets $\{s_1^*, \ldots, s_{n^*}^*\}$, where the min-entropy of s_i^* derived from PIF_i is x_i and $x_1 \le x_2 \le \cdots \le x_{n^*}$. This security notion is captured by Theorem 3.

PIF privacy: ePIF-Privacy$_{\mathcal{A}_2}(\ell)$

```
1: $\{s_1^*, \ldots, s_{n^*}^*\}$ are derived from $\{PIF_1, \cdots, PIF_{n^*}\}$ with F
2: Compute $S = \prod_{i=1}^{n^*} s_i^*$
3: Compute $aux$ (shown in the **Managing** algorithm)
4: Generates a polynomial $p(x) = \prod_{i=1}^{n^*} (x - s_i^*)$
5: Select a set $\{s_{i_1}^*, \ldots, s_{i_{t-1}}^*\}$
6: Generate a PIF set $\overrightarrow{PIF_0}$, where $\overrightarrow{PIF_0} \cap \{PIF_1, \ldots, PIF_{n^*}\} = \emptyset$ and $|\overrightarrow{PIF_0}| = t^* - 1$
7: Derive $\{{s_1^*}', \ldots, {s_{t^*-1}^*}'\}$ from $\overrightarrow{PIF_0}$
8: Set $subS_0 = \{{s_1^*}', \ldots, {s_{t^*-1}^*}'\}$, $subS_1 = \{s_{i_1}^*, \ldots, s_{i_{t-1}}^*\}$
9: $b \overset{\$}{\leftarrow} \{0, 1\}$
10: $b' \leftarrow \mathcal{A}_1(subS_b, s_{i_1}, \ldots, s_{i_{t-1}}, aux)$
11: If $b' = b$, return 1; else, return 0.
```

Figure 6: The ePIF-Privacy$_{\mathcal{A}_2}(\ell)$ game.

Theorem 3. *DRKM protects* $\mathcal{U}$*'s PIFs used to derive* S *against* $\mathcal{A}_2$ *who compromises* $t-1$ *key servers.*

Proof. To prove Theorem 3, we define a PIF-Privacy$_{\mathcal{A}_2}(\ell)$ game, as shown in Figure 6.

Given points $\{(1, s_1), (2, s_1), \ldots, (t-1, s_{t-1})\}$ on the function $f(x)$, $\mathcal{A}_2$ aims to determine whether some function $p'(x)$ (with degree t^*) generate by $subS_0$ as zero points satisfies $f(0) = |p(0)|$. For the PIF-Privacy$_{\mathcal{A}_2}(\ell)$ game, we have

$$\begin{aligned} \Pr[\text{PIF-Privacy}_{\mathcal{A}_2}(\ell) = 1] &= \Pr[b' = b] \\ &= \Pr[b' = 0 | b = 0] + \Pr[b' = 1 | b = 1]. \end{aligned}$$

When $b = 0$, $\mathcal{A}_2$ can win if it guesses out some zero point of $p(x)$ and guesses out the master key S. Therefore, we have

$$\Pr[b' = 0 | b = 0] = \frac{1}{2}(\frac{1}{2} + 2^{-x_i} * 2^{-l}),$$

where l is the min-entropy of S, and x_i is the min-entropy of the zero point (i.e., the PIF).

When $b = 1$, $\mathcal{A}_2$ can win if (1) it guesses out some zero point of $p(x)$ which is included in $subS_0$ or (2) it guesses out some zero point of $p(x)$ which is not included in $subS_0$ but guesses out the master key S. Hence, we have

$$\begin{aligned} &\Pr[b' = 1 | b = 1] \\ &= \frac{1}{2}(\frac{1}{2} + \frac{C_{n^*-1}^{t^*-2}}{C_{n^*}^{t^*-1}} * 2^{-x_i} + (1 - \frac{C_{n^*-1}^{t^*-2}}{C_{n^*}^{t^*-1}}) * 2^{-l}) \\ &= \frac{1}{2}(\frac{1}{2} + \frac{t-1}{n} * 2^{-x_i} + (1 - \frac{t-1}{n}) * 2^{-l}) \\ &\le \frac{1}{2}(\frac{1}{2} + 2^{-l} + 2^{-x_i}). \end{aligned}$$

Therefore, we have

$$\Pr[\text{PIF-Privacy}_{\mathcal{A}_2}(\ell) = 1] \le \frac{1}{2} + negl(\ell).$$

This concludes the proof. □

PIF	Computation delay (ms)
Fingerprint [40]	≈1.60
Face [41]	≈1.50
Iris [42]	≈3.00
Hand geometry [43]	≈7.6
Palmprint [44]	≈6
ECG [46]	≈0.32
Hand gesture [45]	≈90 − 110
SIM card	≈0.003
Hardware token	≈1.36
Intel SGX	≈0.56

Table 1: Computation delay of deriving secrets from PIFs.

5 IMPLEMENTATION AND EVALUATION

We implement a DRKM prototype and conduct a comprehensive performance evaluation. All experiments are conducted on a laptop with an Intel Core i5 processor running at 2 GHz using four cores and 16 GB DDR3 of RAM. Our source code is available on https://github.com/DRKM-code/DRKM.git.

We evaluate the performance from the following aspects.

5.1 Deriving secrets from PIFs

We present the computation delay to derive reconstructable secrets from different PIFs, as shown in Table 1.

Someone the user is. We investigate existing biometric feature recognition and extraction methods. A minutiae-based fingerprint individuality model [40] is utilized to extract features from users' fingerprints. We leverage the scheme [41] for face recognition to extract the face features. We utilize the algorithm in [42] for iris recognition to extract the features of irises. The extracted iris features can be formatted with the biometric standards. We utilize the hand geometry recognition scheme [43], the palmprint recognition scheme [44], and the hand gesture recognition scheme [45] to extract the hand characteristics, respectively. We utilize Deep-ECG [46], a CNN-based biometric approach for ECG signals, to extract the features of ECG.

Something the user has. We simulate the authentication algorithms of a SIM card, hardware token, and Intel SGX on a laptop, and evaluate the performance. The hardware PIFs are dependent on stateful devices. Recalling Section 2, we can derive a reconstructable secret from a device-dependent PIF with the aid of another reconstructable secret.

5.2 Key derivation and management

We investigate existing authentication schemes and choose the applicable methods to derive secrets from PIFs. The computation delay for deriving reconstructable secrets from different PIFs is shown in Section 5.1. If n is larger than 10, $\mathcal{U}$ can derive multiple instances from a type of PIF. For example, if $\mathcal{U}$ needs to derive 20 reconstructable secrets, she/he can choose 2 irises, 2 hand geometries, 2 Palmprints, 1 ECG, 2 SIM cards, 1 hardware token, and 10 fingerprints. In the following, we assume that $\mathcal{U}$ has obtained n^* secrets.

$\mathcal{U}$ computes the master key as $S = \prod_{i=1}^{n^*} s_i^*$. Figure 7(a) shows the computation delay on $\mathcal{U}$ of deriving S from n^* secrets, where we set different n^*. The computation delay is an average of deriving 10 master keys.

$\mathcal{U}$ computes the auxiliary information *aux* which assists $\mathcal{U}$ in recovering the master key from the PIFs. Figure 7(b) shows the computation delay on $\mathcal{U}$ of *aux* with different n^* and t^*. The delay of computing *aux* decreases with t^*. $\mathcal{U}$ shares S among the key servers in a threshold way. Specifically, $\mathcal{U}$ first computes n secret shares $\{s_1, \cdots, s_n\}$. Figure 7(c) shows the delay in computing the secret shares.

$\mathcal{U}$ can use private-key encryption or public-key encryption to encrypt $\vec{K}$. Figure 7(d) and 7(e) show the computational delay for encrypting different size K with different encryption algorithms including RSA-OAEP [47], ElGamal [48], and [CTR]AES. The computation delay for private-key encryption is much less than that for public-key encryption.

In DRKM, $\mathcal{U}$ needs to send the secret shares to the key servers and outsource the auxiliary information and the ciphertexts to $\mathcal{CS}$. In Figure 8(a), we show the communication costs on $\mathcal{U}$ and $\mathcal{KS}_i$. As shown in Figure 8(b), the communication costs on $\mathcal{CS}$ are approximate to the size of K when using private-key encryption. For the same size K, the communication costs on $\mathcal{CS}$ increase with $n^* - t^*$, because the larger $n^* - t^*$ is, the larger size of *aux* is when recovering S. Figure 8(c) shows the communication costs of $\mathcal{CS}$. The communication costs when using public-key encryption are significantly larger than those when using [CTR]AES.

5.3 Key access and recovery

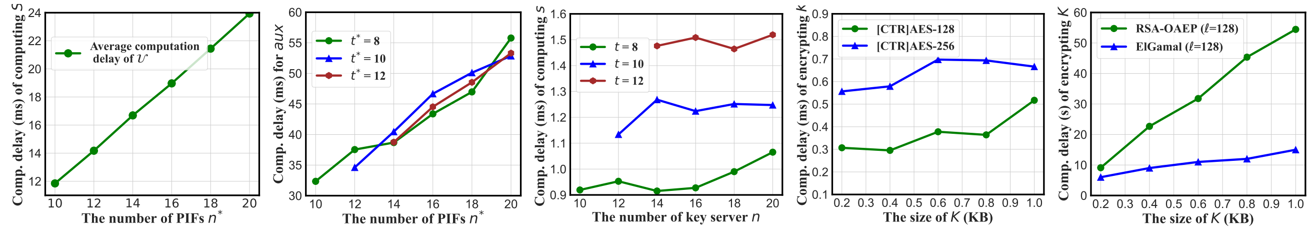
When $\mathcal{U}$ needs to access the master key, $\mathcal{U}$ interacts with the key server via the secure channels to get t secret shares. In Figure 8(d), we show the communication costs on $\mathcal{U}$ and each key server with different n and t. $\mathcal{U}$ also needs to download the ciphertexts from $\mathcal{CS}$. $\mathcal{U}$ can download the ciphertexts on demand when using private-key encryption. Figure 8(e) shows the communication costs of $\mathcal{CS}$ and $\mathcal{U}$, where $\mathcal{U}$ only accesses one key. Figure 9(a) shows the delay on $\mathcal{U}$ to compute S from t secret shares.

If the number of available key servers is less than t, $\mathcal{U}$ has to recover the master by utilizing available PIFs. $\mathcal{U}$ needs to get at least t^* secrets from available PIFs. We assume that $\mathcal{U}$ has derived t^* secrets. Figure 9(b) shows the delay on $\mathcal{U}$ to recover S from t^* secrets and *aux*. The results show that the delay of recovering S from secrets is less than 5 ms.

5.4 Comparison with KDF

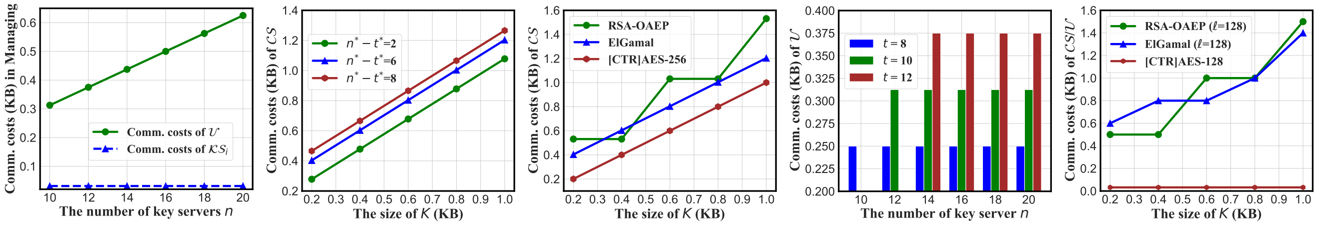
The core of DRKM is to convert a user's PIFs to a reconstructable master key which is utilized to encrypt the user's keys used in various cryptosystems. A natural method to transform a PIF into a key is key derivation functions (KDF). In this subsection, we compare DRKM with different KDFs and further elaborate on the desirable advantages of DRKM.

We start with password-based key derivation functions (PBKDFs) [49–53] that derive keys from users' passwords. The proliferation of PBKDFs provides protection for users without introducing key management problems. Typically, PBKDF1 and PBKDF2 [49] are widely used in network applications, e.g., WPA [54] and WPA2



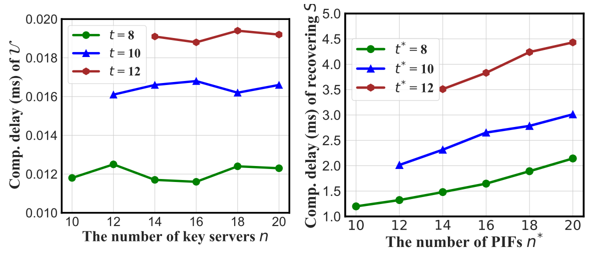
(a) Generating S. (b) Computing aux. (c) Computing s_i. (d) Private-key encryption. (e) Public-key encryption.

Figure 7: The computation delay of $\mathcal{U}$ in the Managing phase.



(a) Sending/receiving s_i. (b) [CTR]AES-128/256. (c) RSA-OAEP/ElGamal. (d) Getting s_i from $\mathcal{KS}_i$. (e) Downloading C.

Figure 8: The communication costs of $\mathcal{U}$, $\mathcal{KS}_i$, and $\mathcal{CS}$.



(a) Accessing S. (b) Recovering S.

Figure 9: The computation delay of $\mathcal{U}$.

Method	Computation delay (ms)
DRKM (Before destruction)	0.012
DRKM (After destruction)	90.72
Threshold MFKDF	89.82

Table 2: Comparison with the threshold MFKDF [27]. We set $t = t^* = 8$ and $n = n^* = 10$.

[55] protocols in wireless communication systems. However, the schemes based on PBKDFs are generally confronted with dictionary guessing attacks, since passwords are inherently low-entropy. Many recent security incidents [56, 57] have shown that utilizing passwords as the sole defense line is indeed insufficient.

Recently, Nair et al. [27] proposed a threshold multi-factor key derivation function (TMFKDF), which allows a user to derive a key from n^* PIFs, and recover the key from any t^* of them. We compare DRKM with the TMFKDF in the following aspects.

Regarding costs, Table 2 shows the comparison between DRKM and TMFKDF in terms of computation delay of accessing the master key with $t = t^* = 8$ and $n = n^* = 10$. The results show that in normal times, the delay using DRKM is 5~6 orders of magnitude less than that of using TMFKDF.

Regarding functionality, the key recovery provided by TMFKDF depends on the availability of metadata, i.e., encrypted shares. Users cannot recover their keys without metadata even if they utilize n^* storage-independent PIFs. In DRKM, users can directly retrieve their keys from n^* PIFs without any storage, if all n^* PIFs are independent of storage.

Regarding security, DRKM provides stronger protection than t^*-of-n^* TMFKDF. In DRKM, a master key is derived from n^* PIFs and then split into n shares, where the shares are independent of the secrets derived from PIFs. However, in [27], once an adversary compromises a PIF, then he can get the corresponding share. Furthermore, in reality, an adversary who compromises t^* PIFs may not recover the master key. Specifically, we assume that a user has n^* PIFs, where n_1^* PIF are storage/device-independent, n_2^* PIFs are device-dependent, and $n^* = n_1^* + n_2^*$. Then the user can derive at least $n_1^* + n_1^* \cdot n_2^*$ (conditional) reconstructable secrets. Hence, the user can generate $C^{n^*}_{n_1^*+n_1^* * n_2^*}$ different master keys. By introducing device-dependent PIFs, recovering some of these master keys needs more than t^* PIFs, since deriving a reconstructable secret from a device-dependent PIF requires another new reconstructable secret.

Regarding practicability, DRKM enables users to balance security and efficiency but the TMFKDF fails to achieve it. Specifically, there are two independent thresholds (t, n), (t^*, n^*) in DRKM. The threshold in TMFKDF is corresponding to (t^*, n^*) in DRKM. Once an adversary compromises t^* shares (i.e., PIFs) of TMFKDF, he can recover the key. In DRKM, such an adversary cannot get any information about the master key, since the user can set $t \geq t^*$ without changing (t^*, n^*) to enhance the security guarantee. However, the larger t is, the more costs the user bears to access the master key in normal times. The user can achieve a trade-off between security and efficiency by adjusting (t, n) while remaining (t^*, n^*).

5.5 Comparison with alternative schemes

Currently, some products and solutions for secure and reliable key management have been proposed, and they might be trivially extended to achieve destruction resistance. We will discuss them in detail, analyze their inherent problems, and compare them with DRKM in the following. Our aim in presenting this section is twofold. The first one is to show the advantages of DRKM in terms of security, functionality, and efficiency. The second one is to demonstrate that designing usable and destruction-resistant key management schemes is very challenging. Generally, existing key management schemes can be categorized into two types: fully local management ones and fully outsourcing management ones.

Fully local management schemes. A user keeps her/his keys in well-guarded devices and keeps them in safe places. Notable examples include YubiKey [3] and Ledger [4]. To achieve destruction resistance, the user can simultaneously utilize YubiKey and Ledger to store the same key (one for general use and the other one for backup) and keep them in different secure locations.

However, the disadvantage of the above scheme is oblivious. The user needs to ensure the security and reliability of the backup key, which requires the user to continuously monitor its condition to minimize the possibility of undetected leakage. Actually, this would cause prohibitive costs for the user. As an empirical observation, individuals do not always have a secure secondary location to store the backup key.

Fully outsourcing management scheme. A user requests key management services from service providers. Typical examples include Keywhiz [58], where the user's key is split into n shares in a threshold way and let each key server store one of them. The user can recover her/his key by interacting with any t of n key servers. Compared with a fully local management scheme, this scheme achieves portability: it enables the user to recover the key as needed without maintaining any secret on local devices. It achieves destruction resistance to some extent: as long as t key servers are available, the key can be recovered.

Obliviously, the destruction resistance provided by this scheme relies on a strong assumption that at least t key servers would not be destroyed. This problem cannot be trivially addressed by requiring the key server to back up the share, as it increases the danger of security breaches significantly [11].

DRKM: a hybrid "local + outsourcing" key management scheme. In DRKM, a user derives a master key from her/his PIFs and shares it among the key servers in a threshold way. DRKM inherits the advantages of the above schemes: in normal times, the user can access the master key by interacting with the key servers; and once all key servers and the user's devices are destroyed, the master key can be recovered from a part of PIFs used for key derivation. DRKM also overcomes their drawbacks: compared with the fully local management scheme, DRKM only requires a *stateless* device and a small number of PIFs to recover the master key after the destruction occurs; compared with the fully outsourcing management one, users in DRKM can retrieve their keys *per se*, even if all key servers are unavailable.

5.6 Applications and compatibility

With storage/device-independent PIFs, we can easily construct a destruction-resistant threshold multiple-factor authentication scheme, where a user only needs to register with a service provider by more than a threshold number of storage/device-independent PIFs. After the destruction occurs, the user can still log in to the service provider via available PIFs. We further emphasize the practical nature of DRKM as well as DRKM-based DRC (i.e., DRKM + commercial backup system) in the following.

One potential application is to manage a secret—say, a key to a bank's vault—that is shared among a board of directors, where the vault is protected by cryptosystems. Anytime when the vault needs to be opened, it should be confirmed and agreed by a majority of members. With DRKM, each member can contribute the secret using her/his PIFs with the (t^*, n^*)-secret aggregation with threshold retrieval mechanism, and then the secret is shared among multiple key servers with the (t^*, n^*)-secret sharing scheme. Each key server is available for a specific member. Before device destruction occurs, the members can recover the secret by interacting with their key servers in a portable way. Once the number of the available key servers is less than t^*, they can cooperatively recover the secret using their PIFs. We stress that regarding device destruction and security, neither Shamir's secret sharing scheme nor the threshold password-hardening protocols [11, 59–66] can achieve it.

Another promising application of DRKM and DRKM-based DRC is to resist ransomware attacks. We notice that the success of ransomware attacks is to make devices unavailable to their users. As such, if the maintenance of the most sensitive information (e.g., the master key) does not rely on any device, a ransomware attack will be doomed to failure (we impliedly require a backup system to store the non-sensitive information, such as ciphertexts, signatures). With DRKM, a user can derive a master key from her/his PIFs and maintain the master key locally or using a set of key servers. Once the master key is hacked by ransomware attacks, the user can recover the master key from the PIFs that are used to derive it. Therefore, DRKM-based DRC can be directly deployed to thwart ransomware attacks.

Recently, many novel PIFs have been proposed, e.g., PCR-Auth [67] and Capacitive Plethysmogram [68]. We stress that DRKM has forward-compatibility with future PIFs. As long as a PIF satisfies Definition 2 and can be represented by a unique binary string, it then can be utilized in DRKM to derive a master key, which potentially applies to newly-discovered PIFs in the future.

6 CONCLUSION

In this paper, we have investigated popular personal identification factors (PIFs) and proposed three concepts (i.e., storage-independent PIFs, device-independent PIFs, and device-dependent PIFs), and given the categorization criteria. We have proposed a series of methods to derive reconstructable secrets from a special class of device-dependent PIFs in tandem with storage/device-independent ones. We have constructed DRKM, a destruction-resistant key management scheme with portability. We have formally proven the security of DRKM. We have implemented a DRKM prototype and conducted a comprehensive performance evaluation to demonstrate its high efficiency.

For the future work, we will explore new storage-independent PIFs with high min-entropy. Furthermore, with such PIFs, it may be possible to construct a key management scheme that simultaneously achieves destruction resistance and portability after the destruction occurs.

REFERENCES

[1] M. Azure, "Key Management in Azure," https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management, 2023.

[2] G. Developer, "Cloud Key Management," https://cloud.google.com/security-key-management/#section-3, 2023.

[3] Yubikey, https://www.yubico.com/products/yubikey-5-overview/, 2023.

[4] Ledger, https://www.ledger.com, 2023.

[5] A. Developer, "Keychain Services," https://developer.apple.com/documentation/security/keychain_services, 2023.

[6] M. Christodorescu, S. Gaddam, P. Mukherjee, and R. Sinha, "Amortized Threshold Symmetric-Key Encryption," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2021, pp. 2758–2779.

[7] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Server-Aided Encryption for Deduplicated Storage," in *Proceedings of Usenix Security Symposium (USENIX Security)*, 2013, pp. 179–194.

[8] G. Poonia, "The Reason not to Throw Away Old Hard Drives Might Be Surprising—There Could be Bitcoin on There," https://www.deseret.com/2021/12/10/22827963/james-howells-threw-away-hard-drives-with-\bitcoin-password, 2021.

[9] C. Osborne, "Colonial Pipeline Ransomware Attack: Everything You Need to Know," https://www.zdnet.com/article/colonial-pipeline-ransomware-attack-everything-you\-need-to-know/, 2021.

[10] J. Reed, "Costa Rica State of Emergency Declared After Ransomware Attacks," https://securityintelligence.com/news/costa-rica-state-emergency-ransomware/, 2022.

[11] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[12] S. K. D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song, "CHURP: Dynamic-Committee Proactive Secret Sharing," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2019, pp. 2369–2386.

[13] Vault, https://developer.hashicorp.com/vault, 2023, HashiCorp Developer.

[14] https://aws.amazon.com/cn/message/41926/, 2017.

[15] N. Agrawal, "Amazon Cloud Service Outage Breaks Parts of the Internet," https://www.latimes.com/business/technology/la-fi-tn-amazon-service-outage-20170228-story.html, 2017.

[16] Wikipedia, "2021–22 Hunga Tonga–Hunga Haapai Eruption and Tsunami," https://en.wikipedia.org/wiki/2021âĂŞ22_Hunga_TongaâĂŞHunga_Ha'apai_eruption_and_tsunami, 2021.

[17] C. Legislature, "California Consumer Privacy Act of 2018 (as amended by the California Privacy Rights Act of 2020)," 2020.

[18] E. Parliament and Council, "General Data Protection Regulation, Regulation (EU) 2016/679 (as amended)." 2016.

[19] A. Takahashi, Y. Koda, K. Ito, and T. Aoki, "Fingerprint Feature Extraction by Combining Texture, Minutiae, and Frequency Spectrum Using Multi-Task CNN," in *Proceedings of IEEE International Joint Conference on Biometrics (IJCB)*, 2020, pp. 1–8.

[20] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "Deepmarks: A Secure Fingerprinting Framework for Digital Rights Management of Deep Learning Models," in *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR)*, 2019, pp. 105–113.

[21] D. Aggarwal, J. Zhou, and A. K. Jain, "Fedface: Collaborative Learning of Face Recognition Model," in *Proceedings of IEEE International Joint Conference on Biometrics (IJCB)*, 2021, pp. 1–8.

[22] https://www.cbsnews.com/essentials/the-best-smart\watches-for-heart-health-monitoring-2023-08-23, 2023.

[23] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted Online Password Guessing: An Underestimated Threat," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2016, pp. 1242–1254.

[24] GoogleDrive, https://www.google.com/intl/en-GB/drive/, 2023.

[25] Dropbox, https://www.dropboxforum.com, 2023.

[26] A. Ben-Israel, "A Cramer Rule for Least-Norm Solutions of Consistent Linear Equations," *Linear Algebra and Its Applications*, vol. 43, pp. 223–226, 1982.

[27] V. Nair and D. Song, "Multi-Factor Key Derivation Function (MFKDF) for Fast, Flexible, Secure, & Practical Key Management," in *Proceedings of USENIX Security Symposium (USENIX Security)*, 2023.

[28] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," in *Proceedings of European Cryptology Conference (EUROCRYPT)*, 2004, pp. 523–540.

[29] A. Juels and M. Sudan, "A Fuzzy Vault Scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[30] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, "Fourth-Factor Authentication: Somebody You Know," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2006, pp. 168–178.

[31] K. Lee, B. Kaiser, J. Mayer, and A. Narayanan, "An Empirical Study of Wireless Carrier Authentication for SIM Swaps," in *Proceedings of Symposium on Usable Privacy and Security (SOUPS)*, 2020, pp. 61–79.

[32] "Edward Snowden: Timeline," https://www.bbc.com/news/world-us-canada-23768248, 2013.

[33] E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo, "True2F: Backdoor-Resistant Authentication Tokens," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019, pp. 398–416.

[34] V. Costan and S. Devadas, "Intel SGX Explained," *Cryptology ePrint Archive*, 2016.

[35] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," in *Proceedings of International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, vol. 10, no. 1, 2013.

[36] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative Technology for CPU Based Attestation and Sealing," in *Proceedings of International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, vol. 13, no. 7, 2013.

[37] Q. Dang, "Recommendation for Existing Application-Specific Key Derivation Functions," *NIST SP 800-135, Revision 1*, 2011.

[38] L. Chen, "Recommendation for Key Derivation Using Pseudorandom Functions," *NIST SP 800-108r1*, 2022.

[39] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," in *Proceedings of International Cryptology Conference (CRYPTO)*, vol. 6223, 2010, pp. 631–648.

[40] J. Chen and Y.-S. Moon, "A Minutiae-Based Fingerprint Individuality Model," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–7.

[41] FingerTec, "FingerTec Face Recognition Technology White Paper," https://www.fingertec.com/download/tips/whitepaper-02.pdf, 2009.

[42] J. Daugman, "How Iris Recognition Works," in *The Essential Guide to Image Processing*, 2009, pp. 715–739.

[43] S. Angadi and S. Hatture, "Hand Geometry Based User Identification Using Minimal Edge Connected Hand Image Graph," *IET Computer Vision*, vol. 12, no. 5, pp. 744–752, 2018.

[44] M. Ahmadi and H. Soleimani, "Palmprint Image Registration Using Convolutional Neural Networks and Hough Transform," *arXiv preprint arXiv:1904.00579*, 2019.

[45] Y. Fang, K. Wang, J. Cheng, and H. Lu, "A Real-Time Hand Gesture Recognition Method," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2007, pp. 995–998.

[46] D. Labati Ruggero, E. Munoz, V. Piuri, R. Sassi, and F. Scotti, "Deep-ECG Convolutional Neural Networks for ECG Biometric Recognition," *Pattern Recognition Letters*, vol. 126, pp. 78–85, 2019.

[47] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption," in *Proceedings of Workshop on the Theory and Application of of Cryptographic Techniques (EUROCRYPT)*, 1994, pp. 92–111.

[48] T. ElGamal, "A Public Key Cryptosystem and A Signature Scheme based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[49] B. Kaliski, "PKCS# 5: Password-Based Cryptography Specification Version 2.0," Tech. Rep., 2000.

[50] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 292–302.

[51] F. F. Yao and Y. L. Yin, "Design and Analysis of Password-Based Key Derivation Functions," in *The Cryptographers' Track at the RSA Conference*, 2005, pp. 245–261.

[52] C. Percival and S. Josefsson, "The Scrypt Password-Based Key Derivation Function," Tech. Rep., 2016.

[53] M. S. Turan, E. Barker, W. Burr, and L. Chen, "Recommendation for Password-Based Key Derivation," *NIST Special Publication*, vol. 800, p. 132, 2010.

[54] I. C. S. L. M. S. Committee *et al.*, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std. 802.11-1997*, 1997.

[55] IEEE, "IEEE Standard for Information Technology: Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Medium Access Control (MAC) Security Enhancements," *IEEE Std. 802.11i-2004*, 2004.

[56] S. Ikeda, "Half a Million Zoom Accounts Compromised by Credential Stuffing, Sold on Dark Web," https://www.cpomagazine.com/cyber-security/half-a-million-zoom-accounts\-compromised-by-credential-stuffing-sold-on-dark-web/, 2020.

[57] M. Kapko, "PayPal Warns 35,000 Customers of Exposure Following Credential Stuffing Attack," https://www.cybersecuritydive.com/news/paypal-credential-stuffing-attack/640804/, 2023.

[58] Keywhiz, https://square.github.io/keywhiz/, 2015.

[59] C. Baum, T. Frederiksen, J. Hesse, A. Lehmann, and A. Yanai, "PESTO: Proactively Secure Distributed Single Sign-on, or How to Trust a Hacked Server," in *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020, pp. 587–606.

[60] Y. Zhang, C. Xu, H. Li, K. Yang, N. Cheng, and X. Shen, "PROTECT: Efficient Password-Based Threshold Single-Sign-on Authentication for Mobile Users against Perpetual Leakage," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2297–2312, 2020.

[61] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "TOPPSS: Cost-Minimal Password-Protected Secret Sharing Based on Threshold OPRF," in *Proceedings of International Conference on Applied Cryptography and Network Security (ACNS)*, 2017, pp. 39–58.

[62] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "PASTA: Password-Based Threshold Authentication," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2018, pp. 2042–2059.

[63] Y. Zhang, C. Xu, N. Cheng, and X. S. Shen, "Secure Password-Protected Encryption Key for Deduplicated Cloud Storage Systems," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[64] R. W. Lai, C. Egger, M. Reinert, S. S. Chow, M. Maffei, and D. Schröder, "Simple Password-Hardened Encryption Services," in *Proceedings of Usenix Security Symposium (USENIX Security)*, 2018, pp. 1405–1421.

[65] C. Jia, S. Wu, and D. Wang, "Reliable Password Hardening Service with Opt-Out," in *Proceedings of International Symposium on Reliable Distributed Systems (SRDS)*, 2022.

[66] L. Chen, Y.-N. Li, Q. Tang, and M. Yung, "End-to-Same-End Encryption: Modularly Augmenting an App with an Efficient, Portable, and Blind Cloud Storage," in *Proceedings of USENIX Security Symposium (USENIX Security)*, 2022, pp. 2353–2370.

[67] L. Huang and C. Wang, "PCR-Auth: Solving Authentication Puzzle Challenge with Encoded Palm Contact Response," in *IEEE Symposium on Security and Privacy (S&P)*, 2022, pp. 1034–1048.

[68] J. Wu, X. Ji, Y. Lyu, X. Luo, Y. Meng, E. Morales, D. Wang, and X. Luo, "Touchscreens Can Reveal User Identity: Capacitive Plethysmogram-Based Biometrics," *IEEE Transactions on Mobile Computing*, 2022.

[69] A. Rényi *et al.*, "On Measures of Entropy and Information," in *Proceedings of Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 547-561, 1961.

[70] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[71] A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2005, pp. 364–372.

[72] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, Name and Bifacial-Security: Understanding Passwords of Chinese Web users," in *Proceedings of USENIX Security Symposium (USENIX Security)*, 2019, pp. 1537–1555.

[73] D. Wang, Q. Gu, X. Huang, and P. Wang, "Understanding Human-Chosen PINs: Characteristics, Distribution and Security," in *Proceedings of ACM on Asia Conference on Computer and Communications Security (ASIA CCS)*, 2017, pp. 372–385.

[74] C.-P. Schnorr, "Efficient Identification and Signatures for Smart Cards," in *Proceedings of Conference on the Theory and Application of Cryptology (ASIACRYPT)*, 1989, pp. 239–252.

[75] A. Figueroa, "Fingerprint Recognition: the Most Popular Biometric," https://www.rootstrap.com/blog/fingerprint-recognition-the-most-popular-biometric/, 2022.

[76] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, 2009.

[77] Y. Tang, F. Gao, J. Feng, and Y. Liu, "FingerNet: An Unified Deep Network for Fingerprint Minutiae Extraction," in *Proceedings of IEEE International Joint Conference on Biometrics (IJCB)*, 2017, pp. 108–116.

[78] C. Wu, K. He, J. Chen, Z. Zhao, and R. Du, "Liveness Is not Enough: Enhancing Fingerprint Authentication with Behavioral Biometrics to Defeat Puppet Attacks," in *Proceedings of Usenix Security Symposium (USENIX Security)*, 2020, pp. 2219–2236.

[79] M. Wang and W. Deng, "Mitigating Bias in Face Recognition Using Skewness-Aware Reinforcement Learning," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9322–9331.

[80] P. Yao, J. Li, X. Ye, Z. Zhuang, and B. Li, "Iris Recognition Algorithm Using Modified Log-Gabor Filters," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, vol. 4, 2006, pp. 461–464.

[81] R. P. Wildes, "Iris Recognition: An Emerging Biometric Technology," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1348–1363, 1997.

[82] H. Proença and J. C. Neves, "IRINA: Iris Recognition (even) in Inaccurately Segmented Data," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 538–547.

[83] A. Boukhayma, R. d. Bem, and P. H. Torr, "3D Hand Shape and Pose from Images in the Wild," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 843–10 852.

[84] Y. Song, Z. Cai, and Z.-L. Zhang, "Multi-Touch Authentication Using Hand Geometry and Behavioral Information," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2017, pp. 357–372.

[85] Y. Han, T. Tan, Z. Sun, and Y. Hao, "Embedded Palmprint Recognition System on Mobile Devices," in *Proceedings of International Conference on Biometrics (ICB)*, 2007, pp. 1184–1193.

[86] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand Keypoint Detection in Single Images Using Multiview Bootstrapping," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1145–1153.

[87] I. Odinaka, P.-H. Lai, A. D. Kaplan, J. A. O'Sullivan, E. J. Sirevaag, and J. W. Rohrbaugh, "ECG Biometric Recognition: A Comparative Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1812–1824, 2012.

[88] M. Li and S. Narayanan, "Robust ECG Biometrics by Fusing Temporal and Cepstral Information," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2010, pp. 1326–1329.

[89] J. Arkko and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)," Tech. Rep., 2006.

[90] J. Arkko, V. Lehtovirta, and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')," Tech. Rep., 2009.

[91] ETSI, "Universal Mobile Telecommunications System (UMTS); LTE; 3G Security; Specification of the MILENAGE algorithm set: An Example Algorithm Set for the 3GPP Authentication and Key Generation Functions f1, f1*, f2, f3, f4, f5 and f5*;," *3GPP TS 35.205 version 10.0.0 Release 10*, 2011.

[92] RSA, "RSA SecurID Hardware Authenticators," https://www.tokenguard.com/RSA-SecurID-Hardware.asp/, 2023.

[93] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," in *Proceedings of Annual International Cryptology Conference (CRYPTO)*, 1996, pp. 1–15.

[94] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "Totp: Time-Based One-Time Password Algorithm," Tech. Rep., 2011.

[95] D. Balfanz, J. Ehrensvard, and J. Lang, "FIDO U2F Raw Message Formats," *FIDO Alliance*, 2017.

[96] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[97] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A Comprehensive Survey," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–36, 2019.

[98] O. Oleksenko, B. Trach, R. Krahn, M. Silberstein, and C. Fetzer, "Varys: Protecting SGX Enclaves from Practical Side-Channel Attacks," in *Proceedings of USENIX Annul Technical Conference (USENIX ATC)*, 2018, pp. 227–240.

[99] J. S. Dwoskin and R. B. Lee, "Hardware-Rooted Trust for Secure Key Management and Transient trust," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2007, pp. 389–400.

[100] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A Secure Database Using SGX," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2018, pp. 264–278.

[101] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "SGXPECTRE: Stealing Intel Secrets from SGX Enclaves via Speculative Execution," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019, pp. 142–157.

[102] R. Maes, V. v. d. Leest, E. v. d. Sluis, and F. Willems, "Secure Key Generation from Biased PUFs," in *Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2015, pp. 517–534.

[103] J.-H. Kim, H.-J. Jo, K.-K. Jo, S.-H. Cho, J.-Y. Chung, and J.-S. Yang, "Reliable and Lightweight PUF-Based Key Generation Using Various Index Voting Architecture," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 352–357.

[104] S. Willassen, "Forensics and the GSM Mobile Telephone System," *International Journal of Digital Evidence*, vol. 2, no. 1, pp. 1–17, 2003.

[105] S. Srinivas, D. Balfanz, E. Tiffany, and A. Czeskis, "Universal 2nd Factor (U2F) Overview," *FIDO Alliance*, 2017.

[106] A. W. Services, "AWS Key Management Service: Developer Guide," https://docs.aws.amazon.com/pdfs/kms/latest/developerguide/kms-dg.pdf, 2023.

[107] S. Agrawal, P. Mohassel, P. Mukherjee, and P. Rindal, "DiSE: Distributed Symmetric-Key Encryption," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2018, pp. 1993–2010.

[108] G. R. Blakley, "Safeguarding Cryptographic Keys," in *Managing Requirements Knowledge, International Workshop on. IEEE Computer Society*, 1979, pp. 313–313.

[109] S. Jarecki, H. Krawczyk, and J. Resch, "Updatable Oblivious Key Management for Storage Systems," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2019, pp. 379–393.

[110] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing or: How to Cope with Perpetual Leakage," in *Proceedings of International Cryptology Conference (CRYPTO)*, 1995, pp. 339–352.

[111] D. A. Schultz, B. Liskov, and M. Liskov, "Mobile Proactive Secret Sharing," in *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*, 2008, pp. 458–458.

[112] R. Vassantlal, E. Alchieri, B. Ferreira, and A. Bessani, "COBRA: Dynamic Proactive Secret Sharing for Confidential BFT Services," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2022, pp. 1528–1528.

[113] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2002, pp. 88–97.

[114] S. Basu, A. Tomescu, I. Abraham, D. Malkhi, M. K. Reiter, and E. G. Sirer, "Efficient Verifiable Secret Sharing with Share Recovery in BFT Protocols," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2019, pp. 2387–2402.

Appendix A PRELIMINARIES

Min-entropy [69]. For adversaries, an attack strategy is guessing random values used in cryptosystems (e.g., the secret keys). The probability of an adversary guessing out the key is determined by the entropy of the key. In information theory, the entropy of a random variable is the level of uncertainty inherent in the random variable's possible outcomes.

Shannon [70] first introduces the concept of information entropy. $A = \{a_1, a_2, \dots, a_n\}$ is a discrete and finite set. Assume that X is a random variable where the value domain is A and the probability of choosing a_i from A is p_i, i.e., $\Pr[X = a_i] = p_i$. The information entropy of X is

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i.$$

The predictability of the random variable X is $\max\{p_i\}_{i\in[1,n]}$, which corresponds to the min-entropy of X is

$$H_\infty(X) = -\log(\max\{p_i\}_{i\in[1,n]}).$$

Shamir's secret sharing [11]. In Shamir's secret sharing scheme, a dealer shares a secret among n parties, and each party has a share. Any t parties can pool their shares and reconstruct the secret. Any $t-1$ parties who collude cannot obtain any information about the secret. Shamir's secret sharing scheme consists of two algorithms, **Split** and **Reconstruction**, which are provided in the following.

Split. A dealer chooses a secret $a \in Z_p$ and split it into n shares.

- The dealer uniformly chooses $a_1, \dots, a_{t-1} \xleftarrow{\$} Z_p$ and generates a polynomial $f(x) = a + a_1 x + \cdots + a_{t-1} x^{t-1}$ over Z_p with degree at most $t-1$.
- The dealer computes shares $s_i = f(i)$ for $i = 1, 2, \dots, n$ and distributes them among n parties, letting each party have a share.

Reconstruction. The dealer recovers the secret a with any t of n shares.

- Upon having t secret shares (denoted by $\{s_{i_1}, \dots, s_{i_t}\}$ for the sake of brevity), the dealer computes $w_{i_l} = \prod_{\substack{1\le j\le t \\ j\neq l}} \frac{i_j}{i_j - i_l}$ for $l = 1, \dots, t$.
- The dealer reconstructs the secret by computing $a = \sum_{l=1}^{t} w_{i_l} s_{i_l}$.

Appendix B COMPARISON WITH SHAMIR'S SECRET SHARING

In this subsection, we elaborate on the key difference between the threshold key derivation mechanism utilized in DRKM and Shamir's secret sharing.

We first abstract the threshold key derivation mechanism from the construction of DKRM (shown in Section 3.5), which consists of **Aggregation** and **Reconstruction**.

Aggregation. Given n^* shares $\{s_1^*, \dots, s_{n^*}^*\}$, a dealer aggregates them into one secret and computes necessary auxiliary information used for reconstruction.

- The dealer computes $a = \prod_{i=1}^{n^*} s_i^*$, where a is the secret by aggregating the shares.
- The dealer computes
$$p(x) = \prod_{i=1}^{n^*} (x - s_i^*),$$
such that $p(s_i^*) = 0$ for $i = 1, 2, \dots, n^*$.
- The dealer outputs the coefficients of $p(x)$ of degree $n^* - 1$ down to t^* as the auxiliary information aux, i.e.,
$$aux = \{-\sum_{i=1}^{n^*} s_i^*, \sum_{\substack{A\subseteq[1,n^*] \\ |A|=2}} (\prod_{i\in A} s_i^*), -\sum_{\substack{A\subseteq[1,n^*] \\ |A|=3}} (\prod_{i\in A} s_i^*),$$
$$\dots, (-1)^{n^*-t^*} \cdot \sum_{\substack{A\subseteq[1,n^*] \\ |A|=n^*-t^*}} (\prod_{i\in A} s_i^*)\}.$$

Reconstruction. Given any t^* of n^* shares and auxiliary information, the dealer can recover the secret.

- Upon having t^* shares (denoted by $\{s_1^*, \dots, s_{t^*}^*\}$ for the sake of brevity), the dealer generates a new polynomial
$$p_1(x) = x^{n^*} + \sum_{i=1}^{n^*-t^*} aux_i \cdot x^{n^*-i},$$
where aux_i denotes the i-th element in aux.
- The dealer solves Equation (3) to obtain $b_{t^*-1}, \cdots, b_0$.
$$\begin{cases} p_1(s_1^*) = & b_0 + b_1 s_1^* + \cdots + b_{t^*-1} s_1^{*\,t^*-1} \\ \quad\cdots & \\ p_1(s_{t^*}^*) = & b_0 + b_1 s_{t^*}^* + \cdots + b_{t^*-1} s_{t^*}^{*\,t^*-1} \end{cases} \tag{3}$$
$|b_0|$ is the reconstructed secret.

From the perspective of construction, the threshold key derivation mechanism contains an **Aggragation** algorithm instead of a **Split** one as in Shamir's secret sharing (provided in Appendix A). This is because the "shares" in the former are pre-determined and aggregated into one secret, while in the latter, the secret is first chosen, and the "shares" are determined by the secret.

Appendix C POPULAR PIF INTRODUCTION

Generally, users' personal identification factors (PIFs) can be divided into three categories: something the user knows, someone the user is, and something the user has [30].

C.1 Something the user knows

This type of PIFs binds a user's identity with a secret only known to the user. As long as the user outputs the correct secrets, she/he can identify herself/himself.

• *Password.* A password is a character string chosen by a user and can be utilized to construct a portable authentication scheme, where the user takes her/his password as the sole input for authentication [62]. Password is memorable and thereby is destruction-resistant. Generally, a cryptographic hash function is used to "obfuscate" the password, and the hashed password serves as the authentication credential. However, the password is inherently low-entropy and is vulnerable to dictionary guessing attacks (DGA) [23, 71, 72].

• *PIN code.* A personal identification number (PIN) code is a set of numbers (which generally consists of four or six digits) generated by the user. PIN codes' entropy is much lower than passwords. Therefore, PIN codes should not be utilized to serve as the sole PIF in the system [73].

• *Private key.* In public-key cryptosystems, a private key is a string uniformly chosen from some set (where the number of elements in the set should be large enough) and is used to generate a corresponding public key. With the pair of keys, an identification scheme can be constructed (e.g., the Schnorr identification scheme [74]). In reality, a private key is always stored in a stateful device or hardware security module (as it is high-entropy), and thereby cannot be recovered if the device (or hardware module) is destroyed.

C.2 Someone the user is

This type of PIFs is essentially the characteristics that can uniquely identify the user. The most widely used PIFs are users' biometric characteristics. Such a PIF is not necessarily maintained in some device. As a consequence, most of these PIFs (discussed in this paper) are destruction-resistant.

A general procedure of biometric characteristic recognition is described as follows. A user collects biometric characteristics with the aid of stateful devices (e.g., obtaining a face image by using a camera), and then some features can be extracted from the characteristics. With the features, some templates can be derived to serve as authentication credentials. We investigate popular biometric PIFs and feature extraction methods. Subsequently, a "fresh" template can be extracted from the user's biometric characteristics to be compared with the pre-generated templates for user authentication. The widely-used biometric characteristics, e.g., fingerprints and faces, can be easily collected using smartphones. With the proliferation of wearable devices, it is also convenient to collect other biometric characteristics, e.g., ECGs [22]. Typical biometric characteristics are introduced in the following.

• *Fingerprint.* Fingerprints are the most widely utilized biometric characteristic for user authentication [75]. A fingerprint can be transformed into a unique image with ridges and valleys, where a ridge is a single curved segment, and a valley is a region between two adjacent ridges, and minutiae-based feature extraction is the most commonly-used method in fingerprint recognition [19, 20, 40, 76, 77]. For users, it is convenient to utilize fingerprints to authenticate herself/himself, since most devices have fingerprint recognition functions. Fingerprints serving as PIF achieve compromise resilience, since the user can use different fingerprints in different systems. However, it is not so hard to collect targeted users' fingerprints for adversaries in the physical world [78].

• *Face.* Facial recognition is also widely used for user authentication. As facial recognition only requires a camera to collect the face image, most smartphones utilize the face as the PIF. In practice, extraction methods based on either geometrical features or statistical features [21, 79] can be utilized to extract features from users' faces. Compared with fingerprints, users' faces fail to achieve compromise resilience.

• *Iris.* An iris camera captures a user's pupil and extracts the iris image from the pupil by using Gabor filters [80]. Compared with fingerprint-based and face-based authentication, iris-based authentication is more accurate for identifying users [81, 82]. Although irises also fail to achieve compromise resilience, collecting users' irises is much harder than collecting fingerprints and faces for adversaries. However, iris recognition requires a specific-purpose device to scan the iris, which causes high costs to deploy the iris-based authentication.

• *Hand.* In reality, the physical characteristics of a hand (e.g., hand geometry [83, 84], palmprints [85], and hand gestures [86]) can also serve as PIFs for user authentication, where hand geometry and hand gestures can be utilized to achieve non-contact recognition. The accuracy rate is significantly affected by the environment, e.g., lighting, and the devices collecting palmprints are generally larger, compared with those collecting fingerprints and faces.

• *Electrocardiographic (ECG).* An ECG records the electrical activity of the heart and mainly consists of P wave, QRS complex, and T wave. ECG is highly personalized and can serve as PIF [87, 88]. ECG-based authentication requires specific types of equipment, e.g., ECG monitors.

C.3 Something the user has

This type of PIFs binds a user's identity with a stateful device or hardware unit. Anyone who possesses the device (or the unit) can pass authentication.

• *Subscriber identity module (SIM) card.* A SIM card is an integrated circuit card, that has been utilized in cellular networks, e.g. 3G and 4G [31]. The authentication protocol is based on message authentication code (MAC) [89, 90]. Specifically, a SIM card stores a universal international mobile subscriber identity (IMSI) number and a 128-bit key which is shared with a server (deployed by the communication service provider). The user and the server initialize the same sequence number SQN. A typical SIM-based authentication procedure is shown in Figure 1, which follows a challenge-response paradigm. When subscribing to cellular networks, the user sends her/his IMSI number as ID to the server. The server first generates a challenge message, including a random number $RAND$ and a MAC for $RAND$ and SQN, and sends it to the user. The user then verifies the received MAC and generates a response message by computing a MAC for $RAND$. The server finally verifies the MAC and allows the user to subscribe if it is valid. In Figure 1, the MAC is based on pseudorandom functions $f1(\cdot)$, $f2(\cdot)$, and $f5(\cdot)$, and are instantiated by AES in practice [91].

With the proliferation of mobile devices, e.g., smartphones, SIM cards are widely used for authentication in daily life, which is convenient for users. This also implies that the utilization of SIM cards has to rely on devices with the cellular communication module.

• *Hardware token.* Hardware tokens can be divided into two categories from the point of the underlying cryptosystem: symmetric-key-based ones and public-key-based ones.

For symmetric-key-based hardware tokens, typical examples include RSA SecurID [92], which is constructed on a hash-based MAC (HMAC) [93]. A simplified authentication procedure is shown in Figure 2. The hardware token stores a seed that is shared with a server. During the sign-on phase, the hardware token and the server invoke a one-time password algorithm (which is based on an HMAC) using the seed and current time as input [94]. The user sends the password generated by the hardware token to the server. The server verifies whether the received password is the same as the locally generated one. If so, the user passes the authentication.

For public-key-based hardware tokens, typical examples include U2F token [33, 95], which is based on ECDSA [96]. Figure 3 shows the authentication procedure. The hardware token stores a private key, and the corresponding public key is stored on the server. In the sign-on phase, the server sends a nonce *Nonce* as a challenge to the hardware token, and the hardware token signs *Nonce* and returns the signature to the server. The server verifies the signature with the corresponding public key. If it is valid, the user passes the authentication.

It is difficult to forge a valid hardware token. However, once an adversary can physically access the hardware token, he can easily impersonate the user to pass the authentication.

• *Trusted execution environment (TEE).* TEEs are isolated private enclaves inside CPU, which are used to protect data. Intel Software Guard Extensions (Intel SGX) is the most widely-used TEE architecture [34]. It can also be utilized to authenticate users by using its two root keys: the root provisioning key (RPK) and the root seal key (RSK), where RPK and RSK are fused in CPU by the manufacturer. Due to the space limitation, please refer to Ref. [34–36] for more details. In addition, except Intel SGX, other TEEs, e.g., ARM TrustZone [97], are also widely used for authentication.

Such hardware units rely on specific manufacturers and are not easily forged, but they may suffer from side-channel attacks [98]. Users and servers have to equip the same TEEs.

Appendix D RELATED WORK

D.1 Key management schemes

Many key management schemes have been proposed in the past few years. We discuss them in the following.

Hardware-based key management schemes. Users can locally manage their keys by utilizing secure hardware devices [99]. For example, Intel Software Guard Extensions (SGX), aiming to protect the confidentiality and integrity of computations on sensitive data performed on a computer, can also be utilized for key management. Priebe et al. proposed a secure database using SGX [100], which can be utilized for key management. However, such a scheme requires the user to possess a device equipped with Intel central processing units (CPUs) that support Intel SGX. In addition, Intel SGX is vulnerable to side-channel attacks [98, 101].

Another approach to key management is to generate keys based on the physical characteristics of a hardware device rather than storing them within it. For example, although the manufacturing process is the same among different ICs, each IC is actually different from others, which is called manufacturing variability and can be utilized for key generation. Maes et al. proposed physical unclonable functions (PUFs) to derive keys by leveraging the variability [102]. Kim et al. proposed a lightweight PUF-based key generation using various index voting architecture [103].

Hardware-based key management schemes essentially shift the problem of managing multiple keys from protecting each individual key to protecting another key that can unlock them. Once an adversary physically accesses the hardware devices, he may recover the user's keys. To resist such an adversary, a widely-used remedy is to introduce an additional authentication mechanism: only the user who passes the authentication can utilize the hardware devices. For example, before using a SIM card, the user needs to input a PIN code to unlock it [104]. The user needs to input the correct password before utilizing the hardware token for authentication [105]. MacOS provides a secure container, called Keychain, which assists users in managing their keys and passwords [5]. MacOS authenticates the users by using passwords, fingerprints, or faces, and only authenticated users can access the Keychain.

Whereas, the functionality and security of hardware-based key management schemes depend on the reliability of the hardware devices. Once the hardware devices are destroyed (e.g., due to wrong formatting), the user would never recover her/his keys. Especially, the hardware devices are individually maintained by the users and are vulnerable to being lost or stolen. In addition, hardware-based key management schemes fail to achieve portable key access.

Software (or extended services) based key management schemes. To free the costs and issues introduced by local key management, users prefer to employ a service provider to achieve key management. For instance, Amazon provides key management services for users [106], where a master key is generated by each user and utilized to encrypt other keys. The master key and the ciphertexts of other keys are well maintained by Amazon. Microsoft Azure [1] and Google cloud platform (GCP) [2] provide the same key management services for users. With the assistance of the service providers, the users are able to manage and access their keys on any device. Nevertheless, for these key management services, service providers rely on hardware security modules (HSM) to manage users' master keys, e.g., Amazon utilizes a distributed fleet of FIPS 140-2 validated HSMs to securely manage the users' master keys [106]. Once the HSMs are destroyed for some reason (e.g., servers are destroyed due to geological disasters), the users' master keys would no longer be recovered.

To eliminate the reliance on security hardware devices, server-aided key management schemes have been proposed in the past few years. A user utilizes a master key to encrypt other keys, and the master key is split among key servers in a threshold way [107, 108]. As long as the number of available key servers is larger than the threshold, the user can recover the key. An adversary who compromises less than a threshold number of key servers cannot obtain anything about the master key. A notably example is Keywhiz, an open-source distributed key management software [58].

To achieve portability, password-hardening protocols [59–66] can also be utilized for key management, where a user hardens her/his password with the aid of the key servers in a threshold and oblivious way. After being hardened, the password is secure against dictionary guessing attacks (DGA), which enhances the security significantly. The hardened password is utilized to compute authentication credentials and generate the master key. Such a password-hardening-based key management scheme is destruction-resistant, since the password is memorable, and any destruction of the user's devices would not cause unavailability of the password. Whereas, it is still unsatisfactory in real-world deployment, since the password serves as the sole secret. Once the password is compromised, no security is guaranteed. In reality, compromising users' passwords is not so hard for a sophisticated adversary, even if they are hardened by the key servers [23, 71, 72].

In addition to directly sharing the master key among multiple key servers, distributed encryption schemes have been proposed as a variant of distributed key management schemes. Agrawal et al. proposed the first formal threshold symmetric-key encryption based on distributed pseudorandom functions in DiSE [107], where the user employs a group of key servers to generate an encryption key for each message. Specifically, the user generates a commitment to a message and sends the commitment to each server, and each server computes a response utilizing its key share for the user. The user aggregates a threshold number of responses and gets the message-specific key to encrypt the message. However, DiSE may introduce heavy computation and communication costs to the user when encrypting a large set of messages. Christodorescu et al. proposed an amortized threshold symmetric-key encryption scheme [6], which enables the user to encrypt a large set of messages using a single interaction. In addition, Jarecki et al. constructed an oblivious key management system based on oblivious pseudorandom functions [109]. This scheme has an updatable encryption capability, and the update procedure of the ciphertexts does not require the user to interact with the key servers.

Whereas, in the above schemes, sophisticated adversaries may corrupt enough key servers given enough time and get the users' master key. To resist such adversaries, multiple proactivization mechanisms for secret sharing have been proposed [12, 110–114]. In these schemes, time is divided into fixed intervals called epochs, and the secret shares are updated with the new ones without changing the master key in different epochs. To further enhance security, the key servers can be replaced by the newly employed ones in different epochs while maintaining the master key.

These server-aided key management schemes are free from reliance on secure hardware devices. However, these schemes cannot resist the destruction of key servers. The functionality and reliability totally rely on that the key servers can provide services for users. Once the key servers are destroyed, the user would never recover their keys.

D.2 Portable authentication

Biometric characteristics provide a convenient and portable way for users to authenticate themselves with servers. Despite the benefits of using biometric characteristics, there exist security issues. Specifically, storing a user's biometric characteristic template on the server side as an authentication credential makes the user's biometric characteristic vulnerable if the credential database is leaked. Furthermore, an attacker may perform trawling attacks to obtain authentication credentials based on different biometric characteristics from different authentication systems. This can enable the attacker to retrieve the user's master key if the user utilizes the same characteristics when deploying DRKM.

A method to mitigate the above attacks is to require the server to store the encrypted templates. The adversary cannot get anything about users' biometric characteristics from the compromised credential database. However, this method cannot resist the internal adversary e.g., the malicious insiders working at the server, who can still get the users' templates.

Recently, the authentication protocol, FIDO U2F [105], has been proposed, which enables users' biometric templates to be stored on the users' devices instead of the server side. Specifically, the user installs an FIDO authenticator on her/his device. During the registration phase, the user unlocks the FIDO authenticator using fingerprint or other biometric characteristics and generates a new public/private key pair. The public key is sent to the server, and the private key and the biometric characteristic templates are stored on the device. During the sign-on phase, the server sends a challenge message, e.g., a nonce, to the user. The user unlocks the FIDO authenticator using the same biometric characteristics as that utilized in the registration phase and signs the challenge message using the private key. The user sends the signature back to the server, and the server verifies it with the stored public key. If it is valid, the user passes the authentication. As such, deploying FIDO authenticators can avoid the leakages of biometric information.

Appendix E ARTIFACT

Abstract. Our artifact consists of a DRKM prototype. DRKM is a portable and destruction-resistant key management system. It can support that a user derives a master key from multiple PIFs and utilizes the master key to manage other cryptographic keys. DRKM also supports the user to retrieve the master key from a part of PIFs utilized for key derivation.

Scope. Our artifact can be used to prove the correctness and feasibility of DRKM and evaluate its performance. Specifically, it demonstrates that DRKM can be deployed in practice and function well. It can be used to evaluate the computation delay and communication costs. It can also be used to validate the evaluation results presented in Section 5.

Content. The artifact comprises the following sub-directories:

-__pycache__, which contains the packaged interfaces.

-key_manager.py, which contains the sourcecode of **Managing**.

-key_recover.py, which contains the sourcecode of **Recovery**.

-key_visit.py, which contains the sourcecode of **Access**.

Hosting. Our artifact is available on the GitHub repository https://github.com/DRKM-code/DRKM.git.

Requirements. We developed and evaluated our artifact on a laptop with an Intel Core i5 CPU and 16 GB LPDDR4X of RAM. The prototype is implemented in Python with the Crypto library. Moreover, to run the prototype correctly, some basic packages including mpmath, pip, pkg_resources, and sympy are required.