Non-Interactive Distributed Key Management with Pre-Determined Shares

Yaqing Song[†], Yuan Zhang[†*], Shiyu Li[‡], Weijia Li[†], Zeqi Lai[§], Qiang Tang[¶]
[†]University of Electronic Science and Technology of China, China
yaqings@163.com, zhangyuan@uestc.edu.cn, tokio_0@163.com
[‡]The University of Virginia, America, tkp2ra@virginia.edu
[§]Tsinghua University, China, zeqilai@tsinghua.edu.cn
[¶]The University of Sydney, Australia, qiang.tang@sydney.edu.au
[*]Corresponding author

***Abstract*—In distributed key management systems, a key can be generated in a centralized or distributed way. However, centralized schemes (e.g., Shamir's secret sharing) require a trusted dealer to perform a private setup to generate a key, which suffers from the single-point-of-failure problem; distributed schemes (e.g., distributed key generation, DKG) require a set of members (who form a committee) to interact with each other to generate the key, which introduces considerable communication costs. Furthermore, neither of them supports pre-determined shares. In this paper, we propose a Non-Interactive Distributed Key Management scheme, dubbed NIDKM, which enables members to generate a key and then recover it with a majority of them in a non-interactive way. We stress that NIDKM is different from existing threshold secret sharing schemes, but the threshold property remains: in NIDKM, the shares are first determined, and the key is derived from the shares; in existing schemes, the key is first generated, and then the shares are computed from the key. In addition, NIDKM does not require the members to interact with each other, which achieves $O(1)$ communication overhead on the member side. We formally prove NIDKM's security, implement its prototype, and conduct a comprehensive performance evaluation to demonstrate its high efficiency.**

***Index Terms*—Key management, Non-interaction, Pre-determined share**

I. Introduction

Highly-valuable assets critically rely on the security protection provided by cryptosystems, where the security of assets is ultimately reduced to the security of the underlying (small) key. Consider a bank vault's physical door controlled by a board of members, who jointly generate and maintain the access key as a credential to open the door. Initially, each member contributes one secret (including but not limited to pressing fingerprints, scanning irises, or inputting long passwords) to generate the key and "register" with the door on-site. Subsequently, a majority of members who submit the original secrets can open the door.

For the above scenario, we need a *distributed* key management mechanism with a threshold property, where each member holds a piece of secret share, and the key can only be reproduced if sufficient members participate. A natural solution is threshold secret sharing with a private setup, where a server uniformly chooses a key and runs (t, n)-threshold secret sharing to compute n shares with a threshold t. Each member maintains a share such that the key can be recovered with any t shares. However, the key is determined by the server alone (which naturally introduces single-point-of-failure problems). Recent incidents show that attackers can bias randomness in hardware or software vulnerabilities to further compromise keys [1]. Notable examples include backdoor attacks [2].

To eliminate reliance on a trusted dealer, a remedy is to utilize distributed key generation (DKG) protocols [3], where for registration, each of n members contributes a secret, and they jointly generate a shared public/private key pair without any single member learning the full private key, and each member finally obtains a share of the key determined by their inputs; during the usage phase, any t members can recover the key with their shares. However, DKG is also unsatisfactory for the scenario considered above due to the following reasons.

The shares maintained by members are not pre-determined. With DKG, the final shares that the members need to keep are completely different from the secrets they initially provided. By contrast, members expect that they only need to provide the secrets as they input during the registration phase for reconstruction. For example, members expect to re-enter the same passwords or present the same biometric features, rather than manage additional cryptographic shares. We refer to this requirement as *pre-determined shares*: the shares are fixed in advance by the members' original secrets and are not generated or changed by the protocol.

Another motivation to keep pre-determined shares for members is to prevent key unavailability due to device destruction in disastrous situations. When running a DKG, members have to utilize devices (e.g., hardware tokens or computers) to store their shares, e.g., YubiKey [4]. Nevertheless, a critical issue—vulnerability against device destruction—arises naturally: as uncontrollable and unpredictable threats towards hardware devices always exist in reality, members have to bear the risk that their shares would be permanently lost if the device is destroyed. Notably, a private key stored in a hard drive was permanently unavailable to its owner due to a hard drive breakdown, which directly caused the 7500 bitcoins (which are worth more than $280 million today) could never be used by the owner [5]. In addition, severe natural disasters would also directly destroy all local devices and make them permanently

unavailable to their users. A notable example is that the eruption of the Tonga volcano in 2022 destroyed critical information infrastructure almost all over the country [6]. Pre-determined shares avoid this risk by eliminating the need for persistent storage of newly generated key material.

DKG is expensive and requires members to interact with each other. Deploying DKG would incur considerable communication and computation overhead, since it requires building a broadcast channel to enable the members to interact with each other. Broadcast channels are actually expensive and not easy to implement in practice, especially when there are malicious participants (latest progress in DKG protocols strive to minimize the broadcast overhead [7], [8]. DKG protocols also require members to conduct expensive cryptographic operations (e.g., certain NIZK proofs) to ensure security [3], [9]. Whereas, in the application scenario we mentioned above, for registration, members can participate on-site and offer their biometrics or passwords using the equipment installed at the vault door. They neither possess nor are expected to maintain additional devices, making interaction among members impractical. Consequently, non-interaction is a fundamental requirement of the setting.

This motivates us to consider the following question:

Can we have non-interactive distributed key management with pre-determined shares while preserving the threshold property?

The key observation behind our work is that there is a natural "server" in the above application, e.g., the door of the bank vaults that controls access, which may remove the need of interactions as in DKG protocols.[1] With the assistance of a server, one may wonder whether a non-interactive distributed key management scheme with pre-determined shares can be trivially constructed: regarding registration, the server collects the secrets from members and then derives a key from them; and for usage, the server can derive the same key from the original secrets. However, the above scheme lacks the threshold property. Either conventional threshold secret sharing or DKG essentially shares the same paradigm: first determine the secret and then split it into multiple shares; any threshold number of shares can reconstruct the secret. Whereas, we expect that the "shares" (i.e., the members' secrets) are pre-determined by members, and the key is then derived from them. As a consequence, both threshold secret sharing and DKG fail to achieve the key recovery from a subset of original secrets, i.e., remain the threshold property. This is the key challenge we address: how to achieve threshold key derivation, such that *the key can be "derived" from all pre-determined "shares" but can be recovered with only the threshold number of them*.

In addition, we show the viability of non-interactive distributed key management for different application scenarios. For instance, it can be utilized to manage the keys of cold wallets. In practice, due to the high value of keys (e.g., the private keys of enterprises, or cryptocurrency exchanges such as Coinbase [10]), they are always backed up by utilizing offline devices to ensure key recovery under some unpredictable circumstances. By deploying the non-interactive distributed key management, the members can generate a key from their secrets that are derived from their biometric features, where the device can be treated as a *server*; subsequently, the key can be recovered with the original secrets by the device. Another example is that such a distributed key management with pre-determined shares can be extended to manage personal keys while achieving destruction resistance and portability. We give more details on how to deploy or extend under these two scenarios in Section IV-D.

[1]We stress that the natural server is totally different from the aforementioned server that executes a private setup. It would not actively engage in malicious behavior, such as opening the vault door without recovering the correct key, as the door is responsible for ensuring the vault's safety.

A. Our contributions

In this paper, we propose a Non-Interactive Distributed Key Management scheme with per-deterministic shares, dubbed NIDKM, which goes one step beyond existing secret sharing and distributed key generation schemes. Specifically, our contributions are summarized as follows.

New concepts about PIFs and secrets. NIDKM assumes that members determine their own secrets, which can be reconstructed without relying on persistent storage. To this end, we 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. Based on these new concepts, we introduce two concepts about secrets—reconstructable secrets and unreconstructable 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 of NIDKM. Each member derives a secret from their PIFs (which include storage/device-independent ones and might include device-dependent ones). NIDKM uses a (t^*, n^*)-threshold key derivation mechanism to derive a master key from the pre-determined secrets of n^* members while generating auxiliary information. There are two methods to recover the key: 1) the key can be recovered from any t^* of the n^* secrets with auxiliary information; 2) when auxiliary information is not available, the key can be directly recovered from n^* original secrets. In addition, we present two approaches to compute the auxiliary information.

We further extend NIDKM to support destruction-resistant and portable key management for personal keys. Specifically, a user can derive n^* secrets from their PIFs during the setup phase, then generate a master key from them, and recover the key with any t^* of the n^* PIFs after destruction occurs. To achieve portability, NIDKM adopts a multi-server-aided paradigm and uses a (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, users can access the master key portably (i.e., they do not need to maintain any secret in local devices).

Formal security proofs and prototype implementation. We provide formal security definitions of NIDKM and prove its security. We prove that an adversary, who compromises $t-1$ shares (generated by conventional threshold secret sharing) and t^*-1 original secrets, cannot get any extra information about the key. We implement an NIDKM prototype and conduct a comprehensive performance evaluation.

We demonstrate the viability of NIDKM for two existing applications that can benefit from the desirable property of destruction resistance in Section VI-D. One of two applications extends to destruction-resistant cryptosystems (DRC). We show how NIDKM supports these applications without changing the current system architecture. Since destruction resistance is a fundamental requirement of any cryptosystem, we believe that NIDKM has further useful applications.

B. Technical overview

We give a technical overview of NIDKM, focusing on the challenges addressed by our work.

Achieving the threshold property for pre-determined shares. Each of the n^* member can derive a reconstructable secret from their PIFs, and then a server aggregates these secrets to obtain a key. The challenge in designing NIDKM is to achieve threshold retrieval for the aggregated key, i.e., a 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, upon receiving n^* secrets from members, the server constructs a n^*-degree polynomial $p(x)$ using the n^* secrets as its roots, and the coefficients of $p(x)$ of degree n^*-1 down to t^* or n^*-t^* negative evaluation points 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 server can compute the coefficients of $p(x)$ from degree t^*-1 down to 0, then find all roots of $p(x)$, and then recover the master from these roots, i.e., the secrets.

Extending to portable key management for personal keys. We also integrate an aggregation-then-split mechanism into NIDKM to achieve portable key access for personal key management. 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 obtain any information about the reconstructable secrets from the secret shares. 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 are unavailable, the user can also recover the master key from n^* original PIFs. A destruction-resistant cryptosystem (DRC) can be developed by directly integrating NIDKM and a full-fledged backup system.

II. Related work

A. Distributed key management

In general, distributed key management (DKM) is built upon threshold secret sharing [11], [12], where a key is split among n members, and any $t \leq n$ shares can reconstruct the key. Users can recover the key as long as the number of available servers exceeds the threshold. Notable implementations include Keywhiz [13]. Many secret sharing schemes with additional features have been proposed. For example, to mitigate long-term compromises, proactive threshold secret sharing schemes have been proposed [14], [15], where shares are periodically refreshed without changing the underlying secret. Other works further enhance robustness by supporting verifiability [16], [17] or error correction [18].

As a fundamental component, DKM has been extended to support various upper-layer cryptographic functionalities. Notably, DKM can be integrated with password hardening protocols [19], [20], [21], [22], [23], [24], [25], [26], [27] to enable users to derive cryptographic keys from their passwords in a portable way. DKM can also be extended into distributed cryptosystems [28]. For example, it can be extended to threshold symmetric-key encryption [29], [30]. Despite these improvements, these threshold sharing schemes follow the same paradigm: the key is first fixed, and then the shares are computed. Recently, Kate et al. [31] provide a key recovery mechanism, where a set of guardians can help a user back up keys without storing anything beyond their own secrets. It partially overlaps with NIDKM's goals. However, in [31], the key is generated by the key owner and subsequently protected by guardians, whereas in NIDKM, the master key is *derived from pre-determined secrets held by multiple members*, and the key can also be reconstructed from the secrets if all extra storage fails.

B. Comparison with concurrent work

A recent work concurrent to NIDKM (i.e., threshold multi-factor key derivation function, short for TMFKDF) proposed by Nair et al. [32] could be a partial solution to construct a non-interactive distributed key management scheme: a server first randomly chooses a key, splits it into multiple shares using a conventional threshold secret sharing scheme, and encrypts each share under a secret determined by each member.

Essentially, TMFKDF inherits the threshold property of the conventional threshold secret sharing scheme, where the master key is generated via a private setup by the server, rather than determined by the members' secrets. Therefore, some metadata, e.g., the encrypted shares, is inherently needed for key recovery, and dedicated storage for metadata is always required. By comparison, NIDKM is completely orthogonal to the conventional sample-share-and-reconstruct idea as in TMFKDF. With NIDKM, the key can be recovered from n^* reconstructable secrets even if any storage is unavailable. In NIDKM, it seems that some metadata is also required in some cases. For instance, if the user derives reconstructable secrets from biometric characteristics [33], some metadata, e.g., error

correcting code [34], is required to ensure the consistency of secrets in different derivations. However, we stress that such metadata in NIDKM is totally different from the ciphertext in TMFKDF due to the following reasons.

The metadata in NIDKM 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 members 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 be free from metadata in NIDKM by utilizing new PIFs, where deriving secrets from them does not require any storage. A detailed comparison with TMFKDF and other methods is provided in Section VI-C.

III. PIFs and reconstructable secrets

A. Preliminaries

Notation. ℓ denotes the security parameter, and $\vec{A}$ 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. $a||b$ denotes the concatenation of two strings a, b. Z_p denotes the set $\{0, 1, \ldots, p-1\}$, where p is a prime.

Min-entropy [35]. For adversaries, an attack strategy is guessing random values used in cryptosystems (e.g., 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 [35] first introduced the concept of information entropy. $A = \{a_1, a_2, \ldots, a_n\}$ 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 $\max\{p_i\}_{i\in[1,n]}$ that corresponds to the min-entropy of X is $H_\infty(X) = -\log_2(\max\{p_i\}_{i\in[1,n]})$.

Key derivation function (KDF). A key derivation function can be utilized to derive a cryptographic key from a random source. We follow the security definition of KDF proposed in [36], as shown in Definition 1.

Definition 1: A 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 than $1/2 + 2^{-m}$.

B. Definitions of PIFs

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, some auxiliary information is required, which would include the directions for use and necessary descriptions. According to the definition of the factor in [32], we formally define a general PIF and its min-entropy as follows.

Definition 2: A PIF is a triple-valued probability distribution (U, A, State) which is generated by an efficient probabilistic algorithm.

Definition 3: A PIF has m-min-entropy if for $\forall(\mu, \alpha, \mathsf{sta})$ which satisfies its probability distribution, $\Pr[U = \mu | A = \alpha, \mathsf{State} = \mathsf{sta}] \leq 2^{-m}$.

For a PIF instance $(\mu, \alpha, \mathsf{sta})$, μ is a private value, and α is public; sta would be public or private, depending on specific PIFs. For clarity, we take passwords, SIM cards [37], and biometric characteristics as examples. For a password, μ is the password, α would be some public information, e.g., the strength requirements, and $\mathsf{sta} = \perp$, since passwords are human-memorable and independent of any extra stateful information. For a SIM card, μ is the secret key fused in it, α includes some public parameters and authentication protocols that are utilized, and sta represents some private *stateful* information (which can be extracted from the SIM card). For a biometric characteristic, μ is the encoded biometric feature, α would contain the recognition and encoding methods, and sta represents some public information (which can be extracted from dedicated equipment, e.g., one used to collect biometric features). With the previous analyses, we can heuristically divide the above PIFs into three categories as follows.

Storage-independent PIFs. If a PIF (U, A, State) satisfies $\mathsf{State} = \perp$, then it is storage-independent. In reality, passwords and PIN codes are storage-independent and can be reconstructed from users' memory and are inherently independent of any personal or public storage.

Device-independent PIFs. If a PIF (U, A, State) satisfies that State is public, then it is device-independent. We observe that biometric characteristics are device-independent, as they are determined by a user per se. However, they may not be totally independent of storage, since some public information, e.g., error correcting code [34], is needed when utilizing them. In cryptographic applications, how to make biometric characteristics free from dedicated storage is a fascinating open problem.

Device-dependent PIFs. If a PIF (U, A, State) satisfies that State is private, then it is device-dependent. SIM cards and hardware tokens are device-dependent PIFs, which rely on specific devices. For device-dependent PIFs (e.g., SIM cards, hardware tokens), once a stateful device is destroyed, the corresponding PIF cannot be recovered by the user per se.

Obviously, storage-independent PIFs are also device-independent, but device-independent ones need public information for reconstruction and are dependent on public storage.

C. 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. We can trivially extend reconstructable values to *reconstructable secrets* by further requiring the value to be kept secret. Similarly, a secret derived from an un-reconstructable value is un-reconstructable.

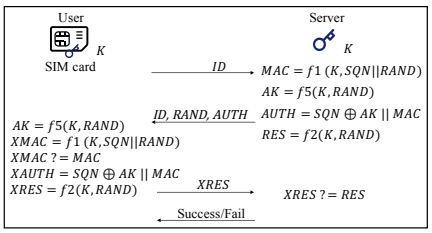


Figure 1: User authentication based on SIM cards.

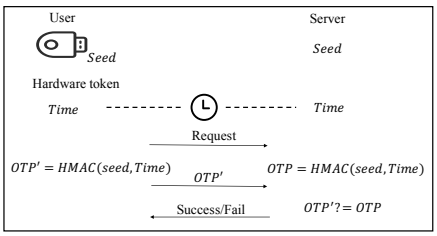


Figure 2: User authentication based on RSA SecurIDs.

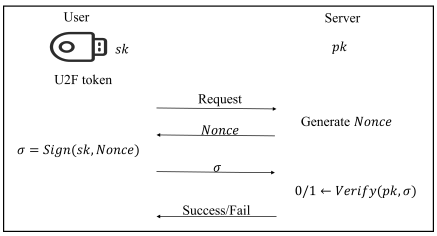


Figure 3: User authentication based on U2F tokens.

In reality, we can utilize storage/device-independent PIFs to serve as reconstructable values to derive reconstructable secrets. For instance, fuzzy extractor algorithms [34] 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.

D. 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, 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 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.

SIM card. Figure 1 shows the authentication protocol based on SIM card, where SQN is the sequence number of the SIM card, $RAND$ is a random number, f_1, f_2 are message authentication functions, and f_5 is a key generation function (referring to [38] for more details). 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 [2], where this approach actually does not need the assistance from the manufacturer [39].

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 [40]. 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 get the secret by calling the *EGETKEY* instruction.

Limitations of the 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.

IV. The proposed NIDKM

A. Definition of NIDKM

There are three algorithms in NIDKM, **Setup**, **KeyGen**, and **Recovery**. To support more general application scenarios, there are also two optional algorithms, **OutSource** and **Access**.

- $PP \leftarrow$ **Setup**(ℓ).

On input the security parameter ℓ, it returns public parameters PP, where two thresholds, i.e., (t, n) and (t^*, n^*) are included. (t, n) is independent of (t^*, n^*).

- $\{S, aux\} \leftarrow$ **KeyGen**$(PIF_1, \ldots, PIF_{n^*})$.

On input n^* PIFs $\{PIF_1, \ldots, PIF_{n^*}\}$, this algorithm returns a key S and public auxiliary information aux.

- $S \leftarrow$ **Recovery**$(PIF_{i_1}, \ldots, PIF_{i_{t^*}}, aux)$.

On input t^* PIFs $\{PIF_{i_1}, \ldots, PIF_{i_{t^*}}\}$ and aux, it reproduces the master key S.

- $\{s_1, \ldots, s_n\} \leftarrow$ **Outsource**(S). (Optional)

On input the key S generated by the algorithm **KeyGen**, it outputs n shares of S.

- $S \leftarrow$ **Access**$(s_{i_1}, \ldots, s_{i_t})$. (Optional)

On input any t of n shares $\{s_{i_1}, \ldots, s_{i_t}\}$, it returns S.

B. Security of NIDKM

The security goals of NIDKM are as follows. 1) Security: an adversary who adaptively compromises up to $t-1$ shares and up to t^*-1 PIFs would not get any additional information about the derived key. 2) Robustness: any t^* PIFs can reconstruct the key. The security property of (t, t^*, n, n^*)-NIDKM is captured by Definition 4 and 5.

Definition 4: Let λ be the security parameter. NIDKM is secure if any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that adaptively compromises at most $t-1$ of the n shares and at most t^*-1 of the n^* secrets, the output key by the **KeyGen** algorithm is computationally indistinguishable from a uniformly-chosen key from $\{0,1\}^\ell$ for $\mathcal{A}$.

Definition 5: NIDKM is robust, if for any set of at least t^* valid secrets derived from the corresponding PIFs, the **KeyGen** algorithm deterministically reconstructs the same key using the auxiliary information aux, except with negligible probability.

C. Construction of NIDKM

$PP \leftarrow$ **Setup**(ℓ).

With the security parameter ℓ, the parameter $PP = \{p, \mathsf{F}, \mathsf{H}, t^*, t, n^*, n\}$ is determined and published, where p is a prime with $\log p \geq \ell$, F is a secure key derivation function,[2] $\mathsf{H}: \{0,1\}^* \rightarrow Z_p$ is a secure hash function, and (t^*, n^*) and (t, n) are two pairs of threshold. n^* is the number of PIFs.

$\{S, aux\} \leftarrow$ **KeyGen**$(PIF_1, \ldots, PIF_{n^*})$.

- Derive n^* secrets from n^* PIF instances $\overrightarrow{PIF} = \{PIF_1, \ldots, PIF_{n^*}\}$ respectively, where these PIFs' distributions are independent. We require that the minimum among the min-entropy values of all PIFs used for key management is not less than ℓ.[3] For $PIF_i \in \overrightarrow{PIF}$ and $i = 1, 2, \ldots, n^*$, if PIF_i is storage/device-independent, a reconstructable secret can be computed from it, and if PIF_i is device-dependent, a conditionally reconstructable secret can be derived from it. The secret can be computed by $seed_i = \mathsf{F}(PIF_i)$ for $i = 1, 2, \ldots, n^*$.
- Compute $s_i^* = seed_i \mod p$ for $i = 1, 2, \ldots, n^*$.
- Generate a polynomial $p(x) = \prod_{i=1}^{n^*}(x - s_i^*)$ over Z_p, such that $p(s_i^*) = 0$ for $i = 1, 2, \ldots, n^*$.
- Compute $S = \mathsf{H}(\mathsf{Sort}(s_1^*, s_2^*, \ldots, s_{n^*}^*))$ as the master key, where $\mathsf{Sort}(\cdot)$ denotes an ascending/descending ordering to ensure the consistency of the derived key.

We present two approaches to generate the public auxiliary information aux.

Approach 1: Compute coefficients. Output the coefficients of $p(x)$ from degree n^*-1 down to t^*, in descending order of degree, as the auxiliary information aux over $\mathbb{Z}_p$.

Approach 2: Compute negative evaluation points. Output $n^* - t^*$ negative evaluation points, i.e., $p(-1), \ldots, p(-(n^* - t^*))$, as the auxiliary information aux.

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

- Derive t^* secrets from available PIFs $\subseteq \overrightarrow{PIF}$. For the sake of brevity, we denote the PIFs by $\{PIF_1, \ldots, PIF_{t^*}\}$.
- Compute $seed_i = \mathsf{F}(PIF_i)$ and $s_i^* = seed_i \mod p$ for $i = 1, 2, \ldots, t^*$.
- Recover other $n^* - t^*$ roots, i.e., $\{s_{t^*+1}^*, \ldots, s_{n^*}^*\}$. We provide the detailed constructions of this step later.
- Compute the master key $S = \mathsf{H}(\mathsf{Sort}(s_1^*, s_2^*, \ldots, s_{n^*}^*))$.

We present two approaches to recover the original $p(x)$, corresponding to two approaches of computing auxiliary information.

Approach 1: Use the coefficients as aux.

- Generate a new polynomial $p_1(x) = x^{n^*} + \sum_{i=1}^{n^*-t^*} aux_i \cdot x^{n^*-i}$ over Z_p, where aux_i denotes the i-th element in aux. $p(x)$ can be represented as $p(x) = p_1(x) + p_2(x)$.
- Utilize Lagrange interpolation to obtain $p_2(x)$. Compute

$$w_i(x) = \prod_{\substack{1 \leq j \leq t^* \\ j \neq i}} \frac{x - s_j^*}{s_i^* - s_j^*}, \; p_2(x) = \sum_{i=1}^{t^*}(-p_1(s_i^*)) \cdot w_i(x).$$

- Obtain $p(x) = p_1(x) + p_2(x)$. Since both $p_1(x)$ and $p_2(x)$ are fully determined, $p(x)$ is uniquely determined.
- Given t^* roots of $p(x)$, other $n^* - t^*$ roots, i.e., $\{s_{t^*+1}^*, \ldots, s_{n^*}^*\}$, can be computed by using Berlekamp's algorithm [41]. Since $p(x)$ is explicitly known over a prime field, standard polynomial factorization algorithms can be used.

[2] We do not specify the length of the input and output of F, since the length depends on which PIF is utilized.

[3] In reality, we can derive a secret from multiple PIFs to achieve a higher security level, i.e., the number of PIFs can be different from that of reconstructable secrets. For the sake of simplicity, we assume that they are equal and only describe how to derive a secret from a single PIF.

Approach 2: Use negative evaluation points as aux.

- Given t^* secrets, obtain a polynomial $q_1(x) = \prod_{i=1}^{t^*}(x - s_i^*)$. Then, the original $p(x)$ can be represented as $p(x) = q_1(x) \cdot q_2(x)$.
- Set $v = -(n^* - t^*)$. Utilize Lagrange interpolation to obtain $q_2(x)$. Compute

$$w_i(x) = \prod_{\substack{v \le j \le -1 \\ j \neq i}} \frac{x-j}{i-j},$$

$$q_2(x) = x^{-v} + \sum_{i=v}^{-1}(p(i)/q_1(i) - i^{-v}) \cdot w_i(x).$$

  Due to the structure of $q_1(x)$, all $q(i)$ for $i = -(n^* - t^*), \ldots, -1$ are not equal to 0. In particular, if $v = -1$, then we define $w_{-1}(x) = 1$.
- Compute the roots of $q_2(x)$ using Berlekamp's algorithm.

$\{s_1, \ldots, s_n\} \leftarrow$ **Outsource**(S).

- Uniformly choose $a_1, \ldots, a_{t-1} \xleftarrow{\$} Z_p$ and generate a polynomial $f(x) = S + a_1 x + \cdots + a_{t-1}x^{t-1}$ over Z_p with degree at most $t-1$.
- Compute $s_i = f(i)$ for $i = 1, 2, \ldots, n$.

$S \leftarrow$ **Access**$(s_{i_1}, \ldots, s_{i_t})$.

- On input t secret shares (denoted by $\{s_1, \ldots, s_t\}$), compute S as $S = \sum_{i=1}^{t} w_i s_i$, where $w_i = \prod_{\substack{1 \le j \le t \\ j \neq i}} \frac{j}{j-i}$.

Discussion. We introduce two approaches for computing the auxiliary information and the corresponding key recovery. Regarding security, we notice that both approaches can satisfy the security definitions provided in Section IV-B. Regarding efficiency, Approach 1 needs to compute t^* interpolations for the given t^* secrets, while Approach 2 requires $n^* - t^*$ interpolations for $[-(n^* - t^*), -1]$. Thus, Approach 1 is more efficient than Approach 2 for key recovery when t^* is small, and Approach 2 outperforms Approach 1 as t^* increases. The detailed performance evaluation is provided in Section VI.

D. Deploying NIDKM for typical scenarios

Manage a key to a bank's vault or a cold wallet. Consider that a vault door key is managed by a committee with n^* members at the bank. For initialization, each member can contribute a (conditionally) reconstructable secret from their own PIFs and input it at the physical door. The door executes the **KeyGen** algorithm to derive a key from the members' secrets, which is then used to protect the vault. To open the vault, any subset of t^* members reproduces their original secrets from the same PIFs and provides them to the door on-site. The door runs the **Recovery** algorithm with the provided secrets and the auxiliary information to reconstruct the key. If the auxiliary information is unavailable, the key can still be recovered by collecting all n^* original secrets and re-running the **KeyGen** algorithm.

The same deployment applies to managing high-value keys in cold wallets. In this setting, the cold wallet (i.e., an offline device) plays the role of the server, which derives the key from the members' secrets during setup and later reconstructs it from a threshold number of original secrets during recovery.

In the above scenarios, the members can jointly generate a key without interacting with each other, and each member per se acts as a source of randomness, thus avoiding biased randomness. Furthermore, when the device destruction occurs, the key can also be recovered by the members' PIFs if they utilize the (conditionally) reconstructable ones.

Portable and destruction-resistant key management for personal keys. With NIDKM, a user can manage their keys in a portable and destruction-resistant way. During the setup phase, the user can derive n^* reconstructable secrets from their PIFs and then use the **KeyGen** algorithm to generate a master key from the secrets with the auxiliary information stored by a backup system. To achieve portability, the user can use a set of key servers, run the **Outsource** algorithm to generate the key shares, and send a share to a key server via a secure channel. As long as any t of the n key servers are available, users can access the master key in a portable way by invoking the **Access** algorithm.[4] For clarity, we refine three cases and how users to reconstruct their key case by case as follows.

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. Users can directly run the **Access** algorithm to recover their key. NIDKM achieves portability in normal times, i.e., enables 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, users can execute the **Recovery** algorithm using any t^* original secrets to reconstruct their key. Thus, NIDKM enables users to recover their master keys under partial destruction if any t^* secrets can be reconstructed.

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, users can run the **KeyGen** algorithm to reconstruct the key using all n^* secrets as input. Therefore, NIDKM ensures the key recovery under full destruction if all n^* PIFs are available.

Users can also utilize the master key to manage their other keys. In reality, there are two methods to deploy NIDKM.

Integrating with encryption. For registered cryptosystems, the user can utilize the master key to encrypt a set of existing cryptographic keys $\vec{K}$ and outsource the ciphertexts to a cloud storage server $\mathcal{CS}$. Subsequently, the user can access the master key with the aid of key servers or with t^* PIFs and decrypt the ciphertexts to obtain $\vec{K}$. The user can use either private-key or public-key encryption, as discussed below.

[4]The user can establish a secure channel with key servers by using an authentication scheme based on other storage/device-independent PIFs.

Private-key encryption. The computation costs of private-key encryption are much less than those of public-key encryption. Users can decrypt $\vec{K}$ on demand when utilizing the CTR mode. However, when a user generates a new key, they must recover the master key S and then encrypt the new key.

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 proxy re-encryption). When updating the master key, the user does not need to decrypt $\vec{C}$ with the previous private key and then encrypt $\vec{K}$ with the newly-generated public key. The user can directly utilize the proxy re-encryption to update the ciphertexts of $\vec{K}$. Furthermore, when generating a new key, the user can directly utilize the public key to encrypt the key without recovering the master key.

Integrating with KDFs. For a new cryptosystem, a more efficient method is to directly derive various new keys from the master key with mature KDFs [42]. In this case, the user does not need to perform encryption or decryption operations.

Key rotation. It is feasible for a sophisticated adversary to corrupt more than t^* PIFs given enough time. Therefore, the master key should be updated after some PIFs are leaked. The user can re-execute the NIDKM 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.

V. Security analysis

An adversary who compromises $t-1$ shares and t^*-1 PIFs cannot get any information about the master key. The security of NIDKM is captured by Theorem 1, and we formally prove its security as follows.

Theorem 1: Assuming that H is a random oracle, F is secure key derivation function following the definition in [36], and every PIF-derived secret has at least ℓ bits of min-entropy, given the public aux, any PPT adversary $\mathcal{A}$ who compromises any $t-1$ of n shares and any t^*-1 of n^* PIFs can distinguish the master key from a uniformly-chosen key from Z_p with probability no more than $\frac{1}{2}+\mathsf{negl}(\ell)$, where ℓ is the security parameter, and negl is a negligible function.

Note that if any unrevealed PIF-derived secret is low-entropy, an adversary who has learned $t-1$ secrets can exhaustively guess this secret and recover the key. Therefore, our security guarantee critically relies on the high entropy of each unrevealed secret.

Proof 1: We prove Theorem 1 with a series of games.

Game 0. An oracle is initialized, where 1) it takes t^*-1 secrets as inputs and randomly generates n^*-t^* PIF-derived secrets each with ℓ bits min-entropy, 2) a master key S is derived from the n^* secrets with **KeyGen** and a random oracle H which uniformly chooses an element in Z_p as output for each different query and records the input/output tuple to ensure the consistency, 3) S is split into n shares with **Outsource**, and 4) a key S^* is uniformly chosen from Z_p. $\mathcal{A}$, querying with t^*-1 secrets, is given any $t-1$ of n shares, the auxiliary information aux, S, and S^*. $\mathcal{A}$ can query the random oracle H at most $q(\ell)$ times. Given S and S^*, if $\mathcal{A}$ can distinguish them, then this game outputs 1, else $\mathcal{A}$ outputs 0. The success probability of $\mathcal{A}$ is $\Pr[\mathsf{Game\ 0}_{\mathcal{A}}(\ell)=1]=\frac{1}{2}+\mathsf{Adv}_0$, where Adv_0 denotes the advantage of $\mathcal{A}$ in this game. We will prove that Adv_0 is negligible.

Game 1. This game is the same as Game 0 except for the generation of the auxiliary information. In this game, the challenger uniformly chooses the auxiliary information as $aux \xleftarrow{\$} Z_p^{n^*-t^*}$ and provides it to the adversary $\mathcal{A}$. We notice that the element in aux is in Z_p, whether using Approach 1 or 2. We utilize Adv_1 to denote $\mathcal{A}$'s advantage on Game 1. In Game 0, aux is determined by the secrets, while it is uniformly chosen in Game 1. Under the assumption that all unrevealed secrets have min-entropy at least ℓ-bit. Thus, the distribution of aux in Game 0 is computationally indistinguishable from uniform from the adversary's view. Therefore, for any PPT $\mathcal{A}$, we have $|\mathsf{Adv}_1-\mathsf{Adv}_0|\leq\mathsf{negl}(\ell)$.

Game 2. This game is the same as Game 1, except that S is uniformly sampled from Z_p, independently of all secrets and aux. Let Adv_2 denote $\mathcal{A}$'s advantage in this game.

Since $\mathcal{A}$ learns at most $t-1$ shares and t^*-1 secrets, and because Shamir's secret sharing is information-theoretically secure, the master key is perfectly hidden from $\mathcal{A}$. Combined with the min-entropy assumption on unrevealed secrets and the random oracle model for H, we have $|\mathsf{Adv}_1-\mathsf{Adv}_2|\leq\mathsf{negl}(\ell)$, and $\mathsf{Adv}_2\leq\mathsf{negl}(\ell)$.

To sum up, we have that the adversary's advantage in Game 0 is also negligible. This concludes the proof. □

Discussion about robustness of NIDKM. The robustness of NIDKM can be trivially ensured by the correctness of our algorithm. We notice that the key can be recovered from any t^* of n^* original secrets with the aid of the auxiliary information or n^* original secrets themselves. However, if we try to recover the key with more than t^* secrets, e.g., t^*+1, but one secret is false, then the key cannot be correctly recovered. The server may need to attempt t^*+1 times to determine which one is false, and recover the key from the remaining correct secrets.

VI. Implementation and evaluation

We implement an NIDKM 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/Yuan-Zhang-uestc/NIDKM.git.

A. Deriving secrets from PIFs

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

Someone the user is. We investigate existing biometric feature recognition and extraction methods. A minutiae-based fingerprint individuality model [43] is utilized to extract features from users' fingerprints. We leverage the scheme [44] for face recognition to extract the face features. We utilize the algorithm in [45] for iris recognition to extract the features of irises. The extracted iris features can be formatted according

to the biometric standards. We utilize the hand geometry recognition scheme [46], the palmprint recognition scheme [47], and the hand gesture recognition scheme [48] to extract the hand characteristics, respectively. We utilize Deep-ECG [49], 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 III, we can derive a reconstructable secret from a device-dependent PIF with the aid of another reconstructable secret.

B. 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 VI-A.

Figure 4 shows the computation costs of **KeyGen** and **Recovery** under the two approaches. We can see that Approach 1 is more efficient than Approach 2 when t^* is small. As the increase of t^*, Approach 2 takes fewer costs than Approach 1, since n^*-t^* becomes smaller, and the number of interpolation operations run by Approach 2 is decreasing gradually.

NIDKM can also be utilized to achieve portable and destruction-resistant key management with the **Outsource** and **Access** algorithms. We assume that a user employs a set of key servers $\vec{\mathcal{KS}}$ to manage the shares of the key and a cloud storage provider $\mathcal{CS}$ to store the auxiliary information. Figure 5 illustrates the user's communication costs of running each algorithm under the fixed n^*, n and the increasing t^*, t. The user takes fewer costs for key generation and key recovery as the increase with t^*. This is because the larger the threshold t^* is, the less auxiliary information is needed. However, as t increases, the communication costs required by the user in the **Access** algorithm become greater, because the number of servers that need to be interacted with also increases.

We test the costs of protecting a key set $\vec{K}$ under different encryption algorithms, including RSA-OAEP, ElGamal, and [CTR]AES. Obviously, the computation delay for private-key encryption is much less than that for public-key encryption. Hence, we focus on communication costs. Figure 6 shows the communication costs on $\mathcal{CS}$ and the costs on $\mathcal{CS}$ and a user $\mathcal{U}$ when $\mathcal{U}$ only recovers one key. The communication costs when using public-key encryption are significantly larger than those when using [CTR]AES. Overall, the ElGamal encryption scheme performs better than RSA-OAEP.

C. Comparison with KDFs

NIDRKM can be considered as a key derivation function, since it can convert users' PIFs to a reconstructable master key. In this subsection, we compare NIDKM with different KDFs and further elaborate on the desirable advantages of NIDKM.

We start with password-based key derivation functions (PBKDFs) [50], [51] that derive keys from users' passwords. The proliferation of PBKDFs provides protection for users without introducing key management problems. Typically, PBKDF1 and PBKDF2 [51] are widely used in network applications, e.g., WPA2 [52] 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 have shown that utilizing passwords as the sole defense line is indeed insufficient.

PIF	Computation delay (ms)
Fingerprint [43]	≈1.60
Face [44]	≈1.50
Iris [45]	≈3.00
Hand geometry [46]	≈7.6
Palmprint [47]	≈6
ECG [49]	≈0.32
Hand gesture [48]	≈90 − 110
SIM card	≈0.003
Hardware token	≈1.36
Intel SGX	≈0.56

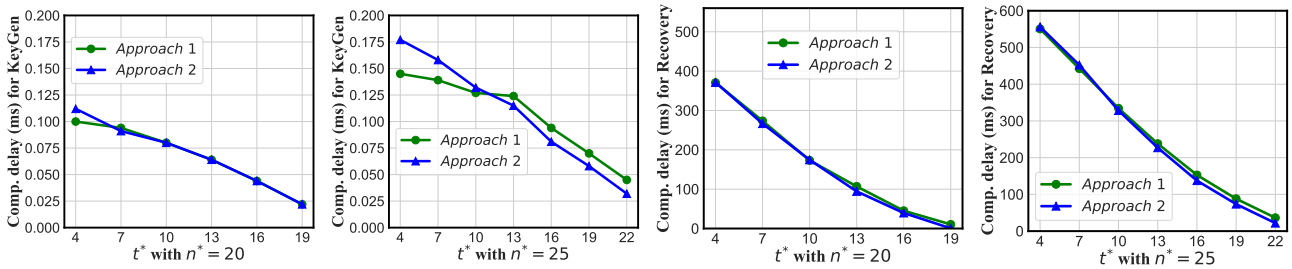
Table I: Computation delay of deriving secrets from PIFs.

Nair et al. [32] 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^*-1 of them. We compare NIDKM with TMFKDF as follows.

Regarding costs, Table II shows the comparison between NIDKM 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 NIDKM is 1∼2 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 NIDKM, users can directly retrieve their keys from n^* PIFs without any storage, if all n^* PIFs are independent of storage.

Regarding security and practicality, TMFKDF needs a private setup to achieve the threshold property on the multiple factors, but NIDKM does not rely on a single party to generate the master key, instead of determined by multiple members. In addition, the shares in NIDKM are independent of PIFs used for key derivation, while in [32], a share is protected by a PIF. In NIDKM, an adversary can compromise any $t-1$ shares and any t^*-1 PIFs, but in [32], it is only allowed to obtain any t^*-1 PIFs and the *corresponding* shares. NIDKM enables users to balance security and efficiency, but the TMFKDF fails to achieve it. Specifically, there are two independent thresholds in NIDKM. The threshold in TMFKDF corresponds to (t^*, n^*) in NIDKM. Once an adversary compromises t^* shares (i.e., PIFs) of TMFKDF, it can recover the key. In NIDKM, such an adversary cannot get any information about the master key, since the user can set $t \geq t^*$. The user can achieve a trade-off between security and efficiency by adjusting (t, n) while remaining (t^*, n^*).



(a) **KeyGen** with $n^* = 20$. (b) **KeyGen** with $n^* = 25$. (c) **Recovery** with $n^* = 20$. (d) **Recovery** with $n^* = 25$.

Figure 4: Comparison of the computational costs of the two approaches of the **KeyGen** and **Recovery** algorithms.

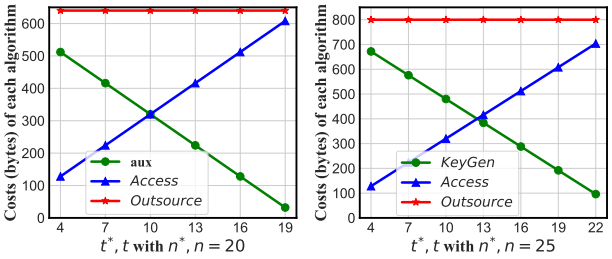


Figure 5: Communication costs of NIDKM.

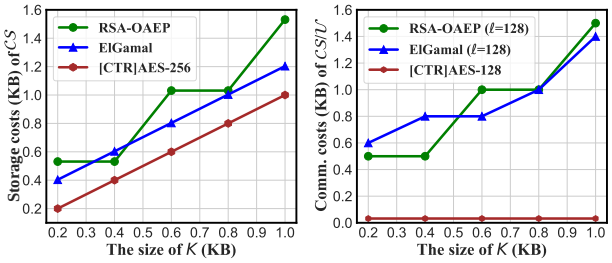


Figure 6: Costs of managing a key set.

Method	Comp. delay (ms)
NIDKM (using **Access**)	0.036
NIDKM (using **Recovery** with Approach 1)	31.75
NIDKM (using **Recovery** with Approach 2)	17.21
Threshold MFKDF	27.86

Table II: Comparison with MFKDF. Set $t = t^* = 8$ and $n = n^* = 10$.

D. Applications and compatibility

One promising application is that we can easily construct a destruction-resistant threshold multiple-factor authentication scheme with storage/device-independent PIFs, 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 NIDKM as well as NIDKM-based DRC (i.e., NIDKM + commercial backup system) in the following.

Another promising application of NIDKM and NIDKM-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 implicitly require a backup system to store the non-sensitive information, such as ciphertexts, signatures). With NIDKM, a user can derive a master key from their PIFs and either maintain it locally or use 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, NIDKM-based DRC can be directly deployed to thwart ransomware attacks.

Recently, many novel PIFs have been proposed, e.g., PCR-Auth [53] and Capacitive Plethysmogram [54]. We stress that NIDKM has forward-compatibility with future PIFs. As long as a PIF is storage-independent and can be represented by a unique binary string, it can then be utilized in NIDKM to derive a master key, which potentially applies to newly-discovered PIFs in the future.

VII. 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 NIDKM, a non-interactive distributed key management scheme with pre-determined shares. We have formally proven the security of NIDKM. We have implemented an NIDKM 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.

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB3106503; in part by the National Natural Science

Foundation of China under Grant 62472074; and in part by the Sichuan Science and Technology Program under Grant 2022ZDZX0038.

REFERENCES

[1] "Sony Fights Spread of Stolen Data by Using "bad seed" Attack on Torrents," https://arstechnica.com/tech-policy/2014/12/sony-fights\--spread-of-stolen-data-by-using-bad-seed-attack-on-torrents/, 2014.

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

[3] J. Katz, "Round-optimal, Fully Secure Distributed Key Generation," in *Proc. CRYPTO*, 2024, pp. 285–316.

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

[5] "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.

[6] "Tonga: Volcanic Eruption and Tsunami," https://reliefweb.int/disaster/vo-2022-000005-ton, 2022.

[7] H. Feng, Z. Lu, and Q. Tang, "Dragon: Decentralization at the Cost of Representation after Arbitrary Grouping and Its Applications to Subcubic DKG and Interactive Consistency," in *Proc. PODC*, 2024, pp. 469–479.

[8] H. Feng, T. Mai, and Q. Tang, "Scalable and Adaptively Secure Any-Trust Distributed Key Generation and All-hands Checkpointing," in *Proc. CCS*, 2024, pp. 2636–2650.

[9] H. Feng and Q. Tang, "Asymptotically Optimal Adaptive Asynchronous Common Coin and DKG with Silent Setup," in *Proc. CRYPTO*, 2025.

[10] "Coinbase," https://wallet.coinbase.com/.

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

[12] G. R. Blakley, "Safeguarding Cryptographic Keys," in *Managing Requirements Knowledge, International Workshop*, 1979, pp. 313–313.

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

[14] 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 *Proc. CCS*, 2019, pp. 2387–2402.

[15] S. K. D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song, "CHURP: Dynamic-Committee Proactive Secret Sharing," in *Proc. CCS*, 2019, pp. 2369–2386.

[16] T. P. Pedersen, "Non-interactive and Information-theoretic Secure Verifiable Secret Sharing," in *Proc. CTYPTO*, 1991, pp. 129–140.

[17] M. Bellare, W. Dai, and P. Rogaway, "Reimagining Secret Sharing: Creating a Safer and More Versatile Primitive by Adding Authenticity, Correcting Errors, and Reducing Randomness Requirements," *Proc. PETS*, 2020.

[18] S. Das, Z. Xiang, A. Tomescu, A. Spiegelman, B. Pinkas, and L. Ren, "Verifiable Secret Sharing Simplified," in *Proc. S&P*, 2025, pp. 633–651.

[19] 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 *EuroS&P*, 2020, pp. 587–606.

[20] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "TOPPSS: Cost-Minimal Password-Protected Secret Sharing Based on Threshold OPRF," in *Proc. ACNS*, 2017, pp. 39–58.

[21] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "PASTA: Password-Based Threshold Authentication," in *Proc. CCS*, 2018, pp. 2042–2059.

[22] Y. Zhang, C. Xu, N. Cheng, and X. S. Shen, "Secure Password-Protected Encryption Key for Deduplicated Cloud Storage Systems," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2789–2806, 2021.

[23] R. W. Lai, C. Egger, M. Reinert, S. S. Chow, M. Maffei, and D. Schröder, "Simple Password-Hardened Encryption Services," in *Proc. USENIX Security*, 2018, pp. 1405–1421.

[24] C. Jia, S. Wu, and D. Wang, "Reliable Password Hardening Service with Opt-Out," in *Proc. SRDS*, 2022, pp. 250–261.

[25] 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 *Proc. USENIX Security*, 2022, pp. 2353–2370.

[26] 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 Trans. Mob. Comput.*, vol. 20, no. 6, pp. 2297–2312, 2020.

[27] Y. Song, C. Xu, Y. Zhang, and S. Li, "Hardening Password-based Credential Databases," *IEEE Trans. Inf. Foren. Sec.*, vol. 19, pp. 469–484, 2023.

[28] Y. Zhang, Y. Wang, G. Xu, Y. Song, and H. Li, "Belt and Braces! Fight against Key Compromising in Single Sign-On Systems," in *Proc. ACSAC*, 2025, pp. 363–376.

[29] S. Agrawal, P. Mohassel, P. Mukherjee, and P. Rindal, "DiSE: Distributed Symmetric-Key Encryption," in *Proc. CCS*, 2018, pp. 1993–2010.

[30] M. Christodorescu, S. Gaddam, P. Mukherjee, and R. Sinha, "Amortized Threshold Symmetric-Key Encryption," in *Proc. CCS*, 2021, pp. 2758–2779.

[31] A. Kate, P. Mukherjee, H. Saleem, P. Sarkar, and B. Roberts, "ANARKey: A New Approach to (Socially) Recover Keys," *ePrint Archive*, 2025.

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

[33] M. Wang and W. Deng, "Mitigating Bias in Face Recognition Using Skewness-Aware Reinforcement Learning," in *Proc. CVPR*, 2020, pp. 9322–9331.

[34] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," in *Proc. EUROCRYPT*, 2004, pp. 523–540.

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

[36] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," in *Proc. CRYPTO*, vol. 6223, 2010, pp. 631–648.

[37] K. Lee, B. Kaiser, J. Mayer, and A. Narayanan, "An Empirical Study of Wireless Carrier Authentication for SIM Swaps," in *Proc. SOUPS*, 2020, pp. 61–79.

[38] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture," Tech. Rep. TS 33.102, 2022.

[39] E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo, "True2F: Backdoor-Resistant Authentication Tokens," in *Proc. S&P*, 2019, pp. 398–416.

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

[41] V. Shoup, *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press. [Online]. Available: https://shoup.net/ntb/ntb-v2.pdf

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

[43] J. Chen and Y.-S. Moon, "A Minutiae-Based Fingerprint Individuality Model," in *Proc. CVPR*, 2007, pp. 1–7.

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

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

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

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

[48] Y. Fang, K. Wang, J. Cheng, and H. Lu, "A Real-Time Hand Gesture Recognition Method," in *Proc. ICME*, 2007, pp. 995–998.

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

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

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

[52] IEEE, "IEEE Standard for Information Technology: Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Specific Requirements." *IEEE Std. 802.11i-2004*, 2004.

[53] L. Huang and C. Wang, "PCR-Auth: Solving Authentication Puzzle Challenge with Encoded Palm Contact Response," in *Proc. S&P*, 2022, pp. 1034–1048.

[54] 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 Trans. Mob. Comput.*, vol. 23, no. 1, pp. 895–908, 2024.