An *et al. Cybersecurity* (2024) 7:42
https://doi.org/10.1186/s42400-024-00236-6

Cybersecurity

RESEARCH

Open Access

Honey password vaults tolerating leakage of both personally identifiable information and passwords

Chao An[1,2,3], YuTing Xiao[1,2*], HaiHang Liu[4], Han Wu[1,2,3] and Rui Zhang[1,2,3]

Abstract

Honey vaults are useful tools for password management. A vault usually contains usernames for each domain, and the corresponding passwords, encrypted with a master password chosen by the owner. By generating decoy vaults for incorrect master password attempts, honey vaults force attackers with the vault's storage file to engage in online verification to distinguish the real vaults, thus thwarting offline guessing attacks. However, sophisticated attackers can acquire additional information, such as personally identifiable information (PII) and partial passwords contained within the vault from various data breaches. Since many users tend to incorporate PII in their passwords, attackers may utilize PII to distinguish the real vault. Furthermore, if attackers may learn partial passwords included in the real vault, it can exclude numerous decoy vaults without the need for online verification. Indeed, both leakages pose serious threats to the security of the existing honey vault schemes. In this paper, we explore two attack variants of the inspired attack scenario, where the attacker gains access to the vault's storage file along with acquiring PII and partial passwords contained within the real vault, and design a new honey vault scheme. For security assurance, we prove that our scheme is secure against one of the aforementioned attack variants. Moreover, our experimental findings suggest enhancements in security against the other attack. In particular, to evaluate the security in multiple leakage cases where both the vault's storage file and PII are leaked, we propose several new practical attacks (called PII-based attacks), building upon the existing practical attacks in the traditional single leakage case where only the vault's storage file is compromised. Our experimental results demonstrate that certain PII-based attacks achieve a 63–70% accuracy in distinguishing the real vault from decoys in the best-performing honey vault scheme (Cheng et al. in Incrementally updateable honey password vaults, pp 857–874, 2021). Our scheme reduces these metrics to 41–50%, closely approaching the ideal value of 50%.

Keywords Honey password vault, Personally identifiable information, Passwords

*Correspondence:
YuTing Xiao
xiaoyuting@iie.ac.cn
Full list of author information is available at the end of the article

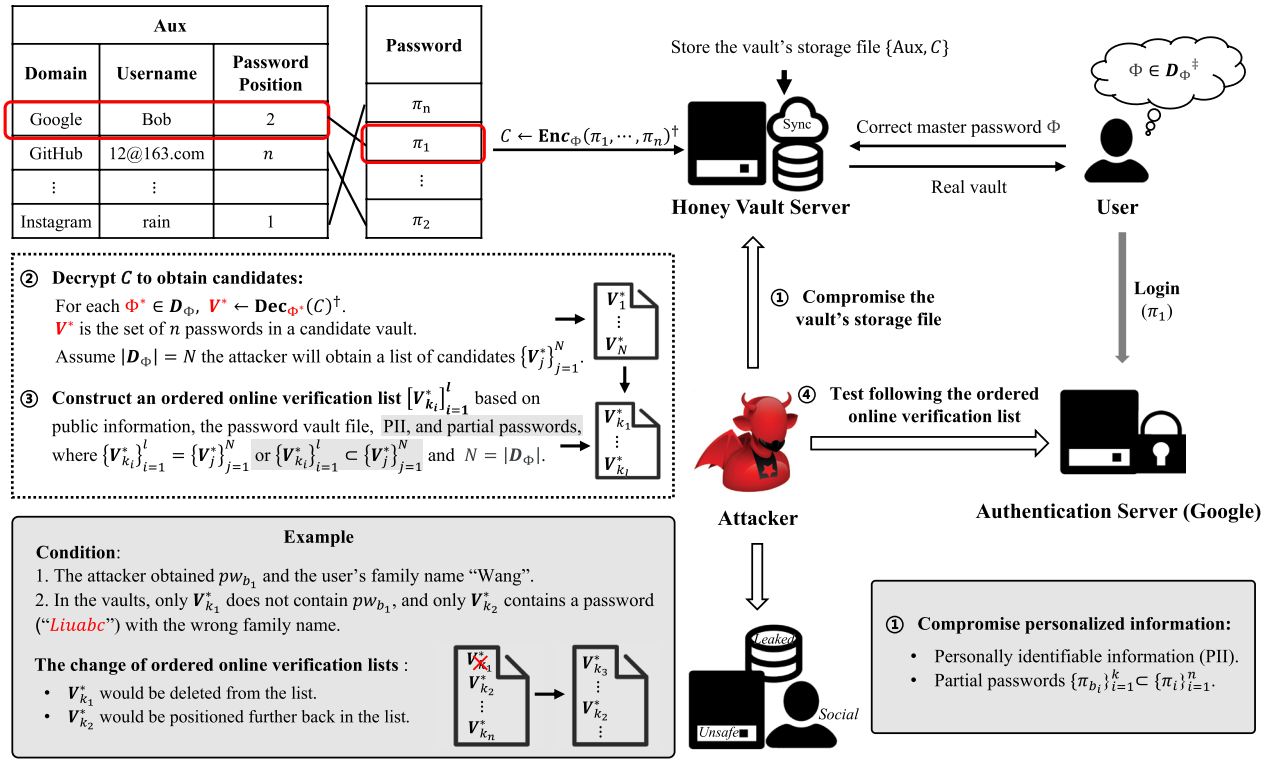


Fig. 1 Attacks variants against honey vault: the original attack [11–13, 18] is the version with all gray text omitted. The attack based on PII and partial passwords in the real vault includes the gray text

Introduction

Passwords are the most widely-used authentication method in practice [1, 2, 8–10, 41] because of their convenience. However, users face increasing challenges in remembering multiple passwords and usernames across services and applications. To tackle this problem, *Password vaults*, also known as wallets or managers, were proposed, where users' passwords are encrypted with a user-selected password, called the master password.

In the real world, it is often necessary to synchronize the password vault across multiple devices, e.g. iCloud keychain. Note that synchronization services provided by the vault applications, such as LastPass and 1Password, or third-party file sync services (like Dropbox and iCloud) may suffer from leakage, which leads to password vault storage (including ciphertext) exposure [23, 24, 39, 40]. Since passwords are usually of low-entropy [7, 49], attackers can efficiently launch offline guessing attacks.

Honey password vault was proposed to address this threat [6, 11–13, 18]. By generating decoy vaults for incorrect master password attempts, honey vaults force attackers with the vault's storage file to engage in online verification to distinguish the real vaults, which is readily detected and countered [16, 19, 37].

Motivations. The primary challenge for honey vaults is to prevent attackers from distinguishing the real vault from decoys. In existing honey vault schemes [6, 11, 13, 18], attackers can obtain the vault's storage file and public information such as password policies, website restrictions, public datasets, probability models, and HE algorithms including the encoder. As shown in Fig. 1 (with gray text omitted), attackers attempt to reveal all passwords $\{\pi_i\}_{i=1}^n$ in the vault as follows:

> *Step 1*: Compromise the vault's storage file $\{\mathsf{Aux}, C\}$, where C is the ciphertext of $\{\pi_i\}_{i=1}^n$ and Aux is the auxiliary information including domains, usernames, and password positions.
> *Step 2*: For each Φ^*[1]$\in \mathcal{D}_\Phi$, where $\mathcal{D}_\Phi$ is the dictionary of master passwords, decrypt C to obtain n passwords V^* in a candidate vault. Assume $|\mathcal{D}_\Phi| = N$, the attacker will obtain a list of candidates $\{\mathsf{V}_i^*\}_{i=1}^N$.

[1] To differentiate the real value of a variable (the master password) from the attacker's guessing value (which may not necessarily be equal to the real value), we use X (Φ) to denote the real value (the real master password) and X^* (Φ^*) to denote the guessing value (the guessing master password) or the value of any other variable derived from the guessing value.

Table 1 Security comparison between our scheme and existing schemes

Scheme		Information an attacker may obtain					
		Public_info	Vault's storage file	PII	Aux_device	Partial passwords	Partial shares of the master password
[11–13, 18]		√	√	×	\	×	\
Our scheme (Sec.4)	Attack-I (Sec.3.2)	√	√	√	√	×	×
	Attack-II (Sec.3.2)	√	√	√	×	$\{\pi_i\}_{i\in\mathcal{K}}$, where $\mathcal{K}\subset\mathcal{I}$.	×
	Supplementary Attack I (Sec.6.1)	√	√	√	×	×	$\{\Phi_i\}_{i\in\mathcal{J}}$, where $\mathcal{J}\subseteq\mathcal{I}, \lvert\mathcal{J}\rvert < t$.
	Supplementary Attack II (Sec.6.1)	√	√	√	×	$\{\pi_i\}_{i\in\mathcal{L}}$, where $\mathcal{L}$ fulfills condition1 †.	$\{\Phi_i\}_{i\in\mathcal{J}}$, where $\mathcal{J}\subseteq\mathcal{I}, \lvert\mathcal{J}\rvert < t-1$.

§ We use $\{\pi_i\}_{i\in\mathcal{I}}$ to denote the passwords in a vault, $\{\Phi_i\}_{i\in\mathcal{I}}$ to denote the shares of the master password Φ, and t to denote the threshold of the secret sharing scheme used in our scheme

§§ The symbol \ indicates that the information in the columns is specifically defined in our scheme, which is not considered in other schemes. The symbol √ (×) indicates that the respective information is (not) accessible to the considered attacker

† Condition 1 is that $\mathcal{L}\subseteq\mathcal{I}$ and any password within $\mathcal{L}\cap\mathcal{J}$ can be deduced by attackers equipped with the vault's storage file and $\{\Phi_i\}_{i\in\mathcal{J}}$

Step 3: Construct an ordered online verification list $\left[\mathsf{V}^*_{k_i}\right]_{i=1}^{N}$ based on public information and the vault's storage file.

Step 4: Test the vaults following the ordered online verification list by logging in to the authentication server (e.g., Google) using the corresponding (Google's) password obtained from each vault.

However, real attackers may possess more power. Due to the numerous website password breaches [3, 20, 35] and the insecure storage of passwords (e.g., plaintext), it is quite likely that attackers may have partial passwords contained within the real vault. As shown in Fig. 1 (with gray text), the attacker, possessing certain passwords (in Step 1), can identify a vault lacking known passwords as a decoy vault, subsequently eliminating it from the ordered online verification list (in Step 3). Therefore, the attacker could discern lots of decoy vaults without online verification. In the extreme case, the attacker can obtain all passwords except one.

Moreover, the personally identifiable information (PII) from sources such as social networks [4] and various breaches [5, 17, 21, 32, 36] makes the situation even worse. Many users create passwords using PII [44, 45], enabling attackers with PII (in Step 1) to construct ordered online verification lists more efficiently (in Step 3), accelerating the discovery of the real vault (Fig. 1 with gray text). For instance, if the target user's family name is "Wang", the vaults containing the password "Liu123" would be positioned further back in the ordered online verification list. Although Cheng et al. [13] acknowledged this threat, they didn't propose a specific scheme.

Indeed, the leakage of PII and partial passwords poses a threat to the security of the existing honey vault schemes [11–13, 18].

Our contributions

In this paper, we explore the vulnerability of existing honey vault schemes in scenarios involving the leakage of PII and partial passwords contained within the real vault. The low entropy of the master password enables attackers to access a group of vaults, including the real one, by using a dictionary of master passwords. Upon obtaining partial passwords, attackers can identify numerous decoy vaults without online verification based on the known passwords. To mitigate the damage caused by the leakage of partial passwords, we introduce a random vector. Naturally, partial passwords and the random vector cannot be leaked simultaneously. To assist user memorization, our honey vault system model incorporates the use of an auxiliary device for storing this random vector.

Attack variants. Building upon the above scenarios and system model, we investigate two attack variants of the inspired attack scenario, where the attacker gains access to the vault's storage file along with acquiring PII and partial passwords contained within the real vault. Due to the risk of losing the auxiliary device and the inability to leak the random vector and partial passwords simultaneously, we consider Attack-I (Table 1), where the vault's storage file, PII, and the random vector are leaked. In the scenario where the auxiliary device is secure, thus countering the damage caused by the leakage of partial passwords, we consider Attack-II (Table 1), where the vault's storage file, PII, and partial passwords are leaked.

A new honey vault scheme. In particular, we design a new honey vault scheme (Sect. 4.3). To evaluate security against Attack-I, we propose PII-based practical attacks considering multiple leakage cases where both the vault's storage file and PII are leaked, building upon existing practical attacks [13, 18] in the traditional single leakage case where only the vault's storage file is compromised. Our experimental results reveal that our PII-based single password attack, PII-based hybrid attack, and PII-based KL divergence attack achieve an accuracy of 63%-70% in distinguishing the real vault from decoys in the best-performing honey vault scheme [13]. Our scheme reduces the metric's value to 41%-50%, closely approaching the ideal value of 50%. For Attack-II, we formally define security against Attack-II and prove that our scheme is secure against it.

Further discussion. As a further discussion, we consider two supplementary attacks for our scheme. In our scheme, we first segment the master password into different shares using a (t, n)-threshold secret sharing scheme (Sect. 2), where n denotes the number of passwords in the vault, and $t < n$. Then, each password in the vault is encrypted with the corresponding share after encoding. Therefore, considering the potential leakage of some shares of the master password during the calculation processes, we define Supplementary Attack-I and Supplementary Attack-II (Table 1). We prove that our scheme provides the same security against Supplementary Attack-I as against Attack-I, and it is secure against Supplementary Attack-II.

Related work

Honey encryption (HE). Juels and Ristenpart introduced honey encryption [22], which can resist brute-force attacks by generating a seemingly credible message for any wrong password. HE employs the distribution transforming encoder (DTE) to encode a message M, conforming to a distribution $\mathcal{M}$, into a string S indistinguishable from randomness. This string is encrypted using carefully selected password-based encryption (PBE) with K, such as AES in CTR mode with PBKDF. Decryption using incorrect key K' produces a random bit string S', decoded back into a decoy message M' sampled from $\mathcal{M}$.

Honey password vault. The design of decoy vaults originates from Kamouflage proposed by Bojinov et al. [6]. They pre-generated a fixed set of decoy vaults (e.g., 1000) along with corresponding decoy master passwords. This method exposes the real master password structure. In 2015, Chatterjee et al. [11] proved that Kamouflage reduces overall security compared to traditional PBE.

In 2015, Chatterjee et al. [11] proposed a honey vault scheme NoCrack based on HE. HE-based honey vault schemes correlate M and K with the vault and the master password, respectively. The scheme encodes the vault into a seemingly random bit string seed via the probabilistic encoder and further encrypts the seed using PBE. If an incorrect master password is used to decrypt and decode, a decoy vault is generated. For the probability model of the vault, Chatterjee et al. used probabilistic context-free grammars (PCFG) to describe the probability model of the single password distribution and sub-grammars to simulate password similarity. For the encoder, they constructed natural language encoders (NLE). NoCrack can resist basic machine-learning attacks.

Golla et al. [18] utilized the Markov model and extended it by the reuse-rate approach to construct the probability model. They proposed the adaptive natural language encoder (ANLE) adjusting the encoder based on the vault's storage file to bring the decoy closer to the real vault. This honey vault scheme can resist Kullback-Leibler divergence attacks, unlike NoCrack.

However, both NLE and ANLE remain vulnerable to encoding attacks. To address the problem, Cheng et al. [12] proposed a probability model transforming encoder against encoding attacks.

Cheng et al. [13] designed a generic construction of honey vaults based on a multi-similar-password model, the conditional probability model transforming encoder (CPMTE), and an incremental update mechanism. With the mechanism, the honey vault can resist intersection attacks. To evaluate the security when the vault's storage file leaks, they proposed the theoretically optimal strategy for online verifications and practical attacks. These attacks can effectively distinguish the real vault from decoys for the existing honey vault schemes excluding their scheme.

Targeted online password guessing. The compromise of Personally Identifiable Information (PII) and sister passwords can enable attackers to conduct targeted online password guessing, wherein they attempt to guess a specific victim's password for a service [15, 28, 33, 43–46, 48]. However, the vulnerability of honey vault security to the leakage of PII and partial passwords has not been extensively explored. While Cheng et al. [13] recognized this threat, they did not propose a specific scheme to address it.

To effectively utilize Personally Identifiable Information (PII) for targeted online password guessing, Wang et al. [43] classified PII into two types: type-1 and type-2. Type-1 PII, which includes information such as names and birthdays, can directly contribute to password generation. Conversely, type-2 PII, such as gender and education [30], may influence password generation behavior but is typically not directly incorporated into passwords. They introduced several PII tags (e.g., $N_1 \sim N_7$

representing name tags, with N_1 indicating the usage of the full name) to extend the original tags as in PCFG [47], and constructed TarPCFG. Additionally, they employed a password-reuse-based context-free grammar to conduct online password guessing for a target user at one service when provided with a leaked sister password of the same user from another service.

In subsequent developments, representative models like Markov [29] and List [42] were transformed into targeted versions, namely TarMarkov and TarList [46], using a similar methodology.

Preliminary

In this section, we review some useful notations and notions.

Notations. We use $\lambda \in \mathbb{N}$ to denote the security parameter. We use PPT to denote probabilistic polynomial time. We use $|\cdot|$ to denote the cardinality of a set or the bit length of a string. We use "||" to denote the concatenation of strings. We use "$\leftarrow\$$" to denote a randomized process, and "$\leftarrow$" to denote a deterministic process. For a deterministic algorithm DAlg, $y \leftarrow \mathrm{DAlg}(x)$ denotes running it with x as input, yielding output y. For a probabilistic algorithm PAlg, $y \leftarrow\$ \mathrm{PAlg}(x)$ denotes running it with x as input, yielding output y. A probabilistic algorithm will become deterministic once its internal randomness r is explicitly specified, which is denoted as $y \leftarrow \mathrm{PAlg}(x, r)$.

Threshold secret sharing. A (t, n)-threshold secret sharing scheme is a fundamental cryptographic technique that divides a secret into n shares. Any t or more shares are sufficient to reconstruct the secret. Shamir [38] constructed a simple and elegant threshold scheme that ensures perfect privacy [31]. The scheme includes the following three algorithms:

- $\mathrm{Gen}(p, t)$: this probabilistic algorithm takes input a random prime p and a threshold t and returns a random polynomial $f(x)$ of degree $t-1$: $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1} (\mathrm{mod}\ p)$, where a_0 is the secret y, $a_i (1 \le i \le k-1)$ is randomly generated form $\mathbb{Z}_p$, and the random vector $\overrightarrow{r_t} = (a_1, \ldots, a_{t-1})$.
- $\mathrm{SS}(\overrightarrow{r_t}, y)$: this deterministic algorithm takes input a secret y and $\overrightarrow{r_t}$ and returns $(y_1, y_2, \cdots, y_n)$, where $y_i = f(i)$ denotes the i-th share of y.
- $\mathrm{Recon}\left(p, \{y_j\}_{j\in\mathcal{J}}\right)$: this deterministic algorithm takes input p and arbitrary t shares $\{y_j\}_{j\in\mathcal{J}}$ and returns the secret $y = \sum_{j\in\mathcal{J}} y_j \lambda_j (\mathrm{mod}\ p)$, where λ_j is the lagrange interpolation coefficient for $j \in \mathcal{J}$ and $\lambda_j = \prod_{l\in J, l\neq j} \frac{-l}{j-l} (\mathrm{mod}\ p)$.

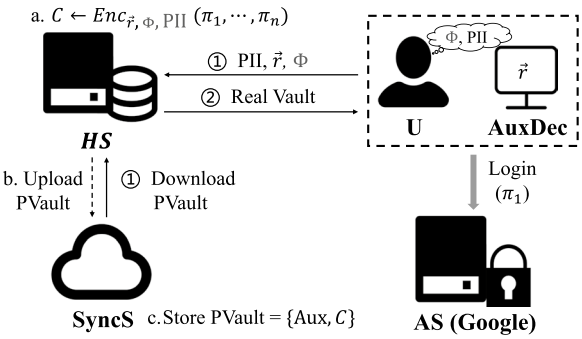


Fig. 2 Our system model

Our model

In this section, we introduce our system model and the security model.

The system model

As shown in Fig. 2, our system involves the following entities:

- *User* U, who wants to store some self-selected[2] passwords $\{\pi_i\}_{i\in\mathcal{I}}$ and has the master password Φ selected from the master password dictionary $\mathcal{D}_\Phi$. Moreover, U has an auxiliary device AuxDec[3] to store the random vector $\overrightarrow{r}$. Using a honey vault, U stores $\{\pi_i\}_{i\in\mathcal{I}}$ and Aux, where Aux is the auxiliary information and $\{\pi_i\}_{i\in\mathcal{I}}$ is encrypted with Φ, $\overrightarrow{r}$ and Personally Identifiable Information (PII) to the ciphertext C. In particular, $\mathsf{Aux} = \{\mathsf{AS}_i\}_{i\in\mathcal{I}}$, where Aux_i includes identity information ADom_i of AS_i, username Un_i and the password position for more convenient retrieval π_i. We require Φ to be independent of the passwords in the vault, as the existing schemes [11–13, 18], and PII.
- *Authentication servers* $\{\mathsf{AS}_i\}_{i\in\mathcal{I}}$. For each $i \in \mathcal{I}$, U sets a password π_i to authenticate with the respective authentication server AS_i.
- *Honey vault server* HS, who provides the password management service for U, with support from AuxDec.
- *Synchronization server* SyncS, who offers the synchronization service and stores the vault's storage file $\mathsf{PVault} = \{\mathsf{Aux}, C\}$.

[2] Pearman et al. [34] indicates that only a small fraction of users use password managers with password generators.

[3] Users can set other trusted auxiliary devices by securely transferring secret information ($\overrightarrow{r}$ and PII) from the trusted device to the new one using methods such as NFC.

The system encompasses four phases:

Initialization phase: U selects Φ and initiates the authentication register protocol with HS. The protocol generates $\overrightarrow{r}$ for encryption, which is then stored in AuxDec.

Store phase: Based on $\overrightarrow{r}$, Φ, $\{\pi_i\}_{i\in\mathcal{I}}$, PII, and Aux offered by U, HS encrypts $\{\pi_i\}_{i\in\mathcal{I}}$ into C and Aux is stored as plaintext [13]. Then PVault is uploaded to SyncS.

Query phase: When U queries passwords with $\overrightarrow{r}^*$, PII*, and Φ^*, HS decrypts C with these inputs. If correct, the real vault is returned; otherwise, a decoy vault is returned.

Update phase: HS downloads and updates PVault based on Φ and changes provided by U, which include password changes, auxiliary information changes, and $\overrightarrow{r}$ changes.

The security model

We assume that an attacker can obtain the following information, which is reasonable as discussed in Sect. 1:

- *All public information* includes password policies, website password restrictions, public datasets, probability models, and HE algorithms including the encoder.
- *Vault's storage file* $\mathsf{PVault} = \{\mathsf{Aux}, C\}$ can be leaked when using sync services.
- *Random vector* $\overrightarrow{r}$ could be obtained through side-channel attacks or from lost AuxDec.
- *PII* could be obtained from social networks and various data breaches.
- *Partial passwords* can be obtained through shoulder surfing attacks, data breaches, or vulnerabilities in websites. We consider the extreme case where the attacker can obtain all but one of the passwords in the vault.

To evaluate the security of our password vault scheme in multiple leakage scenarios where public information, the vault's storage file, PII and partial passwords may leak, we consider two attacks denoted as Attack-I and Attack-II (Table 1).

Attack-I. For Attack-I, we allow the attacker to access the vault's storage file, public information, PII, and $\overrightarrow{r}$. Using the vault's storage file, PII, and $\overrightarrow{r}$, the attacker attempts to decrypt C by employing all master passwords in $\mathcal{D}_\Phi$, generating a list of candidate vaults, where at most one is the real vault. Utilizing public information and PII, the attacker constructs an ordered online verification list. Subsequently, the attacker tests the vaults following the ordered list to confirm their correction by logging into the authentication server (e.g., Google) using the respective (Google's) password obtained from each vault.

The success of the attack depends on two main factors: the offline guessing order of master passwords which is linked to the strength of the master password and the ordered list dictated by a priority function (i.e. the indistinguishability of real and decoy vaults). We take the same research direction as existing schemes [11–13, 18], focusing on the security of encoders.

Attack-II. For Attack-II, we allow the attacker to access the vault's storage file, public information, PII, and potentially all passwords except one contained in the vault. Unlike Attack-I, the attacker in the case of Attack-II is constrained from attempting all possible $\overrightarrow{r}$ values to obtain all candidate vaults containing the actual vault. We assume that the attacker's objective is to compromise one unknown password in the vault. To define the security against Attack-II, we define the following experiment:

Setup Phase: Initialize passwords $\{\pi_i\}_{i\in\mathcal{I}}$ in a vault and an empty list $\mathcal{L}_{corr}$.

Query Phase: In this phase, the attacker is allowed to adaptively query the following oracles:

- Leak(λ): this oracle returns C and PII. This query models the attacker's ability to obtain the real ciphertext and PII.
- Corrupt(k): If $k \notin \mathcal{L}_{corr}$ and $|\mathcal{L}_{corr}| < |\mathcal{I}| - 1$, return π_k and add k to $\mathcal{L}_{corr}$. This query models the attacker's ability to obtain a limited number of passwords.
- RePV(C, Φ^*, PII, $\overrightarrow{r}^*$): this oracle returns $\{\pi_i^*\}_{i\in\mathcal{I}}$. This query models the interaction between the attacker and HS. If Φ^* and $\overrightarrow{r}^*$ are correct, decrypting C reveals the real passwords provided to the attacker. Otherwise, a decoy vault is provided instead.
- OnTest(i, π^*): If $\pi^* = \pi_i$, return 1, otherwise, return 0. This query models the attacker's online password verification with AS_i. For each i, this oracle can be queried at most q[4] times. If the number of logins exceeds this limit, the account will be locked.

Challenge Phase: The attacker picks a target i^* and outputs a guess π^*. If $i^* \notin \mathcal{L}_{corr}$ and $\pi^* = \pi_{i^*}$, the $\mathcal{A}$ wins the experiment.

Definition 1 A honey vault scheme is secure against Attack-II if for any PPT attacker $\mathcal{A}$ in the above experiment, there exists a negligible function ***nelg*** s.t.:

$$\Pr[\,\mathcal{A}\text{ wins}\,] \le \max\{\Pr_G[\,\pi_{i^*}\,], \frac{1}{|\mathcal{D}_\Phi|}\} + \textbf{\textit{nelg}}(\lambda)$$

[4] Considering the target online password guess, Wang et al. [46] recommend that q be set to a small value (e.g. 3).

where the master password Φ is independently and uniformly generated from $\mathcal{D}_\Phi$ and independent of the passwords contained within the vault and PII, and $\Pr_G[\pi_{i^*}] = \Pr\left[\pi_{i^*} \mid \text{PII}, \{\pi_i\}_{i\in\mathcal{L}_{\text{corr}}}, q\right]$ is the probability of success in guessing the target password π_{i^*} online within q times based on PII and $\{\pi_i\}_{i\in\mathcal{L}_{corr}}$.

Definition 1 indicates that a honey vault scheme is secure against Attack-II if the attacker in the experiment doesn't have an advantage over an attacker who guesses the target password online based on PII and partial passwords.

Our honey vault scheme

In this section, we introduce our honey vault scheme. First, we modify PII tags [43] and construct the PII-based probability model. Then, we construct our honey vault scheme based on the PII-based probability model, Shamir's secret sharing [38], and the conditional probability model transforming encoder (CPMTE)[5] [13].

PII tags

We denote the passwords in a vault as V, where $\mathsf{V} = \{\pi_i\}_{i=1}^n$. Inspired by TarPCFG [43], we parse π_i to π_i^T with PII tags, which can capture PII semantics. The number of PII tags and their specific definitions depend on the nature of the PII to be trained and on the granularity the attacker prefers. Here we define our PII tags for attacking Chinese users.

Our PII tags retain the PII tags (name: $N_1, N_2, \cdots, N_7$, birthday: $B_1, B_2, \cdots, B_{10}$, email prefix: E_1, E_2, E_3, phone number: P_1, P_2, and Chinese National Identification number: I_1, I_2, I_3) proposed by Wang et al. [43] and add some new tags including N_8 for family name + given name (e.g., "wangjianguo"), N_9 as the abbr. of N_8 (e.g., "wjg"), and N_{10} for the given name with the first letter capitalized (e.g., "Jianguo"). Since we match password datasets by email to generate our password vault dataset, which indicates that the username selected by the user for the password vault is unknown, we do not consider the username here. We use the above tags to parse the corresponding PII usages in passwords. For instance, "wangjianguo@123" is parsed into N_8@123.

PII-based probability model

We drew on the password generation methods in Cheng et al.'s work [13]: "reusing" parsed old passwords $\pi_1^T, \cdots, \pi_i^T$ and generating a new one. Then we construct a PII-based probability model $\Pr_{\text{PII}}$. Then, the probability $\Pr_{\text{PII}}[\mathsf{V} \mid \text{PII}]$ for the passwords $\mathsf{V} = \{\pi_i\}_{i=1}^n$ in a vault can be expanded as

$$\begin{aligned}\Pr_{\text{PII}}[\mathsf{V} \mid \text{PII}] &= \prod_{i=0}^{n-1} \Pr_{\text{PII}}\left[\pi_{i+1} \mid \{\pi_{i'}\}_{i'=1}^{i}, \text{PII}\right] \\ &= \prod_{i=0}^{n-1} \Pr_{\text{PII}}\left[\pi_{i+1}^T \mid \{\pi_{i'}^T\}_{i'=1}^{i}\right],\end{aligned}$$

where

$$\begin{aligned}&\Pr_{\text{PII}}\left[\pi_{i+1}^T \mid \{\pi_{i'}^T\}_{i'=1}^{i}\right] \\ &= \left(\frac{f(i)}{i}\sum_{i'=1}^{i}\Pr_{\text{pss}}\left[\pi_{i+1}^T \mid \pi_{i'}^T\right] + \big(1-f(i)\big)\Pr_{\text{ps}}\left[\pi_{i+1}^T\right]\right),\end{aligned}$$

where $\Pr_{\text{pss}}$, $\Pr_{\text{ps}}$, and $f(i)$ represent the PII-based single-similar password model, the PII-based single password model, and the reused probability function. The new generation and the reusing are captured by $\Pr_{\text{ps}}$ and $\Pr_{\text{pss}}$. And $f(i)$ captures the probability of reusing the first i parsed old passwords to generate π_{i+1}^T. Considering that a PII tag may represent more than two normal characters (ASCII codes), we define that two parsed passwords are reused if the longest common substring distance (LCSStrD) [13] is at least $\frac{1}{5}$, where LCSStrD is equal to the length of their longest common substring divided by their maximum length.

PII-based single-similar password model. We use $\{\pi_A^T, \pi_B^T\}$ to denote a reused parsed password pair of a user for different authentications. We match passwords in different password datasets (Table 2) by email to construct a list of reused password pairs. We assume that π_A^T can be generated by reusing π_B^T through tail deletion (*td*), tail insertion (*ti*), head deletion (*hd*), and head insertion (*hi*), which are the most common reuse habits of users [14].

During the training phase, the first step is to use LCSStrD [13], Manhattan-distance (MD) [25], and Levenshtein-distance (LD) [27] to measure the similarity score $d_D^1 = D\left(\pi_A^T, \pi_B^T\right)$.

We then employ an operation, denoted as OP, following the order of *hd*, *td*, *ti*, *hi* to generate $\pi_{A^1}^T$ by reusing π_B^T. This implies that we resort to tail deletion only if the similarity score does not increase through head deletion. The path is considered effective if $d_D^2 = D\left(\pi_A^T, \pi_{A^1}^T\right)$ fulfills the following conditions:

- For delete operation (*hd* or *td*) in the path, the distance needs to satisfy (1) $d_{LD}^2 < d_{LD}^1$ or (2) $d_{LD}^2 \le d_{LD}^1$ and $d_{MD}^2 < d_{MD}^1$.
- For delete operation (*hi* or *ti*) in the path, $d_{LD}^2 < d_{LD}^1$ and $d_{LCSStrD}^2$ $d_{LCSStrD}^1$.

[5] Naturally, our scheme inherits the traits of resistance to encoding attacks, intersection attacks, and attacks on adaptive encoders.

If the validity of paths is determined by a single method, we may miss some effective paths. Subsequently, π_A^T is updated to $\pi_{A^1}^T$. The process is repeated until $\pi_{A^k}^T = \pi_B^T$.

Based on all effective paths for all parsed password pairs, we compute the probability of the existence of the insert operation, the probability of the existence of the delete operation, the probability of the number of operations, and the probability of adding the operation character including PII tags and normal characters in 95 printable ASCII code. Let l_{OP} be the number of the operation OP. Since over 99% of passwords are less than 17 characters long [29], and very few are shorter than 4 characters, then $l_{hd} + l_t td < \min\{\frac{4}{5} \times |\pi_A^T|, |\pi_A^T| - 4\}$ and $l_{hi} + l_{ti} < \min\{4 \times (|\pi_A^T| - l_{hd} - l_{td}), 16 - |\pi_A^T| - l_{hd} - l_{td}\}$.

Then, $\Pr[\,\text{wjgwords} \mid \text{5words67}, \text{PII}\,] = \Pr_{\text{PII}}[\,\text{N}_8\text{words} \mid \text{5words67}\,] = \Pr_I[\,1\,] \times \Pr_D[\,1\,] \times \times \Pr_{hdn}[\,1\,] \times \Pr_{tdn}[\,2\,] \times \Pr_{hin}[\,1\,] \times \Pr_{hic}[\,\text{N}_8\,]$. Here, $\Pr_I[\,1\,]$ ($\Pr_D[\,1\,]$) is the probabilities that insertion (deletion) exists; $\Pr_{hdn}[\,1\,]$, $\Pr_{tdn}[\,2\,]$ and $\Pr_{hin}[\,1\,]$ are the probabilities of deleting 1 head character, deleting 2 tail characters, and adding 1 head character, respectively; $\Pr_{hic}[\,\text{N}_8\,]$ is the probabilities of adding the character "N_8" to the head, respectively.

PII-based single password model. Taking into account the rarity of passwords shorter than 4 characters, we presume that a parsed password with a length of less than 4 includes at least one PII tag. To conveniently meet this condition, we utilize the TarList model [46] with add-$k_s = 10^{-8}$ smoothing as the probability model. However, considering the limitations and small sizes of password datasets with PII, we can't rely solely on list-based methods.

Therefore, we use the TarList model for parsed passwords with lengths of less than 4 and opt for a 1-order TarMarkov model [46] with Laplace smoothing for parsed passwords with lengths of more than 3. It's worth noting when using the TarMarkov model to calculate probabilities: since every parsed password in Chin contains 3 or fewer PII tags, we impose a limit to avoid excessive length post-restoration-parsed passwords cannot contain 4 or more PII tags. This necessitates us to calculate probabilities under multiple conditions. Furthermore, the probability of a parsed password with a length greater than 3 is the product of the parsed password probability based on the above method and the initial coefficient. The initial coefficient is the sum of the probabilities of parsed passwords with lengths greater than 3.

Reused function. We train $f(i)$ based on Chin as Cheng et al. [13]. As shown in Fig. 3, we use $\frac{1}{1+e^{-3.134i+4.033}}$ to simulate $f_{\mathsf{Chin}}(i)$.

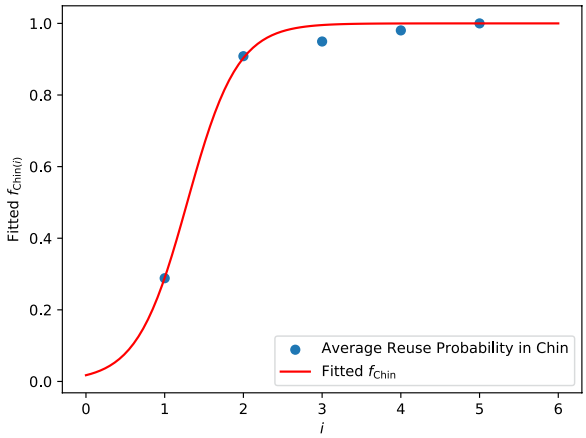


Fig. 3 Reuse function $f_{\mathsf{Chin}}(i)$

Our scheme

Our honey vault scheme consists of the following ingredients: PII-based password probability model (Sect. 4.2), Shamir's secret sharing [38], AES in CTR mode with PBKDF as the PBE scheme, the incremental update mechanism [13], and CPMTE [13].

Initialization phase. The honey vault scheme is initialized as follows:

1. $p \leftarrow\$ \mathrm{Init}(\lambda, Max)$: Given λ and the maximum capacity Max of the honey vault, this algorithm outputs a prime number $p > Max$.
2. $\overrightarrow{r_t} \leftarrow\$ \mathrm{Gen}(p, t)$: This algorithm is the same as the Gen algorithm in Sect. 2 and the random vector $\overrightarrow{r_t}$ will be stored in AuxDec.

Store phase. When U wants to store the passwords $\{\pi_i\}_{i\in\mathcal{I}}$, based on $\overrightarrow{r_t}$, the master password Φ, PII, and the auxiliary information Aux offered by U, HS follows the steps below:

1. $\{S_i\}_{i\in\mathcal{I}} \leftarrow\$ \mathrm{Encode}\big(\{\pi_i\}_{i\in\mathcal{I}}, \mathrm{PII}\big)$: Based on PII, $\{\pi_i\}_{i\in\mathcal{I}}$ is parsed into $\{\pi_i^T\}_{i\in\mathcal{I}}$. With CPMTE, $\big(\pi_i \mid \{\pi_{i'}\}_{i'=1}^{i-1}, \mathrm{PII}\big)$ is encoded to S_i for each $i \in \mathcal{I}$.
2. $\{\Phi_i\}_{i\in\mathcal{I}} \leftarrow \mathrm{SS}\big(\Phi, \overrightarrow{r_t}\big)$: Using the SS algorithm in Sect. 2 taking input a secret $\mathsf{H}(\Phi) \in \mathbb{Z}_p$[6] and $\overrightarrow{r_t}$, HS obtains the i-th share of $\mathsf{H}(\Phi)$ as Φ_i.
3. $C \leftarrow\$ \mathrm{Enc}\big(\{\Phi_i\}_{i\in\mathcal{I}}, \{S_i\}_{i\in\mathcal{I}}\big)$: For each $i \in \mathcal{I}$, HS uses the PBE scheme to encrypt S_i with Φ_i and gets C_i. The ciphertext C is $C_1 || \cdots || C_{|\mathcal{I}|}$ and the password file $\mathsf{PVault} = \{\mathsf{Aux}, C\}$ is updated to SyncS.

[6] Note that any password dictionary can be hashed into $\mathbb{Z}_p$ using a collision-resistant hash $\mathsf{H}(*)$.

Query phase. When U wants to query a password, HS follows the steps below:

1. $\{\Phi_i^*\}_{i\in\mathcal{I}} \leftarrow \mathrm{SS}(\Phi^*, \overrightarrow{r_t}^*)$: Using SS algorithm taking input Φ^* and $\overrightarrow{r_t}^*$, HS obtains $\{\Phi_i^*\}_{i\in\mathcal{I}}$.
2. $\{S_i^*\}_{i\in\mathcal{I}} \leftarrow \mathrm{Dec}(\{\Phi_i^*\}_{i\in\mathcal{I}}, C)$: After splitting C to $\{C_i\}_{i\in\mathcal{I}}$, HS uses the PBE scheme to decrypt C_i using Φ_i^* and gets S_i^*.
3. $\{\pi_i^*\}_{i\in\mathcal{I}} \leftarrow \mathrm{Decode}(\{S_i^*\}_{i\in\mathcal{I}}, \mathrm{PII})$: With CPMTE, $\left(S_i^* \mid \{\pi_{i'}^{T*}\}_{i'=1}^{i-1}, \mathrm{PII}\right)$ is decoded in sequential order from $i = 1$ to $|\mathcal{I}|$ and obtains $\{\pi_i^{T*}\}_{i\in\mathcal{I}}$, which can convert to $\{\pi_i^*\}_{i\in\mathcal{I}}$ using PII. And $\{\pi_i^*\}_{i\in\mathcal{I}}$ is returned to U.

Update phase. When U wants to update a password, HS choose one step below:

- Adding a new password: when U adds a new password to the vault, U has the option to increase the threshold t. If U increases the threshold, Init algorithm will be executed to generate $\overrightarrow{r_{t+1}}$ and $\mathcal{I}$ is updated to $\mathcal{I} \cup \{|\mathcal{I}| + 1\}$. Then HS re-executes the algorithms in the other phases.
- Deleting an old password: mark the password as deleted (in Aux) without changing C.
- Changing an old password: delete the old password and add a new password as in the previous two steps. Then update the password position for the corresponding account.

Security analysis

We compare the security of our scheme with the existing schemes in Table 1. The experimental results show that our scheme enhances resistance against Attack-I. Further analysis reveals that our scheme is secure against Attack-II.

Security against Attack-I

Cheng et al.'s [13] proposed the theoretically optimal strategy and practical attacks to evaluate the security of existing honey vault schemes in the traditional single leakage case where only the vault's storage file is compromised. To evaluate the security of our honey vault scheme against Attack-I, we propose a new theoretically optimal strategy to launch Attack-I and several new practical attacks (called PII-based practical attacks), building upon the existing attacks.

Theoretical optimal strategy

To reveal passwords from the vault's storage file, the attacker decrypts C with $\mathcal{D}_\Phi = \{\Phi^{j*}\}_{j=1}^{N}$, where $\mathcal{D}_\Phi$ is the dictionary of master passwords, and obtains a group of vaults. We use V_j^* to denote the set of the passwords obtained by decrypting C with Φ^{j*}, where $1 \le j \le N$. Assuming the attacker tests vaults in a descending order defined by a priority function f_{prio}, we apply the Bayesian theorem to derive the following theorem. The proof of Theorem 1 is postponed to Appendix A.

Theorem 1 *If the encoder is seed-uniform and the master password Φ is independent of the passwords contained in the vault and PII, then*

$$\Pr\left[\Phi^{j*} \mid \overrightarrow{r_t}, C, \mathrm{PII}\right] = k \times \Pr\left[\Phi^{j*}\right] \times \frac{\Pr_{\mathrm{real}}\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]}{\Pr_{\mathrm{decoy}}\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]},$$

where $1 \le j \le N$ and k is a constant.

According to Theorem 1, without considering $\Pr[\Phi^{j*}]$ [13], the theoretically optimal online verification order is the descending order of $\frac{\Pr_{\mathrm{real}}[\mathsf{V}_j^*|\mathrm{PII}]}{\Pr_{\mathrm{decoy}}[\mathsf{V}_j^*|\mathrm{PII}]}$. We parse the passwords in V_j^* and use V_j^{T*} to denote the set of parsed passwords. The priority function f_{prio} is estimated as $\frac{\Pr_{\mathrm{real}}[\mathsf{V}_j^{T*}]}{\Pr_{\mathrm{decoy}}[\mathsf{V}_j^{T*}]}$.

PII-based practical attacks

Based on the PII-based strategy, we extend Cheng et al.'s practical attacks [13] to several PII-based attacks naturally. Furthermore, we consider other existing attacks and extend the Kullback–Leibler (KL) divergence attack [18] to the PII-based KL divergence attack. We instantiate the attacks according to the particularity of PII.

PII-based single-password attack. The attack captures the differences between real and decoy conditional single-password distributions, denoted as $\Pr_{\mathrm{real}}[\pi^T]$ and $\Pr_{\mathrm{decoy}}[\pi^T]$. Assuming passwords in V_j^* are independent, the priority function is estimated as $f_{prio}^{\mathrm{CS}}\left(\mathsf{V}_j^*\right) = \prod_{\pi^{T*}\in\mathsf{V}_j^{T*}} \frac{\Pr_{\mathrm{real}}[\pi^{T*}]}{\Pr_{\mathrm{decoy}}[\pi^{T*}]}$.

To estimate $\Pr_{\mathrm{decoy}}[\pi^{T*}]$, we utilize the PII-based single password model (Sect. 4.2). For $\Pr_{\mathrm{real}}[\pi^{T*}]$, the TarList model with add-$k_s = 10^{-8}$ smoothing is preferred since the list-based attacks are the most effective in targeted online password guessing [46].

PII-based password-similarity attack. The attack captures the difference in similarity distribution between real and

decoy vaults based on two features: feature M and feature I. We define that a vault has feature M, if there exist two passwords (π_1^T, π_2^T) in the vault that LCSStrD of the passwords is at least $\frac{1}{5}$. A vault has a feature I if there exist two passwords (π_1^T, π_2^T) meet at least one of the following conditions: MD is at most $\frac{1}{5}$; at least one of the similarity scores defined by LD and longest common subsequence (LCS) [14] is at least $\frac{1}{5}$; the similarity scores defined by Overlap [26] at least $\frac{1}{4}$.[7] We define that $\mathrm{M}\backslash\mathrm{I}\left(\mathsf{V}_i^{T*}\right) = 1$, if V_i^{T*} has feature M but no feature I. The definition of I\M is similar to the above. The priority function is estimated as $f_{prio}^{\mathrm{S}} = \frac{\mathrm{Pr}_{\mathrm{real}}\left[\mathrm{M}\backslash\mathrm{I}\left(\mathsf{V}_j^{T*}\right)\right]}{\mathrm{Pr}_{\mathrm{decoy}}\left[\mathrm{M}\backslash\mathrm{I}\left(\mathsf{V}_j^{T*}\right)\right]} \times \frac{\mathrm{Pr}_{\mathrm{real}}\left[\mathrm{I}\backslash\mathrm{M}\left(\mathsf{V}_j^{T*}\right)\right]}{\mathrm{Pr}_{\mathrm{decoy}}\left[\mathrm{I}\backslash\mathrm{M}\left(\mathsf{V}_j^{T*}\right)\right]}$.

PII-based hybrid attack. The attack combines the above two attacks. The priority function is estimated as $f_{prio}^{\mathrm{H}} = f_{prio}^{\mathrm{csp}} \times f_{prio}^{\mathrm{ps}}$.

PII-based KL divergence attack. KL divergence attack [18] outperforms the support vector machine (SVM) attack [13]. So we only extend the KL divergence attack. The priority function of the PII-based KL divergence attack is estimated as $f_{prio}^{\mathrm{KL}} = \sum_{i=1}^{s} f_i \log \frac{f_i}{\mathrm{Pr}_{\mathrm{decoy}}\left[\pi_i^{T*}\right]}$, where $\{\pi_i^{T*}\}_{i=1}^{s}$ are the unique passwords of the vault and f_i the frequency of π_i^{T*} in the vault.

Experimental settings

Datasets containing passwords and PII as shown in Table 2 were obtained through hacking incidents or insider exposure, leading to their public availability on the internet. By matching these datasets via email, we generated the Chinese vault dataset, denoted as Chin (Table 2). The sizes of the vaults in Chin range from 2 to 6.

To train the PII-based single password model, the PII-based single-password attack, and the PII-based KL divergence attack, we randomly select 80% of data (passwords and PII) from the 12306 datasets in Chin as the training set for passwords. We use $\mathcal{L}_{Email}$ to denote the set of emails in the training set for passwords.

To train the PII-based single-similar password model for Chinese passwords, we select the data (password pairs and PII) associated with emails in $\mathcal{L}_{Email}$ from the 12306 and Email datasets in Chin. Because Email and 12306 exhibited the highest number of matches among the datasets (Table 2).

To train the reused probability function and the PII-based password-similarity attack, we select the vaults associated with emails in $\mathcal{L}_{Email}$ in Chin as the training set, while the remaining portion served as the testing set. The vaults in the testing set will be treated as real vaults.

[7] To ensure the effectiveness of M\I, the limit value of similarity score under LCSStr is less than LCS.

Table 2 Datasets with PII

Dataset	Type of datasets	Size	Types of PII
12306	Passwords	127,779	Name, Birthday, Phone Number, NID, Email, Username
Email[a]	Passwords	201,017,211	Email
CSDN	Passwords	6,378,780	Email, Username
178	Passwords	2,632,422	Email
DODONEW	Passwords	16,214,712	Email, Username
renren	Passwords	4,130,129	Email
Chin	Vaults	117,917	Name, Birthday, Phone Number, NID, Email

[a] The passwords in Email are mainly from 163, QQ, and Hotmail

Regarding the probabilities related to decoy vaults required for attacks in Sect. 5.1.2, attackers could compute these probabilities using stolen encoders, specifically by leveraging the decoy vaults generated by the stolen encoders.

For a fair and comprehensive comparison, we utilized the same datasets in Cheng et al.'s scheme [13], with 12306 as the password dataset and Chin as the password vault dataset.

In this setting, we employ honey vault schemes to generate decoys and execute attacks to determine the rank of each vault in the testing set.

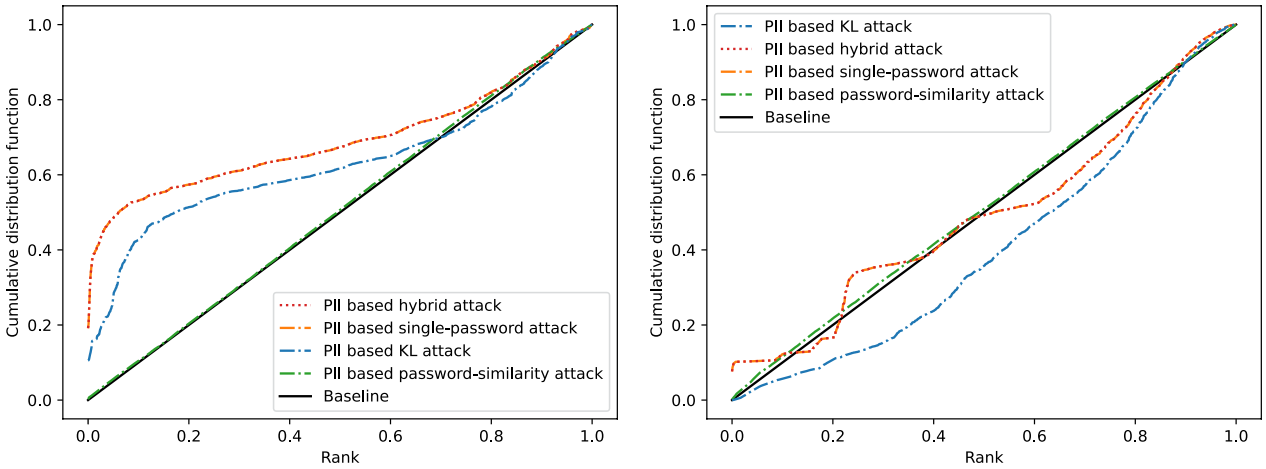
Security metrics. We employ the average rank $\bar{r}$ and accuracy α to indicate the security of a honey vault scheme against attacks, as in [13]. The rank is defined as the ratio of the position in the order to the number of decoys, where the number is 999. Then $\bar{r}$ and α are estimated as

$$\bar{r} = 1 - \int_0^1 F(x)dx, \alpha - 1 - \bar{r},$$

where $F(x)$ is the cumulative distribution function of the ranks. As discussed in [13], a perfectly secure honey vault scheme guarantees that $F_U(x) = x$ and $\alpha = \bar{r} = 0.5$. So we use $F_U(x)$ as the baseline for comparison.

Experimental results

From Fig. 4 and Table 3, we observed that PII-based single password attacks, PII-based hybrid attacks, and PII-based KL divergence attacks achieve an accuracy range of 63% to 70% when distinguishing the real vault from decoys in Cheng et al.'s honey vault scheme [13], which is the existing best-performing scheme. In our scheme, these values are reduced to 41% to 50%, closely approaching the ideal value of 50%. Our experimental results showcase that attackers



(a) Cheng et al.'s [13] (b) Ours

Fig. 4 RCDFs for honey vault schemes under PII-based attacks

Table 3 $\bar{\tau}$ of real vaults under attacks

Attack scheme	Cheng et al.'s [13]	Ours
PII-based single-password attack	30%	49%
PII-based password-similarity attack	49%	49%
PII-based hybrid attack	31%	50%
PII-based KL divergence attack	37%	59%

would need approximately 1.6 times more online verifications to compromise our scheme. PII-based password-similarity attack achieves 49% accuracy in both our scheme and Cheng et al.'s scheme [13]. This suggests that using PII to prase passwords has minimal to no effect on the probability of features M\I and I\M. Consequently, these experimental results indicate an improvement in our scheme's resilience against Attack-I.

Security against Attack-II

We have the following theorem per Theorem 1. The proof of Theorem 2 is postponed to Appendix B.

Theorem 2 *Our honey vault scheme ("Our scheme" section) is secure against Attack-II, assuming the master password* Φ *is independently and uniformly selected from* $\mathcal{D}_\Phi$ *and independent of the passwords in the vault.*

Discussions and extensions

In this section, we present extended attacks to assess the security of our scheme (Sect. 4.3) under the potential leakage case where some shares of the master password are exposed during the calculation processes. Furthermore, we introduce a simplified version of our scheme that does not rely on an auxiliary device.

Supplementary attacks

In practice, attackers can launch side-channel attacks during the calculation processes to obtain crucial information in our scheme, such as some shares of the master password Φ. And their compromise has not been considered in previous attacks. In this section, we explore a potential leakage case where attackers can obtain some shares $\{\Phi_i\}_{i\in\mathcal{J}}$ ($\mathcal{J}\subseteq\mathcal{I}$). We present two extended attacks, denoted as Supplementary Attack-I and Supplementary Attack-II (refer to Table 1), to assess the security of our scheme under such compromises.

We assume that the attacker's goal is to obtain the unknown target password π_{i^*} ($i^*\notin\mathcal{J}$).

Supplementary Attack-I. For Supplementary Attack-I, the attacker can obtain the vault's storage file, public information, PII, and at most $t-1$ shares. This limitation is imposed to prevent the attacker from deducing Φ, $\overrightarrow{r_t}$, and consequently all passwords.

The correctness of a (t,n)-threshold secret sharing scheme implies that $\{\Phi_i\}_{i\in\mathcal{J},|\mathcal{J}|=t-1}$ and $\overrightarrow{r_t}$ correspond one-to-one for any Φ. Therefore, the security achieved by our scheme against Supplementary Attack-I is the same as against Attack-I.

Supplementary Attack-II. For Supplementary Attack-II, the attacker can obtain the vault's storage file, public information, PII, at most $t-2$ shares $\{\Phi_i\}_{i\in\mathcal{J}}$, and the partial passwords $\{\pi_i\}_{i\in\mathcal{L}}$ ($\mathcal{L}\subseteq\mathcal{I}$), where $\mathcal{L}$ fulfills condition that any password within $\mathcal{L}\cap\mathcal{J}$ can be deduced by attackers equipped with the vault's storage file and

$\{\Phi_i\}_{i\in\mathcal{J}}$. This limitation is imposed to prevent the attacker from guessing π_{i^*} ($i^* \notin \mathcal{J} \cup \mathcal{L}$) when the attacker obtains π_i and Φ_i, and π_i is generated by reusing π_{i^*}.

According to the security of the (t, n)-threshold secret sharing scheme:

$$\Pr\left[\ \Phi \mid \{\Phi_i\}_{i\in\mathcal{J},|\mathcal{J}|=t-2}\ \right] = \frac{1}{p}.$$

The successful probability for the attacker to guess π_{i^*} is estimated as

$$\begin{aligned}&\Pr[\,\pi_{i^*} \mid C, \mathrm{PII}, \{\pi_i\}_{i=\mathcal{L}}, \{\Phi_i\}_{i\in\mathcal{J}}\,]\\ \le &\max\{\frac{1}{p\times|\mathcal{D}_\Phi|}, \frac{q}{|\mathcal{D}_{\pi_{i^*}}|}\} \le \frac{q}{|\mathcal{D}_{\pi_{i^*}}|} + \mathbf{nelg}(\lambda)\end{aligned}$$

Accordingly, our scheme resists Supplementary Attack-II.

A simplified version

In this section, we delve into scenarios where users either lack auxiliary devices or prefer not to use them. For instance, when users need to access the honey vault on different devices at any time, requiring an additional device as an auxiliary tool would entail users to carry the device with them at all times. This could increase the difficulty of use for users, leading them to prefer not to use auxiliary devices. In such situations, our scheme can revert to a simpler version. This simplified version incorporates the PII-based password probability model (Sect. 4.2), AES in CTR mode with PBKDF serving as the PBE scheme, the incremental update mechanism [13], and CPMTE [13]. Notably, in contrast to our main scheme (Sect. 4.3), it omits the need for a secret sharing scheme and an auxiliary device to store $\overrightarrow{r}$. Despite these simplifications, this version still exhibits strong security performance against attackers who gain access to the vault's storage file, public information, and PII. However, it lacks resilience against attacks where the vault's storage file, public information, PII, and partial passwords are leaked.

The store and query phases are outlined below:

Store phase

- Encode the passwords $\{\pi_i\}_{i\in\mathcal{I}}$ in the password vault into $\{S_i\}_{i\in\mathcal{I}}$.
- Utilize the master password Φ to encrypt the seed S into the ciphertext C, where the seed $S = S_1||\cdots||S_{|\mathcal{I}|}$.

Query phase

- Decrypt C into S^* using Φ^*.
- Split S^* into $\left\{S_i^*\right\}_{i\in\mathcal{I}}$, which is decoded to $\{\pi_i^*\}_{i\in\mathcal{I}}$ with CPMTE.

The update phase remains the same as in [13].

The security of the simplified version against attackers who gain access to the vault's storage file, public information, and PII mirrors the security of our original scheme (Sect. 4.3) against attackers with $\overrightarrow{r}$ in the case of Attack-I. This is because both attackers can obtain candidate vaults by decrypting C with $\mathcal{D}_\Phi$, and the same encoder is employed in both schemes.

Conclusion

Our study is the first exploration of honey vault security in multiple leakage scenarios including the leak of PII and partial passwords contained within the real vault, apart from the compromise of the vault's storage file in the traditional single leakage scenarios. We propose various attack variants catering to multiple leakage scenarios. We construct a honey vault scheme and demonstrate its efficacy in thwarting these diverse attacks.

Appendix A: Proof of Theorem 1

Proof of Theorem 1 According to Theorem 3 in [12], $\Pr\left[S \mid \mathsf{V}_j^*, \mathrm{PII}\right] = \frac{k_1}{\Pr_{\mathrm{decoy}}\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]}$. Then, we have

$$\begin{aligned}&\Pr(\Phi^{j*} \mid \overrightarrow{r_t}, C, \mathrm{PII})\\ =\ &\frac{\Pr\left[\Phi^{j*}, \overrightarrow{r_t}, C, \mathrm{PII}\right]}{\Pr\left[\overrightarrow{r_t}, C, \mathrm{PII}\right]}\\ =\ &\frac{\Pr\left[\Phi^{j*}, \mathsf{V}_j^*, \overrightarrow{r_t}, C, \mathrm{PII}\right]}{\Pr\left[\overrightarrow{r_t}, C, \mathrm{PII}\right]}\\ =\ &\frac{\Pr\left[C \mid \Phi^{j*}, \mathsf{V}_j^*, \overrightarrow{r_t}, \mathrm{PII}\right]}{\Pr\left[\overrightarrow{r_t}, C, \mathrm{PII}\right]} \times \Pr\left[\Phi^{j*}, \mathsf{V}_j^*, \overrightarrow{r_t}, \mathrm{PII}\right],\end{aligned}$$

where

$$\begin{aligned}&\Pr\left[C \mid \Phi^{j*}, \mathsf{V}_j^*, \overrightarrow{r_t}, \mathrm{PII}\right]\\ =\ &\Pr\left[S \mid \mathsf{V}_j^*, \mathrm{PII},\right] \times \Pr\left[C \mid S, \Phi^{j*}, \overrightarrow{r_t}\right],\end{aligned}$$

$$
\begin{aligned}
&\Pr\left[\Phi^{j*}, \mathsf{V}_j^*, \overrightarrow{r_t}, \mathrm{PII}\right] \\
=\ &\Pr\left[\Phi^{j*}, \mathsf{V}_j^* \mid \mathrm{PII}\right] \times \Pr[\mathrm{PII}] \\
=\ &\Pr\left[\Phi^{j*} \mid \mathrm{PII}\right] \times \Pr\left[\mathsf{V}_j^* \mid \mathrm{PII}\right] \times \Pr[\mathrm{PII}] \\
=\ &\Pr\left[\Phi^{j*}\right] \times \Pr\left[\mathsf{V}_j^* \mid \mathrm{PII}\right] \times \Pr[\mathrm{PII}],
\end{aligned}
$$

where $\Pr\left[C \mid S, \Phi^{j*}, \overrightarrow{r_t}\right]$, $\Pr\left[\overrightarrow{r_t}, C, \mathrm{PII}\right]$, and $\Pr[\mathrm{PII}]$ are constants. The events $\left(\overrightarrow{r_t}, C, \mathrm{PII}\right)$ and PII are known and fixed facts when we attack. And $\Pr\left[C \mid S, \Phi^{j*}, \overrightarrow{r_t}\right]$ depends on the PBE scheme and the secret sharing scheme. Then

$$
\begin{aligned}
&\Pr(\Phi^{j*} \mid \overrightarrow{r_t}, C, \mathrm{PII}) \\
=\ &k \times \Pr\left[\Phi^{j*}\right] \times \frac{\Pr\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]}{\Pr_{\mathrm{PII}}\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]} \\
=\ &k \times \Pr\left[\Phi^{j*}\right] \times \frac{\Pr_{\mathrm{real}}\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]}{\Pr_{\mathrm{decoy}}\left[\mathsf{V}_j^* \mid \mathrm{PII}\right]}.
\end{aligned}
$$

□

Appendix B: Proof of Theorem 2

Proof of Theorem 2 Based on Theorem 3 presented in [12], the probability of encode π_i to S_i is estimated as

$$
\begin{aligned}
&\Pr_{\mathrm{encode}}\left[S_i \mid \{\pi_{i'}\}_{i'=1}^{i}, \mathrm{PII}\right] \\
=\ &\frac{1}{2^{ln_{\max}} \Pr_{\mathrm{PII}}\left[\pi_i \mid \{\pi_{i'}\}_{i'=1}^{i-1}, \mathrm{PII}\right]},
\end{aligned}
$$

where l is the storage overhead parameter, and $n_{\max}$ is the maximum length of generating sequences of π_i in the condition of $\{\pi_{i'}\}_{i'=1}^{i-1}$ and PII for $i \in \mathcal{I}$.

The attacker picks and guesses a target π_{i^*}. Let $\mathcal{I}^* = \mathcal{I} \backslash \{i^*\}$, then

$$
\begin{aligned}
&\Pr[\Phi \mid \mathsf{V}, \mathrm{PII}, C] \\
=\ &\max_{\substack{\mathcal{B} \subset \mathcal{I}^* \\ |\mathcal{B}| = t}} \left(\Pr[(\Phi_i)_{i \in \mathcal{B}} \mid \{\pi_i\}_{i \in \mathcal{I}}, \mathrm{PII}, C]\right) \\
=\ &\max_{\substack{\mathcal{B} \subset \mathcal{I}^* \\ |\mathcal{B}| = t}} \left(\Pr[(S_i)_{i \in \mathcal{B}} \mid \{\pi_i\}_{i \in \mathcal{I}}, \mathrm{PII}]\right) \\
\le\ &\sum_{\substack{\mathcal{B} \subset \mathcal{I}^* \\ |\mathcal{B}| = t}} \left(C_{n-1}^{t}\right)^{-1} \prod_{i \in \mathcal{B}} \Pr_{\mathrm{encode}}\left[S_i \mid \{\pi_{i'}\}_{i'=1}^{i}, \mathrm{PII}\right] \\
\le\ &\max_{i \in \mathcal{I}^*} \left(\Pr_{\mathrm{encode}}\left[S_i \mid \{\pi_{i'}\}_{i'=1}^{i}, \mathrm{PII}\right]\right)^{t} \\
=\ &\max_{i \in \mathcal{I}^*} \left(2^{ln_{\max}} \Pr\left[\pi_i \mid \{\pi_{i'}\}_{i'=1}^{i-1}, \mathrm{PII}\right]\right)^{-t} \\
=\ &\left(2^{ln_{\max}} \Pr_{\min_{i \in \mathcal{I}^*}}\left[\pi_i \mid \{\pi_{i'}\}_{i'=1}^{i-1}, \mathrm{PII}\right]\right)^{-t} \\
=\ &\left(2^{l} \min\{\Pr_{\mathrm{OP}}[*]\}\right)^{-n_{\max} t},
\end{aligned}
$$

where $\frac{1}{2^l} \ll \min\{\Pr_{\mathrm{OP}}[*]\}$ and $\{\Pr_{\mathrm{OP}}[*]\}$ is a set of all probabilities concluding the probability of the existence of the insert operation, the existence of the delete operation, the number of operations, and the operation character.

Then

$$
\begin{aligned}
&\Pr[\mathcal{A}wins] \\
\le\ &\max\{\Pr_G[\pi_{i^*}], \Pr[\Phi \mid \mathsf{V}, \mathrm{PII}, C], \frac{1}{p^{t-2} \times |\mathcal{D}_\Phi|}\} \\
\le\ &\max\{\Pr_G[\pi_{i^*}], \frac{1}{|\mathcal{D}_\Phi|}\} + \mathbf{negl}(\lambda).
\end{aligned}
$$

□

Acknowledgements
The authors would like to thank the reviewers for their valuable time.

Author contributions
Chao An and YuTing Xiao proposed the new honey vault scheme and drafted the manuscript. Rui Zhang participated in problem discussions and improvements of the manuscript. HaiHang Liu, Han Wu, and Chao An implemented the proposed scheme and attacks. All authors read and approved the manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (Nos. 62172404, 62172411, 61972094, 62202458).

Availability of data and materials
Due to ethical restrictions, supporting data is not available.

Declarations

Competing interest
The authors declare that they have no competing interest.

Author details
[1]Key Laboratory of Cyberspace Security Defense, No.19 Shucun Road, Haidian District, Beijing 100084, China. [2]Institute of Information Engineering, Chinese Academy of Sciences, No. 19 Shucun Road, Haidian District, Beijing 100084, China. [3]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China. [4]Independent, Hefei, China.

Received: 18 January 2024 Accepted: 21 March 2024
Published online: 04 October 2024

References

1. (2016) The password is dead, long live the password! https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2016/october/the-password-is-dead-long-live-the-password/
2. (2017) Passwords are not lame and they're not dead. https://it.toolbox.com/blogs/itmanagement/passwords-are-not-lamea nd-theyre-not-dead-heres-why-072417
3. (2018) All data breach sources. https://breachalarm.com/allsources
4. Abdelberi C, Ács G, Kâafar MA (2012) You are what you like! Information leakage through users' interests
5. Adowsett F (2016) What has been leaked: impacts of the big data breaches. https://rantfoundry.wordpress.com/2016/04/19/what-hasbeen-leaked-impacts-of-the-big-data-breaches/
6. Bojinov H, Bursztein E, Boyen X et al (2010) Kamouflage: loss-resistant password management, pp 286–302
7. Bonneau J, Schechter SE (2014) Towards reliable storage of 56-bit secrets in human memory, pp 607–623
8. Bonneau J, Herley C, van Oorschot PC et al (2012) The quest to replace passwords: a framework for comparative evaluation of web authentication schemes, pp 553–567
9. Bonneau J, Herley C, van Oorschot PC et al (2015) Passwords and the evolution of imperfect authentication. Commun ACM 58(7):78–87
10. Burnett M (2016) Is there life after passwords? https://medium.com/unhackable/is-there-life-after-passwords-290d50fc6f7d
11. Chatterjee R, Bonneau J, Juels A et al (2015) Cracking-resistant password vaults using natural language encoders, pp 481–498
12. Cheng H, Zheng Z, Li W et al (2019) Probability model transforming encoders against encoding attacks, pp 1573–1590
13. Cheng H, Li W, Wang P et al (2021) Incrementally updateable honey password vaults, pp 857–874
14. Das A, Bonneau J, Caesar M et al (2014) The tangled web of password reuse
15. Dong Q, Wang D, Shen Y et al (2022) Pii-psm: a new targeted password strength meter using personally identifiable information. In: International conference on security and privacy in communication systems. Springer, pp 648–669
16. Freeman D, Jain S, Dürmuth M et al (2016) Who are you? A statistical approach to measuring user authenticity
17. Goldman J (2013) Chinese hackers publish 20 million hotel reservations. http://www.esecurityplanet.com/hackers/chinese-hackerspublish-20-million-hotel-reservations.html
18. Golla M, Beuscher B, Dürmuth M (2016) On the security of cracking-resistant password vaults, pp 1230–1241
19. Grassi PA, Fenton JL, Newton EM et al (2017) Digital identity guidelines: authentication and lifecycle management. Technical report
20. Hackett R (2017) Yahoo raises breach estimate to full 3 billion accounts, by far biggest known. http://fortune.com/2017/10/03/yahoo-breach-mail/
21. Holmes A (2021) 533 million facebook users' phone numbers and personal data have been leaked online. https://www.businessinsider.com/stolen-data-of-533-million-facebook-users-leaked-online-2021-4
22. Juels A, Ristenpart T (2014) Honey encryption: security beyond the brute-force bound, pp 293–310
23. Kincaid J (2011) Dropbox security bug made passwords optional for four hours. https://techcrunch.com/2011/06/20/dropbox-security-bug-made-passwords-optional-for-four-hours/
24. Kincaid J (2014) iCloud data breach: hacking and celebrity photos. https://www.forbes.com/sites/davelewis/2014/09/02/icloud-data-breach-hacking-and-nude-celebrity-photos/
25. Krause EF (1986) Taxicab geometry: an adventure in non-Euclidean geometry. Courier Corporation
26. Levandowsky M, Winter D (1971) Distance between sets. Nature 234(5323):34–35
27. Levenshtein VI et al (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady, Soviet Union, pp 707–710
28. Li Y, Li Y, Chen X et al (2022) Pg-pass: targeted online password guessing model based on pointer generator network. In: 2022 IEEE 25th international conference on computer supported cooperative work in design (CSCWD). IEEE, pp 507–512
29. Ma J, Yang W, Luo M et al (2014) A study of probabilistic password models, pp 689–704
30. Mazurek ML, Komanduri S, Vidas T et al (2013) Measuring password guessability for an entire university, pp 173–186
31. Mignotte M (1983) How to share a secret? pp 371–375
32. Morris C (2021) Massive data leak exposes 700 million linkedin users information. https://fortune.com/2021/06/30/linkedin-data-theft-700-million-users-personal-information-cybersecurity/
33. Pal B, Daniel T, Chatterjee R et al (2019) Beyond credential stuffing: password similarity models using neural networks, pp 417–434
34. Pearman S, Zhang SA, Bauer L et al (2019) Why people (don't) use password managers effectively. In: Fifteenth symposium on usable privacy and security (SOUPS 2019), pp 319–338
35. Pham T (2015a) Anthem breached again:hackers stole credentials. http://duo.sc/2ene0Pr
36. Pham T (2015b) Four years later, anthem breached again: Hackers stole credentials. http://duo.sc/2ene0Pr
37. Pinkas B, Sander T (2002) Securing passwords against dictionary attacks, pp 161–170
38. Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613
39. Siegrist J (2015) LastPass hacked C identified early & resolved. https://blog.lastpass.com/2015/06/lastpass-security-notice.html/
40. Turner K (2016) Hacked dropbox login data of 68 million users is now for sale on the dark web. https://www.washingtonpost.com/news/the-switch/wp/2016/09/07/hacked-dropbox-data-of68-million-users-is-now-or-sale-on-the-dark-web/
41. Ur B (2016) Supporting password-security decisions with data
42. Wang D, Jian G, Huang X et al (2014) Zipf's law in passwords. Cryptology ePrint Archive, Report 2014/631. https://eprint.iacr.org/2014/631
43. Wang D, Zhang Z, Wang P et al (2016) Targeted online password guessing: an underestimated threat, pp 1242–1254
44. Wang D, Cheng H, Wang P et al (2018) A security analysis of honeywords
45. Wang D, Wang P, He D et al (2019) Birthday, name and bifacial-security: understanding passwords of Chinese web users, pp 1537–1555
46. Wang D, Zou Y, Dong Q et al (2022) How to attack and generate honeywords, pp 966–983
47. Weir M, Aggarwal S, de Medeiros B et al (2009) Password cracking using probabilistic context-free grammars, pp 391–405
48. Xie Z, Zhang M, Yin A et al (2020) A new targeted password guessing model, pp 350–368
49. Yan J, Blackwell A, Anderson R et al (2004) Password memorability and security: empirical results. IEEE Secur Privacy Mag 2(5):25–31

Yuting Xiao is the corresponding author of this paper. She is an assistant researcher in the Key Laboratory of Cyberspace Security Defense, Beijing, China, and the Institute of Information Engineering, Chinese Academy of Sciences, China. She obtained her PhD from the Institute of Information Engineering, Chinese Academy of Sciences. Her main research interests include authenticated key exchange and multi-party computing.