Delft University of Technology

Document Version
Final published version

Citation (APA)
Xu, P., Rao, T., Wang, W., Lu, Z., & Liang, K. (2025). Power of union: Federated honey password vaults against differential attack. *Computers and Security*, *157*, Article 104592. https://doi.org/10.1016/j.cose.2025.104592

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states "Dutch Copyright Act (Article 25fa)", this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Green Open Access added to TU Delft Institutional Repository
as part of the Taverne amendment.

More information about this copyright law amendment
can be found at https://www.openaccess.nl.

Otherwise as indicated in the copyright section:
the publisher is the copyright holder of this work and the
author uses the Dutch legislation to make this work public.

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full length article

Power of union: Federated honey password vaults against differential attack

Peng Xu[a,b], Tingting Rao[a], Wei Wang[c,*], Zhaojun Lu[a,b], Kaitai Liang[d]

[a] Hubei Key Laboratory of Distributed System Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[b] Jinyinhu Laboratory, Wuhan 430040, China
[c] Cyber-Physical-Social Systems Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
[d] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 Delft, The Netherlands

ARTICLE INFO

Keywords:
Privacy
Password security
Honey password vault
Distribution transforming encoder
Differential attack

ABSTRACT

The honey password vault is a promising method for managing user passwords and mitigating password-guessing attacks by creating plausible-looking decoy password vaults. Recently, various methods, such as Chatterjee-PCFG (IEEE S&P'15), Golla-Markov (ACM CCS'16), and Cheng-IUV (USENIX Security'21), have been proposed to construct the cornerstone of honey password vaults, known as the distribution transforming encoder (DTE). These innovations significantly enhance the security and functionality of each kind of DTE. However, our findings indicate that when users employ multiple honey password vaults of distinct DTEs to manage their passwords, a passive attacker can easily compromise user passwords by exploiting differences among those DTEs. Consequently, we propose the *differential attack* targeting existing honey password vaults. The extensive experimental results confirm the effectiveness of this attack, distinguishing real from decoy password vaults with accuracy from 99.13% to 100.00%. In response, we design a novel, collaborative approach to train DTE, called *federated DTE model*, and construct a secure honey password vault. This strategy markedly bolsters security, reducing the differential attack's distinguishing accuracy to approximately 52.41%, nearing the ideal threshold of 50.00%. Our findings emphasize the need for collaborative strategies to maintain password security to combat advanced cyber threats.

1. Introduction

Users widely use passwords to prove their identities and obtain authorization to access Internet resources (Wang et al., 2023c; David and Wool, 2021; Xie et al., 2024; Galbally et al., 2017; Zhang et al., 2021). In practice, users usually leverage password vaults to organize their passwords (Lyastani et al., 2018; Gasti and Rasmussen, 2012). Meanwhile, according to the survey (Munyendo et al., 2023), most of the users maintain more than one password vault. For example, a user can manage his/her passwords in a mobile vault application, like Apple Keychain (2024); at the same time, he/she also stores passwords in a web-based vault, like Chrome Password Manager (Google, 2025), to ensure that he/she can access his/her passwords conveniently at any time. The traditional password vault encrypts users' passwords using password-based encryption (PBE) with the users' master password (1password, 2021; Enpass, 2025; Han et al., 2016). However, the tendency to create weak master passwords by users significantly heightens the risk of vault breaches (Wang et al., 2023b; Xu et al., 2023; Wang et al., 2023a; Pasquini et al., 2020; Zhang et al., 2022). Upon obtaining an encrypted password vault, an attacker can compromise the user's passwords by brute-force guessing all possible master passwords (Li et al., 2014, 2024). Specifically, if the guessing master password is incorrect, the attacker will obtain meaningless random characters (Li et al., 2023); oppositely, he/she can obtain a meaningful password vault and identify the correct master password. In real applications, password vaults consistently experience security breaches (PasswdTeam, 2025; Islam et al., 2023). For instance, 90% of passwords stored in Chrome Password Manager suffer data leakage (Chrome, 2022; Thomas et al., 2019) while 2.6 billion iPhone users' passwords have been compromised in two years (AppleReport, 2023).

The state-of-the-art solution against the brute-force guessing attack is using honey password vault (Chatterjee et al., 2015; Golla et al., 2016; Cheng et al., 2019, 2021; Rao et al., 2023). In this solution, when an attacker guesses incorrect master passwords and encrypts the vault, he/she will obtain plausible-looking decoy password vaults indistinguishable from the real one. Distribution transforming encoder (DTE) is the main component of honey password vaults to achieve

* Corresponding author.
E-mail addresses: xupeng@mail.hust.edu.cn (P. Xu), raott@hust.edu.cn (T. Rao), viviawangwei@hust.edu.cn (W. Wang), lzj_cse@hust.edu.cn (Z. Lu), kaitai.liang@tudelft.nl, kaitai.liang@utu.fi (K. Liang).

https://doi.org/10.1016/j.cose.2025.104592
Received 16 January 2025; Received in revised form 29 May 2025; Accepted 4 July 2025
Available online 23 July 2025
0167-4048/© 2025 Elsevier Ltd. All rights reserved, including those for text and data mining, AI training, and similar technologies.

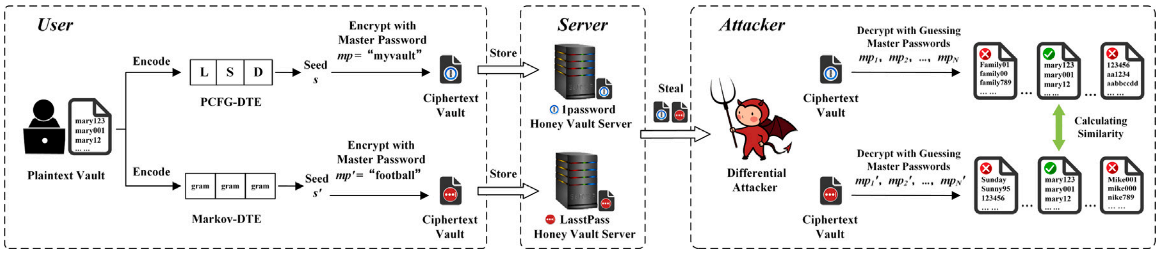


Fig. 1. The differential attack scenario against honey password vaults. Different honey vault systems 1*password* and *LastPass* encrypt the same plaintext password vault from a user. Suppose 1*password* and *LastPass* use the PCFG-DTE (Chatterjee et al., 2015) and Markov-DTE (Golla et al., 2016), respectively. The PCFG-DTE encodes passwords using a trained grammar tree, and the Markov-DTE encodes passwords using an n-gram model. (More details about the differences between PCFG-DTE and Markov-DTE are described in Section 2.3). 1*password* and *LastPass* generate and store their ciphertext password vaults C^1 and C^2 in two cloud servers, respectively.

this indistinguishability (Jaeger et al., 2016; Juels and Ristenpart, 2014; Juels and Rivest, 2013). DTE runs through encoder–decoder configurations. Initially, the encoder transforms the plaintext password vault into a seed. PBE encrypts this seed into a ciphertext. Upon decryption, PBE converts the ciphertext back to a seed. The decoder subsequently transforms into a plaintext password vault. Importantly, if the decryption process involves an incorrect master password, the DTE is designed to produce decoy vaults that appear valid and are entirely fictitious.

Our Motivation. Various password applications providing diverse services have recently led numerous users to store their passwords in multiple vaults for convenience (Oesch and Ruoti, 2020; Hou and Wang, 2023). As illustrated in Fig. 1, we find that when a user stores his/her plaintext vault in two honey password vault systems constructed with diverse DTEs, such as PCFG-DTE (Chatterjee et al., 2015) and Markov-DTE (Golla et al., 2016), these honey password vault systems are vulnerable to the differential attack. Although DTEs have advantages in generating plausible-looking decoys, we found significant differences in the decoys produced by various DTEs, which introduce severe security risks to honey password vaults. When an attacker obtains various ciphertext password vaults from multiple applications, he/she can compare the decoys decrypted by these DTEs and identify the correct vault with considerable accuracy. We have identified that the discrepancies among DTEs allow attackers to immediately discern the real password vault from the decoys. The major novelty of this paper is to discover the *differential attack* and propose a new and secure strategy to train DTEs for various honey password vaults.

All current honey password vault systems are vulnerable to differential attacks because the DTEs employed by different honey password vaults exhibit great diversity. The initial implementation of a honey password vault leveraging the DTE framework was proposed by Chatterjee et al. (2015). They construct a DTE using the PCFG model (Han et al., 2021; Houshmand et al., 2015). Subsequently, Golla et al. (2016) designed an adaptive DTE to generate decoy password vaults. After that, Cheng et al. (2021) investigated the incrementally updated measure for honey vault utilizing a conditional probability model. The SMART scheme (Rao et al., 2023) designed by Rao et al. focuses on the dynamic of the master password, and the DTE of SMART is the same as the work of Cheng et al. (2021). The above various DTEs depend on their models and training datasets: (1) different DTE models have different ways of generating decoy password vaults; (2) DTEs with different training datasets generate decoy password vaults having statistical differences, even if these DTEs have the same models. By analyzing the similarity between the generated vaults, attackers can identify the real vault.

Our Ideas. The essence of addressing differential attacks is eliminating the discrepancies among DTEs to create uniform decoy password vaults. We propose a federated framework for training different DTEs to counteract the differential attack. The federated framework removes the variances among the existing DTE models and their training datasets and constructs a general DTE for various honey password vaults. The primary challenge is integrating multiple types of DTEs. We first normalize all kinds of DTEs as a generic conditional probability model. Subsequently, we construct a federated learning method to train the generic model according to the various DTEs. In contrast to the traditional honey password vault, our method significantly mitigates the differential attack. Moreover, our federated DTE model learns the datasets of various honey password vaults, enhancing generalization performance and bolstering security.

Our Contributions. We propose the *differential attack* against honey password vaults according to their DTEs' differences. Since the variances among DTEs make honey password vaults generate the decoy vaults with different distributions. Such differences enable attackers to identify the decoy vaults and find the correct master password. Specifically, the differential attack is effective for two scenarios in real deployments. In the first scenario called Example I, once attackers obtain the ciphertext password vaults from any two honey password vaults, where their DTEs use the same model but distinct datasets, they can immediately distinguish the real password vault from decoys. As for the second scenario Example II, if two DTEs use distinct models and training datasets, attackers can also identify the real password vault by comparing the differences between the decoy password vaults generated by those two DTEs.

As a countermeasure to the differential attack, we construct a new framework to build the federated DTE from a number of different DTEs and establish the *federated honey password vaults* as shown in Fig. 2. This framework mainly consists of four parts: (1) each participant vault trains the local DTE model using its local dataset; (2) each participant vault uploads the trained model parameters to the server; (3) the server aggregates the trained model parameters of all participant vaults into a global DTE model; (4) the server transmits the global DTE model to all participant vaults, and then each participant vault uses the global DTE model to construct its federated honey password vault as the traditional vault does. This approach leverages the collective model parameters of multiple participant honey password vaults while safeguarding the privacy of their respective local datasets.

We conduct empirical experiments on the differential attack and its countermeasure. The differential attack can achieve 99.13%–100.00% accuracy in distinguishing the real password vault from decoys, yielding a significant attack performance against current honey password vaults. Our federated honey password vault reduces the performance of the differential attack to 52.41% (the optimal result is 50.00%). We also assess the capability of our federated honey password vault against previous attacks, such as the KL divergence attack (Golla et al., 2016), the encoding attack (Cheng et al., 2019), and the intersection attack (Cheng et al., 2021). All evaluations confirm that our work has achieved significant security improvement.

The remainder of this paper is structured as follows. Section 2 provides background and related works, introducing honey encryption and analyzing various honey password vaults and their limitations. Section 3 defines the differential attack model and its priority function and presents two cases to implement the differential attack. Section 4

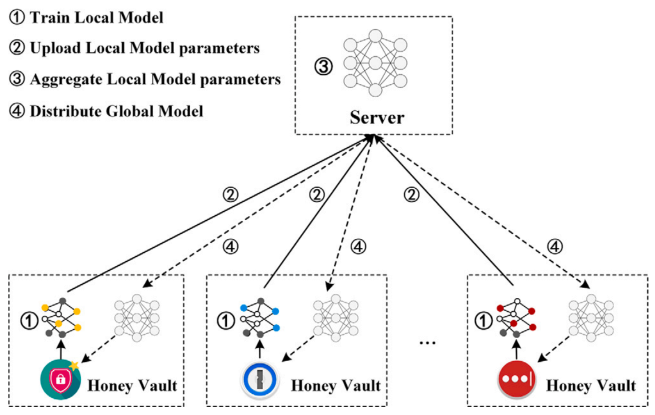

① Train Local Model
② Upload Local Model parameters
③ Aggregate Local Model parameters
④ Distribute Global Model
③
Server
②
②
②
④
④
④
①
Honey Vault
①
Honey Vault
...
①
Honey Vault

Fig. 2. The Federated DTE Framework.

details our proposed federated DTE model. Finally, Section 5 evaluates the effectiveness of both the differential attack and the federated DTE model by comparing their performance with the current leading works.

2. Backgrounds and related work

2.1. Password-based encryption

Password-based encryption (PBE) is the most widely used technique to encrypt plaintext with an encryption key derived from the user-created password (Dong et al., 2021). PBE generates an encryption key by algorithm $\mathsf{KDF}(pwd, sa)$, where sa is a random salt, pwd is a user-created password, and KDF is a key derivation function based on the user's passwords, utilizing the hashing algorithm SHA-256. Algorithm $\mathsf{KDF}(pwd, sa)$ usually executes the SHA-256 function c times, where c is a constant number, e.g., $c = 10{,}000$. PBE generates a ciphertext C with the encryption key. Current research works (Wang et al., 2016) show that PBE is highly susceptible to brute-force attacks. By conducting q times brute-force guessing, the attacker can successfully decrypt a single ciphertext with a probability of $q/c2^{\mu}$, where the guessing passwords are selected from a distribution with a min-entropy of μ. For example, given a PBE ciphertext $C = \mathsf{PBE.Enc}(pwd, msg)$ with the inputs of message msg and password pwd, where pwd and msg usually satisfy some known distributions. The brute-force guessing attacker recovers msg by decrypting C with every guessing password candidates $\mathcal{PWD} = \{pwd_1, pwd_2, \ldots, pwd_N\}$ and yield the corresponding candidate messages $\mathcal{MSG} = \{msg_1, msg_2, \ldots, msg_N\}$. Suppose that msg represents an 11-digit phone number encoded via ASCII. The probability of $msg_i \neq msg$ being a valid ASCII encoding of an 11-digit string is negligible, less than $(10/256)^{11} < 2^{-74}$. Hence, once the attacker cracks to get the junk characters, it can tell that the pwd_i is incorrect.

2.2. Honey encryption

Honey encryption (Juels and Ristenpart, 2014) is an advanced cryptographic primitive to resist the above brute-force guessing attack. Honey encryption can prevent attackers from successfully recovering the real message, even if they try to decrypt a ciphertext with all possible guessing passwords (Tian et al., 2023; Li et al., 2021). The main framework of honey encryption is *DTE-then-PBE*. The DTE is made up of an encoder and a decoder. When inputting a message msg, the encoder samples a seed S. Then, PBE encrypts S with an input password pwd and yields a ciphertext C. Honey encryption uses reverse procedures when decrypting a ciphertext. Suppose a brute-force guessing attacker wants to recover the message of a honey encryption ciphertext C with every guessing password candidate $\mathcal{PWD} = \{pwd_1, pwd_2, \ldots, pwd_N\}$, where the guessing passwords satisfy a distribution with min-entropy μ. The attacker will yield the corresponding candidate messages $\mathcal{MSG} = \{msg_1, msg_2, \ldots, msg_N\}$. The attacker cannot identify the real message from $\mathcal{MSG}$ since all messages in $\mathcal{MSG}$ have the same distribution. Consequently, honey encryption limits the attacker's probability of guessing the correct password to be $1/2^{\mu}$. And honey encryption is more secure than PBE. Moreover, a well-designed DTE that fits the specific applications will make honey encryption more effective.

2.3. Honey password vault

Honey password vault (Duan et al., 2024) is a specialized application of honey encryption to enhance the security of password storage systems. The main concept is to generate convincing decoy password vaults when attackers attempt to brute-force decrypt the ciphertext by guessing the user's master password. Hence, they cannot confirm whether the decrypted password vault is real and fail to find the correct master password.

Chatterjee et al. (2015) used the probabilistic context-free grammars (PCFG) to model the first DTE. Then, they constructed the cracking-resistant password vault Nocrack. The PCFG model is a parse tree that divides a password into three grammar structures: letters, symbols, and numbers (Li and Zeng, 2021). The probability of each grammar structure relies on the training password datasets. The PCFG-DTE encodes a password vault into a seed according to the probabilities of all grammar structures of the password. To decode a seed, the PCFG-DTE converts the seed to several grammar structures and obtains the password. Since the parse tree is small, the PCFG-DTE is more likely to generate the same grammar structures and obtain similar passwords.

Golla et al. (2016) utilized the n-gram Markov model to construct a DTE called Markov-DTE. The n-gram Markov model divides each password into several grams of length n (Yang et al., 2018; Li et al., 2017). The probability of each gram also relies on the training password datasets. The Markov-DTE encodes a password vault into a seed according to a sequence of probabilities of several grams of the password. To decode a seed, the Markov-DTE converts the seed to several grams and obtains the password. In order to generate more real decoy vaults, the Markov-DTE uses an adaptive mechanism that adjusts the distribution of decoy vaults to be more similar to that of the real password vault. As a result, the attacker is unable to differentiate between the distributions of the real password vault and decoys.

Cheng et al. (2021) designed an incremental updateable honey password vault based on the conditional probability DTE. The conditional probability DTE encodes a password vault using the reuse method. The reuse method divides a password into a base and a suffix. The base is the prefix substrings that often appear in most passwords, and the suffix is the other part, except for the base. The probabilities of the base and the suffix rely on the training password datasets. The conditional probability DTE encodes a password vault into a seed according to a sequence of probabilities of the base and the suffix of the password. To decode a seed, the conditional probability DTE converts the seed to the base and the suffix and obtains the password. In order to realize the incremental updateable mechanism, the conditional probability DTE encrypts the added or modified password and then directly adds the generated ciphertext to the tail of the ciphertext password vault. The incremental update mechanism removes an old password by marking it as deleted in plaintext, without modifying the ciphertext of the password vault.

3. Differential attack

The security of a honey password vault scheme depends on its DTE, which is constructed with a chosen model, like PCFG, Markov, and conditional probability, and its training dataset. This section will introduce the details of the differential attack and two examples to explain why this attack is effective.

Algorithm 1 Differential Attack Process.

```
DifferentialAttack($\mathcal{C}^1$, $\mathcal{C}^2$, DTE$_1$, DTE$_2$)
 1: Create two candidate master password lists $\mathcal{MP}^1$ and $\mathcal{MP}^2$, each containing $N$;
 2: for $i = 1$ to $N$ do
 3:     Decrypt each ciphertext $\mathcal{C}^1[j] \in \mathcal{C}^1$ by algorithm PBE.Dec($\mathcal{C}^1[j], \mathcal{MP}^1[i]$) and generate a seed list $\mathcal{S}^1[i]$;
 4:     Decrypt each ciphertext $\mathcal{C}^2[j] \in \mathcal{C}^2$ by algorithm PBE.Dec($\mathcal{C}^2[j], \mathcal{MP}^2[i]$) and generate a seed list $\mathcal{S}^2[i]$;
 5:     Decode each seed $\mathcal{S}^1[i,j] \in \mathcal{S}^1[i]$ by decoder DTE$_1$.Decoder($\mathcal{S}^1[i,j]$) and generate a plaintext password vault $\mathcal{V}^1[i]$;
 6:     Decode each seed $\mathcal{S}^2[i,j] \in \mathcal{S}^2[i]$ by decoder DTE$_2$.Decoder($\mathcal{S}^2[i,j]$) and generate a plaintext password vault $\mathcal{V}^2[i]$;
 7: end for
 8: For any two password vaults $\mathcal{V}^1[i] \in \mathcal{V}^1$ and $\mathcal{V}^2[j] \in \mathcal{V}^2$, Compute their similarity value $r_{i,j}$ by priority function Priority($\mathcal{V}^1[i], \mathcal{V}^2[j]$, DTE$_1$, DTE$_2$);
 9: Let $r_{x,y}$ be the maximum one in $\{r_{i,j} | i,j \in [1,N]\}$;
10: return $\mathcal{MP}^1[x]$ and $\mathcal{MP}^2[y]$ as the guessing results;

Priority($\mathcal{V}^1[i], \mathcal{V}^2[j]$, DTE$_1$, DTE$_2$)
 1: Compute the encode probability of each password in $\mathcal{V}^1[i]$ and $\mathcal{V}^2[j]$ with DTE$_1$ and DTE$_2$, respectively;
 2: Let $\mathcal{P}^1[i]$ and $\mathcal{P}^2[j]$ denote the encode-probability lists of $\mathcal{V}^1[i]$ and $\mathcal{V}^2[j]$, respectively;
 3: Compute the similarity value $r_{i,j}$ between $\mathcal{P}^1[i]$ and $\mathcal{P}^2[j]$ using Jensen–Shannon divergence;
 4: return $r_{i,j}$;
```

3.1. Attack process

Without loss of generality, suppose a user stores his/her password vault in two honey password vaults of different DTEs DTE$_1$ and DTE$_2$, where DTE$_1$ is trained with model $model_1$ and training dataset d_1, and DTE$_2$ is trained with model $model_2$ and training dataset d_2. The size of his/her password vault is M. Let $\mathcal{C}^1$ and $\mathcal{C}^2$ be the two ciphertext vaults generated by those two honey password vaults with two master passwords, respectively. Note that the user chose these two master passwords (could be the same), and both DTE$_1$ and DTE$_2$ are public in practice.

Given two ciphertext vaults $\mathcal{C}^1$ and $\mathcal{C}^2$, a differential attacker does the following steps to guess the user's master passwords and break out the user's password vault:

1. Create two lists of candidate master passwords $\mathcal{MP}^1$ and $\mathcal{MP}^2$ targeting two ciphertext password vaults $\mathcal{C}^1$ and $\mathcal{C}^2$, respectively (as shown in Algorithm 1, Step 1; the generate method is similar with the previous works, like (Golla et al., 2016; Cheng et al., 2019, 2021); our experiment will mention the details in Section 5);
2. Recovery all possible plaintext password vaults $\mathcal{V}^1$ and $\mathcal{V}^2$ from $\mathcal{C}^1$ and $\mathcal{C}^2$ using every candidate master password in $\mathcal{MP}^1$ and $\mathcal{MP}^2$, respectively (as shown in Algorithm 1, Steps 2–7);
3. Compute the similarity of any two plaintext password vaults $\mathcal{V}^1[i] \in \mathcal{V}^1$ and $\mathcal{V}^2[j] \in \mathcal{V}^2$ by computing their Jensen–Shannon divergence and returning the two vaults having the maximum similarity as the final guessing results (as shown in Algorithm 1, Steps 8 and 9).

In practice, a user may store his/her passwords in more than two honey password vaults. We will illustrate that this case can be effectively reduced to the following Examples I and II, and experiment with this case in Section 5.6.

3.2. Priority function

In the differential attack as shown in Algorithm 1, the key step of the priority function Priority($\mathcal{V}^1[i], \mathcal{V}^2[j]$, DTE$_1$, DTE$_2$) is to compute the similarity of two possible plaintext password vaults by calculating the Jensen–Shannon divergence of these two vaults' encode probabilities. Given two encode-probability lists $\mathcal{P}^1[i]$ and $\mathcal{P}^2[j]$, the details to compute their Jensen–Shannon divergence (Menéndez et al., 1997) are as follows.

Given a password $p \in \mathcal{V}^1[i] \cup \mathcal{V}^2[j]$, let $\mathcal{P}^1[i](p)$ and $\mathcal{P}^2[j](p)$ denote the probabilities of p in $\mathcal{P}^1[i]$ and $\mathcal{P}^2[j](p)$, respectively. The Jensen–Shannon divergence of $\mathcal{P}^1[i]$ and $\mathcal{P}^2[j]$ is to compute

$$JS(\mathcal{P}^1[i] \parallel \mathcal{P}^2[j])$$
$$= \frac{1}{2} \sum_{p \in \mathcal{V}^1[i] \cup \mathcal{V}^2[j]} \mathcal{P}^1[i](p) \log\left(\frac{2 \cdot \mathcal{P}^1[i](p)}{\mathcal{P}^1[i](p) + \mathcal{P}^2[j](p)}\right)$$
$$+ \frac{1}{2} \sum_{p \in \mathcal{V}^1[i] \cup \mathcal{V}^2[j]} \mathcal{P}^2[j](p) \log\left(\frac{2 \cdot \mathcal{P}^2[j](p)}{\mathcal{P}^1[i](p) + \mathcal{P}^2[j](p)}\right).$$

Furthermore, the similarity value between $\mathcal{V}^1[i]$ and $\mathcal{V}^2[j]$ is defined as

$$r_{i,j} = \exp(-JS(\mathcal{P}^1[i] \parallel \mathcal{P}^2[j])),$$

where exp is a natural exponential function. The higher similarity value indicates that the corresponding vaults are more likely to be real.

3.3. Two examples

DTE is the fundamental element of the honey password vault. Notable DTE models include the PCFG model, Markov model, and conditional probability model. Consider a user who stores his passwords in two honey password vaults with distinct DTEs DTE$_1$ and DTE$_2$, respectively. In Example I, we suppose that DTE$_1$ and DTE$_2$ have the same model but different datasets. In Example II, we suppose that DTE$_1$ and DTE$_2$ have different models and training datasets. Note that in practice, two different honey password vaults necessarily have different training datasets. Further elaboration on scenarios involving more than two honey password vaults is discussed in Section 5.6.

Example I. Suppose the models of both DTE$_1$ and DTE$_2$ are the PCFG model. The PCFG model parses a password as a combination of letters, numbers, and symbols. Let L, S, and D denote the successive letters, symbols, and numbers, respectively. For example, the password "MyVault@456" is denoted by $L_8 S_1 D_3$, where the subscripts indicate the number of successive letters, symbols, and numbers, respectively. The PCFG model encodes the password "MyVault@456" according to the probabilities of $\Pr(L_8 S_1 D_3)$, $\Pr(L_8 = MyVault)$, $\Pr(S_1 = @)$, and $\Pr(D_3 = 456)$, and these probabilities are decided by the corresponding training dataset.

Suppose differential attackers guess the correct master passwords; they will decrypt out the correct seeds from those two honey password vaults. Then, DTE$_1$ and DTE$_2$ can decode the same and correct password vaults. Otherwise, he/she will obtain two random seeds according to the PBE's security. Since DTE$_1$ and DTE$_2$ have different training datasets, they will decode two different decoy password vaults. Consequently, dataset differences perpetuate inconsistencies in decoy password vaults, underscoring the effectiveness of differential attacks against DTEs trained on different datasets of the same DTE model.

Example II. Suppose the models of DTE$_1$ and DTE$_2$ are the PCFG and Markov models, respectively. In contrast to the above PCFG model, the Markov model processes the password "MyVault@456" utilizing the n-grams ($n = 4$) model. It divides the password into several segments as $\{w_1 = MyVa, w_2 = yVau, w_3 = Vaul, w_4 = ault, w_5 = ult@, w_6 = lt@4, w_7 = t@45, w_8 = @456, w_9 = \wedge MyV, w_{10} = 456\$\}$, where $\wedge$ and \$ represent the start and the end symbols, respectively. The

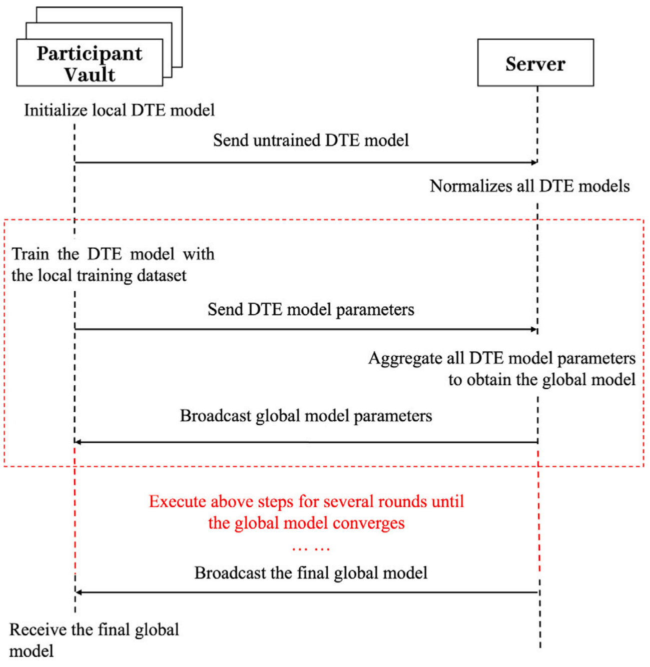

Participant Vault
Server
Initialize local DTE model
Send untrained DTE model
Normalizes all DTE models
Train the DTE model with the local training dataset
Send DTE model parameters
Aggregate all DTE model parameters to obtain the global model
Broadcast global model parameters
Execute above steps for several rounds until the global model converges
… …
Broadcast the final global model
Receive the final global model

Fig. 3. The federated DTE model.

probability of a segment only depends on n prefixes segments. The DTE$_2$ encode the password "MyVault@456" according to the probability $\Pr(w_1 w_2 \dots w_{10}) = \Pr(w_1) \cdot \Pr(w_2|w_1) \cdot \Pr(w_3|w_1 w_2) \dots \Pr(w_9|w_5 \dots w_8) \cdot \Pr(w_{10}|w_6 \dots w_9)$. These probabilities rely on the corresponding training dataset.

Suppose differential attackers guess the correct master passwords; they will decrypt out the correct seeds, and DTE$_1$ and DTE$_2$ can decode the same and correct password vaults. However, if the guesses are incorrect, he/she will obtain two random seeds, and the divergence in the models between DTE$_1$ and DTE$_2$ and their different training datasets results in significantly distinct decoy password vaults. DTE$_1$ parses and encodes a password based on grammatical structures. DTE$_2$ decomposes a password into fixed-length segments. Compared with Example I, the different models and training datasets give the differential attacker more confidence to distinguish a real password vault from a decoy one.

4. Federated DTE

Previous studies proposed multiple types of DTE. When employing these different DTEs to build honey password vaults, all those vaults are vulnerable to the differential attack. This section innovatively proposes a federated DTE model to resist the differential attack. In practice, all honey password vaults can employ the federated DTE model to build their systems. The employment measure is the same as the traditional method introduced in Sections 2.2 and 2.3. Hence, this section pays attention to the construction of the federated DTE.

4.1. Federated DTE model

The federated DTE model consists of several key steps, as shown in Fig. 3. At first, each honey password vault initializes and sends its untrained DTE model to a server, and the server normalizes the received untrained DTE models. Secondly, each honey password vault trains its initialized DTE model with the local training dataset and uploads the trained DTE model parameters to the server. Thirdly, the server aggregates the received trained DTE models' parameters to obtain a global model. After several rounds of the local training of honey password vaults and the server's aggregation, the server gradually converges the model parameters and obtains the final global model. Finally, the server returns the final global model to all vaults. Each vault uses the final global model to construct its honey password vault as the traditional method does.

Algorithm 2 Federated DTE Training Process

```
FederatedTraining($\{\mathcal{V}^i\}$; $\mathcal{A}$)
Setup Phase:
  1: Each vault $\mathcal{V}^i$ initializes and sends the untrained local model $model_i$ to the server $\mathcal{A}$;
  2: The server $\mathcal{A}$ normalizes the received models $\{model_i\}$ from $\{\mathcal{V}^i\}$ to initialize a global DTE model $\mathcal{G}^0$;
The $t$th Round Training Phase:
  1: Each vault $\mathcal{V}^i$ implements the local training $\mathcal{G}^{t,i} = \mathsf{LocalTraining}(\mathcal{G}^{t-1}, model_i, d_i)$ and send $(\mathcal{G}^{t,i}, n_i)$ to the server $\mathcal{A}$, where $n_i$ is the size of the local training dataset for vault $\mathcal{V}^i$;
  2: The server $\mathcal{A}$ aggregates $\{\mathcal{G}^{t,i}\}$ and generates a new global DTE model $\mathcal{G}^t = \mathsf{GlobalAggregating}(\{(\mathcal{G}^{t,i}, n_i)\})$;
  3: The server $\mathcal{A}$ send $\mathcal{G}^t$ to all vaults;
Final Phase:
  1: Each vault $\mathcal{V}^i$ constructs its honey password vault system with the final global DTE model $\mathcal{G}^T$;
```

According to the above main ideas, we can define our federated DTE model as follows. The model has two kinds of parties: Vaults and a Server $\mathcal{A}$. Let $\mathcal{V}^i$ denote the ith vault, and $\{\mathcal{V}^i\}$ denote all vaults joining the federated DTE model. To generate the final global model, the federated DTE model defines a T-round federated training process between all vaults and the server, named $\mathsf{FederatedTraining}(\{\mathcal{V}^i\}; \mathcal{A})$ as shown in Algorithm 2. All participant vaults $\{\mathcal{V}^i\}$ aim to construct the federated DTE model without exposing their training datasets to the other vaults or the server. $\mathsf{FederatedTraining}(\{\mathcal{V}^i\}; \mathcal{A})$ consists of the following three phases:

1. In the setup phase, each vault $\mathcal{V}^i$ takes its selected passwords as the local training dataset d_i and chooses a DTE model $model_i$ as the traditional honey password vault does. Note that $model_i$ can be a well-known model, such as the PCFG model, the Markov model, and the conditional probability model, and d_i can be a public password dataset, such as Pastebin, RockYou, Yahoo, Myspace, Gmail, and LinkedIn. Upon receiving all vaults' models, the server $\mathcal{A}$ normalizes these models to an initial global DTE model $\mathcal{G}^0$;
2. In the training phase, all vaults, and the server implement a T-round training to obtain a final (or convergence) global DTE model $\mathcal{G}^T$. In each round $t \in [1, T]$, each vault $\mathcal{V}^i$ implements a local training process, namely $\mathcal{G}^{t,i} = \mathsf{LocalTraining}(\mathcal{G}^{t-1}, model_i, d_i)$, where $\mathcal{G}^{t-1}$ denotes the current global DTE model. After receiving all vaults' $\{(\mathcal{G}^{t,i}, n_i)\}$, the server $\mathcal{A}$ aggregates $\{\mathcal{G}^{t,i}\}$ and generates a new global DTE model by running algorithm $\mathcal{G}^t = \mathsf{GlobalAggregating}(\{(\mathcal{G}^{t,i}, n_i)\})$, where n_i denotes the size of the local training dataset of vault $\mathcal{V}^i$; finally, the server $\mathcal{A}$ sends $\mathcal{G}^t$ to all vaults;
3. In the final phase, each vault $\mathcal{V}^i$ uses $\mathcal{G}^T$ to establish its honey password vault system with PBE as the traditional method does.

The following sections introduce three essential components in the federated DTE model training process: model normalization, local model training, and global model aggregating.

4.2. Model normalization

To construct an initialized global DTE model $\mathcal{G}^0$, the server normalizes each received untrained DTE model $model_i$ from the participant

Algorithm 3 Local Training Process

```
LocalTraining($\mathcal{G}^{t-1}$, $model_i$, $d_i$)
1: Initialize $\mathcal{G}^{t,i} = \mathcal{G}^{t-1}$;
2: for each local training epoch $e = 1$ to $E$ do
3:     for each batch $B \subset d_i$ do
4:         Compute gradient $\mathcal{P}_i = \nabla\mathcal{L}(\mathcal{G}^{t,i}, B)$;
5:         Set $\mathcal{G}^{t,i} = \mathcal{G}^{t,i} - \eta\mathcal{P}_i$, where $\eta$ is the learning rate;
6:     end for
7: end for
8: return $\mathcal{G}^{t,i}$;
```

Algorithm 4 Global Aggregating Process

```
GlobalAggregating($\{(\mathcal{G}^{t,i}, n_i)\}$)
1: Compute the total training dataset size $n = \sum_i n_i$;
2: Generate the aggregated model by computing $\mathcal{G}^t = \sum_i \frac{n_i}{n}\mathcal{G}^{t,i}$;
3: return $\mathcal{G}^t$;
```

vaults $\mathcal{V}^i$ to a conditional probability model. The conditional probability model parses a password as a base and a suffix. The base is the prefix substrings that often appear in most passwords, and the "suffix" is the other part except for the base. For example, given a password "$acdf2345$", the conditional probability model sets "$acdf$" as the base and "2345" as the suffix, respectively. The conditional probability model encodes the password according to the probability of "$acdf2345$", where $\Pr(acdf2345) = \Pr(acdf) \cdot \Pr(2345|acdf)$, and the probabilities $\Pr(acdf)$ and $\Pr(2345|acdf)$ depend on all participant vaults' datasets and will be computed in the training phase.

Suppose the model $model_i$ chosen by $\mathcal{V}^i$ is a PCFG one. The server normalizes $model_i$ to a conditional probability model as follows. In a PCFG model, the probability of a password is the product of the probabilities of grammar structures. The conditional probability of any grammar structure is independent of others. For example, the server normalizes the PCFG model as $\Pr(2345|abcdef) = \Pr(2345)$. In the training phase, the vault $\mathcal{V}^i$ will compute the probabilities $\Pr(2345)$ and $\Pr(abcdef)$ according to its dataset and send these probabilities to the server.

When the chosen model $model_i$ is an n-gram Markov one, the server normalizes $model_i$ to a conditional probability model as follows. In an n-gram Markov model, the conditional probability of a gram only depends on the last n grams. For example, the server normalizes the Markov model as $\Pr(abcdef) = \Pr(abcd) \cdot \Pr(bcde|abcd) \cdot \Pr(cdef|abcde)$, where all these probabilities and $\Pr(2345|abcdef)$ will be computed by the vault $\mathcal{V}^i$ in the training phase and sent to the server. For more details on model normalization, see Appendix.

4.3. Local training

For each vault $\mathcal{V}_i$, the local model training process $(\mathcal{G}^{t-1}, model_i, d_i)$ takes the current global model parameters $\mathcal{G}^{t-1}$, the chosen local model $model_i$, and the local dataset d_i as inputs and constructs a local contribution $\mathcal{G}^{t,i}$, which will be utilized by the server $\mathcal{A}$ to generate the next-round global model. Before the training, the vault $\mathcal{V}^i$ divides its local dataset d_i into several batches according to the total size n_i of d_i. Algorithm 3 shows the details of vault $\mathcal{V}^i$, which are as follows.

1. Vault $\mathcal{V}^i$ initials the tth round local contribution with the received current global DTE model $\mathcal{G}^{t-1}$ from the server $\mathcal{A}$ (as shown in Algorithm 3, Step 1);
2. Vault $\mathcal{V}^i$ trains the local model using its dataset d_i multi times (as shown in Algorithm 3, Steps 2–7). Let E denote the times. At each time of training, vault $\mathcal{V}^i$ repeats to compute gradient $\mathcal{P}_i = \nabla\mathcal{L}(\mathcal{G}^{t,i}, B)$ and then update $\mathcal{G}^{t,i} = \mathcal{G}^{t,i} - \eta\mathcal{P}_i$ for all batches, where B denotes a batch or subset of the local training dataset d_i, $\mathcal{L}$ denotes the cross-entropy loss function, ∇ denotes the process to minimize the loss function, and η denotes the learning rate;
3. Finally, vault $\mathcal{V}^i$ returns its local contribution $\mathcal{G}^{t,i}$ to the server $\mathcal{A}$. (as shown in Algorithm 3, Step 8).

Note that parameters E, B, and η are assigned according to the specific dataset. We will set them in Section 5.4.

4.4. Global aggregating

The server $\mathcal{A}$ aggregates and generates the global DTE model $\mathcal{G}^t$ by executing GlobalAggregating($\{(\mathcal{G}^{t,i}, n_i)\}$) in the tth round training phase. Algorithm GlobalAggregating takes all vaults' contributions and dataset sizes $(\mathcal{G}^{t,i}, n_i)$ as inputs and constructs a new global DTE model $\mathcal{G}^t$. As shown in Algorithm 4, the server $\mathcal{A}$ performs the following processes:

1. Computes the total size of all vaults' datasets via $n = \sum_i n_i$;
2. Aggregates all local contributions to obtain a new global DTE model $\mathcal{G}^t = \sum_i \frac{n_i}{n}\mathcal{G}^{t,i}$, where $\frac{n_i}{n}$ denotes the weight of vault $\mathcal{V}^i$;
3. Sends the new global DTE model $\mathcal{G}^t$ back to all participant vaults for the next-round training phase.

4.5. Security analysis against the differential attack

The federated DTE effectively mitigates all participant vaults' statistical and model differences to resist the differential attack. Furthermore, the federated DTE aggregates all participant vaults' datasets, resulting in a high-quality DTE model with superior generalization performance while enhancing password security.

In the federated DTE model, each participant vault $\mathcal{V}^i$ uses the same global model $\mathcal{G}^T$ to construct its DTE and honey password vault system. Using the same DTE model by all participant vaults achieves the consistent statistical distribution of the generated decoy password vaults. Suppose a user utilizes two different password vaults in practice. When decrypting these two ciphertext vaults with the incorrect master passwords, the differential attacker will obtain two decoy password vaults having the same distribution as the real password vault. Hence, the federated DTE model can resist the differential attack. The global model $\mathcal{G}^T$ is a conditional probability model that can also withstand prior attacks, such as the KL divergence attack (Golla et al., 2016), the encoding attack (Cheng et al., 2019), and the intersection attack (Cheng et al., 2021). We will evaluate the capability of the federated DTE model against the differential attack and the prior attacks in Section 5.

In addition, the server $\mathcal{A}$ does not access any password data of vaults directly. Moreover, our federated DTE model can adopt the existing privacy-preserving methods, including secure multi-party computing (Bonawitz et al., 2017), differential privacy (Wei et al., 2020), and homomorphic encryption (Phong et al., 2018), to prevent the server from extracting the participant vaults' passwords during the training phase. Hence, we omit this work in this paper since maintaining password security during the training phase is not our novel contribution.

5. Evaluation

Based on real-world datasets, we assess the effectiveness of the differential attack and our proposed federated honey password vault scheme under prior attacks such as KL divergence attacks (Golla et al., 2016), encoding attacks (Cheng et al., 2019), and intersection attacks (Cheng et al., 2021). The experimental results indicate that our framework is practical and has considerable security improvements.

5.1. Experiment setting

We conducted experiments on both the traditional honey password vaults and the federated honey password vault within a uniform experimental setup, utilizing Python 3.6 coupled with Cryptography 37. Consistent with methodologies employed in prior research (Chatterjee et al., 2015; Golla et al., 2016; Cheng et al., 2021; Rao et al., 2023), our encryption process incorporated AES encryption in CTR mode along with SHA-256 in PBKDF2 for key derivation.

5.2. Dataset

For a fair comparison with (Chatterjee et al., 2015; Golla et al., 2016; Cheng et al., 2021), we utilize the most widely used datasets Pastebin, RockYou, Yahoo, Myspace, Gmail, and LinkedIn leak datasets, to evaluate the differential attack and train local DTEs. The password vault dataset Pastebin, which contains 276 plaintext password vaults, is the only real-world dataset in existence. Pastebin's vault size ranges from 2 to 50. The RockYou is one of the largest password lists publicly available, consisting of 32.6 million passwords, with 14 million unique passwords obtained by an SQL injection attack. The Yahoo dataset was leaked by the hacker group D33DS in 2012 and has around 442,800 password samples. The Myspace dataset obtained in 2006 by a phishing attack contains about 50,000 passwords, of which 37,100 are unique. The Gmail dataset contains over 5 million Gmail credentials. The LinkedIn dataset consists of 6 million total passwords.

In the initial phase of our evaluation, we partitioned password datasets sourced from RockYou, Yahoo, Myspace, Gmail, and LinkedIn into two distinct subsets: 90% for training and 10% for testing. Following that, we trained DTE models using the training set. Following the research works (Golla et al., 2016; Cheng et al., 2021; Bojinov et al., 2010; Chatterjee et al., 2015; Rao et al., 2023), we only use the above public and outdated password lists for research purposes, and we do not imperil any Internet user's security and privacy.

5.3. Security benchmarks

This paper introduced two essential security benchmarks to assess the efficacy of honey password vault systems: the average rank of the real password vault, denoted as $\bar{r}$, and the attack accuracy α ($\alpha = 1 - \bar{r}$). The metric $\bar{r}$ ranges between 0 and 1. To facilitate a fair comparison across different implementations, we employed the method from Cheng et al. (2021) for estimating the cumulative distribution function $F(x)$ of the real password vault's rank, which encapsulates the overall attack results. Assuming a uniform distribution as a baseline, expected values for both $\bar{r}$ and α are 0.5. We further explored the distribution's specific quartiles, $F(0)$, $F(1/4)$, $F(1/2)$, and $F(3/4)$, which represent the probabilities of correctly identifying real vaults at different ranks: immediately and at each quarter of the total range. Ideally, for a linear $F(x)$ reflecting a uniform distribution, the values are 0, 0.25, 0.5, and 0.75, respectively.

In practice, attackers may generate a large guessing list based on the leaked passwords (Sahin et al., 2023). Suppose the real master password is not in this list, attackers regenerate it again. However, the spaces of master passwords and decoy password vaults are enormous, as shown in Wang et al. (2022). Calculating the rank of a real password vault is complicated by generating all decoys. Therefore, we choose to calculate the relative rank of the real password vault in candidate lists, which is common in previous works (Chatterjee et al., 2015; Golla et al., 2016; Cheng et al., 2021). The performance of the attack remains unaffected by the sampling experiments. Because the success rate only depends on the attacker's ability to rank the real password vault near the top rank for online verification, rather than guessing the master password successfully. Online verification is resource-intensive as it is easily detected by remote servers and blocked by some existing mechanisms (Freeman et al., 2016).

5.4. Experiments for federated DTE model

The federated DTE training process is shown in Algorithm 2. Detailed steps are as follows:

1. In the setup phase, we set $K (K = 9, 99, 999, 9999)$ as the number of participant vaults to implement the federated DTE training process. Participant vaults construct DTEs of participant vaults with $model_1$, $model_2$, and $model_3$. Define the $model_1$, $model_2$, and $model_3$ as PCFG, Markov, and the conditional probability model, respectively. The number of each model is equal. Set the datasets d_1, d_2, d_3, d_4, and d_5 as RockYou, Yahoo, Gmail, and LinkedIn.
2. In the training phase, we define parameters of learning rate η, epochs E, and batch B. The learning rate η parameter dictates the impact of the loss gradient on the model parameter updates, with a fixed value of 0.0003 used for training. The number of epochs E defines the times the model processes the entire training dataset, set to 3. The data batch B is according to the training dataset, and we set B as 1000.
3. In the final phase, each vault uses the global DTE model to establish its honey password vault system. The federated honey password vault consists of two main components: federated DTE and PBE. The federated DTE is constructed from the received global model, and PBE remains the same as the traditional method. The federated DTE transforms the vault to obtain the seed. After that, PBE encrypts the seed using the master password to produce the ciphertext password vault. As for the encryption process, PBE derives the encryption key using the master password and a uniform salt by password-based key derivation function (KDF). Here, KDF is instantiated with SHA-256.

5.5. Experiments for differential attacks

In real-world scenarios, different vault applications provide various services to users. Therefore, a number of users store their passwords in different types of honey password vaults. Based on the above scenario, suppose that different attackers obtain ciphertext password vaults from different honey password vault applications. The two ciphertexts contain a large number of passwords from the same user. The realization of the differential attack is outlined in Algorithm 1. Detailed steps are as follows:

1. Create two candidate master password lists. The size of the candidate master password lists is N $(N = 10, 100, 1000, 10000)$. The attacker analyzes the user's password-generating habits in the context of the existing leaked password dataset to generate two lists of candidate master password lists $\mathcal{MP}^1$ and $\mathcal{MP}^2$ for ciphertext password vaults.
2. Initialize DTEs. To perform Example I, initialize DTE_1 and DTE_2 with two same models and then train DTE_1 and DTE_2 using different datasets d_1 and d_2. To perform Example II, initialize DTE_1 and DTE_2 with two distinct models and use different datasets d_1 and d_2.
3. Obtain ciphertext vaults from different DTEs-based honey password vaults. Denote the ciphertext password vaults as $\mathcal{C}^1$ and $\mathcal{C}^2$ encrypted from the same vault $\mathcal{V}$, respectively. Set the size of the $\mathcal{V}$ as M $(M \in [2, 50])$.
4. Decrypt ciphertext vaults and generate candidate vaults. Decrypt ciphertexts $\mathcal{C}^1$ and $\mathcal{C}^2$ utilizing the candidate master passwords in $\mathcal{MP}^1$ and $\mathcal{MP}^2$, correspondingly, output the seeds $\mathcal{S}^1$ and $\mathcal{S}^2$. Then, decode the seeds to produce the candidate vaults $\mathcal{V}^1$ and $\mathcal{V}^2$.
5. Rank password vaults with an attack priority function and output the top-ranked vault. Compute the score list r_i of $\mathcal{V}^1$ and $\mathcal{V}^2$ with $\mathsf{Priority}()$ designed in Section 3.2. Sort the score list in decreasing order to obtain the ranked list. The higher rate indicates a higher probability of the candidate password vault being

Table 1
Evaluation: the differential attack on various honey vault systems. Symbols $model_1$, $model_2$, and $model_3$ denote PCFG, Markov, and conditional probability models, respectively. Symbols d_1, d_2, d_3, d_4, and d_5 denote the RockYou, Yahoo, Myspace, Gmail, and LinkedIn datasets, respectively.

Attack	Scheme	$\bar{r}$	α	$F(0)$	$F(1/4)$	$F(1/2)$	$F(3/4)$
	DTE$(model_1, d_1)$|DTE$(model_1, d_2)$	0.29%	99.71%	97.61%	99.83%	99.83%	99.83%
	DTE$(model_1, d_1)$|DTE$(model_1, d_3)$	0.24%	99.76%	98.98%	99.83%	99.83%	99.83%
	DTE$(model_1, d_1)$|DTE$(model_1, d_4)$	0.83%	99.17%	96.42%	98.98%	98.98%	99.66%
	DTE$(model_1, d_1)$|DTE$(model_1, d_5)$	0.51%	99.49%	97.72%	98.21%	99.86%	99.89%
	DTE$(model_1, d_2)$|DTE$(model_1, d_3)$	0.69%	99.31%	96.93%	99.32%	99.32%	99.49%
	DTE$(model_1, d_2)$|DTE$(model_1, d_4)$	0.87%	99.13%	97.44%	98.80%	99.15%	99.32%
	DTE$(model_1, d_2)$|DTE$(model_1, d_5)$	0.63%	99.37%	96.90%	97.31%	98.80%	99.89%
	DTE$(model_1, d_3)$|DTE$(model_1, d_4)$	0.48%	99.52%	98.12%	99.27%	99.76%	99.81%
	DTE$(model_1, d_3)$|DTE$(model_1, d_5)$	0.56%	99.44%	97.39%	98.80%	99.46%	99.78%
	DTE$(model_2, d_1)$|DTE$(model_2, d_2)$	0.10%	99.90%	98.98%	100.00%	100.00%	100.00%
Example I	DTE$(model_2, d_1)$|DTE$(model_2, d_3)$	0.72%	99.28%	96.93%	99.32%	99.32%	99.49%
	DTE$(model_2, d_1)$|DTE$(model_2, d_4)$	0.57%	99.43%	96.42%	99.15%	99.66%	100.00%
	DTE$(model_2, d_2)$|DTE$(model_2, d_3)$	0.70%	99.30%	96.59%	99.15%	99.32%	99.83%
	DTE$(model_2, d_2)$|DTE$(model_2, d_4)$	0.42%	99.58%	97.95%	99.39%	99.63%	99.73%
	DTE$(model_2, d_2)$|DTE$(model_2, d_5)$	0.43%	99.57%	97.77%	99.14%	99.42%	99.90%
	DTE$(model_3, d_1)$|DTE$(model_3, d_2)$	0.75%	99.25%	97.64%	99.16%	99.16%	99.66%
	DTE$(model_3, d_1)$|DTE$(model_3, d_3)$	0.58%	99.42%	97.10%	99.32%	99.32%	99.83%
	DTE$(model_3, d_1)$|DTE$(model_3, d_4)$	0.64%	99.36%	96.24%	99.15%	99.49%	99.80%
	DTE$(model_3, d_2)$|DTE$(model_3, d_3)$	0.47%	99.53%	97.27%	99.49%	99.49%	99.83%
	DTE$(model_3, d_2)$|DTE$(model_3, d_4)$	0.56%	99.44%	97.95%	99.32%	99.46%	99.76%
	DTE$(model_3, d_3)$|DTE$(model_3, d_5)$	0.63%	99.37%	98.15%	99.11%	99.45%	99.93%
	DTE$(model_1, d_1)$|DTE$(model_2, d_2)$	0.12%	99.88%	94.56%	100.00%	100.00%	100.00%
	DTE$(model_1, d_3)$|DTE$(model_2, d_4)$	0.28%	99.72%	92.35%	99.72%	99.81%	99.92%
	DTE$(model_1, d_5)$|DTE$(model_3, d_1)$	0.11%	99.89%	94.89%	100.00%	100.00%	100.00%
	DTE$(model_1, d_2)$|DTE$(model_3, d_1)$	0.27%	99.73%	95.40%	98.88%	99.90%	99.90%
	DTE$(model_2, d_1)$|DTE$(model_3, d_2)$	0.42%	99.58%	96.59%	99.66%	99.66%	99.13%
Example II	DTE$(model_2, d_1)$|DTE$(model_3, d_3)$	0.14%	99.86%	96.59%	100.00%	100.00%	100.00%
	DTE$(model_2, d_1)$|DTE$(model_3, d_4)$	0.33%	99.67%	96.25%	99.66%	99.66%	100.00%
	DTE$(model_2, d_1)$|DTE$(model_3, d_5)$	0.22%	99.78%	96.59%	99.83%	99.83%	100.00%
	DTE$(model_2, d_2)$|DTE$(model_3, d_3)$	0.25%	99.75%	96.42%	99.83%	99.83%	100.00%
	DTE$(model_2, d_3)$|DTE$(model_3, d_4)$	0.27%	99.73%	96.59%	99.83%	99.83%	99.83%
	DTE$(model_2, d_4)$|DTE$(model_3, d_5)$	0.30%	99.70%	96.93%	97.68%	99.37%	99.77%

correct. The ranked list provides an ordering of the candidate password vaults based on their similarity rates. The top-ranked vault is the most promising candidate for a real password vault.

5.6. *Experimental results*

We evaluate the differential attack against three state-of-the-art honey password vault schemes, of which DTEs are based on $model_1$, $model_2$, and $model_3$, respectively, as shown in Table 1. Define the $model_1$, $model_2$, and $model_3$ as PCFG, Markov, and the conditional probability model, respectively. Set the datasets d_1, d_2, d_3, d_4, and d_5 as RockYou, Yahoo, Myspace, Gmail, and LinkedIn. The security metrics used to assess the attack performance are defined in Section 5.3.

Evaluation on Example I. We designed attack scenarios, for Example I, and the results are shown in 1. The attack scenario of DTE$(model_1, d_1)$|DTE$(model_1, d_2)$ performs Example I against two honey password vaults using the same model $model_1$ to construct DTEs, and trains models using d_1 and d_2, respectively. From Table 1, we see that the $\bar{r}$ (real password vault's rank) under Example I is 0.29%, and the accuracy α is 99.71%. $F(0)$, $F(1/4)$, $F(1/2)$, and $F(3/4)$ are 97.61%, 99.83%, 99.83%, and 99.83%, respectively. The attackers successfully rank 97.61% of the real password vaults in the first position.

In scenario of DTE$(model_1, d_2)$|DTE$(model_1, d_5)$, the honey password vaults are constructed with $model_1$ and the datasets are d_2 and d_5, respectively. The experimental findings reveal that the real password vaults in Example I have an average rank $\bar{r}$ of 0.63%, and an attack accuracy α of 99.37%. Specifically, the values for $F(0)$, $F(1/4)$, $F(1/2)$, and $F(3/4)$ are 96.90%, 97.31%, 98.80%, and 99.89%, respectively. Attackers are able to rank 96.90% of the real vaults in the top position, while 97.31% are placed in the top 25% of candidate vaults. Additionally, 98.80% of the real vaults appear in the top 50%, and 99.89% are found in the top 75% of the candidate vaults.

In scenario of DTE$(model_1, d_3)$|DTE$(model_1, d_4)$, two honey password vaults created using the same $model_1$ for constructing DTEs. The datasets used for training are models that are datasets d_3 and d_4, respectively. For the attack scenario of Example I, the real password vaults have an average rank $\bar{r}$ of 0.48%, and the attack accuracy α is 99.52%, suggesting that the real vaults are often correctly identified by the attacker. The cumulative distribution values $F(0)$, $F(1/4)$, $F(1/2)$, and $F(3/4)$ are 98.12%, 99.27%, 99.76%, and 99.81%, respectively. According to the results, 98.12% of real password vaults are at the top rank, and 99.27% of real password vaults are ranked in the top quarter of the candidate vaults. Note that Table 1 shows experimental results for $model_1$ with five datasets and the partial results of $model_2$ and $model_3$. The conclusions of the experiments for $model_2$ and $model_3$ are the same as $model_1$. Due to the page limitation, we will show the full table in the full version of this paper. According to the experimental results, the average rank of real password vaults is very close to the top among candidate vaults. Therefore, the differential attack is effective in Example I.

Evaluation on Example II. We designed scenarios for Example II. The scenario of DTE$(model_1, d_1)$|DTE$(model_2, d_2)$ in Table 1 perform the Example II against two DTEs constructed with $model_1$ and $model_2$ and train models use d_1 and d_2, respectively. The presented results show that attackers rank the real password vault upper 0.12% in the candidate vaults. And the attack accuracy achieved 99.88%. In this scenario, $F(0)$, $F(1/4)$, $F(1/2)$, and $F(3/4)$ are 94.56%, 100%, 100%, and 100%, respectively. The attackers place 94.56% of the real password vaults at the top and 100% of the real vaults in the top quarter of the candidate vaults.

In the scenario DTE$(model_1, d_5)$|DTE$(model_3, d_1)$, two honey password vaults created DTEs using $model_1$ and $model_3$, trained on datasets d_5 and d_1, respectively. The results indicate that attackers are able to position real password vaults within the top 0.11% of candidate vaults. A closer examination of the $F(x)$ for these rankings shows that $F(0)$ is 94.89%, meaning that 94.89% of real vaults are placed as the first candidate vault. Moreover, $F(1/4) = F(1/2) = F(3/4) = 100\%$,

Table 2
Performance of differential attack against existing honey password vaults with different experimental parameters. The real password vault is sourced from Pastebin. We set $N_1 = 100, N_2 = 1000, N_3 = 10{,}000$ (M is of size each vault and N is the size of the candidate set).

	Vault size	Example I		Example II	
	M	$\bar{r}$	α	$\bar{r}$	α
N_1	2–3	0.67%	99.33%	0.13%	99.87%
	4–8	0.00%	100.00%	0.00%	100.00%
	9–50	0.00%	100.00%	0.00%	100.00%
	All (2–50)	0.22%	99.78%	0.04%	99.96%
N_2	2–3	0.73%	99.27%	0.24%	99.76%
	4–8	0.00%	100.00%	0.00%	100.00%
	9–50	0.00%	100.00%	0.00%	100.00%
	All (2–50)	0.24%	99.76%	0.08%	99.92%
N_3	2–3	0.84%	99.16%	0.26%	99.74%
	4–8	0.00%	100.00%	0.00%	100.00%
	9–50	0.00%	100.00%	0.00%	100.00%
	All (2–50)	0.28%	99.72%	0.08%	99.92%

implying that all real password vaults are ranked within the top 25% of the candidate vaults.

The models are different when constructing DTEs in scenario DTE$(model_2, d_1)$|DTE$(model_3, d_2)$ to create two honey password vaults in Table 1. The attacker ranks the real password vault at the top 0.42%. Moreover, 96.59% of the real password vaults are ranked first, and 99.66% of the real password vaults are ranked in the upper quartiles of candidate vaults. Therefore, the differential attack can effectively identify every real password vault from decoy password vaults in Example II.

Discussion on Experimental Parameters. We investigate how the size of the vault (M) and the number of candidate vaults (N) affect the differential attack, with the results in Table 2. The experiments reveal that a larger vault size increases the diversity of passwords, which reduces the similarity among candidate vaults, thereby improving the attack's accuracy. Furthermore, an increase in the number of candidate vaults leads to a slight rise in the number of decoy vaults with all the same passwords. We evaluated Examples I and II for the differential attack, respectively. The size of the real password vaults is $M \in [2, 50]$. The size of candidate vault is $N \in \{100, 1000, 10000\}$. In Example II, the average rank $\bar{r}$ ranges from 0.13% to 0.26%, corresponding to an accuracy between 99.74% and 99.87%. While M is set to 2–3, the real password vaults are ranked in the upper 0.67%–0.84% of the candidate vaults in Example II. The results show that the attack effectiveness may slightly decrease for tiny vault sizes, but it still maintains a high ranking for the real password vaults. When M is larger than 3, the differential attacks all achieve 100% accuracy in Examples I and II.

Differential Attack for Multiple Honey Password Vaults. In reality, most Internet users employ more than two password vaults to store their passwords, such as three or four password vaults (Ma et al., 2019; İşler et al., 2019). This section explores the extension of the differential attack to scenarios where users utilize multiple honey password vaults. The multiple honey password vault scenario also suffers from a differential attack against any two pairs of vaults. The differential attacker can identify the correct master password using the Algorithm 1 for any two password vaults.

Suppose a user has N honey password vaults, denoted as $\mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^N$, where $N > 2$. The differential attack against multiple honey password vaults is as follows: Firstly, the differential attacker generates candidate master password lists $\mathcal{MP}^1, \mathcal{MP}^2, \ldots, \mathcal{MP}^N$ for each of the N vaults, respectively. Then, the differential attacker decrypts each password vault $\mathcal{V}^i$ using its corresponding candidate master password, yielding N candidate password vaults $\mathcal{DV}^1, \mathcal{DV}^2, \ldots, \mathcal{DV}^N$. For each pair of candidate password vaults $\mathcal{DV}^i$ and $\mathcal{DV}^j$, the differential attacker computes the similarity rate of candidate password vaults using the priority function in Section 3.2 and obtains a ranked list

Table 3
Evaluation: the differential attack on multiple honey vault systems. Symbols $model_1$, $model_2$, and $model_3$ denote PCFG, Markov, and conditional probability models, respectively. Symbols d_1, d_2, d_3, d_4, and d_5 denote the RockYou, Yahoo, Myspace, Gmail, and LinkedIn datasets, respectively.

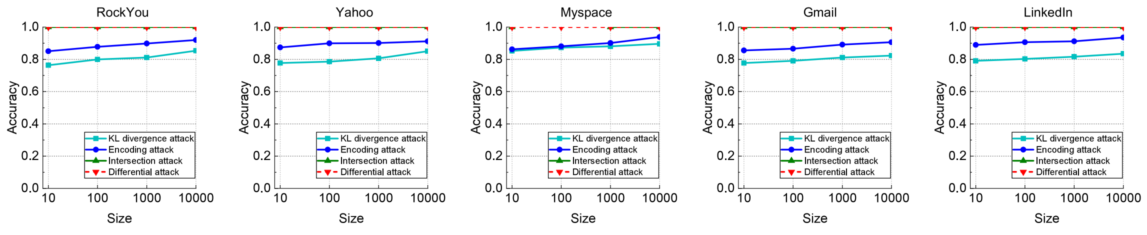
Scheme	$\bar{r}$	α
DTE$(model_1, d_1)$|DTE$(model_2, d_2)$|DTE$(model_3, d_3)$	0.11%	99.89%
DTE$(model_1, d_2)$|DTE$(model_2, d_3)$|DTE$(model_3, d_4)$	0.20%	99.80%
DTE$(model_1, d_3)$|DTE$(model_2, d_4)$|DTE$(model_3, d_5)$	0.13%	99.87%
DTE$(model_2, d_1)$|DTE$(model_3, d_2)$|DTE$(model_1, d_3)$	0.14%	99.86%
DTE$(model_2, d_2)$|DTE$(model_3, d_3)$|DTE$(model_1, d_4)$	0.12%	99.88%
DTE$(model_2, d_3)$|DTE$(model_3, d_4)$|DTE$(model_1, d_5)$	0.20%	99.80%
DTE$(model_3, d_1)$|DTE$(model_1, d_2)$|DTE$(model_2, d_3)$	0.10%	99.90%
DTE$(model_3, d_2)$|DTE$(model_1, d_3)$|DTE$(model_2, d_4)$	0.08%	99.92%
DTE$(model_3, d_3)$|DTE$(model_1, d_4)$|DTE$(model_2, d_5)$	0.03%	99.97%

of candidate password vaults. The highest-ranked vault in the sorted list corresponds to the most likely combination of the correct master password. Therefore, the differential attack extends to the case where the user employs multiple vaults, and our federated honey password vault framework resists the differential attack.

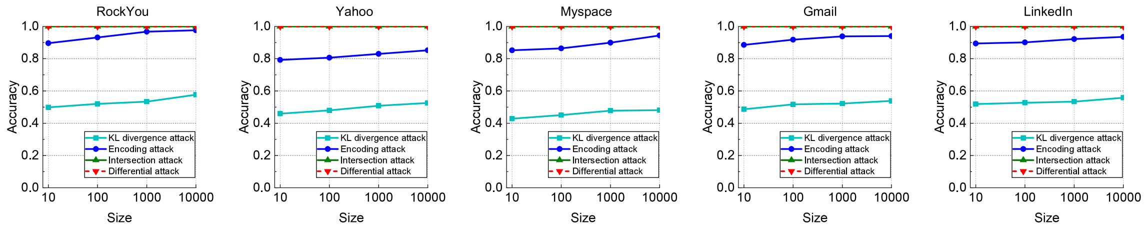
To prove our conclusions, we experimentally assumed that the user stores passwords in three types of honey password vaults. Table 3 shows the effect of differential attacks on three honey password vaults with different DTEs constructed by the $model_1$ (PCFG model), the $model_2$ (Markov model), and the $model_3$ (conditional probability model), respectively. The above models are trained with datasets d_1 (RockYou), d_2 (Yahoo), d_3 (Myspace), d_4 (Gmail), and d_5 (LinkedIn), respectively. Each row in Table 3 such as DTE$(model_1, d_1)$|DTE$(model_2, d_2)$| DTE$(model_3, d_3)$ represents a scenario in which a user stores his/her password vaults in three honey password vaults with three different DTEs. Experimental results are the average rank of the real password vault $\bar{r}$ and the attack accuracy α. For example, the DTE$(model_1, d_1)$ |DTE$(model_2, d_2)$|DTE$(model_3, d_3)$ utilizes the $model_1$, $model_2$ and $model_3$ to construct DTEs with datasets d_1, d_2, and d_3, respectively. The average rank of the real password vault $\bar{r}$ is 0.11%, and the attack accuracy α is 99.89%. The experimental results show that 99.89% of the real password vaults are ranked first. Therefore, the differential attack on three honey password vaults is highly precise in identifying the real password vault. The experimental results of the above attacks show that if a user stores his/her password vault in more than three password vaults, the differential attack against multiple password vaults is effective.

Evaluation on Federated Honey Password Vaults. We have shown that the differential attack can effectively distinguish the real and decoy password vaults against current honey vault schemes (Chatterjee et al., 2015; Golla et al., 2016; Cheng et al., 2021). Next, we also evaluate the performance of our federated honey password vault scheme, comparing it against previous schemes (Chatterjee et al., 2015; Golla et al., 2016; Cheng et al., 2021) under KL divergence attacks (Golla et al., 2016), encoding attacks (Cheng et al., 2019), and intersection attacks (Cheng et al., 2021). We briefly outline the core mechanisms of the above attacks proposed in previous works. The KL divergence attack distinguishes distribution differences between the decoy and real password vaults (Golla et al., 2016), the encoding attack compares the encoding path of the decoy and real password vaults (Cheng et al., 2019), and the intersection attack analyzes overlaps in password vaults before and after updating (Cheng et al., 2021).

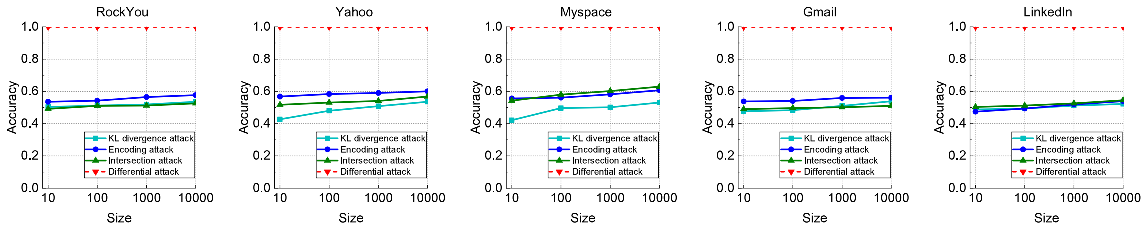
In the following, we compare the performance of the instantiated federated honey password vault scheme with previous works such as Chatterjee-PCFG (Chatterjee et al., 2015), Golla-Markov (Golla et al., 2016), and Cheng-IUV (Cheng et al., 2021). We calculate the average outcomes for the aforementioned attack scenarios. Experimental results prove that our scheme can resist the above attacks, as shown in Fig. 4. We conduct scenarios for the differential attack and calculate the average of the attack results. In Fig. 4, the federated honey password vault scheme effectively resists the differential attack and decreases



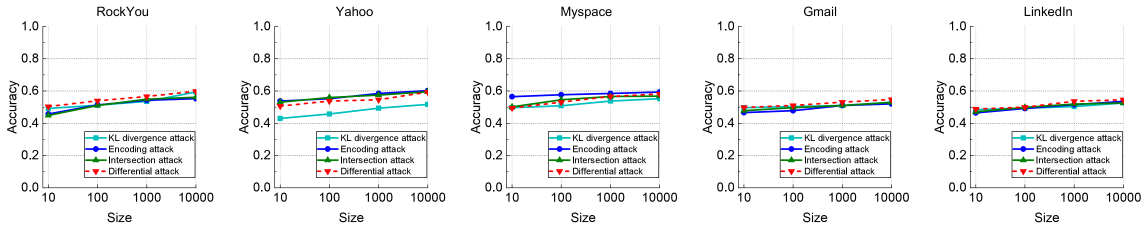
(a) Chatterjee-PCFG.



(b) Golla-Markov.



(c) Cheng-IUV.



(d) Our Scheme.

Fig. 4. Evaluation: Differential attack and previous attack methods (Golla et al., 2016; Cheng et al., 2019, 2021) against different vault sizes $M \in [2, 50]$. The $N = 1{,}000$. We train DTEs using RockYou, Yahoo, Myspace, Gmail, and LinkedIn datasets.

the accuracy to 50.34%–59.86%, 50.68%–59.35%, 49.48%–58.16%, 0.49%–0.54%, and 0.48%–0.54% for five datasets, which are approaching the optimal value 50%. Our federated honey password vault scheme is also compatible with SMART and resists the MPGA (Rao et al., 2023) against master passwords. Note that MPGA is an attack against the master password, and the other attacks focus on the security of the DTE. Therefore, MPGA is not discussed with the other attacks in this paper. As the genetic countermeasure of MPGA, the SMART technique can be directly combined with our federated honey password vault. Therefore, our experiments prove that the federated honey password vault framework is effective against differential attacks as well as previously proposed attacks such as KL divergence attacks (Golla et al., 2016), encoding attacks (Cheng et al., 2019), intersection attacks (Cheng et al., 2021), and MPGA (Rao et al., 2023). Our framework significantly improves security.

6. Conclusion

This paper first proposes the differential attack against state-of-the-art honey password vault schemes. Our detailed experiments demonstrate the capability of our proposed attacks to recognize the real password vault from decoy password vaults with nearly 100.00% accuracy. To counter the vulnerability and resist the differential attack, we further design a novel federated DTE model where honey password vaults jointly train a global DTE and then use the DTE to construct a federated honey password vault. Our explicit security analysis proves that the federated honey password vault provides superior security guarantees. Moreover, our extensive experiments illustrate that the distinguishing accuracy reduces to 52.41%, which is close to the ideal 50%.

CRediT authorship contribution statement

Peng Xu: Writing – review & editing, Methodology. **Tingting Rao:** Writing – original draft, Methodology. **Wei Wang:** Formal analysis, Supervision. **Zhaojun Lu:** Writing – review & editing, Formal analysis. **Kaitai Liang:** Writing – review & editing, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and valuable suggestions. This work was supported by the National Natural Science Foundation of China (Grant No. 62272186 and 62372201) and the EU Horizon Europe Research and Innovation Program under grant agreements 101073920 (TENSOR), 101070052 (TANGO), 101070627 (REWIRE), and 101092912 (MLSysOps).

Appendix. Model normalization

The PCFG-DTE (Chatterjee et al., 2015) is designed with password generation rules and calculates password probabilities using a parse tree. This model treats passwords as combinations of letters, numbers, and symbols. To encode the vault, the PCFG-DTE analyzes the sub-grammar of each password within the vault and converts passwords into seeds, and then concatenates these seeds for output. Each PCFG-DTE-based participant vault in the federated DTE model independently generates its password-generation rules and corresponding probabilities, which serve as initial model parameters. During the federated training process, these parameters are transmitted from each vault to the server.

The Markov-DTE (Golla et al., 2016) models password vault using the n-gram Markov model. In the n-gram Markov model, the probability of the next token in a sequence is derived from a prefix of length n. The Markov-DTE fixes an $n - gram$ to compute the transition probabilities of each prefix gram for passwords. Then, the Markov-DTE fixes the base password of the password vault and computes the transition probabilities by the reuse rate of the base password. Each Markov-DTE-based participant vault in the federated DTE model computes the transition probabilities with local datasets and sets each probability as initial DTE model parameters.

The conditional probability model-based DTE (Cheng et al., 2021) applies an incremental update mechanism of the password vault. This type of DTE encodes passwords one by one. The probability of a vault $\mathcal{V}$ is defined as $\Pr(\mathcal{V}) = \prod_{i=0}^{M-1} \Pr(pw_{i+1} pw_1, pw_2, \ldots, pw_i)$, where $\Pr(pw_{i+1} | (pw_{i'})_{i'=1}^{i-1})$ is the conditional probability of generating a new password pw_{i+1} given the i number of old passwords $pw_1, pw_2, \ldots, pw_i$. This mechanism efficiently facilitates password updates within the vault. The DTE, based on the conditional probability model, encodes the new password using the conditional encoder and appends it to the end. Each conditional probability model-based participant vault in the federated DTE model sets the conditional probabilities as the initial DTE model parameters.

The server normalizes the PCFG model and Markov model to the conditional probability model. The normalized method is designed as follows: In the PCFG model, the probability of a password is the product of the probabilities of grammar structures. The conditional probability of a grammar structure does not depend on any previous structure. Hence, the global model normalizes the PCFG model to the conditional probability model as $\Pr(r_i | r_1 r_2 \ldots r_{i-1}) = \Pr(r_i)$ where r_i is the grammar structure and $\Pr(r_i)$ is trained on a training dataset. The PCFG model contributes the probabilities of the grammar structure model parameters to the global model. To normalize the Markov model of n-gram, the conditional probability of a gram only depends on the last n grams. Therefore, the conditional probability is defined as $\Pr(r_i | r_1 r_2 \ldots r_{i-1}) = \Pr(r_i | r_{i-n} r_{i-n+1} \ldots r_{i-1}) (i > n)$ where r_i is each gram and $\Pr(r_i)$ is trained on a training dataset. The Markov model contributes the probabilities of grams as model parameters to the global model.

Data availability

Data will be made available on request.

References

1password, 1Password Security Design. https://1passwordstatic.com/files/security/1password-white-paper.pdf.

AppleReport, 2.6 billion personal records compromised by data breaches in past two years. https://www.apple.com/newsroom/2023/12/report-2-point-6-billion-records-compromised-by-data-breaches.

Bojinov, H., Bursztein, E., Boyen, X., Boneh, D., 2010. Kamouflage: Loss-resistant password management. In: Gritzalis, D., Preneel, B., Theoharidou, M. (Eds.), ESORICS 2010. In: LNCS, vol. 6345, Springer, pp. 286–302. http://dx.doi.org/10.1007/978-3-642-15497-3_18.

Bonawitz, K.A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K., Practical secure aggregation for privacy-preserving machine learning. In: ACM CCS. pp. 1175–1191.

Chatterjee, R., Bonneau, J., Juels, A., Ristenpart, T., 2015. Cracking-resistant password vaults using natural language encoders. In: IEEE S&P. pp. 481–498.

Cheng, H., Li, W., Wang, P., Chu, C.-H., Liang, K., 2021. Incrementally updateable honey password vaults. In: USENIX Security. pp. 857–874.

Cheng, H., Zheng, Z., Li, W., Wang, P., Chu, C.-H., 2019. Probability model transforming encoders against encoding attacks. In: USENIX Security. pp. 1573–1590.

Chrome, 90% of passwords stored in chrome found in data breach. https://www.reddit.com/r/chrome/comments/vpph9k/90_of_passwords_stored_in_chrome_found_in_data/.

David, L., Wool, A., 2021. An explainable online password strength estimator. In: Bertino, E., Schulmann, H., Waidner, M. (Eds.), ESORICS 2021. In: LNCS, vol. 12972, Springer, pp. 285–304. http://dx.doi.org/10.1007/978-3-030-88418-5_14.

Dong, Q., Jia, C., Duan, F., Wang, D., 2021. RLS-PSM: A robust and accurate password strength meter based on reuse, leet and separation. IEEE Trans. Inf. Forensics Secur. 16, 4988–5002. http://dx.doi.org/10.1109/TIFS.2021.3107147.

Duan, F., Wang, D., Jia, C., 2024. A security analysis of honey vaults. In: IEEE S&P. pp. 230–247.

Enpass, Security Whitepaper. https://enpass.io.

Freeman, D., Jain, S., Dürmuth, M., Biggio, B., Giacinto, G., 2016. Who are you? A statistical approach to measuring user authenticity. In: NDSS. pp. 21–24.

Galbally, J., Coisel, I., Sanchez, I., 2017. A new multimodal approach for password strength estimation - part I: theory and algorithms. IEEE Trans. Inf. Forensics Secur. 12 (12), 2829–2844. http://dx.doi.org/10.1109/TIFS.2016.2636092.

Gasti, P., Rasmussen, K.B., 2012. On the security of password manager database formats. In: Foresti, S., Yung, M., Martinelli, F. (Eds.), ESORICS 2012. In: LNCS, vol. 7459, Springer, pp. 770–787. http://dx.doi.org/10.1007/978-3-642-33167-1_44.

Golla, M., Beuscher, B., Dürmuth, M., 2016. On the security of cracking-resistant password vaults. In: ACM CCS. pp. 1230–1241.

Google, Google Chrome Privacy Whitepaper. https://www.google.com/chrome/privacy/whitepaper.html.

Han, W., Li, Z., Yuan, L., Xu, W., 2016. Regional patterns and vulnerability analysis of Chinese web passwords. IEEE Trans. Inf. Forensics Secur. 11 (2), 258–272. http://dx.doi.org/10.1109/TIFS.2015.2490620.

Han, W., Xu, M., Zhang, J., Wang, C., Zhang, K., Wang, X.S., 2021. TransPCFG: Transferring the grammars from short passwords to guess long passwords effectively. IEEE Trans. Inf. Forensics Secur. 16, 451–465. http://dx.doi.org/10.1109/TIFS.2020.3003696.

Hou, Z., Wang, D., 2023. New observations on Zipf's law in passwords. IEEE Trans. Inf. Forensics Secur. 18, 517–532. http://dx.doi.org/10.1109/TIFS.2022.3176185.

Houshmand, S., Aggarwal, S., Flood, R., 2015. Next gen PCFG password cracking. IEEE Trans. Inf. Forensics Secur. 10 (8), 1776–1791. http://dx.doi.org/10.1109/TIFS.2015.2428671.

Islam, M., Bohuk, M.S., Chung, P., Ristenpart, T., Chatterjee, R., 2023. Araña: Discovering and characterizing password guessing attacks in practice. In: USENIX Security. pp. 1019–1036.

İşler, D., Küpçü, A., Coskun, A., 2019. User perceptions of security and usability of mobile-based single password authentication and two-factor authentication. In: Pérez-Solà, C., Navarro-Arribas, G., Biryukov, A., García-Alfaro, J. (Eds.), ESORICS 2019. In: LNCS, vol. 11737, Springer, pp. 99–117. http://dx.doi.org/10.1007/978-3-030-31500-9_7.

Jaeger, J., Ristenpart, T., Tang, Q., 2016. Honey encryption beyond message recovery security. In: Fischlin, M., Coron, J. (Eds.), EUROCRYPT. In: LNCS, vol. 9665, Springer, pp. 758–788. http://dx.doi.org/10.1007/978-3-662-49890-3_29.

Juels, A., Ristenpart, T., 2014. Honey encryption: Security beyond the brute-force bound. In: Nguyen, P.Q., Oswald, E. (Eds.), EUROCRYPT. In: LNCS, vol. 8441, Springer, pp. 293–310. http://dx.doi.org/10.1007/978-3-642-55220-5_17.

Juels, A., Rivest, R.L., 2013. Honeywords: Making password-cracking detectable. In: ACM CCS. pp. 145–160.

Keychain, iCloud Keychain. https://support.apple.com/en-us/109016.

Li, Z., He, W., Akhawe, D., Song, D., 2014. The {emperor's} new password manager: Security analysis of web-based password managers. In: USENIX Security. pp. 465–479.

Li, X., Tang, Q., Zhang, Z., 2021. Fooling an unbounded adversary with a short key, repeatedly: The honey encryption perspective. In: Tessaro, S. (Ed.), 2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference. In: LIPIcs, vol. 199, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 23:1–23:21. http://dx.doi.org/10.4230/LIPICS.ITC.2021.23.

Li, W., Wang, P., Liang, K., 2023. HPAKE: honey password-authenticated key exchange for fast and safer online authentication. IEEE Trans. Inf. Forensics Secur. 18, 1596–1609. http://dx.doi.org/10.1109/TIFS.2022.3214729.

Li, Y., Wang, H., Sun, K., 2017. Personal information in passwords and its security implications. IEEE Trans. Inf. Forensics Secur. 12 (10), 2320–2333. http://dx.doi.org/10.1109/TIFS.2017.2705627.

Li, W., Zeng, J., 2021. Leet usage and its effect on password security. IEEE Trans. Inf. Forensics Secur. 16, 2130–2143. http://dx.doi.org/10.1109/TIFS.2021.3050066.

Li, S., Zhang, Y., Song, Y., Cheng, N., Yang, K., Li, H., 2024. Blockchain-based portable authenticated data transmission for mobile edge computing: A universally composable secure solution. IEEE Trans. Comput. 73 (4), 1114–1125. http://dx.doi.org/10.1109/TC.2024.3355759.

Lyastani, S.G., Schilling, M., Fahl, S., Backes, M., Bugiel, S., 2018. Better managed than memorized? Studying the impact of managers on password strength and reuse. In: USENIX Security. pp. 203–220.

Ma, S., Bertino, E., Nepal, S., Li, J., Ostry, D., Deng, R.H., Jha, S., 2019. Finding flaws from password authentication code in android apps. In: Sako, K., Schneider, S.A., Ryan, P.Y.A. (Eds.), ESORICS 2019. In: LNCS, vol. 11735, Springer, pp. 619–637. http://dx.doi.org/10.1007/978-3-030-29959-0_30.

Menéndez, M.L., Pardo, J., Pardo, L., Pardo, M., 1997. The jensen-shannon divergence. J. Franklin Inst. 334 (2), 307–318.

Munyendo, C.W., Mayer, P., Aviv, A.J., 2023. "I just stopped using one and started using the other": Motivations, techniques, and challenges when switching password managers. In: CCS. pp. 3123–3137.

Oesch, S., Ruoti, S., 2020. That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In: USENIX Security. pp. 2165–2182.

Pasquini, D., Ateniese, G., Bernaschi, M., 2020. Interpretable probabilistic password strength meters via deep learning. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (Eds.), ESORICS 2020. In: LNCS, vol. 12308, Springer, pp. 502–522. http://dx.doi.org/10.1007/978-3-030-58951-6_25.

PasswdTeam, The largest password leaks of all time. https://passwd.team/blog/largest-password-leaks/.

Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S., 2018. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. 13 (5), 1333–1345. http://dx.doi.org/10.1109/TIFS.2017.2787987.

Rao, T., Su, Y., Xu, P., Zheng, Y., Wang, W., Jin, H., 2023. You reset I attack! a master password guessing attack against honey password vaults. In: Tsudik, G., Conti, M., Liang, K., Smaragdakis, G. (Eds.), ESORICS 2023. In: LNCS, vol. 14346, Springer, pp. 141–161. http://dx.doi.org/10.1007/978-3-031-51479-1_8.

Sahin, S., Al-Roomi, S.A., Poteat, T., Li, F., Investigating the Password Policy Practices of Website Administrators. In: IEEE S&P. pp. 552–569.

Thomas, K., Pullman, J., Yeo, K., Raghunathan, A., Kelley, P.G., Invernizzi, L., Benko, B., Pietraszek, T., Patel, S., Boneh, D., Bursztein, E., 2019. Protecting accounts from credential stuffing with password breach alerting. In: USENIX Security. pp. 1556–1571.

Tian, Y., Li, L., Peng, H., Wang, D., Yang, Y., 2023. Honeywords generation mechanism based on zero-divisor graph sequences. IEEE Trans. Serv. Comput. 16 (6), 4567–4579. http://dx.doi.org/10.1109/TSC.2023.3329013.

Wang, D., Shan, X., Dong, Q., Shen, Y., Jia, C., 2023a. No single silver bullet: Measuring the accuracy of password strength meters. In: USENIX Security. pp. 947–964.

Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X., 2016. Targeted online password guessing: An underestimated threat. In: ACM CCS. pp. 1242–1254.

Wang, D., Zou, Y., Dong, Q., Song, Y., Huang, X., 2022. How to attack and generate honeywords. In: IEEE S&P. pp. 489–506.

Wang, D., Zou, Y., Xiao, Y.-A., Ma, S., Chen, X., 2023b. Pass2Edit: A Multi-Step generative model for guessing edited passwords. In: USENIX Security 2023. pp. 983–1000.

Wang, D., Zou, Y., Zhang, Z., Xiu, K., 2023c. Password guessing using random forest. In: USENIX Security. pp. 965–982.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q.S., Poor, H.V., 2020. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Trans. Inf. Forensics Secur. 15, 3454–3469. http://dx.doi.org/10.1109/TIFS.2020.2988575.

Xie, Z., Shi, F., Zhang, M., Ma, H., Wang, H., Li, Z., Zhang, Y., 2024. GuessFuse: Hybrid password guessing with multi-view. IEEE Trans. Inf. Forensics Secur. 19, 4215–4230. http://dx.doi.org/10.1109/TIFS.2024.3376246.

Xu, M., Yu, J., Zhang, X., Wang, C., Zhang, S., Wu, H., Han, W., 2023. Improving real-world password guessing attacks via bi-directional transformers. In: USENIX Security. pp. 1001–1018.

Yang, S., Ji, S., Beyah, R.A., 2018. DPPG: a dynamic password policy generation system. IEEE Trans. Inf. Forensics Secur. 13 (3), 545–558. http://dx.doi.org/10.1109/TIFS.2017.2737971.

Zhang, Y., Xu, C., Cheng, N., Shen, X., 2022. Secure password-protected encryption key for deduplicated cloud storage systems. IEEE Trans. Dependable Secur. Comput. 19 (4), 2789–2806. http://dx.doi.org/10.1109/TDSC.2021.3074146.

Zhang, Y., Xu, C., Li, H., Yang, K., Cheng, N., Shen, X., 2021. PROTECT: efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage. IEEE Trans. Mob. Comput. 20 (6), 2297–2312. http://dx.doi.org/10.1109/TMC.2020.2975792.

Peng Xu received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2010. He was a Post-Doctoral researcher with the Huazhong University of Science and Technology from 2010 to 2013; and an Associate Research Fellow with the University of Wollongong, Australia, from 2018 to 2019. He is currently a Full Professor with the Huazhong University of Science and Technology. He has authored 70 research papers, 20 patents, and three books. He was a PI in ten grants, including four NSF projects. His research interest is in the field of cryptography.

Tingting Rao is currently pursuing the Ph.D. degree in cyberspace security at the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. Her research interests include cryptography, password security, and data privacy.

Wei Wang received the B.E. and Ph.D. degrees in electronic and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2011, respectively. She is currently an Associate Professor with the Cyber-Physical-Social Systems Laboratory, Huazhong University of Science and Technology. She has authored 40 papers in international journals and conferences. Her research interests include cryptography and data privacy.

Zhaojun Lu received the Ph.D. degree in microelectronic and solid-state electronics from the Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently a Lecturer at the School of Cyber Science and Engineering, Huazhong University of Science and Technology. His current research interests include hardware security, cryptographic chip design, and fully homomorphic encryption.

Kaitai Liang received the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2014. His research, featured in international information security journals and conferences like USENIX Security, NDSS, Asiacrypt, ESORICS (with a best research paper award), IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Dependable and Secure Computing, addresses cybersecurity challenges using information security and cryptographic tools. As a principal investigator in various EU funded projects, he has demonstrated real-world security impact through collaborations with academic and industrial partners. He has served as a TPC member, general chair, and steering committee for international security and privacy conferences, e.g., USENIX Security, IEEE Euro S&P, ESORICS, IEEE CSF, and PoPETs, and as an associate editor for international journals. He has also contributed to ISO standards as a member of the standards committee 381027 "Cybersecurity & Privacy" at NEN.