This work has been accepted and will appear in the proceedings of IEEE S&P 2024.

A Security Analysis of Honey Vaults

Fei Duan*, Ding Wang*†‡, Chunfu Jia*†‡
*College of Cyber Science, Nankai University, Tianjin, China; {wangding, cfjia}@nankai.edu.cn
†Key Laboratory of Data and Intelligent System Security (NKU), Ministry of Education, Tianjin, China
‡Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin, China

***Abstract*—Honey encryption (HE) protected password vaults (called honey vaults) are promising tools that allow a user to store multiple passwords (called a password vault) and encrypt them with a master password using HE. In case password vaults are somehow leaked and the attackers launch offline password guessing, honey vaults can yield decoy password vaults for incorrect guesses, forcing an offline guessing attacker to interact with the authentication server to identify whether passwords in decrypted vaults are correct or not. Therefore, honey vaults transform the *offline* guessing attacker into an *online* guessing attacker, i.e., *honey vault distinguishing attacker*.**

In online guessing, attackers can adopt various attacks to perform *multiple* guesses against *multiple* vaults, but the existing theoretical message recovery (MR) security for HE only focuses on the advantage of *one-time* guess against a *single* vault, which cannot accurately model realistic attackers and thus can not provide practical advice for users' vault security. To address this issue, we propose a theoretically-grounded optimal strategy for distinguishing attackers, and manage to derive a much tighter upper bound on the advantage against MR security. Particularly, we provide much tighter upper/lower bounds for advantage against HE-related cryptographic security games, i.e., the security of distribution transforming encoder (DTE), known message attack, and known side information attack. This provides a better understanding of the actual security of *honey encryption*.

To better understand the security of honey vault systems, we instantiate our optimal strategy into three practical attacks and propose an encoding attack. Extensive experiments against two major honey vault systems demonstrate that our four attacks can improve the attack success rate by 1.15-4.35 times compared with their counterparts. For the intersection attack, we propose a feature attack against Cheng et al.'s incremental update mechanism (at USENIX SEC'21), and our attack can breach their mechanism with *87%-93% advantage*.

1. Introduction

While the number of accounts that users manage constantly increases (e.g., with ordinary Internet users reported to have 80-107 online accounts [15], [27]), the memory capacity of human brains remains stable. Due to human memory limitations, users often choose popular passwords [34], [35] or reuse passwords [15], [25], [37], which leads to the vulnerability of conventional password authentication.

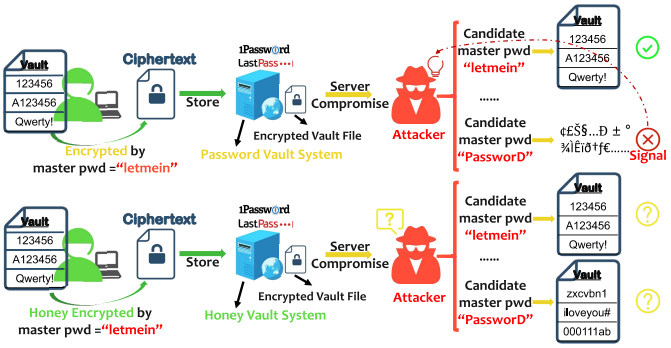


Figure 1: An illustration of the honey vault system. For the conventional password vault system (the upper part), once the authentication server is compromised, the attacker can perform offline password guessing against the user's master password. The decrypted Non-ASCII random junk strongly signals the incorrectness of the candidate master password. While in a honey vault system (the lower part), a wrong decryption can still yield plausible looking passwords to confuse the attacker. Therefore, honey vault systems weaken the offline guessing attacker into an online guessing attacker.

To ease the burden of remembering multiple passwords, password vaults (also called password managers) are widely recommended by both academia and industry [7], [13], [22].

Generally, the password vault system stores vault files on the server (as shown in Fig. 1). Vault files contain users' frequently used passwords (encrypted by master passwords) and related information (e.g., domains and usernames). Once the server is compromised, attackers can perform offline guessing on the users' master passwords to recover all password vaults, i.e., attackers use candidate master passwords to trial-decrypt the vault one by one. The decrypted random junk is a strong signal to attackers that the wrong candidate master password is tried, and attackers can immediately rule out the candidate master passwords. Attackers can implement 10^9-10^{12} guesses per day using existing password cracking software [28], [30]. Therefore, offline guessing poses a major threat to conventional password vault systems.

To address this issue, *honey encryption (HE)* techniques have been introduced into password vaults [4], [5], [6], [12]. Honey encryption [18] contains a conventional symmetric encryption scheme Se that uses low-entropy keys (e.g., pass-

words) and a randomized encoder: *distribution-transforming encoder (DTE)*. Encryption first uses DTE to encode the message into a bit string (called code), and then encrypts the code with Se. Decryption is the opposite. We call a password vault system deployed with honey encryption as a *honey vault system*. When decrypted with a wrong master password, the honey vault system can yield decoy vaults instead of random junk to confuse attackers. In an offline guessing attack, a plausible vault is generated per trial-decryption, forcing the attacker to implement online verification. That is, the attacker needs to enter the passwords in every plausible vault on some websites and check whether the login is successful. Since online guessing can be detected and blocked effectively [11], [13], [29], rational attackers need to determine an optimal verification order for plausible vaults to reduce the time cost of verification. It weakens the offline guessing attacker's ability to distinguish the real vault from decoy vaults. Therefore, honey encryption significantly improves the security of vault systems.

1.1. Motivations

The goal of a honey vault distinguishing attacker is to distinguish the real vault from decoy vaults by online login attempts. However, the modeling of attackers/adversaries in existing works [18] is far from realistic. Juels and Ristenpart [18] derive a *theoretical* upper bound on the attacker's success rate under *one guess and one user* in a formal cryptographic security game against honey encryption. However, in practice, a variety of attacks can be launched (see [5], [6], [12]), and these attacks are hard to be covered by the theoretical model in [18]. More realistically, attackers can often perform multiple online guesses on multiple users [4], [5], [6], [12]. All this makes it impossible for theories in [18] to accurately model the actual security of honey vault systems, and thus users' passwords cannot be secured. *In all, it remains a challenge to accurately model the attacker's advantage, and provide practical upper/lower bounds on various realistic attacks*.

Distribution transforming encoder (DTE) is the core component of a honey encryption (HE) scheme. The security of DTE is quantified by the difference between the distribution of the binary strings outputted by DTE and the uniform random distribution. The more similar the two distributions are, the more secure the DTE is. However, existing security proof techniques are relatively direct (i.e., using statistical distance [4], [6], [18]) and cannot accurately measure the security of DTE. Similar problems exist in other HE-related cryptographic studies (we will detail them in Sec. 2). A much more accurate upper/lower bound can help us understand the security of existing systems and avoid choosing unnecessarily large security parameters, such as the encryption round and the code length. To our knowledge, there is a lack of relevant studies on these issues.

Most prior art on honey vaults (see [4], [5], [6], [12]) mainly use heuristic distinguishing attacks (without solid theoretical foundation) to evaluate the security of honey vault systems. For example, Cheng et al. [5] assume the attacker is unaware of the salt when guessing and has excessive knowledge of the side-channel information of each password in a vault. Thus, existing attacks cannot be used to accurately evaluate the honey vault systems' actual security and may lead to biased security advice. *It is necessary to design realistic and theoretically-grounded attacks that help us better evaluate honey vault systems' actual security.*

1.2. Background and Related work

In this section, we give a brief summary of some key concepts and important prior arts related to honey encryption and honey vaults, enabling a wide audience to access our work. Further, we show technical details in Sec. 2.

Message recovery security (MR security). Juels and Ristenpart [18] first proposed honey encryption (HE) and its core component, the distribution transforming encoder. The basic security property of HE is message recovery security (MR security), which is defined in the MR security game; i.e., the user randomly picks a message and a low-entropy key, then obtains the ciphertext by encryption, and sends the ciphertext to the MR adversary $\mathcal{A}$, who wins if $\mathcal{A}$ outputs the real message. We use $\mathcal{A}$'s success rate in the MR game (called MR advantage) to measure the MR security of HE.

The security of distribution transforming encoder. The security of distribution transforming encoder (DTE) [18] is defined against the distinguishing game, i.e., the distinguisher $\mathcal{B}$ tries to tell the difference between the distribution of the bit string outputted by the DTE and the random uniform distribution on bit strings. The higher the $\mathcal{B}$'s advantage, the lower the security of the DTE.

Inverse sampling distribution transforming encoder (IS-DTE). Juels and Ristenpart [18] built an instantiation of DTE, i.e., IS-DTE, utilizing the inverse sampling technique. IS-DTE can encode message distributions into bit strings.

Message recovery security under Known Message Attack (MR-KMA security). In 2016, Jaeger et al. [17] proposed MR-KMA security for HE. The only difference with MR security is that MR-KMA adversaries are capable of accessing the public encryption oracle. Also, they proved that HE with a low-entropy key setting is unable to satisfy MR-KMA security. In addition, Jaeger et al. [17] further proposed target-distribution semantic (TDSS) security and target-distribution non-malleability (TDNM) security. They upper bounded TDSS and TDNM advantage using similar proof techniques originating from the Ball&Bin game (a mathematical model, which we illustrate in Sec. 2.1).

Message recovery security with Side Information (MR-SI security). Oprisanu et al. [24] introduced MR-SI security for honey encryption. The key distinction of MR-SI security is that adversaries know partial information about the challenge message, as opposed to MR security.

Honey vault systems. At IEEE S&P'15, Chatterjee et al. [4] proposed the first honey vault system: NoCrack, and its core is a carefully designed natural language encoder (NLE) based on the PCFG model [32], [38], i.e., NoCrack's NLE. The NoCrack's NLE [4] are specifically for encoding passwords into random-looking codes. Due to the honey nature of the NLE, a random seed trial-decrypted

by arbitrarily wrong master passwords can be decoded into a decoy password (or decoy vault). In 2016, Golla et al. proposed Golla-NLE [12], which is similar to NoCrack's NLE [4] except that the former employs the Markov model [19]. They also proposed an adaptive mechanism [12], which can adjust generated decoy vaults according to the real vaults to behave statistically closer to the real vault distribution.

Distinguishing attacks against honey vault systems. The goal of distinguishing attacks is to tell the real vault from a set of decoy vaults. In 2016, Golla et al. proposed the Kullback-Leibler divergence attack [12], which estimates the distance between the decrypted vault and decoy passwords sampled from the NLE. A closer vault is more likely to be identified as a decoy. In 2019, Cheng et al. found that NoCrack's NLE [4] and Golla-NLE [12] are vulnerable to encoding attacks [6], i.e., attackers can check some simple features in the code of the vault to tell the real vault from decoy vaults without extra information. Cheng et al. [6] also proposed a generic NLE design idea resistant to encoding attacks. However, their design idea is impractical due to the high complexity of enumerating all the generating rules in NoCrack's NLE [4] and Golla-NLE [12]. In 2021, Cheng et al. [5] proposed a distinguishing attack strategy and then instantiated this strategy into five realistic attacks.

Intersection attacks and Incrementally Updateable mechanism. The goal of intersection attacks is also to distinguish the user's real vault from decoy vaults. However, unlike distinguishing attacks, intersection attacks mainly consider that the attacker obtains the ciphertext pair from both the user's current *new vault* and the *old vault*. Typically, a user's new vault and old vault differ by one or more passwords at the end. A correct decryption will obtain two real vaults with some password overlap, while an incorrect decryption will obtain two *completely different* decoy vaults with a high probability. To resist intersection attacks, Cheng et al. [5] propose the Incrementally Updateable mechanism (IU mechanism). IU mechanism [5] uses prefix-preserving encryption to guarantee that any decrypted old and new vaults contain a large number of overlapping passwords as a way to confuse attackers.

1.3. Our contributions

The contributions of this work are as follows:

- **New honey encryption theories.** We propose an optimal attack strategy for MR adversaries and provide an accurate upper bound for MR advantage using our proposed optimal strategy, which fills a gap between the theory of HE and practical applications/attacks. To accurately estimate the security of IS-DTE [18], we provide a much tighter upper bound for the IS-DTE's security. Additionally, we significantly improve the lower bound for MR-KMA advantage, the new lower bound is about 60 times better than Jaeger et al.'s result [17] in the evaluation of real-world datasets. Based on the same technique, we provide a lower bound for MR-SI advantage.
- **New attacks against honey vaults and extensive evaluation**. We propose a new encoding attack, i.e., *super encoding attack*. It is more comprehensive than the strong/weak encoding attack [6], as it exploits the code distribution of Chatterjee et al's DTE output [4]. In our evaluation using the real-world dataset, the super encoding attack improves the attack success rate by 1.15-4.35 times compared with its counterparts. In addition, we instantiate three theoretically-grounded attacks using our optimal strategy. Extensive experiments reveal that our three attacks improve by 1.23-2.05 times compared with their counterparts.
- **Further Exploration**. We analyze the current adaptive mechanism in Golla-NLE [12] and reveal that it can not effectively enhance the security of Markov model-based NLE. Additionally, we propose a feature attack against Cheng et al.'s Incrementally Updateable mechanism (IU mechanism). By obtaining the ciphertexts of a user's new and old vaults, our attack achieves a significant advantage of nearly 90% in breaching the IU mechanism [5].

2. Preliminaries

2.1. Notation and Definitions

Notations. We denote by $y \leftarrow \mathcal{A}(x)$ that given a randomized algorithm $\mathcal{A}$ on input x, setting y equal to random output of $\mathcal{A}$. When $\mathcal{A}$ on input x is a deterministic algorithm, we write $y := \mathcal{A}(x)$. If G is a security game, we let $\Pr(\mathsf{G}{=}1)$ and $\Pr(\mathsf{G} \Rightarrow \mathsf{true})$ denote G outputs 1 and true, respectively. We use calligraphic uppercase to denote a set/space, e.g., $\mathcal{S}$. A distribution on set $\mathcal{S}$ is a function $p_{\mathcal{S}}$ or $\Pr_{\mathcal{S}}(\cdot) : \mathcal{S} \rightarrow [0,1]$, such that $\sum_{s \in \mathcal{S}} \Pr_{\mathcal{S}}(s){=}1$. By $s \leftarrow\!\!\!{}^{p_{\mathcal{S}}} \mathcal{S}$ we denote sampling an element $s \in \mathcal{S}$ according to the distribution $p_{\mathcal{S}}$ and $s \leftarrow\!\!\!{}^{\$} \mathcal{S}$ we denote sampling uniformly at random. For ease, we further simplify $s \leftarrow\!\!\!{}^{p_{\mathcal{S}}} \mathcal{S}$ to $s \leftarrow \mathcal{S}$. We denote by $|\mathcal{S}|$ the size of $\mathcal{S}$. For $a, b \in \mathbb{Z}$, we denote by $(a, b]$ the set $\{a{+}1, \ldots, b\}$, and by $[a, b]$ the set $\{a, \ldots, b\}$.

For honey encryption security definitions, such as message recovery security (MR security), we use the MR game to denote the cryptographic security game defining MR security, the MR adversary to denote the adversary participating in the MR game, and the MR advantage to denote the advantage obtained by the MR adversary in the MR game. For other HE-related security definitions such as MR-KMA, and MR-SI, we also use the same abbreviation convention.

Distance measures. Let μ and ν be two distributions on a finite event space Ω; the *statistical distance* between μ and ν is defined as $\|\mu\text{-}\nu\| := \sum_{x \in \Omega} \max\{0, \mu(x)\text{-}\nu(x)\}$. The *Kullback Leibler (KL) divergence* is defined as $\Delta_{\mathsf{KL}}(\mu, \nu) := \sum_{x \in \Omega} \mu(x) \ln(\frac{\mu(x)}{\nu(x)})$. The *chi-squared divergence* is $\chi^2(\mu, \nu) := \sum_{x \in \Omega} \frac{(\mu(x)\text{-}\nu(x))^2}{\nu(x)}$. Note that $\Delta_{\mathsf{KL}}(\mu, \nu)$ and $\chi^2(\mu, \nu)$ is well-defined because we require μ is full support. The relation among the three measures can be captured by Lem. 1 (Pinsker's inequality). The proof for Lem. 1 is given in Appendix A.1.

Lemma 1. **(Pinsker's inequality)** Let μ and ν be two distributions on a finite event space Ω. Based on the above definition, we have

$$(\|\mu-\nu\|)^2 \le \frac{1}{2}\Delta_{\mathsf{KL}}(\mu,\nu); \quad \Delta_{\mathsf{KL}}(\mu,\nu) \le \chi^2(\mu,\nu). \tag{1}$$

Information-theoretic indistinguishability. The distinguishing game is used to estimate the differences between two systems $\mathbf{S}_0$/$\mathbf{S}_1$. For a distinguisher $\mathcal{B}$ with access to one of two systems, many security proofs require an upper bound on

$$\mathsf{Adv}^{\mathsf{dist}}_{\mathbf{S}_0,\ \mathbf{S}_1}(\mathcal{B}) = |\Pr(\mathcal{B}(\mathbf{S}_0)=1) - \Pr(\mathcal{B}(\mathbf{S}_1)=1)|. \tag{2}$$

In symmetric encryption, there exist some ways upper bounding $\mathsf{Adv}^{\mathsf{dist}}_{\mathbf{S}_0,\ \mathbf{S}_1}(\mathcal{B})$ using statistical distance [2], [21]. More specifically, $\mathcal{B}$ performs a query u, and the system $\mathbf{S}_0$/$\mathbf{S}_1$ sends a response v. Let (u,v) be a query-response pair. $\mathcal{B}$ can perform multiple queries to $\mathbf{S}_0$/$\mathbf{S}_1$, and the system will answer each query. Let the number of queries by $\mathcal{B}$ be q (related to $\mathcal{B}$'s capability), we denote the ordered sequence of q query-response pairs as $\mathbf{Z}_q = ((u_1,v_1),\ldots,(u_q,v_q))$ and u_i and v_i denote the i^{th} query and the system $\mathbf{S}_0$/$\mathbf{S}_1$'s response, respectively. Let $\Pr_{\mathbf{S}_0}(\mathbf{Z}_q)/\Pr_{\mathbf{S}_1}(\mathbf{Z}_q)$ be the probability of the sequence of q query-response pairs in system $\mathbf{S}_0$/$\mathbf{S}_1$, then we have

$$\mathsf{Adv}^{\mathsf{dist}}_{\mathbf{S}_0,\ \mathbf{S}_1}(\mathcal{B}) \le \sum_{\mathbf{Z}_q} \|\Pr_{\mathbf{S}_0}(\mathbf{Z}_q) - \Pr_{\mathbf{S}_1}(\mathbf{Z}_q)\|, \tag{3}$$

where $\mathbf{Z}_q$ contains all sequence of q query-response pairs.

Ball&Bin game. There are m balls with mass distribution p_{ball} and n bins. Each ball enters a bin randomly according to the distribution of the bin p_{bin}. Each bin mass is called the load of the bin. We focus on the expectation of the maximum load, denoted by $\mathbb{E}[L_{p_{\mathsf{ball}},p_{\mathsf{bin}}}]$ where p_{ball} and p_{bin} are parameters. The MR security of HE can be modeled using this game. We mainly discuss this game in Sec. 3.1.

2.2. Honey encryption

Distribution transforming encoder (DTE). The DTE is an algorithm pair (encode, decode), where encode is a randomization algorithm. encode's input and output are defined on the message space $\mathcal{M}$ and the seed/code space $\mathcal{S}=\{0,1\}^l$, where $l=l(\lambda)$ is code length, λ is the security parameter and usually can be ignored. decode is vice versa. A secure DTE must satisfy following properties.

1) Correctness. We require

$$\Pr(\mathsf{decode}(\mathsf{encode}(m))=m : m\leftarrow\mathcal{M}) = 1. \tag{4}$$

2) Pseudorandomness. For all (even computationally unbounded) distinguishers $\mathcal{B}$, we require $\mathcal{B}$'s advantage $\mathsf{Adv}^{\mathrm{dte}}_{\mathsf{DTE},\ p_{\mathcal{M}}}(\mathcal{B})$ to be

$$|\Pr(\mathsf{SAMP0}^{\mathcal{B}}_{p_d}=1) - \Pr(\mathsf{SAMP1}^{\mathcal{B}}_{p_{\mathcal{M}}}=1)| \le \mathsf{negl}(\lambda) \tag{5}$$

where the $\mathsf{SAMP0}^{\mathcal{B}}_{p_d}$ and $\mathsf{SAMP1}^{\mathcal{B}}_{p_{\mathcal{M}}}$ are defined in Fig. 2. According to Eq. 5, the pseudorandomness required for DTE is that it is difficult for $\mathcal{B}$ to distinguish between $m\in\mathcal{M}$ that is chosen according to $p_{\mathcal{M}}$ and chosen by firstly picking a seed s uniformly at random and then applying $\mathsf{decode}(s)$ (gray part in $\mathsf{SAMP0}^{\mathcal{B}}_{p_d}$ in Fig. 2).

Noted the subscript of $\mathsf{SAMP0}^{\mathcal{B}}_{p_d}$, we let $p_d/\Pr_d(\cdot)$ be the distribution on $\mathcal{M}$ induced by the DTE. Formally,

$$\Pr_d(m) = \Pr(m=\mathsf{decode}(s) : s\xleftarrow{\$}\mathcal{S}) \tag{6}$$

Instantiations of DTE. At Eurocrypt'14, Juels and Ristenpart [18] built a DTE instantiation using the inverse sampling technique, i.e., IS-DTE = (is-encode, is-decode). Let $\mathrm{CDF}(\cdot)$ be the cumulative distribution function of a message distribution $p_{\mathcal{M}}$ according to some order $\{m_1,m_2,...,m_{|\mathcal{M}|}\}$ and define $\mathrm{CDF}(m_0)$:=0, let the code length l and factor ρ satisfy $\rho := \frac{1}{2^l} \ll \min_m \Pr_{\mathcal{M}}(m)$. Next, define the representation function $\mathsf{rep}_\rho(\cdot)$, such that for any value $a\in[0,1]$, $\mathsf{rep}_\rho(a) := \arg\min_{b\in\mathbb{N}} |a-b\rho|$.

For is-encode$(\cdot)$, we have

$$s\xleftarrow{\$}[\mathsf{rep}_\rho(\mathrm{CDF}(m_{i-1})), \mathsf{rep}_\rho(\mathrm{CDF}(m_i))). \tag{7}$$

For is-decode$(\cdot)$, the message distribution $p_{\mathcal{M}}$ is public even to the adversary/attacker, thus one can simply determine the location of the code s, and then recover m.

In addition, there also exists DTE for special data, such as Chatterjee et al.'s DTE [4] (Chatterjee-DTE for short). Chatterjee-DTE [4] is specific for encoding fractions of the form $\frac{p}{q}$, where $p,q\in\mathbb{N}$ and $p\le q$. Usually, the fraction $\frac{p}{q}$ is the cumulative-frequency of a message [4]. We detail Chatterjee-DTE [4] in Sec. 4.2. Exploiting the code distribution of Chatterjee-DTE output [4] can help us implement attacks against honey vault systems.

$\mathsf{SAMP0}^{\mathcal{B}}_{p_d}$	$\mathsf{SAMP1}^{\mathcal{B}}_{p_{\mathcal{M}}}$
$s\xleftarrow{\$}\mathcal{S}$	$m\leftarrow\mathcal{M}$
$m:=\mathsf{decode}(s)$	$b\leftarrow\mathcal{B}(m)$
$b\leftarrow\mathcal{B}(m)$	**return** b
return b	

Figure 2: Games for DTE security. The left is SAMP0 and the right is SAMP1. The difference comes from the message distribution (The gray part).

Honey encryption. An HE scheme (HEnc, HDec) is a symmetric encryption scheme $\mathsf{Se}=$ (enc, dec) with a DTE DTE = (encode, decode). As shown in Fig. 3, in encryption, the message m is first encoded into a code s then encrypted using Se.enc (Hash based symmetric encryption scheme in Fig. 3). Decryption is the opposite. The difference between HE and conventional symmetric encryption is that the former needs to be tailored according to the message distribution.

$\mathsf{HEnc}(k,m)$	$\mathsf{HDec}(k,c')$
$s\leftarrow\mathsf{encode}(m)$;	$(c,r):=c'$
$r\xleftarrow{\$}\{0,1\}^*$	$s:=c\oplus\mathsf{h}(r\|k)$
$c:=\mathsf{h}(r\|k)\oplus s$	$m:=\mathsf{decode}(s)$
$c':=(c,r)$	**return** m
return c'	

Figure 3: Hash-based honey encryption (HE) scheme, which is the HE scheme using a cryptographic hash h, i.e., the symmetric encryption scheme Se is based on h (typically modeled as a random oracle).

MR & MR-KMA security. Juels and Ristenpart [18] first instantiated HE schemes that achieve message recovery security (MR security). MR game is defined in Fig. 4. For a given HE scheme and message distribution $p_{\mathcal{M}}$, key distribution $p_{\mathcal{K}}$, the MR adversary $\mathcal{A}$ (even computationally unbounded) enters as input a ciphertext c of a challenge message m and outputs a guess m^*, where the challenge message m and key k are picked according to distributions

MR	MR-KMA	**Enc**
$m \leftarrow \mathcal{M}; k \leftarrow \mathcal{K}$	$m \leftarrow \mathcal{M}; k \leftarrow \mathcal{K}$	$m \leftarrow \mathcal{M}$
$s \leftarrow \mathsf{encode}(m)$	$s \leftarrow \mathsf{encode}(m)$	$c \leftarrow \mathsf{HEnc}(m, k)$
$c \leftarrow \mathsf{enc}(s, k)$	$c \leftarrow \mathsf{enc}(s, k)$	**return** m, c
$m^* \leftarrow \mathcal{A}(c)$	$m^* \leftarrow \mathcal{A}^{\mathbf{Enc}}(c)$	
return $m{=}{=}m^*$	**return** $m{=}{=}m^*$	

Figure 4: Games defining message recovery security (MR security) and message recovery security under a known message attack (MR-KMA security). The left is the MR game, the middle is the MR-KMA game, and the right is a public encryption oracle used in the MR-KMA game.

$p_{\mathcal{M}}$ and $p_{\mathcal{K}}$. $\mathcal{A}$ wins iff $m^*{=}{=}m$. We measure the MR advantage by

$$\mathsf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) := \Pr(\mathrm{MR}^{\mathcal{A}}_{p_{\mathcal{M}},p_{\mathcal{K}}} \Rightarrow \mathsf{true}). \tag{8}$$

Jaeger et al. [17] proposed message recovery security for HE under Known Message Attack (MR-KMA security). The MR-KMA game is shown in Fig. 4 The MR-KMA adversary $\mathcal{A}$ is able to access the encryption oracle **Enc**. With a low-entropy key setting, $\mathcal{A}$ has an infinite number of **Enc** queries. We measure the MR-KMA advantage by

$$\mathsf{Adv}^{\text{mr-kma}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) := \Pr(\text{MR-KMA}^{\mathcal{A}}_{p_{\mathcal{M}},p_{\mathcal{K}}} \Rightarrow \mathsf{true}). \tag{9}$$

2.3. Honey vault system

Natural language encoder (NLE). The core of the honey vault system is the NLE, which encodes passwords/vaults into random-looking bit strings (i.e., codes/seeds) and decodes arbitrary bit strings into passwords/vaults. More specifically, the NLE usually contains two sub-modules, i.e., a password probability model (PPM, such as PCFG model [32], [38] and Markov model [19], [34]) and DTEs. When encoding a vault, the PPM first parses each password into multiple ordered rules, each of which is assigned a probability, and then encodes these rules using DTE. The decoding is the opposite. According to Kerchoff's principle, both PPM and DTEs are public. The security of the NLE directly affects the security of the honey vault system.

The two existing honey vault systems are NoCrack [4] and Golla-NLE [12]. The password probability model (PPM) in NoCrack's NLE [4] is PCFG model [38], while in Golla-NLE [12] is Markov model [19]. For ease, we say that NoCrack's NLE [4] is based on PCFG model [38], while Golla-NLE [12] is based on Markov model [19]. Both the two NLEs use Chatterjee-DTE [4] to encode the rules. Meanwhile, Golla-NLE [12] uses an adaptive mechanism to generate more real decoy vaults [12], the idea is to heuristically increase the frequency of some rare rules in the Markov model, thus distorting the rule distribution in the Markov model to confuse semantic-aware attackers.

2.4. Security model

The fundamental security goal of honey vault systems is that, when given a vault ciphertext and wrong master passwords, the honey vault system will generate a number of decoy vaults, making them indistinguishable from the real vault. This goal is defined against a honey vault distinguishing attacker (HV attacker) $\mathcal{A}$ who has obtained the encrypted vault files (such as the online synchronization service from the password manager). Then $\mathcal{A}$ can try (almost) all possible candidate master passwords to trial-decrypt the vault ciphertext and obtain n plausible-looking vaults, where n is the number of candidate master passwords tried by $\mathcal{A}$, and distinguish the real vault from n plausible-looking vaults.

REMARK. HV attackers inherently differ from MR/MR-KMA adversaries against honey encryption and DTE distinguishers $\mathcal{B}$, because the latter three are theoretical adversaries against HE, while HV attackers are realistic attackers against honey vault systems. To avoid ambiguity, when we use the adversary/attacker notation $\mathcal{A}/\mathcal{B}$, we will specify it in advance.

Honey vault distinguishing attacker (HV attacker). In this work, we focus on distinguishing attacks against honey vault systems (i.e., HV attackers). Given any user U_i's encrypted vault file, the attack process of the HV attacker $\mathcal{A}$ is as follows.

1. $\mathcal{A}$ uses n candidate master passwords to trial-decrypt the vault ciphertext, then gets n plausible-looking vaults.

2. $\mathcal{A}$ assigns a score to each of the n vaults through a score function $\mathsf{s}(\cdot)$, and logs in to the corresponding domains according to the rank of scores to implement online verification.

In the distinguishing attack, we mainly focus on the order of online verification determined by the score function, which can reflect the attack success rate and the security of the honey vault system. Moreover, we assume that $\mathcal{A}$ can always recover the real vault within n=1000 attempts. This practice closely follows the best experimental setup in the main-stream honey vault studies [4], [5], [6], [12]. Additionally, we have also tested larger values of n = 10,000, 1,000,000, which yielded similar results but significantly increased the overall experimental time cost.

Attacker capabilities. In our evaluation, we assume that the HV attacker $\mathcal{A}$ has somehow obtained the encrypted vault files (i.e., vault ciphertexts), the algorithms of the NLE and the DTE, etc., and knows all public information such as the publicly leaked third-party password datasets. All distinguishing attacks are based on a carefully-designed score function $\mathsf{s}(\cdot)$ used to score the vaults, and the order of $\mathcal{A}$'s online verification is based on the scores. Also, $\mathcal{A}$ can use the publicly leaked password datasets to train some data-driven score functions.

Other attackers: Intersection attacks. The goal of intersection attacks is to distinguish the user's real vault from decoy vaults, similar to the honey vault distinguishing attacks. In intersection attacks, the attacker $\mathcal{A}$ obtains the ciphertext pair (c^n, c^o) from the user's both current *new vault* and the *old vault*. Typically, the new vault and old vault differ by one or more passwords at the end. In each trial decryption of the ciphertext pair (c^n, c^o), $\mathcal{A}$ obtains two plausible vaults (v^n, v^o). A correct master password will decrypt (c^n, c^o) into two real vaults with some password overlap, while an incorrect master password will decrypt it into two *completely different* decoy vaults with a high probability.

2.5. Evaluation metrics

This work adopts the following metrics to equally the security of a honey vault system.

Average rank $\bar{r}$ and accuracy α are proposed by Chatterjee et al. [4], where $\bar{r}$ is the average rank of all the real vaults given a honey vault system and an attack. The α is the accuracy of distinguishing a real vault from a decoy one. The two metrics measure the $average\text{-}case$ performance of a honey vault system and a practical attack. An ideal honey vault system has $\bar{r}$ of 0.5 and α of 0.5, while an ideal attack has $\bar{r}$ of 0 and α of 1.

Cumulative distribution graph (CDG) is proposed by Cheng et al. [5], [6], which plots the ratio y of successfully cracked vaults, when the attacker makes x online verifications. This metric measures the global picture of the attack success rate. Golla et al. [12] proposed the discrete metric Q_k where $k = 0.25, 0.5$, which means that k fraction of vaults are among the top-ranked Q_k of the vaults. Thus, Q_k can also be represented in the CDG by x when y is k.

2.6. Our datasets and ethics considerations

Datasets. Our evaluation relies on Pastebin, which is the only publicly accessible password vault dataset as far as we know. Pastebin may have been leaked before June 2011 and appears to have been collected by malware running on many clients. Pastebin was first used by Chatterjee et al. [4] to evaluate NoCrack's NLE [4] and make the dataset available along with the NoCrack source code. Pastebin contains the usernames/domains-password pairs in every vault. Since the honey vault systems we evaluate do not require personal information, we only focus on the passwords in Pastebin. Pastebin contains 276 vaults of sizes 2 to 50. In our evaluation, we use these 276 vaults as our testset.

To make our experiments fairer, we use the `Rockyou` dataset (`Rockyou` for short) as an auxiliary password dataset as [4], [5], [6], [12], i.e., using `Rockyou` to train some data-driven attacks. `Rockyou` was leaked in December 2009 [1], which is one of the largest password datasets disclosed early and widely used in recent password security studies [16], [19], [20], [25], [26], and contains 32.6 million passwords. Therefore, `Rockyou` can provide large enough password samples that an attacker can learn early user password behaviors.

In addition, to make our evaluation more comprehensive, we also incorporate a recently leaked dataset, `Wishbone`, as another auxiliary password dataset. `Wishbone` was leaked in January 2020 [3] and may reflect the most current user password behaviors. It is a popular mobile app that allows users to compare two items in a simple poll. The Wishbone dataset is originally leaked in MD5 (not plaintext), and costs us 9 months to recover 95.61% of it using various password guessing models [28], [30], [35], [37]. These recovered passwords encompass diverse compositions (`@Ss7596953`, `Avarose#101`), semantics (`watermelon1`, `l0vemet0day`), and many randomly looking passwords (`$oftb@ll#1`). Therefore, the current set of plaintext passwords we recovered well represents the distribution of the whole dataset. Following the data cleaning method used by [34], [35], we removed non-password strings from the original dataset, including titles, descriptions, footnotes, and non-ASCII strings. We also removed passwords with length > 30, as they may have been randomly generated by password managers or spam.

Ethics considerations. We are aware that although publicly available and widely used in the existing literature (e.g., [4], [5], [6], [12], [16], [19], [20], [25], [26]), these datasets are private data. Therefore, for non-plaintext data recovery (i.e., retrieving `Wishbone` plaintext passwords from MD5 hashes), our workstations are kept strictly independent from the external network during the recovery, which ensures that our password recovery is done only locally and independently and in privacy. For all the plaintext password data, we only report aggregated statistical information and treat each individual account as confidential so that their use in our study does not increase the risk to the corresponding victim. We have consulted with privacy experts several times and obtained approval from our center's IRB. In addition, these datasets may be exploited by attackers as training samples, and our use facilitates the academic community to understand the realistic security of honey vault systems. Since our datasets are publicly available on the Internet, the results of this work are fully reproducible.

3. Revisiting honey encryption

In this section, we propose a theoretically-grounded optimal attack strategy for the MR adversary and further derive a new upper bound on the MR advantage. Besides, we give a tighter upper bound on the security of IS-DTE [18] and further analyze MR-KMA and MR-SI security for honey encryption and give much tighter bounds for them.

3.1. Advantages against MR security

Revisiting JR's proof. Juels and Ristenpart [18] first upper bounded the MR advantage, their proof contains three steps:
1) **Estimating the difference between MR game (Fig. 4) and Game_2 (Fig. 5).** The message distributions in MR game and Game_2 are $p_{\mathcal{M}}$ and p_d (gray part) separately. Therefore, the difference between the MR game and Game_2 is, at most, the upper bound of the DTE distinguisher $\mathcal{B}$'s advantage.
2) Game_2 **is equal to** Game_3**.** Assumption of the random oracle model for the conventional encryption scheme enables us to assume ciphertexts (in the box) to be uniform (i.e., the same as in Game_3), so Game_2 is equivalent to Game_3.
3) Game_3 **is a Ball&Bin game.** According to our assumptions, the keys are considered as balls and the messages are considered as bins, and the maximum expected loading $\mathbb{E}[L_{\mathsf{HE},p_{\mathcal{K}}}]$ of the Ball&Bin game upper bounds the advantage of Game_3. Finally, we use the hybrid argument to get the advantage of MR adversary to satisfy

$$\mathsf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_{\mathcal{M}}}(\mathcal{A}) + \mathbb{E}[L_{\mathsf{HE},p_{\mathcal{K}}}], \tag{10}$$

where $\mathbb{E}[L_{\mathsf{HE},p_{\mathcal{K}}}]$ is the maximum expected loading of a Ball&Bin game, the subscript is the parameters of Game_2.

Although the MR advantage based on DTE security is given, there are still some shortages in real-world applications: the encoding techniques used in some specific DTEs [4], [12] (e.g., Chatterjee et al.'s DTE [4]) are not the same as [18], we cannot directly estimate the MR advantage of the schemes based on these DTEs. Moreover, attackers against real-world applications (e.g., honey vault distinguishing attackers [5], [6], [12]) usually implement some attack strategies that cannot be captured by the security model in [18] (Even exploiting some vulnerabilities in the system implementation [6]). To address this issue, we give a theoretically-grounded optimal strategy for the MR game and derive a much tighter upper bound of MR advantage based on our strategy. *Meanwhile, our strategy can provide a theoretical basis for the design of attacks against different honey vault systems in Sec. 4.4.*

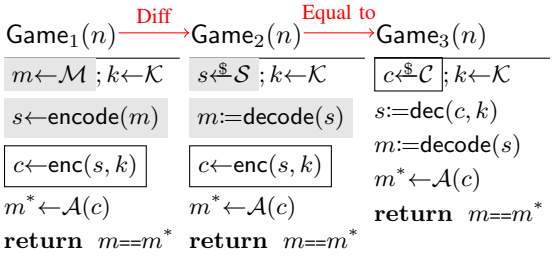


Figure 5: Games used in JR's proof [18]. Game$_1$ is the MR Game in Fig. 4. The difference between Game$_1$ and Game$_1$ is upper bounded by the distinguishing advantage; Game$_2$ and Game$_3$ are equal under the random oracle model; Game$_3$ is a Ball&Bin game under the random oracle model, the expected maximum loading is $\mathbb{E}[L_{\mathsf{HE},p_\mathcal{K}}]$.

Our optimal strategy against MR Game. To construct our optimal strategy against the MR game (shown in Fig. 4), we use the Bayesian method to estimate the posterior probability $\Pr(m|c)$ that the message picked by the user is m for a given ciphertext c. This requires us to detail the encryption process of the HE scheme. When the message m is picked, let $\mathcal{S}_m$ be the set containing all possible codes of m in the IS-DTE [18]. The IS-DTE [18] outputs a code $s\in\mathcal{S}_m$ uniformly at random. Then the user picks $k\in\mathcal{K}$ according to the distribution $p_\mathcal{K}$ and encrypts the code s using the symmetric encryption scheme Se. Let $\Pr(\mathsf{enc}(k,s)=c)$ be the probability that Se.enc outputs the ciphertext c. Since the code s and the key k are picked randomly, we need to enumerate all cases of $s\in\mathcal{S}_m$ and $k\in\mathcal{K}$ to accumulate the posterior probability of yielding the given ciphertext c for a given message m. That is, $\sum_{s\in\mathcal{S}_m}\sum_{k\in\mathcal{K}}\Pr_\mathcal{K}(k)\Pr(\mathsf{enc}(k,s)=c)$. According to the above analysis, we first present our Thm. 1, which serves as the theoretical foundation for our optimal attack strategy against the HE scheme.

Theorem 1. Let HE be the honey encryption scheme with an IS-DTE [18]. Given the ciphertext c, if the guess of the MR adversary $\mathcal{A}$ is m (shown in Fig. 4), then the MR advantage is at most

$$\sum_{s\in\mathcal{S}_m}\frac{\Pr_\mathcal{M}(m)}{|\mathcal{S}|\Pr_d(m)\Pr_\mathcal{C}(c)}\sum_{k\in\mathcal{K}}\Pr_\mathcal{K}(k)\Pr(\mathsf{enc}(k,s)=c), \tag{11}$$

where $\Pr_\mathcal{M}(\cdot)$, $\Pr_\mathcal{K}(\cdot)$, $\Pr_\mathcal{C}(\cdot)$ denote the probability distribution of the message, low-entropy key, and ciphertext. $\Pr_d(\cdot)$ denotes the decoy message distribution modeled by the IS-DTE [18]. $\mathcal{S}_m$ contains all possible codes of m and $\mathcal{S}$ denotes code space. $\Pr(\mathsf{enc}(k,s)=c)$ denotes the probability that Se.enc encrypts the code s to the ciphertext c. Thm. 1 indicates that if $\mathcal{A}$ tries the message m as the guess for the real message, her expected attack success rate can be expressed by Eq. 11. The proof is given in Appendix A.2.

Explaining our optimal attack strategy. According to Thm. 1, our optimal attack strategy is that, the MR adversary $\mathcal{A}$ first calculates the Bayesian posterior probability of each message $m\in\mathcal{M}$ given the ciphertext c, using Eq. 11 in Thm. 1. $\mathcal{A}$ then picks the message with the highest posterior probability as her guess (denoted as $m^{(1)}$). Note that the superscript $(i)(i\in[1,|\mathcal{M}|])$ indicates its ranking is i^{th} according to the Bayesian posterior probability Eq. 11. From $\mathcal{A}$'s view, $m^{(1)}$ is the most likely message picked by the user. If this guess fails, $\mathcal{A}$ continues guessing subsequent messages in descending order, i.e., $\{m^{(2)},\ldots,m^{(|\mathcal{M}|)}\}$ until $\mathcal{A}$ succeeds. Overall, Thm. 1 is a Bayesian method to the MR game, estimating the probability of the message m given the ciphertext c, thus our attack strategy is optimal.

Comparing to the result in [18]. Based on our optimal strategy reflected in Thm. 1, we derive a much tighter upper bound for the MR advantage, i.e., Thm. 4 in Appendix A.2. Compared to the upper bound of the MR advantage in [18], our optimal attack strategy has two main advantages:

(1) In [18], Juels and Ristenpart construct two additional games (i.e., Game$_2$, Game$_3$ in Fig. 5), using the hybrid argument and scaling techniques for the Ball&Bin Game, to obtain a relaxed upper bound on the MR advantage. Compared to [18], our Bayesian method in Thm. 4 is the theoretically optimal attack itself. Therefore, Thm. 4 provides a more accurate MR advantage.

(2) For relevance to real-world attacks, our Thm. 1 can be used to design theoretically-grounded attacks against honey vault systems (e.g., [4], [12] see in Sec. 4.4), thus guiding honey vault distinguishing attackers (HV attackers) to implement effective distinguishing attacks. More specifically, Eq. 11 can be transformed into an effectively calculated score function for HV attackers. Both $\Pr_\mathcal{M}(\cdot)$ and $\Pr_d(\cdot)$ can be efficiently estimated by a password model and a natural language encoder, respectively. For the term $\Pr_\mathcal{K}(k)\Pr(\mathsf{enc}(k,s)=c)$, we can make a reasonable approximation based on a specific honey vault system. However, the upper bound given in [18] cannot achieve this. Our attack strategy fills the gap between the theory and practical security of honey encryption and honey vault systems.

3.2. Analysis of DTE security

The security of DTE directly affects the MR security of honey encryption (Recall Eq. 10). To estimate the MR advantage, it is necessary to estimate the DTE security accurately. However, Existing proof methods are rather crude and often do not provide tight bounds (see [6], [18]), especially when faced with some sophisticated concatenations of

multiple DTEs (e.g., NoCrack's NLE [4], Golla-NLE [12]). In this section, we give a much tighter bound on IS-DTE security using new proof techniques. Our techniques can also be used for the concatenation of multiple IS-DTEs [6].

Much tighter upper bounds for the IS-DTE [18] security. Our main proof techniques build on the Chi-Squared method, first proposed by Dai et al. [9], which can provide a more accurate upper bound on statistical distance. The main case considered by Dai et al. [9] is the adaptive query, i.e., the current response depends on previous ones. However, in our setting, we consider the non-adaptive distinguisher against DTE security, so we extend the Chi-square method to non-adaptive distinguishers, i.e., Lem. 2. The proof of Lem. 2 is given in Appendix A.3.

Lemma 2. **([9] Lemma 3, adapted)** For a distinguisher $\mathcal{B}$, capable of non-adaptively performing q queries on the system $\mathbf{S}_0$ (resp. $\mathbf{S}_1$), we obtain that

$$\sum_{\mathbf{Z}_q} \|\Pr_{\mathbf{S}_0}(\mathbf{Z}_q) - \Pr_{\mathbf{S}_1}(\mathbf{Z}_q)\| \leq (\frac{1}{2}\sum_{i=1}^{q}\chi_i^2)^{\frac{1}{2}}, \tag{12}$$

where χ_i^2 is the chi-squared divergence of the i$^{\text{th}}$ pair distribution in the two systems (SAMP0/SAMP1), i.e., $\sum_z \frac{(\Pr_{\mathbf{S}_0}(z)-\Pr_{\mathbf{S}_1}(z))^2}{\Pr_{\mathbf{S}_1}(z)}$. We get Thm. 2 using Lem. 1.

Theorem 2. Let $\mathcal{B}$ be a distinguisher distinguishing between the two systems shown in Fig. 2. The advantage is

$$\mathsf{Adv}^{\text{dte}}_{\text{IS-DTE},p_{\mathcal{M}}}(\mathcal{B}) \leq \frac{\sqrt{|\mathcal{M}|/2}}{2^l} \leq \frac{|\mathcal{M}|}{2^l}. \tag{13}$$

The proof is in Appendix A.4. Here we show the proof sketches. We first estimate the chi-squared divergence of the two systems' response distributions at one query. Then Lem. 2 is used to upper bound the divergence at q queries. For computationally unbounded distinguishers, we apply q=$|\mathcal{M}|$ to get the Eq. 13. Although the above process seems intuitive, there are still some technical difficulties, such as how to estimate the difference in the probability of "one query" z=(u, v) in two systems SAMP0/SAMP1. In summary, the key point is combining the estimation of IS-DTE [18]'s response distribution and Lem. 1.

Implications. Thm. 2 is useful to realistic honey vault system design, as it can reduce the code length of the original system by $\log(\sqrt{2|\mathcal{M}|})$ bits without weakening the system security. Reducing the code length can not only improve the system's computational performance but also reduce the storage cost of encrypted vault files.

3.3. Revisting the MR-KMA security

MR-KMA security is crucial to HE schemes. Jaeger et al. [17] state that in the low-entropy key setting, an MR-KMA adversary can achieve the advantage at least $\frac{1}{2\kappa^2}$ when **Enc** query number is at most κ=$\lceil \log |\mathcal{K}| \rceil$, where $\mathcal{K}$ is the low-entropy key space. However, the advantage $\frac{1}{2\kappa^2}$ is too small to enable users to understand the real-world MR-KMA security. Meanwhile, a global expression of how the advantage varies with the query number is needed, as it can better portray the global picture of the MR-KMA security.

Lower bounded MR-KMA security for hash-based HE. We propose a new attack against the MR-KMA game in the low-entropy key setting, and give a much tighter lower bounds for the MR-KMA advantage. Our attack enables the MR-KMA adversary to gain an overwhelming advantage and is far better than Jaeger et al.'s result [17]. Our result is shown below.

Theorem 3. For a hash-based HE scheme in the low-entropy key setting, there exists an MR-KMA adversary $\mathcal{A}$, for any distribution $p_{\mathcal{M}}$, $p_{\mathcal{K}}$ with at most $q \geq 1$ queries to oracle **Enc**, the MR-KMA advantage satisfies

$$\mathsf{Adv}^{\text{mr-kma}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) \geq 1 - |\mathcal{K}|\omega_m^q, \tag{14}$$

where $|\mathcal{K}|$ is key space size, and ω_m:=$\max_m \Pr_d(m)$. The proof is in Appendix A.6. Here we briefly show our proof idea. Let the user-picked key be k^*. $\mathcal{A}$ obtains q plaintext-ciphertext pairs $\{(c_i, m_i)\}_{i=1}^q$ through oracle **Enc**. For i$^{\text{th}}$ pair, $\mathcal{A}$ decrypts c_i to m_i using the key k∈$\mathcal{K}$ ($k \neq k^*$) with probability $\Pr_d(m) \leq \omega_m$. We call the key that makes all q ciphertexts decrypted correctly as the *consistent key*, and the probability of such an event is less than ω_m^q. Using the union bound for all keys except k^*, the probability of no consistent key is at least $1 - |\mathcal{K}|\omega_m^q$. This indicates that $\mathcal{A}$ can rule out all keys except the real key k^*.

Further, we use the `Rockyou` password dataset (see Sec. 2.6) to model both of the distributions of master passwords and common passwords (encrypted by master passwords) in honey vault applications. Let the `Rockyou` be $\mathcal{D}$, $|\mathcal{D}| = 32,581,870$, $\omega_m \approx 9 \times 10^{-3}$. When the **Enc** query number q=4, the MR-KMA adversary has an advantage of at least 78.6%. While applying Jaeger et al.'s result [17], the advantage is at least about 1.3%. Therefore, our new result can increase the lower bound of the MR-KMA advantage by about 60 times in the evaluation of the `Rockyou` dataset.

Additional works. To understand MR-KMA security for HE schemes, we further improve Jaeger et al.'s proof technique, breaking the limitation that Jaeger et al. [17] can only estimate MR-KMA advantage given some specific query numbers. We derive a global expression of MR-KMA advantage for general HE schemes (see Thm. 5 in Appendix A.5). Our MR-KMA advantage is much more precise than that of Jaeger et al.'s. Moreover, using a similar strategy, we further analyze the MR security under known side information attack (MR-SI security), giving an exact lower bound for MR-SI advantage (see Appendix A.7).

4. Security Analysis of Existing Honey Vaults

In this section, we analyze the security of existing honey vault systems using our proposed HE-related theorems in Sec. 3. We propose an encoding attack, i.e., *super encoding attack* and instantiate our optimal strategy in Sec. 3.1 into three theoretically-grounded attacks, i.e., *List-based attack* against NoCrack's NLE [4], *TarGuess-II-based attack* and *adaptive TarGuess-II-based attack* against Golla-NLE [12]. The overview of our all attacks is shown in Table 1.

4.1. Formalizing the process of NLE

The NLE is an essential module in the honey vault system. The security of the NLE directly affects the security of the honey vault system [4], [5], [6]. The NLE contains

TABLE 1: AN OVERVIEW OF OUR PROPOSED PRACTICAL ATTACKS AGAINST HONEY VAULT SYSTEMS

Targeted honey vault systems	attack	How best to instantiate $\Pr_{\mathrm{V}}(\cdot)$/$\Pr_{\mathrm{PW}}(\cdot)$		Instantiate $\Pr_d(\cdot)$	Alternative models†
		Password (reuse) model	Score function		
All systems [4], [12]	Super encoding attack	-	Eq. 16	-	-
NoCrack's NLE [4]	List-based attack	List model	Eq. 18	NoCrack's NLE [4]	Hybrid models*
Static Golla-NLE [12]	TarGuess-II-based attack	TarGuess-II [35]	Eq. 19	Static Golla-NLE [12]	Pass2path [25]
Adaptive Golla-NLE [12]	Adaptive TarGuess-II-based attack	TarGuess-II [35]	Eq. 19 with Pre-processing	Adaptive Golla-NLE [12]	Pass2path [25]

†These models have also been used to instantiate $\Pr_{\mathrm{V}}(\cdot)$/$\Pr_{\mathrm{PW}}(\cdot)$, but according to the results of our additional experiments Fig. 8, they are not as effective as List model and TarGuess-II [35]. Therefore, we mainly choose List model and TarGuess-II [35] to instantiate $\Pr_{\mathrm{V}}(\cdot)$/$\Pr_{\mathrm{PW}}(\cdot)$.
*The hybrid models αList +βPCFG+γMarkov ($\alpha, \beta, \gamma \in [0, 1]$ and $\alpha + \beta + \gamma = 1$) means: α fraction of probability are from List model, β from PCFG, Etc..

two sub-modules, i.e., a password probability model (PPM) and DTEs. We first formalize the process of the NLE. The goal of the NLE is to encode the vault into a code (aka. seed). Generally, the NLE parses each password in the vault into corresponding *password generation rules* using the PPM, then encodes these rules using Chatterjee-DTE [4], and then concatenates the codes of all rules as the code of the whole vault. For example, in the PCFG model, the generation rules of the password `abc123` are {`(S→WD)`, `(W→abc)`, `(D→123)`}, where `S`, `W`, `D` denote the start symbol, word symbol, and digital symbol, separately; in the 3-order Markov model, generation rules are {`(L→6)`, `(***, a)`, `(**a, b)`, ..., `(c12, 3)`}, where (L→6) denotes the password length, and the rest rules come from the Markov chain.

Let a vault v containing n passwords be v=$(pw_1, ..., pw_n)$. For the code, let $\mathbf{s}_v$ be the code of the vault v, let $\mathbf{s}_{pw_i}$ be the code of a password $pw_i \in v$, and let s be the code of a generation rule belonging to a password. For the rules, let r be a rule, let the $\mathbf{r}_{pw_i}$ be the rule set of a password $pw_i \in v$, and let $\mathbf{r}_v$ be the rule set of a vault. More specifically, for the i[th] password pw_i in the vault v, we denote the password rule set as $\mathbf{r}_{pw_i}$. If the password rule set $\mathbf{r}_{pw_i}$ contains k_i=$|\mathbf{r}_{pw_i}|$ ordered generation rules, we write it as $\mathbf{r}_{pw_i}$=$\{r_{i1}, ..., r_{ik_i}\}$. Thus, the vault rule set is $\mathbf{r}_v$=$\{\mathbf{r}_{pw_1}, \ldots, \mathbf{r}_{pw_n}\}$. After encoding, we let the code of the rule r_{ij} be s_{ij}, let the code of pw_i be $\mathbf{s}_{pw_i}$=$(s_{i1}, ..., s_{ik_i})$, and the code of the vault is $\mathbf{s}_v$=$(\mathbf{s}_{pw_1}, \ldots, \mathbf{s}_{pw_n})$.

For example, in a PCFG model, the rules for i[th] password `abc123` are (`(S→WD)`, `(W→abc)`, `(D→123)`), thus we have k_i=3, r_{i1} = `(S→WD)`,..., r_{i3} = `(D→123)`. If the code length is 4, and the corresponding codes are s_{i1} = `1100`, s_{i2} = `1001`, s_{i3} = `0011`, the code of the password `abc123` is $\mathbf{s}_{pw_i}$ = (`1100`, `1001`, `0011`). For ease, we assume all rules are encoded as the same length l and use the integers in [0, 2^l] to denote the code corresponding to the rule, so we can write the code of `abc123` as $\mathbf{s}_{pw_i}$ = (12, 9, 3).

We emphasize that the rules in a (password/vault) rule set need to be kept in order so that the NLE can use the ordered rules to recover the password and the vault. A password rule set can only correspond to one password, but a password may correspond to multiple password rule sets (According to the ambiguity of password probabilistic models in the NLE). For example, in a PCFG model, `password` may have two rule sets $\mathbf{r}^1$= {`S→W`, `W→password`)} and $\mathbf{r}^2$= {`S→WW`, `W→pass`, `W→word`}.

Here we take NoCrack's NLE as an example to show the whole process. NoCrack's NLE [4] is based on the PCFG model, which uses the sub-Grammar method [4] to capture the similarity of passwords in a vault. For a vault v = (`password`, `123456`), the encoding process consists of three steps:

1) Parsing the vault v into generation rules, we obtain two possible vault rule sets for v, i.e., $\mathbf{r}_v^1$= {`S→W`, `W→password`, `S→D`, `D→123456`} and $\mathbf{r}_v^2$= {`S→WW`, `W→pass`, `W→word`, `S→D`, `D→123456`}.

2) Selecting the rule set with the highest probability under the PCFG model $\Pr_{\mathsf{PCFG}}(\cdot)$. Assuming that $\Pr_{\mathsf{PCFG}}(\mathbf{r}_v^1) > \Pr_{\mathsf{PCFG}}(\mathbf{r}_v^1)$, we choose $\mathbf{r}_v^1$ as the rule set for v.

3) Encoding the sub-Grammar and the rules. We normalize the frequencies of each rule in $\mathbf{r}_v^1$ to construct the sub-Grammar, and then encode the sub-Grammar and $\mathbf{r}_v^1$ using Chatterjee-DTE [4]. Decoding is in the opposite direction.

The process of Golla-NLE [12] is similar to that of NoCrack's NLE [4], except for a Markov model. Essentially, the NoCrack's NLE [4] and Golla-NLE [12] are more comprehensive, we simplify it only for ease of explanation.

Notice that the PCFG model in NoCrack's NLE [4] is ambiguous, i.e., the vault v can generate two rule sets $\mathbf{r}_v^1$ and $\mathbf{r}_v^2$. But NoCrack's NLE [4] selects the rule set with the highest probability. This case provides the chance for the weak encoding attack [6]. Generally, if the HV attacker $\mathcal{A}$ obtains a vault code $\mathbf{s}$ by trial-decrypt a vault ciphertext c, $\mathcal{A}$ first decodes the code $\mathbf{s}$ into the vault rule set, say $\mathbf{r}^d$ = {`S→WW`, `S→D`, `W→pass`, `W→word`, `D→123456`}, then get the vault v = (`password`, `123456`) using $\mathbf{r}^d$. In order to identify whether the recovered vault v a real vault, $\mathcal{A}$ can parse the vault v in the encoding process and obtain another vault rule set $\mathbf{r}^e$ = {`S→W`, `W→password`, `S→D`, `D→123456`}. Noted that $\mathbf{r}^d \neq \mathbf{r}^e$, $\mathcal{A}$ can immediately identify v as a decoy (as the decoded rule set $\mathbf{r}^d$ is not obtained in the encoding process). For ease, we refer to the vault rule set $\mathbf{r}^d$ obtained by decoding the vault code $\mathbf{s}$ as the *decoding rule set*, and $\mathbf{r}^e$ obtained by parsing the vault v (recovered by the decoding rule set $\mathbf{r}_d$) as the *encoding rule set*. The weak encoding attack [6] checks whether the decoding rule set is the same as the encoding rule set. The score function $\mathsf{s}_{\mathsf{weak}}(v) = 1$ if yes, 0 otherwise.

The strong encoding attack [6] is a further attack, which uses the probability of the obtained rule set as the score function, i.e., $\mathsf{s}_{\mathsf{strong}}(v) = \frac{\mathsf{s}_{\mathsf{weak}}(v)}{\Pr(\mathbf{r}^e)}$, where $\mathbf{r}^e$ is the encoding rule set of the vault v, $\Pr(\mathbf{r}^e)$ is the probability of $\mathbf{r}^e$ in a password probabilistic model in the NLE, e.g., a PCFG-based NLE $\Pr_{\mathsf{PCFG}}(\mathbf{r}^e)$.

4.2. The process of Chatterjee-DTE

In this section, we detail the encoding process of Chatterjee-DTE [4], i.e., how to convert a rule into a code. *We do so because our attack exploits the encoding process of Chatterjee-DTE [4].* Chatterjee-DTE [4] is specific for encoding fractions of the form $\frac{p}{q}$, where $p, q \in \mathbb{Z}$ and $p \leq q$. The fraction $\frac{p}{q}$ is the cumulative frequency of a rule. For example, there are three rules: r_1, r_2, r_3, with frequencies 2, 3, and 5 respectively. Then, their corresponding fractional frequencies are $\frac{2}{10}$, $\frac{3}{10}$, $\frac{5}{10}$. In the cumulative distribution, we let the fraction interval $[\frac{0}{10}, \frac{2}{10})$ represent rule r_1, $[\frac{2}{10}, \frac{5}{10})$ represent r_2, and $[\frac{5}{10}, \frac{10}{10})$ represent r_3. When encoding r_2, we *uniformly randomly* pick a fraction of form $\frac{p}{10}, (p \in \mathbb{N})$ from $[\frac{2}{10}, \frac{5}{10})$, say $\frac{4}{10}$. This case means rule r_2 has $p = 4$ and $q = 10$, and its cumulative-frequency is $\frac{4}{10}$. *That is, for a given rule r, q is constant and public while p is a random variable on the corresponding interval.*

Next, we let b be the code length and l, h be two fixed values that satisfy $h = 2^b \mod q$ and $lq + h = 2^b$. Random variable X is on the uniform distribution $U(0, 2^b)$, random variable Y is induced by X, i.e., $Y = X - (X \mod q)$. Finally, we get the code s of the rule r, i.e., $s = Y + p$. The decoding is the opposite, i.e., $p = s \mod q$ (Note that the constant q is public). Notice that the code s contains two random variables, i.e., p (picked uniformly from the interval corresponding to the rule r) and Y. Accordingly, the conditional code probability of the rule r is

$$\Pr_{\mathsf{encode}}(s = kq + p | r) = \begin{cases} \frac{q}{2^b |\Delta_r|}, & 0 \leq k < l \\ \frac{h}{2^b |\Delta_r|}, & k = l, \end{cases} \tag{15}$$

where Δ_r is the set of all fractions of the form $\frac{p}{q}$ in the interval corresponding to the rule r. For the same example just above, the value $|\Delta_{r_2}|$ for rule r_2 is 3, i.e., $\frac{2}{10}$, $\frac{3}{10}$, $\frac{4}{10}$.

4.3. Our proposed super encoding attack

We propose *super encoding attack* that is aware of the non-uniformity of the code distribution of Chatterjee-DTE output [4]. More specifically, the super encoding attack is able to exploit the difference between the uniformly random seeds and the code distribution of Chatterjee-DTE output [4]. Our attack idea is that the code of Chatterjee-DTE [4] output is non-uniformly distributed, which reveals that lower probability codes are less likely to be outputted, while a uniformly random seed has the same probability. For a given vault, if the code of the vault has a lower conditional probability in Chatterjee-DTE output [4], attackers can utilize the strong signal to infer that the vault is more likely to be a decoy. Hence, we can estimate the conditional probability of the code $\mathbf{s}$ given the rule set $\mathbf{r}$ of a vault v using Eq. 15, which is similar to the idea of maximum likelihood estimation. That is, the higher the conditional probability of the code under the vault v, the more real the code and the corresponding vault are. Formally, for a vault $v = (pw_1, ..., pw_n)$, let $\mathbf{r}_{pw_i} = \{r_{i1}, .., r_{ik_i}\}$ be the rule set of the i[th] password $pw_i \in v$, where r_{ij} be the j[th] rule belonging to pw_i, and $k_i = |\mathbf{r}_{pw_i}|$ be the number of rules belonging to pw_i. Let $\mathbf{s}_v = (\mathbf{s}_{pw_1}, ..., \mathbf{s}_{pw_n})$ be the vault code, where $\mathbf{s}_{pw_i} = (s_{i1}, .., s_{ik_i})$ is the code of pw_i, and s_{ij} is the code of the rule r_{ij}. Based on the above attack idea, our score function for the super encoding attack is

$$\mathsf{s}_{\mathsf{super}}(v) = \prod_i \prod_j \Pr(s_{ij} | r_{ij}) = \frac{1}{\prod_i \prod_j \Pr_{\mathsf{encode}}(s_{ij} | r_{ij}) \cdot |\Delta_{r_{ij}}|}, \tag{16}$$

where the index i denote the i[th] password in the vault v, and the index j denote the j[th] rule belong to the i[th] password. The term $\Pr(s_{ij} | r_{ij})$ denotes the conditional probability of a code s_{ij} for a given rule r_{ij} (see Eq. 15 for the explicit expressions). $\Delta_{r_{ij}}$ is the set of all fractions corresponding to the rule r_{ij}. $\mathsf{s}_{\mathsf{super}}(v)$ is constructed similarly to $\mathsf{s}_{\mathsf{strong}}(v)$, but considers the outputted distribution of Chatterjee-DTE [4], therefore $\mathsf{s}_{\mathsf{super}}(v)$ can estimate the conditional probability of $\mathbf{s}_v$ under a vault v more accurately.

4.4. Our proposed theoretically-grounded attacks

The honey vault system is based on honey encryption (HE). Thus we can use the MR adversary's optimal attack strategy (Thm. 1) to provide a theoretical basis for the design of HV attackers. More specifically, we can instantiate Thm. 1 as score functions, i.e., the vault with higher conditional probability is more real given the vault ciphertext. In this way, we relate Thm. 1 to HV attackers.

NoCrack's NLE [4] and Golla-NLE [12] are based on different password models (i.e., PCFG model and Markov model), and they are different in modeling the vault distribution. For example, NoCrack's NLE [4] assumes that passwords in a vault are independent of each other, while Golla-NLE [12] assumes that there are correlations among them. Since the vault distribution directly affects the attack effectiveness of Thm. 1, it requires us to implement different instantiation strategies for the two existing NLEs.

Our attacks against NoCrack's NLE. In NoCrack's NLE [4], each password is independently encoded and then concatenated together. We focus on modeling the individual passwords in the vault, and estimating the master password distribution; Thus, the score of a single password pw is (rephrase Thm. 1 in honey vault systems)

$$k \cdot \frac{\Pr_{\mathrm{PW}}(pw)}{\Pr_d(pw)} \sum_{\mathbf{s}_{pw} \in \mathcal{S}_{pw}} \sum_{mp \in \mathcal{W}} \Pr_{\mathrm{MP}}(mp) \Pr(\mathsf{enc}(mp, \mathbf{s}_{pw}) = c), \tag{17}$$

where $k = \frac{1}{|\mathcal{S}| \Pr_{\mathcal{C}}(c)}$ is a constant we can neglect. $\mathcal{S}$ is the code (seed) space, $\mathcal{W}$ is the password space, c is the ciphertext of pw and $\mathcal{S}_{pw}$ is a code set contains all codes of pw. $\Pr_{\mathrm{MP}}(\cdot)$ is the master password distribution, $\Pr_{\mathrm{PW}}(\cdot)$ is the password distribution, while $\Pr_d(\cdot)$ is the password distribution induced by the targeted NLE. In our practical application, we consider enc as an encryption algorithm, and the salt and the initial vector used for encryption are public. To efficiently estimate the score, we make the following considerations.

A password pw may have the same ciphertext under different master passwords and different codes (i.e., $\mathsf{enc}(\mathbf{s}^*_{pw}, mp^*) = \mathsf{enc}(\mathbf{s}^-_{pw}{}^*, \bar{mp}^*)$, and $\mathbf{s}^*_{pw}, \mathbf{s}^-_{pw}{}^* \in \mathcal{S}_{pw}$). However, this case happens only with very low probability, which is also argued by Cheng et al. [5]. To avoid extremely

low computational performance, we ignore this negligible probability and assume that for a pw and a ciphertext c, there exists only one master password mp^* and only one code $\mathbf{s}^*_{pw}$ guaranteeing $\mathsf{enc}(\mathbf{s}^*_{pw}, mp^*)$=$c$.

Because the salt and the initial vector are public, enc becomes a deterministic algorithm. It also guarantees that for any master password mp, any ciphertext c and any password code $\mathbf{s}_{pw}$, $\Pr(\mathsf{enc}(\mathbf{s}_{pw}, mp) = c)$ has only two possible values, i.e, 0 and 1. Therefore, we can further simplify Eq. 17 as $\frac{\Pr_{\mathrm{PW}}(pw)\Pr_{\mathrm{MP}}(mp)}{\Pr_d(pw)}$. We cannot directly estimate the master password distribution because we lack a priori data set of master passwords. To efficiently estimate $\Pr_{\mathrm{MP}}(mp)$, we use the following strategy to approximate. We assume the passwords in a vault and the master password are picked by the same user, so the two password distributions are approximately equal, i.e., $\Pr_{\mathrm{PW}}(\cdot)\approx\Pr_{\mathrm{MP}}(\cdot)$.

We make this assumption mainly for two reasons. Firstly, if the passwords in the user's vault are generated by a password manager as random passwords, as described in [4], we can directly design a basic natural language encoder (NLE), which is called UNIF. More specifically, when encoding a password, UNIF encodes each character c_i in this password into a random number x_i satisfying x_i=$\bar{c_i}$ mod 96, where $\bar{c_i}$ is the rank of the character c_i under a canonical (e.g., alphabetical) order. Decoding is the opposite. In this case, when the HV attacker trial-decrypts with wrong master passwords, UNIF yields randomly looking decoy passwords that are indistinguishable from real random passwords. Therefore, UNIF can achieve almost ideal security. However, the above simple UNIF cannot handle the complicated case of user-chosen passwords. Therefore, as described in [4], [5], [6], [12], honey vault systems (such as NoCrack [4] and Golla-NLE [12]) are primarily designed to address complicated yet realistic user-chosen passwords (i.e., passwords in the vault and the master password used for encrypting the vault are both chosen by the user).

Secondly, although password managers can generate random passwords, recent studies [23], [39] have shown that the adoption rate of the auto-generation feature of password managers by users is low, and users tend to prefer using their own chosen passwords. Additionally, a recent user survey [31] targeting 1,012 American password manager users revealed that only 27% of users utilize random password generators of password managers to generate passwords. Overall, considering the significant number of ordinary users lacking sufficient security knowledge and the limited adoption rate of random passwords, our assumption is reasonable.

According to the above assumption, we use passwords in the vault as prior samples to approximate the probability of mp. Using the geometric average of the probabilities of the passwords, we can estimate the probability of mp. That is, for a given master password mp and a vault v=$(pw_1, ..., pw_n)$, we have $\Pr_{\mathrm{MP}}(mp)$=$\sqrt[n]{\prod_{pw\in v}\Pr_{\mathrm{PW}}(pw)}$. The score function is

$$\mathsf{s}(v) = \prod_{pw\in v}\frac{\Pr_{\mathrm{PW}}(pw)}{\Pr_d(pw)}\Pr_{\mathrm{MP}}(mp) = \prod_{pw\in v}\frac{(\Pr_{\mathrm{PW}}(pw))^2}{\Pr_d(pw)}. \quad (18)$$

Our attacks against Static Golla-NLE. For Static Golla-NLE [12], our treatment is similar to that of NoCrack's NLE [4]. Since Static Golla-NLE [12] implements encoding on the whole vault, we need to consider the whole vault instead the individual password. Based on the previous assumptions, we also use the geometric average to estimate the probability of mp, i.e., $\Pr_{\mathrm{MPW}}(mp) = \sqrt[n]{\Pr_{\mathrm{V}}(v)}$, where $\Pr_{\mathrm{V}}(\cdot)$ is the vault distribution. The score function against Static Golla-NLE [12] is

$$\mathsf{s}(v) = \frac{\Pr_{\mathrm{V}}(v)}{\Pr_d(v)}\cdot\sqrt[n]{\Pr_{\mathrm{V}}(v)} = \frac{(\Pr_{\mathrm{V}}(v))^{\frac{n+1}{n}}}{\Pr_d(v)}. \quad (19)$$

Our attacks against Adaptive Golla-NLE. Adaptive Golla-NLE [12] is the Static Golla-NLE with an adaptive mechanism [12]. The adaptive mechanism essentially distorts the 4-gram (i.e., (`abc, d`)) distribution in the Markov model to confuse semantics-aware attackers. Golla et al. argued that their adaptive mechanism improves the security of static NLE [12]. Its process is as follows: (1) For each password in the vault, we randomly choose a 4-gram from that password and multiply the frequency by 5 (boosted). (2) For the unboosted 4-grams, we multiply the frequency by 5 with a probability of 0.2. (3) Normalize the probability. After that, we encode the password with the modified Markov dictionary and note that this dictionary is public. Thus Adaptive Golla-NLE [12] contains a distorted 4-gram distribution compared to Static Golla-NLE [12]. For security purposes, Golla et al. propose a stronger attacker, i.e., the attacker is aware of boosted 4-gram sets [12]. In this work, we have the same setting as Golla et al [12]. Our treatment is similar to that of Static Golla-NLE [12], in addition to a *pre-processing*: we first use the boosted 4-grams to rule out some simple decoys. i.e., if there exists at least one password in the vault whose set of 4-grams does not intersect with the boosted 4-gram set. After pre-processing, we again use the score function, i.e., Eq. 19.

Instantiate the above attacks. NoCrack's NLE [4] treats the passwords in a vault independently, while Golla-NLE [12] considers password reuse, i.e., old passwords influence new passwords. We adopt different strategies in estimating $\Pr_{\mathrm{PW}}(\cdot)$ for NoCrack [4] and $\Pr_{\mathrm{V}}(\cdot)$ for Golla-NLE [12].

(1) Modeling $\Pr_{\mathrm{PW}}(\cdot)$ in NoCrack's NLE [4], using conventional password models are a direct method. Password models such as PCFG model [38], Markov model [19], and List model [33] are widely used in password security [5], [6], [33], [36]. However, all current password models seem flawed, which leads to the misestimation of password probabilities. For example, Markov overestimates password fragments with some basic semantics, e.g., `110120130`, because the model only considers local information among character relations and cannot be extended to the global information [33]. An inaccurate estimation of the score can significantly weaken the attack success rate. Therefore, we design the hybrid models of PCFG [38], Markov [19], and List [33] (e.g., $\frac{1}{3}$List + $\frac{1}{3}$PCFG +$\frac{1}{3}$Markov), with a total number of 7 ($\binom{3}{1}+\binom{3}{1}+1$). As seen in Fig. 8(d) and 8(e), the List-based attack (i.e., using List model [33]) always has the best attack success rate among all the 7 attacks.

(2) For $\Pr_{\mathrm{V}}(\cdot)$ in Golla-NLE [12]. Golla-NLE [12] is designed with the password reuse model, i.e., the encoding

of new passwords depends on the old passwords. Password reuse models have been intensively studied in recent years, and some well-known password reuse models, such as TarGuess-II [35] and Pass2path [25], have been widely used. Therefore, a simple and direct method is to instantiate $\Pr_{\mathrm{V}}(\cdot)$ by using password reuse models. As shown in Figs. 8(f), 8(g), 8(h), and 8(i), the attack success rate of TarGuess-II-based method is better than Pass2path [25], so we mainly use TarGuess-II [35] to instantiate $\Pr_{\mathrm{V}}(\cdot)$.

5. Evaluation experiments

vspace-0.5em We now evaluate the security of the natural language encoder (NLE) in existing honey vault systems by practical attacks via real-world datasets, and examine the attack efficiencies of our proposed four attacks in Sec. 4.

5.1. Evaluation setup

Evaluation process. Our evaluation includes two attack types, i.e., heuristic and theoretical. The idea of heuristic attacks is to exploit vulnerabilities in the system implementation, while theoretical attacks are based on the optimal strategy of MR adversaries. Heuristic attacks can be blocked by a carefully-designed NLE, while any NLE based on a password probability model (PPM) certainly suffers from theoretical attacks. Given a honey vault system and an attack A and A's score function $\mathsf{s}_{\mathsf{A}}(\cdot)$, the setup follows the attack process in Sec. 2.3, which contains three steps.
1) For each vault with size ≥ 2 in `Pastebin`, we randomly sample 999 decoy vaults using the targeted NLE and then shuffle the 999 decoys and the real vault as one *test list*.
2) Use the public dataset to train the attack A. For each sweet vault list, use $\mathsf{s}_{\mathsf{A}}(\cdot)$ to score all $1,000$ vaults and rank the vaults according to the scores in descending order.
3) For each *test list*, check the ranking of the real vault, then calculate the evaluation metrics in Sec. 2.5.

Here are the heuristic and theoretical attacks involved in the comparison. The theoretical attacks (the last four) are all proposed by Cheng et al. [5] and based on the same general score function, i.e., $\frac{\Pr_{\mathrm{V}}(v)}{\Pr_{d}(v)}$. The difference between these attacks is the heuristic methods instantiating $\Pr_{\mathrm{V}}(v)$.
Strong/weak encoding attack [6]. Cheng et al. [6] proposed the two attacks, which are shown in Sec. 4.1.
KL divergence attack [12]. This attack was proposed by Golla et al. [12]; it uses KL divergence of the frequencies of the passwords in the vault and the decoy passwords (sampled from the targeted NLE) as the score of the vault.
Single password attack [5]. This attack assumes passwords in one vault are independent of each other, and estimates the password probability using a heuristic statistical model.
Password similarity attack [5]. This method assumes that the passwords in the vault are correlated and models the vault distribution using some heuristic password features.
Adaptive extra attack [5]. This method mainly targets Adaptive Golla-NLE [12], exploits the boosted 4-grams in the adaptive mechanism [12] and uses heuristic binomial distribution to model the leaked information. Note that this method uses the unrealistic assumption that the attacker knows the passwords corresponding to each boosted 4-gram. However, we still incorporate this method in the comparison.
(Adaptive) Hybrid attack [5]. This method is a hybrid of the above three attacks, and its score function is the product of the score functions of the above three methods.

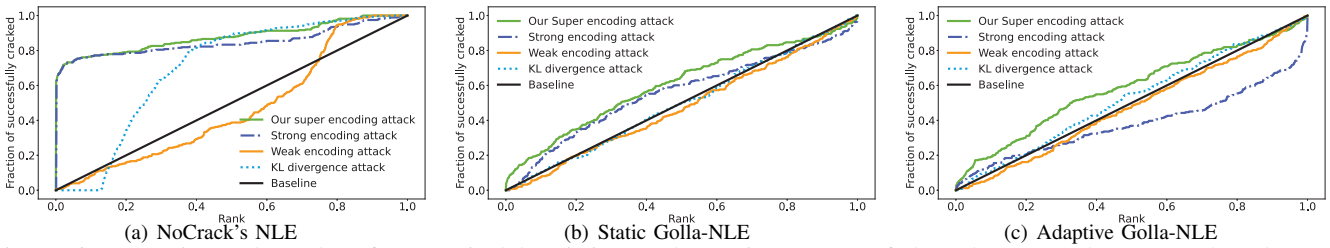
5.2. Results on heuristic attacks

The performance of our attacks. As shown in Fig. 6, our proposed *super encoding attack* performs better against all heuristic attacks. For all the static NLEs, the super encoding attack achieves 87.75% accuracy α against NoCrack's NLE [4] and 58.62%-60.50% against Golla-NLE [12]. The average rank $\bar{r}$ for NoCrack's NLE [4] is 12.25%, and those for Golla-NLE [12] are 39.50%-41.38%.

Overall, our super encoding attack demonstrates significantly higher attack success rates compared to other counterpart attacks. Taking NoCrack's NLE [4] as an example, our super encoding attack achieves an average rank $\bar{r}$ at 12.25%, compared to 15.18%-53.33% for counterpart attacks, making it 1.24 (=15.18/12.25) to 4.35 (=53.33/12.25) times more efficient than existing attacks. This translates to requiring 21.37% to 76.13% fewer online verification, greatly enhancing the attack success rate. Additionally, our attack improves the attack success rate by 1.15-1.32 times for static Golla-NLE [12], and 1.15-1.49 times for adaptive Golla-NLE [12] compared with its foremost counterparts.

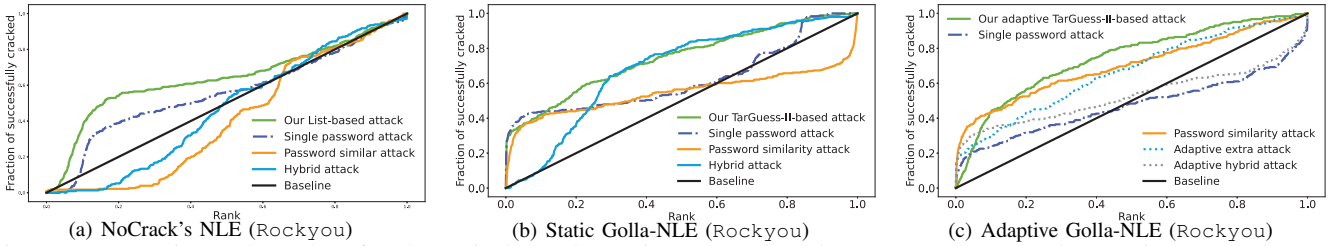
For strong encoding attack [6], its α and $\bar{r}$ are 84.82% and 15.18% for NoCrack's NLE [4]; 38.28%-55.74%, 44.26%-61.72% for Golla-NLE [12], which are less effective than our super encoding attack. This reveals that considering the distribution of the output code can improve the accuracy of modeling the conditional probability of the code.

For KL divergence attack [12], it has significant attack success rate 69.84% for α and 30.16% for $\bar{r}$ against NoCrack's NLE [4]. Dell'Amico and Filippone [10] pointed out that the distribution of samples from the PCFG model tends to head aggregation, which is significantly different from human password distribution. If a weak password appears in a vault, the vault is more likely to be considered as sampled by the PCFG model (i.e., decoys). When facing Golla-NLE [12], passwords sampled from the Markov model exhibit a more uniform distribution compared to those from the PCFG model. As a result, each decrypted vault contains passwords of varying strengths in a nearly uniform manner. This uniformity hinders the ability of KL divergence to distinguish between the real vault and the decoys, leading to a loss of the attack success rate.
The performance of honey vault systems. The Fig. 6 shows that NoCrack's NLE [4] is vulnerable to encoding attacks. NoCrack's NLE [4] uses grammatical rules with ambiguity and a deterministic rule set selection, which results in multiple encoding rule sets for one password. These features are easily captured by encoding attacks. While for Golla-NLE [12], the encoding rule set is unique, and the only information that encoding attacks can exploit is the conditional probability of the code. Hence, Golla-NLE [12] can resist weak encoding attack [6]. In summary, the attack



(a) NoCrack's NLE (b) Static Golla-NLE (c) Adaptive Golla-NLE

Figure 6: Experimental results of 4 practical heuristic attacks against NLEs of three honey vault systems based on a real-world password vault dataset Pastebin, where the *baseline* curve represents the random guess. Strong/weak encoding attack is proposed in [6], and KL divergence attack is proposed in [12].



(a) NoCrack's NLE (Rockyou) (b) Static Golla-NLE (Rockyou) (c) Adaptive Golla-NLE (Rockyou)

Figure 7: Experimental results of 3 theoretical attacks against NLEs based on Rockyou. Attacks on Fig. 7(a), 7(b) and 7(c) are trained on Rockyou. In Fig.7(a), our new attack is *List-based attack*, and in Fig. 7(b), 7(c), our new attack is *TarGuess-II-based attack*. For adaptive Golla-NLE [12], our *adaptive TarGuess-II-based attack*, adaptive extra attack [5] and adaptive hybrid attack [5] consider the boosted 4-grams.

success rate of encoding attacks against Golla-NLE [12] is reduced by 3.22-3.38 times compared with NoCrack [4].

5.3. Results on theoretically-grounded attacks

The performance of our attacks. As shown in Fig. 7, all of our proposed new attacks, i.e., *List-based attack*, *TarGuess-II-based attack*, and *adaptive TarGuess-II-based attack* all (seen these **green** curve) perform better against all their counterparts. The experimental results on the Wishbone dataset are shown in Fig. 8(a), 8(b) and 8(c). Taking the Rockyou dataset as an example, for NoCrack's NLE [4], our List-based attack achieves 63.76% for accuracy α, and 36.24% for average rank $\bar{r}$. For Golla-NLE [12], our TarGuess-II-based attack has 74.22%/74.34% against static/adaptive Golla-NLE [12]. As the $\bar{r}$ for static Golla-NLE [12] is 25.78% and that for adaptive Golla-NLE [12] is 25.66%. In summary, our List-based attack can improve the attack success rate by 1.23-1.62 times compared to its foremost counterparts; our TarGuess-II-based attack and adaptive TarGuess-II-based attack can improve the attack success rate by 1.23-1.75 times and 1.30-2.05 times.

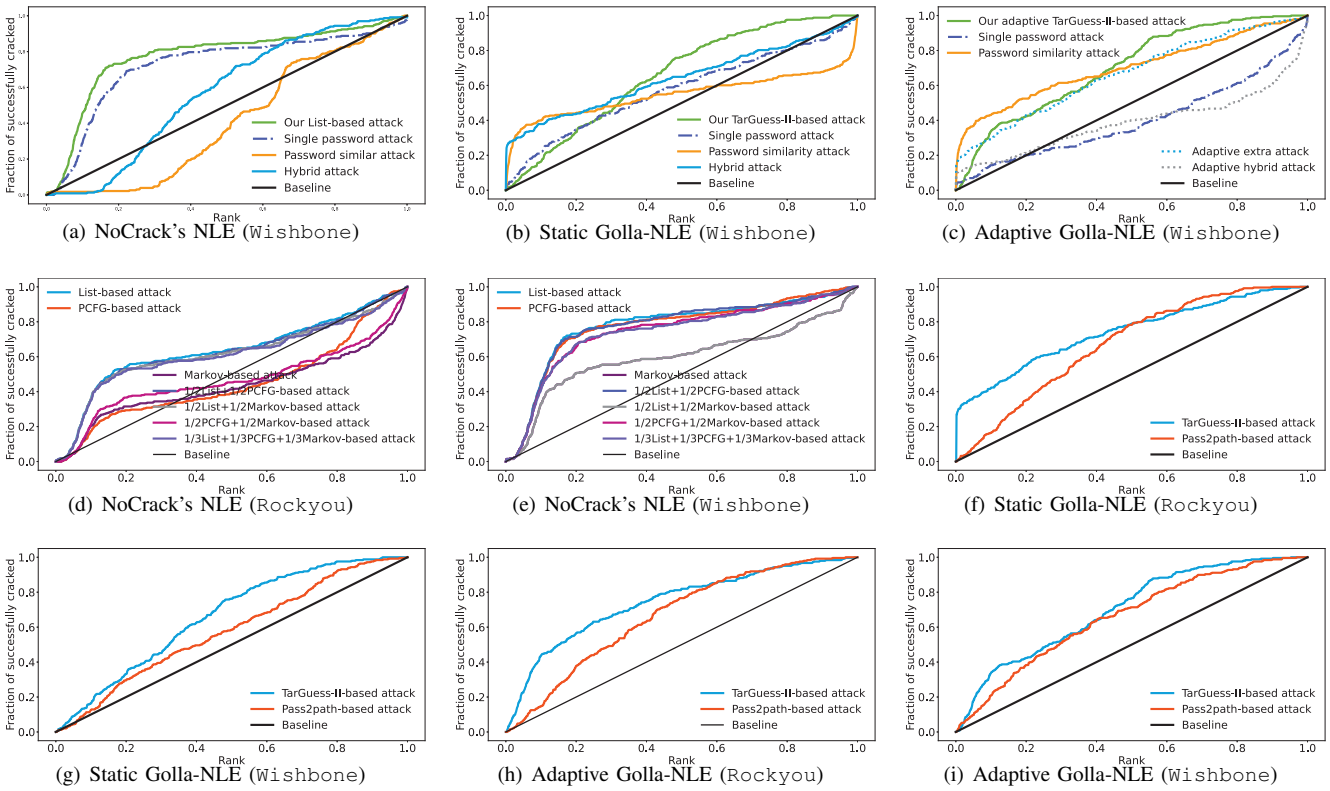
For single password attack [5], its α and $\bar{r}$ are 72.41% and 27.59% against NoCrack's NLE [4], and 47.40%-62.48%, 37.52%-57.60% against Golla-NLE [12], which are less effective than our attacks. One possible reason is the used single password model [5] does not match the real password distribution, and loses accuracy facing passwords with a low frequency and a low decoy probability (i.e., $\mathrm{Pr}_d(pw)$), and these passwords account for a large proportion (69.25%).

For password similarity attack [5], its α and $\bar{r}$ are 40.98% and 59.02% against NoCrack's NLE [4] and 40.98%-70.24%, 29.76%-59.02% against Golla-NLE [12], which are less effective than our TarGuess-II-based attack. Note that it performs slightly better against Golla-NLE [12] than NoCrack's NLE [4], because the used password similarity model [5] is more closely to the Golla-NLE [12], i.e., it considers the password reuse. But the password similarity model estimates the password vault with a coarse-grained method, considering some simple local statistics features, which significantly weakens the efficiency. For the (adaptive) hybrid attack [5], it mixes the above two attacks, i.e., it mixes the single password model [5] and password similarity model [5]. But according to our above analysis, the (adaptive) hybrid attack [5] still has shortages in modeling password features because it contains two inaccurate models.

5.4. Potential threats

Our feature attack against IU mechanism. To resist intersection attacks, Cheng et al. [5] propose the Incrementally Updateable mechanism (IU mechanism) adopting a prefix-preserving symmetric encryption scheme PPSE. IU mechanism [5] encodes the old vault v^o and the new vault v^n into the code $\mathbf{s}_{v^o}$ and $\mathbf{s}_{v^n}$, ensuring $\mathbf{s}_{v^o}$ is a prefix of $\mathbf{s}_{v^n}$, and then applies PPSE to encrypt. However, IU mechanism [5] still has security vulnerabilities and we propose a feature attack that can completely break IU mechanism [5] with a probability close to 90%. Our feature attack is as follows.

For a decrypted new and old vaults code pair $(\mathbf{s}_v^n, \mathbf{s}_v^o)$, the decoded vault pair is (v^n, v^o). We first identify the same bit string prefix of $(\mathbf{s}_v^n, \mathbf{s}_v^o)$, and denote the bit string prefix as bsp. Our attack idea is that a bsp from a real vault code pair should be decoded into some complete passwords exactly. Hence, if bsp can be exactly decoded into some passwords, i.e., all the bits in bsp are utilized. In this case, we determine the score of the vault pair (v^n, v^o) as 1. Conversely, if some remaining bits at the end of the bsp cannot be decoded to a



(a) NoCrack's NLE (Wishbone) (b) Static Golla-NLE (Wishbone) (c) Adaptive Golla-NLE (Wishbone)

(d) NoCrack's NLE (Rockyou) (e) NoCrack's NLE (Wishbone) (f) Static Golla-NLE (Rockyou)

(g) Static Golla-NLE (Wishbone) (h) Adaptive Golla-NLE (Rockyou) (i) Adaptive Golla-NLE (Wishbone)

Figure 8: Figs. 8(a), 8(b) and 8(c) show the results of three theoretical attacks against three NLEs based on Wishbone. For adaptive Golla-NLE [12], the *adaptive TarGuess-II-based attack*, adaptive extra attack [5] and adaptive hybrid attack [5] consider the boosted 4-grams. *For Golla-NLE [12], notice that in Fig. 7(b) and 8(b), the success rate curves are the same for password similarity attack [5], and it is similar for adaptive extra attack [5] in Fig. 7(c) and 8(c). This is because these two attacks do not require the additional password datasets* Rockyou *and* Wishbone *for training.* Figs.8(d) and 8(e) are our attacks based on 7 (mix) password probability models (PPM) against NoCrack's NLE on two real-world datasets. Attacks on Figs. 8(d) and 8(e) are trained on Rockyou and Wishbone, respectively. *List-based attack* performs better than its counterparts. Figs.8(f), 8(g), 8(h), and 8(i) are our new attack based on two PPMs against Golla-NLE. Attacks on Fig. 8(f) and 8(h) are trained on Rockyou and Fig. 8(g) and 8(i) are trained on Wishbone. Our *TarGuess-II-based attack* and *adaptive TarGuess-II-based attack* perform better than their counterparts.

complete password (i.e., can only be decoded into password fragments), the score is 0.

Experimental results. We combine the IU-mechanism with existing NLEs to obtain three new NLEs. We pick vaults in Pastebin (with size $\geq$2) as the testset. We use the last password as a newly added one. In this way, we can get the new and old versions of each vault. Then, we use IU-mechanism to generate decoy new/old vaults for the real new/old vault simultaneously. The rest of the experiment setup is the same as in Sec. 5.1. As shown in Fig. 9, for the three NLEs with Cheng et al.'s IU-mechanism [5], our feature attack can achieve 87.31%-93.22% for accuracy α. To conclude, existing NLEs with IU-mechanism [5] are all vulnerable to our proposed feature attack.

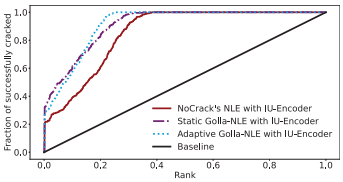


Figure 9: Experimental results of our feature attack.

5.5. Security discussion

How to design a secure honey vault system. Our security analysis highlights the following countermeasures.
(1) Based on our optimal attack strategy (Thm. 1) and the instantiated attacks (Eqs. 18&19) in Sec. 4.4, To make the honey vault system resist our instantiated attacks (Eqs. 18&19), the password probability model used by the NLE (which is the core component of the honey vault system) should accurately model the user's real password distributions, i.e., ensuring $\Pr_d(\cdot) \approx \Pr_V(\cdot)$ to resist attacks exploiting differences between real and decoy vaults (e.g., our instantiated attacks). This is because when the decoy vault distribution $\Pr_d(\cdot)$ modeled by the NLE satisfies $\Pr_d(\cdot) \approx \Pr_V(\cdot)$, our instantiated attacks (Eqs. 18&19) degrade into random guessing according to vault probabilities.
(2) Based on our analysis of the workflow of existing NLEs [4], [12] in Secs. 4.1-4.2, to enable the honey vault system

to resist our super encoding attack, the password probability model (PPM) in the NLE should not comprise ambiguous rules. This is because if the password probability model used by the NLE does not include ambiguous rules, the super encoding attack will be unable to exploit multiple different rules from the same vault, effectively degrading into random guessing. This countermeasure can also apply to Cheng et al.'s encoding attack [6].

Security implications of high-entropy vault passwords. If all passwords in the vault v are high-entropy passwords generated by password managers, since these high-entropy passwords are typically independent of each other, the probability of this vault can be expressed as $\Pr_{\mathrm{V}}(v)=\prod_{pw\in v}\Pr_{\mathrm{PW}}(pw)$. Compare to user-chosen passwords, due to the presence of high-entropy passwords in v, this leads to a lower vault probability $\Pr_{\mathrm{V}}(v)$. According to our theoretically-grounded attacks' score functions (see Eqs. 18&19), the lower vault probability further results in a corresponding lower score of vault v, causing v to rank higher among all n plausible vaults. Consequently, HV attackers need to expend more online verification attempts to successfully guess this vault. Thus, the presence of high-entropy passwords in the vault significantly reduces the attack success rate of HV attackers.

The impacts of authentication system's security mechanisms on the attack effectiveness. The impact of security mechanisms deployed in real-world authentication systems on our proposed practical attacks is marginal. This is primarily attributed to two reasons. First, various techniques are available to circumvent security mechanisms deployed by websites. For instance, when faced with the k-strike lockout mechanism, the attacker can stop attempts after trying k-1 online verifications in the lockout mechanism's permitted time phase. The attacker then waits for the lockout mechanism to reset in the next time phase (e.g., typically 24 hours later or after the legitimate user's login), and continues attempting k-1 times, and so forth. Additionally, widely deployed CAPTCHAs can also be bypassed [14]. Second, even when online security mechanisms are effective, our results demonstrate that distinguishing attacks remain highly effective: Based on our experimental results (Figs. 6&7), within just 10 online verifications, the real vault can be distinguished with probabilities of 41.83% for NoCrack's NLE, 27.35% for Golla-NLE, and 25.21% for adaptive Golla-NLE, respectively.

Experimental results of higher n. A higher n will have little effect on our results. We have conducted experiments with higher n, and the results were consistent with our findings. For instance, in our representative List-based attack against NoCrack's NLE [4] using the `Rockyou` dataset, we set n to 10,000, 100,000, 1,000,000, and 10,000,000. The average rank $\bar{r}$ of real vaults are approximately 36.02%, 35.77%, 35.93%, and 36.74%, respectively. These results are consistent with $n = 1,000$ in this work where $\bar{r}$ is 36.24%.

6. Conclusion

We have provided significantly tighter upper/lower bounds for HE-related security games, and propose four attacks against honey vault systems. Extensive experiments show that our four attacks drastically outperform their counterparts. Particularly, we propose a feature attack against the incrementally updateable mechanism at USENIX SEC'21, which makes intersection attacks a damaging threat against most of the existing honey vault systems. We believe our work constitutes an important step forward in this direction and will trigger interest in new honey technique research. Considering the prevalence and catastrophic impacts of offline password guessing attacks, we believe our work constitutes an important step forward in this direction.

Acknowledgement

The authors are grateful to the shepherd and anonymous reviewers for their invaluable comments. Ding Wang is the corresponding author. We thank Zhenduo Hou for insightful discussions. This research was in part supported by the National Natural Science Foundation of China under Grants Nos. 62222208 and 62172240, and Natural Science Foundation of Tianjin, China under Grant No. 21JCZDJC00190.

References

[1] C. Allan, *32 million Rockyou passwords stolen*, Dec. 2009, http://www.hardwareheaven.com/news.php?newsid=526.

[2] D. J. Bernstein, "How to stretch random functions: The security of protected counter sums," *J. Cryptol.*, vol. 12, no. 3, p. 185–192, 1999.

[3] C. Catalin, *Hacker leaks 40 million user records from popular Wishbone app*, May. 2020, https://www.zdnet.com/article/hacker-selling-40-million -user-records-from-popular-wishbone-app/.

[4] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, "Cracking-resistant password vaults using natural language encoders," in *Proc. IEEE S&P 2015*, pp. 481–498.

[5] H. Cheng, W. Li, P. Wang, C.-H. Chu, and K. Liang, "Incrementally updateable honey password vaults," in *Proc. USENIX SEC 2021*, pp. 857–874.

[6] H. Cheng, Z. Zheng, W. Li, P. Wang, and C. Chu, "Probability model transforming encoders against encoding attacks," in *Proc. USENIX SEC 2019*, pp. 1573–1590.

[7] C. Clifford and P. Sharon, *Keep Your Passwords Strong and Secure With These 9 Rules*, May. 2022, https://www.cnet.com/tech/mobile/keep-your-passwords-strong-and-secure-with-these-9-rules/.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[9] W. Dai, V. T. Hoang, and S. Tessaro, "Information-theoretic indistinguishability via the chi-squared method," in *Proc. CRYPTO 2017*, pp. 497–523.

[10] M. Dell'Amico and M. Filippone, "Monte carlo strength evaluation: Fast and reliable password checking," in *Proc. ACM CCS 2015*, pp. 158–169.

[11] M. Dürmuth, D. Freeman, S. Jain, B. Biggio, and G. Giacinto, "Who are you? A statistical approach to measuring user authenticity," in *Proc. NDSS 2016*, pp. 1–15.

[12] M. Golla, B. Beuscher, and M. Dürmuth, "On the security of cracking-resistant password vaults," in *Proc. ACM CCS 2016*, pp. 1230–1241.

[13] P. A. Grassi, E. M. Newton, R. A. Perlner, and et al., "NIST 800-63B digital identity guidelines: Authentication and lifecycle management," McLean, VA, Tech. Rep., June 2017.

[14] M. Guerar, L. Verderame, M. Migliardi, F. Palmieri, and A. Merlo, "Gotta captcha 'em all: A survey of 20 years of the human-or-computer dilemma," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–33, 2021.

[15] A. Hanamsagar, S. S. Woo, C. Kanich, and J. Mirkovic, "Leveraging semantic transformation to investigate password habits and their causes," in *Proc. ACM CHI 2018*, pp. 1–10.

[16] B. Hitaj, P. Gasti, G. Ateniese, and F. Pérez-Cruz, "PassGAN: A deep learning approach for password guessing," in *Proc. ACNS 2019*, ser. LNCS, vol. 11464, pp. 217–237.

[17] J. Jaeger, T. Ristenpart, and Q. Tang, "Honey encryption beyond message recovery security," in *Proc. EUROCRYPT 2016*, pp. 758–788.

[18] A. Juels and T. Ristenpart, "Honey encryption: Security beyond the brute-force bound," in *Proc. EUROCRYPT 2014*, pp. 293–310.

[19] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE S&P 2014*, pp. 689–704.

[20] W. Melicher, B. Ur, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean and accurate: Modeling password guessability using neural networks," in *Proc. USENIX SEC 2017*, pp. 1–17.

[21] M. Nandi, "A simple and unified method of proving indistinguishability," in *Proc. INDOCRYPT 2006*, ser. LNCS, vol. 4329, 2006, pp. 317–334.

[22] S. Oesch and S. Ruoti, "That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers," in *Proc. USENIX SEC 2020*, pp. 2165–2182.

[23] S. Oesch, S. Ruoti, J. Simmons, and A. Gautam, ""it basically started using me:" an observational study of password manager usage," in *Proc. ACM CHI 2022*, pp. 1–23.

[24] B. Oprisanu, C. Dessimoz, and E. De Cristofaro, "How much does genoguard really "guard"? an empirical analysis of long-term security for genomic data," in *Proc. WEPS 2019*, pp. 93–105.

[25] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE S&P 2019*, pp. 417–434.

[26] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving password guessing via representation learning," in *Proc. IEEE S&P 2021*, pp. 1382–1399.

[27] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's go in for a closer look: Observing passwords in their natural habitat," in *Proc. ACM CCS 2017*, pp. 295–310.

[28] A. Peslyak, *John the Ripper password cracker*, Feb. 1996, http://www.openwall.com/john/.

[29] B. Pinkas and T. Sander, "Securing passwords against dictionary attacks," in *Proc. CCS 2002*, pp. 161–170.

[30] J. Steube, *Hashcat*, 2018, https://hashcat.net/hashcat/.

[31] S. Team, *AMERICA'S Password Habits 2021*, Aug. 2023, https://www.security.org/resources/online-password-strategies/.

[32] R. Veras, C. Collins, and J. Thorpe, "On the semantic patterns of passwords and their security impact," in *Proc. NDSS 2014*, pp. 1–16.

[33] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A security analysis of honeywords," in *Proc. NDSS 2018*, pp. 1–16.

[34] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of Chinese web users," in *Proc. USENIX SEC 2019*, pp. 1537–1555.

[35] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM CCS 2016*, pp. 1242–1254.

[36] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE S&P 2022*, pp. 966–983.

[37] D. Wang, Y. Zou, X. Yuan-An, and M. siqi, "Pass2edit: A multi-step generative model for guessing edited passwords," in *Proc. USENIX SEC 2023*, pp. 983–1000.

[38] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. IEEE S&P 2009*, pp. 391–405.

[39] S. Zibaei, D. R. Malapaya, B. Mercier, A. Salehi-Abari, and J. Thorpe, "Do password managers nudge secure (random) passwords?" in *Proc. SOUPS 2022*, pp. 581–597.

Appendix A. Omitted Proofs

A.1. Proof for Lemma 1

In Lem. 1, the first inequality is the Pinsker's inequality [8]. Here we primarily provide the proof for the second inequality: Because $\ln(x)$ is concave, we have

$$\sum_{x\in\Omega}\mu(x)\ln(\frac{\mu(x)}{\nu(x)}) \le \ln(\sum_{x\in\Omega}\frac{\mu(x)^2}{\nu(x)}). \tag{20}$$

Meanwhile

$$\begin{aligned}\chi^2(\mu,\nu) &= \sum_{x\in\Omega}\frac{(\mu(x)-\nu(x))^2}{\nu(x)}\\ &= \sum_{x\in\Omega}\frac{\mu(x)^2}{\nu(x)} - \sum_{x\in\Omega}(2\mu(x)-\nu(x)) = \sum_{x\in\Omega}\frac{\mu(x)^2}{\nu(x)} - 1.\end{aligned} \tag{21}$$

Using the inequality $x-1\ge\ln(x)$, we have

$$\chi^2(\mu,\nu) = \sum_{x\in\Omega}\frac{\mu(x)^2}{\nu(x)} - 1 \ge \ln(\sum_{x\in\Omega}\frac{\mu(x)^2}{\nu(x)}) \ge \Delta_{\mathsf{KL}}(\mu,\nu). \tag{22}$$

Here we complete our proof.

A.2. Proof for Theorem 1 and Theorem 4

We first present the Thm. 4. The upper bound of MR advantage in Thm. 4 is obtained by our optimal strategy Thm. 1.

Theorem 4. Given the same settings as Thm. 1, for any MR adversary $\mathcal{A}$, we have the MR advantage

$$\begin{aligned}&\max_{\mathcal{A}}\mathsf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) \le\\ &\sum_{c\in\mathcal{C}}\max_m\frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)|\mathcal{S}|}\sum_{s\in\mathcal{S}_m}\sum_{k\in\mathcal{K}}\Pr_{\mathcal{K}}(k)\Pr(\mathsf{enc}(k,s)=c).\end{aligned} \tag{23}$$

Since Thm. 1 and Thm. 4 are closely related, in this section, we prove them at the same time. We consider the MR game shown in Fig. 4. Given the ciphertext c, we apply the Bayesian method directly, the probability that m is a real message is

$$\begin{aligned}\Pr(m|c) &= \frac{1}{\Pr_{\mathcal{C}}(c)}\Pr(m,c)\\ &= \frac{\Pr_{\mathcal{M}}(m)}{\Pr_{\mathcal{C}}(c)}\sum_{s\in S_m}\Pr(s|m)\sum_{k\in\mathcal{K}}\Pr_{\mathcal{K}}(k)\Pr(\mathsf{enc}(k,s)=c).\end{aligned} \tag{24}$$

Most existing honey encryption schemes [5], [6], [17], [18] use IS-DTE [18] that can output uniformly random codes, so we have $\Pr(s|m) = \frac{1}{|\mathcal{S}_m|} = \frac{1}{\Pr_d(m)|\mathcal{S}|}$. We can write Eq. 24 as

$$\Pr(m|c) = \frac{1}{\Pr_{\mathcal{C}}(c)}\Pr(m,c)$$
$$= \frac{1}{\Pr_{\mathcal{C}}(c)} \sum_{s\in\mathcal{S}_m} \frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)|\mathcal{S}|} \sum_{k\in\mathcal{K}} \Pr_{\mathcal{K}}(k)\Pr(\mathsf{enc}(k,s)=c). \tag{25}$$

Therefore, for each given ciphertext c, the optimal strategy is that $\mathcal{A}$ chooses the message m with the largest conditional probability as the guess. According to this strategy, we rewrite Eq. 25 as

$$\max_m \Pr(m|c) =$$
$$\frac{1}{\Pr_{\mathcal{C}}(c)} \max_m \sum_{s\in\mathcal{S}_m} \frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)|\mathcal{S}|} \sum_{k\in\mathcal{K}} \Pr_{\mathcal{K}}(k)\Pr(\mathsf{enc}(k,s)=c). \tag{26}$$

According to Eq. 26, we finally get that for any MR adversary $\mathcal{A}$, the MR advantage is at most

$$\mathsf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) \le \sum_{c\in\mathcal{C}} \Pr_{\mathcal{C}}(c) \max_m \Pr(m|c)$$
$$= \sum_{c\in\mathcal{C}} \max_m \frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)|\mathcal{S}|} \sum_{s\in\mathcal{S}_m} \sum_{k\in\mathcal{K}} \Pr_{\mathcal{K}}(k)\Pr(\mathsf{enc}(k,s)=c). \tag{27}$$

Here we complete the proof. Note that we have omitted the probability of random bits used by the random oracle model in the above equations.

A.3. Proof for Lemma 2

According to Lem. 1, we have

$$\begin{aligned}
&\frac{1}{2}\|\Pr_{\mathbf{S}_0} - \Pr_{\mathbf{S}_1}\|^2 \le \sum_{\mathbf{Z}_q} \Pr_{\mathbf{S}_1}(\mathbf{Z}_q)\ln(\frac{\Pr_{\mathbf{S}_1}(\mathbf{Z}_q)}{\Pr_{\mathbf{S}_0}(\mathbf{Z}_q)})\\
&= \sum_{\mathbf{Z}_q}\sum_{i=1}^{q} \Pr_{\mathbf{S}_1}(\mathbf{Z}_q)\ln(\frac{\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^i)}{\Pr_{\mathbf{S}_0}(\mathbf{Z}_q^i)})\\
&= \sum_{\mathbf{Z}_q}\sum_{i=1}^{q}(\prod_{j=1}^{q}\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^j))\ln(\frac{\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^i)}{\Pr_{\mathbf{S}_0}(\mathbf{Z}_q^i)})\\
&= \sum_{i=1}^{q}\sum_{\mathbf{Z}_q}(\prod_{j=1}^{q}\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^j))\ln(\frac{\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^i)}{\Pr_{\mathbf{S}_0}(\mathbf{Z}_q^i)})\\
&= \sum_{i=1}^{q}\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^j)\ln(\frac{\Pr_{\mathbf{S}_1}(\mathbf{Z}_q^i)}{\Pr_{\mathbf{S}_0}(\mathbf{Z}_q^i)}) \le \sum_{i=1}^{q}\chi_i^2
\end{aligned} \tag{28}$$

A.4. Proofs for Theorem 2

Before our proof, we first recall the following concepts: the seed/code space $\mathcal{S}=\{0,1\}^l$, the cumulative distribution function of m is $\mathrm{CDF}_{\mathcal{M}}(m)$, and a fraction $a\in[0,1]$ is represented by value b such that $b:=\mathsf{rep}_\rho(a)$. $\mathsf{round}(\cdot)$ is the round function. For any message m_i, $\mathsf{is\text{-}encode}(m_i)$ randomly picks $s\xleftarrow{\$}(\mathsf{rep}_\rho(\mathrm{CDF}(m_{i-1})), \mathsf{rep}_\rho(\mathrm{CDF}(m_i))]$ as the code of m_i. The process of $\mathsf{is\text{-}decode}(m_i)$ is reverse.

Our proof contain the following steps. We first introduce a new concept: representation error.

Representation error. First we consider the *representation error* of IS-DTE [18]. We set code length to be l and assume that l is appropriate, i.e., $2^{-l} \ll \min_m \Pr_{\mathcal{M}}(m)$, and let $\rho:=2^{-l}$. We denote the representation error as ϵ and obtain

$$\begin{aligned}
\epsilon &= \max_{a\in\mathsf{img}(\mathrm{CDF}(\cdot))} |a - \mathsf{round}(\frac{a}{\rho})\cdot\rho|\\
&\le \max_{a\in\mathsf{img}(\mathrm{CDF}(\cdot))} |a - (\frac{a}{\rho}-\frac{1}{2})\cdot\rho| = \frac{\rho}{2},
\end{aligned} \tag{29}$$

where $\mathsf{round}(\cdot)$ is the round function. Next, using the representation error ϵ, we estimate the chi-squared divergence between real message distribution $\Pr_{\mathcal{M}}(\cdot)$ in SAMP1 and the IS-DTE [18] induced message distribution $\Pr_d(\cdot)$ in SAMP0 (as shown in Fig. 2).

Estimating chi-squared divergence. The responses of system SAMP0 and SAMP1 at one query conform the distribution $\Pr_{\mathcal{M}}(\cdot)$ and $\Pr_d(\cdot)$, thus, for any m we have

$$\begin{aligned}
\frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)} &= \frac{a_i - a_{i-1}}{b_i\rho - b_{i-1}\rho} = \frac{b_i\rho + \delta_i - b_{i-1}\rho - \delta_{i-1}}{b_i\rho - b_{i-1}\rho}\\
&= \frac{b_i\rho - b_{i-1}\rho + (\delta_i - \delta_{i-1})}{b_i\rho - b_{i-1}\rho} = 1 + \frac{(\delta_i - \delta_{i-1})}{b_i\rho - b_{i-1}\rho}.
\end{aligned} \tag{30}$$

which uses that $\Pr_{\mathcal{M}}(m) = a_i - a_{i-1}$ and $\Pr_d(m) = b_i\rho - b_{i-1}\rho$ (Recall **Instantiating DTE** in Sec. 2.2). We let $\delta_i = a_i - b_i\rho$, and have $|\delta_i| \le \epsilon$ (Absolute value inequalities, the same as δ_{i-1}) and $|\delta_i - \delta_{i-1}| \le 2\epsilon$. We obtain the relation $\left|\frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)} - 1\right| \le 2\epsilon$. Then we compute the χ_i^2 at one query to SAMP0/SAMP1 as follows

$$\begin{aligned}
\chi_i^2(p_{\mathcal{M}}, p_d) &= \sum_{m\in\mathcal{M}} \frac{(\Pr_{\mathcal{M}}(m) - \Pr_d(m))^2}{\Pr_d(m)}\\
&= \sum_{m\in\mathcal{M}} (\frac{\Pr_{\mathcal{M}}(m)}{\Pr_d(m)} - 1)^2\Pr_d(m) \le \sum_{m\in\mathcal{M}} 4\epsilon^2\Pr_d(m) = 4\epsilon^2.
\end{aligned} \tag{31}$$

Combining the chi-squared divergence and Lem. 2. Finally, According to Lem. 2 and Eq. 31, we have

$$\mathsf{Adv}^{\mathrm{dte}}_{\mathsf{IS\text{-}DTE},p_{\mathcal{M}}}(\mathcal{B}) \le \|\Pr_{\mathbf{S}_0}(\cdot) - \Pr_{\mathbf{S}_1}(\cdot)\| \le \sqrt{(\frac{1}{2}\sum_{i=1}^{q}\chi_i^2)} \le \epsilon\sqrt{2q}. \tag{32}$$

According to the low-entropy key setting, the distinguisher $\mathcal{B}$'s query number q is set to $|\mathcal{M}|$. We apply $q{=}|\mathcal{M}|$ and $\epsilon{=}2^{-l-1}$ to Eq. 32, and have the final result $\mathsf{Adv}^{\mathrm{dte}}_{\mathsf{IS\text{-}DTE},p_{\mathcal{M}}}(\mathcal{B}) \le \frac{\sqrt{|\mathcal{M}|/2}}{2^l}$, which is much tighter than the upper bound $\mathsf{Adv}^{\mathrm{dte}}_{\mathsf{IS\text{-}DTE},p_{\mathcal{M}}}(\mathcal{B}) \le \frac{|\mathcal{M}|}{2^l}$ given in [18].

A.5. Much more precise MR-KMA lower bound

We derive a global expression for MR-KMA advantage that is general for all HE schemes. Compared to Jaeger et al.'s result [17], our MR-KMA advantage is much more precise. According to our result, *we reveal that any systems based on HE schemes are vulnerable to MR-KMA adversaries, i.e., MR-KMA adversaries will gain an overwhelming advantage.* We first show our new MR-KMA result: Thm. 5.

Theorem 5. For a HE scheme in a low entropy key setting, there exists an MR-KMA adversary $\mathcal{A}$, for any distribution $p_{\mathcal{M}}$, $p_{\mathcal{K}}$ with at most $q-1 (q \ge 1)$ queries to oracle **Enc** and the advantage satisfies

$$\mathsf{Adv}^{\text{mr-kma}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A}) \ge \frac{1}{\sqrt[q]{|\mathcal{K}|}}. \tag{33}$$

Our proof. We let the user-chosen message be m^* and the challenge ciphertext be c^*. The MR-KMA adversary $\mathcal{A}$ can get at most q-1 message-ciphertext pairs by q-1 **Enc** queries. Let the q-1 message-ciphertext pairs be $\{(m_i,c_i)\}_{i=1}^{q-1}$. Taking q-1 pairs as an example, we enumerate all the low-entropy keys in $\mathcal{K}$ and use the q-1 pairs as a filter. Here we stress that since HE schemes are primarily designed for

low-entropy key settings, $\mathcal{A}$ can enumerate all low-entropy keys $k\in\mathcal{K}$. Let $E_{q\text{-}1}$ be the set of keys $k\in\mathcal{K}$ that satisfies the q-1 pairs, which means that for any $i\in[1,q\text{-}1]$, $\mathcal{A}$ can decrypt c_i into m_i correctly with any $k\in E_{q\text{-}1}$. We know that $E_{q\text{-}1}$ must contain the keys that can yield the user message m^* when decrypting c^*, Further, we denote the set of keys that can decrypt the challenge ciphertext c^* into m^* as E_q. Therefore, the average success rate is $\frac{1}{q}\sum_{i=1}^{q}\frac{|E_i|}{|E_{i-1}|}$, and the MR-KMA advantage is $\frac{1}{q}\sum_{i=1}^{q}\mathbb{E}\frac{|E_i|}{|E_{i-1}|}]\geq\mathbb{E}[\sqrt[q]{\frac{|E_q|}{|E_0|}}]$, which is greater than $\frac{1}{\sqrt[q]{|\mathcal{K}|}}$. Formally, $\mathcal{A}$ makes at most q-1 **Enc** queries, and the MR-KMA advantage satisfies $\mathsf{Adv}^{\text{mr-kma}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A})\geq\sum_{i=1}^{q}\frac{1}{q}\mathbb{E}[\frac{|E_i|}{|E_{i-1}|}]$, where $E_0\subseteq\mathcal{K}$ and $E_q\neq\emptyset$. For any q, we have $E_{q+1}\subseteq E_q$, thus we have

$$\mathsf{Adv}^{\text{mr-kma}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A})\geq\sum_{i=1}^{q}\frac{1}{q}\mathbb{E}[\frac{|E_q|}{|E_{q-1}|}]=\mathbb{E}[\frac{1}{q}\sum_{q=1}^{q}\frac{|E_q|}{|E_{q-1}|}]$$
$$\geq\mathbb{E}[\sqrt[q]{\prod_{i=1}^{q}\frac{|E_q|}{|E_{q-1}|}}]\geq\mathbb{E}[\sqrt[q]{\frac{|E_q|}{|E_0|}}]\geq\mathbb{E}[\sqrt[q]{\frac{1}{|\mathcal{K}|}}]=\sqrt[q]{\frac{1}{|\mathcal{K}|}}. \tag{34}$$

A.6. Proof for Theorem 3

In the MR-KMA game, given the message distribution $\Pr_{\mathcal{M}}(\cdot)$ and the decoy message distribution $\Pr_d(\cdot)$ modeled by the DTE, we denote $\omega_m:=\max_m\Pr_d(m)$. We let r^* be the user's salt value, m^* be the chosen message (challenge message), k^* be the chosen key, and (r^*,c^*) be the ciphertext. Meanwhile (r^*,c^*) is sent to the MR-KMA adversary $\mathcal{A}$. $\mathcal{A}$'s goal is to guess the user's chosen real message m^* based on the user ciphertext (r^*,c^*) and the q message-ciphertext pairs obtained from q queries, denoted as $Q_q=\{(m_i,(r^i,c^i))\}_{i=1}^{q}$. We use the set $\mathcal{Q}_q$ to represent all possible q message-ciphertext pairs, i.e., $Q_q\in\mathcal{Q}_q$.

Proof idea. We first fix the q message-ciphertext pairs $Q_q=\{(m_i,(r^i,c^i))\}_{i=1}^{q}$, and then we give a lower bound on $\mathcal{A}$'s success rate in the fixed Q_q (say, a constant value denoted as C). Furthermore, in $\mathcal{A}$'s view, Q_q follow a distribution on $\mathcal{Q}_q$, which could be determined by the system ($\mathcal{A}$ is non-adaptive) or jointly determined by $\mathcal{A}$ and the system ($\mathcal{A}$ is adaptive). In any case, since $\mathcal{A}$'s attack strategy is independent of $\mathcal{Q}_q$ distribution, $\mathcal{A}$'s expected success rate (MR-KMA advantage) must be greater than the previously obtained constant C.

Our Proof. First, we assume that the salt values obtained from the q **Enc** queries are different. For a given Q_q, we say if a key k is consistent with the i$^{\text{th}}$ **Enc** query, it means k can decrypt the ciphertext (r^i,c^i) into the correct message m_i. According to the process in Fig. 3, for any key $k\neq k^*$, if $\mathcal{A}$ decrypts the i$^{\text{th}}$ ciphertext (r^i,c^i) using k, the HE scheme generates a seed s from $c_i\oplus\mathsf{h}(r_i||k)$. Here we stress that since HE schemes are primarily designed for low-entropy key settings, $\mathcal{A}$ can enumerate all low-entropy keys $k\in\mathcal{K}$.

Since we model the hash function h as a random oracle model, s is on a uniform distribution on $\{0,1\}^l$, and the probability of $\mathsf{decode}(s)=m_i$ is $\Pr_d(m_i)$. Hence, the probability that a key k is consistent with i$^{\text{th}}$ **Enc** query is $\Pr_d(m)\leq\omega_m$. The probability that the key $k\neq k^*$ is consistent with the q **Enc** queries is at most ω_m^q. We denote A_k as the event that the key $k\neq k^*$ is consistent with q **Enc** queries, and by $\overline{A_k}$ we denote that there exists at least one **Enc** query that the key k is inconsistent with. Thus, we have an MR-KMA advantage

$$\Pr(\cap_{k\neq k^*}\overline{A_k})=1-\Pr(\cup_{k\neq k^*}A_k)\geq1-\sum_{k\neq k^*}\Pr(A_k)\geq1-|\mathcal{K}|\omega_m^q. \tag{35}$$

Eq. 35 shows that if all the keys except the real key k^* are inconsistent at least one of the q **Enc** queries, $\mathcal{A}$ can easily rule out all keys except the real key and get the real message, thus winning the game. Considering the above analysis at q different salt values, we obtain a final lower bound for the MR-KMA advantage $1-|\mathcal{K}|\omega_m^q-\frac{2q}{2^l}$.

A.7. Analysis on MR-SI security

Oprisanu et al. [24] propose the MR-SI security for HE schemes. They claim that if the MR-SI adversary $\mathcal{A}$ has $\kappa=\lceil\log|\mathcal{K}|\rceil$ characters of the challenge message, $\mathcal{A}$'s advantage is $\mathsf{Adv}^{\text{mr-si}}_{\mathsf{HE},p_{\mathcal{M}},p_{\mathcal{K}}}(\mathcal{A})\geq\frac{1}{2\kappa^2}$, which is similar to Jaeger et al.'s MR-KMA result [17]. However, in Oprisanu et al.'s result [24], E_q is the key set satisfying q-1 characters, and $\mathbb{E}[\frac{|E_q|}{|E_{q\text{-}1}|}]$ is the probability that $\mathcal{A}$ only finds a message matching q$^{\text{th}}$ characters.

In this section, we revisit the MR-SI security of HE scheme. In the MR-SI game, we let m^* be the challenge message, k^* be the chosen key, and c^* be the challenge ciphertext. For a given side information $I\in\mathcal{I}$, $\mathcal{A}$ can determine the messages set $M^I\subset\mathcal{M}$ that contains the message in $\mathcal{M}$ matching I. $\mathcal{A}$ does as follows: (1) $\mathcal{A}$ decrypts the ciphertext c^* using a key $k\in\mathcal{K}$, and if the decrypted message is in M^I, add the key to the key set K^I; (2) After enumerating all keys, $\mathcal{A}$ randomly picks a key from K^I and decrypts the ciphertext. The probability of a key being in K^I is $\Pr_d(M^I):=\sum_{m\in M^I}\Pr_d(m)$. Here we stress that since HE schemes are primarily designed for low-entropy key settings, $\mathcal{A}$ can enumerate all low-entropy keys $k\in\mathcal{K}$. Therefore, $\mathcal{A}$'s success rate given side information I is

$$\begin{aligned}&\sum_{i=0}^{|\mathcal{K}|-1}\binom{|\mathcal{K}|-1}{i}\Pr_d(M^I)^i(1-\Pr_d(M^I))^{|\mathcal{K}|-i-1}\\&=\frac{1}{\Pr_d(M^I)|\mathcal{K}|}\cdot\left[1-(1-\Pr_d(M^I))^{|\mathcal{K}|}\right].\end{aligned} \tag{36}$$

Further, for all side information $I\in\mathcal{I}$, $\mathcal{A}$'s expected success rate (MR-SI advantage) is

$$\begin{aligned}&\mathbb{E}\left[\frac{1}{\Pr_d(M^I)|\mathcal{K}|}\cdot\left[1-(1-\Pr_d(M^I))^{|\mathcal{K}|}\right]\right]\\&=\mathbb{E}\left[\frac{1}{\Pr_d(M^I)|\mathcal{K}|}\right]\cdot\mathbb{E}\left[1-(1-\Pr_d(M^I))^{|\mathcal{K}|}\right]\\&\geq\frac{\alpha}{|\mathcal{K}|}\cdot\mathbb{E}\left[\frac{1}{\Pr_d(M^I)}\right]\geq\frac{\alpha}{|\mathcal{K}|\cdot\mathbb{E}\left[\Pr_d(M^I)\right]},\end{aligned} \tag{37}$$

where α is a constant less than 1, representing the minimal value of $\left[1\text{-}(1\text{-}\Pr_d(M^I))^{|\mathcal{K}|}\right]$ for side information $I\in\mathcal{I}$. The obtained lower bound is related to the $\Pr_d(\cdot)$ which is induced by a DTE. This lower bound may be useless if $\mathbb{E}[\Pr_d(M^I)]$ is large, which is showing that the system may have been carefully designed for the defense purpose.

Appendix B.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

This paper investigates honey encryption, which is an encryption scheme used to protect password vaults wherein instead of simply failing to decrypt when provided the wrong password, the system instead yields a decoy vault that can confuse an attacker and waste time. Specifically, a plausible decoy vault is generated per trial-decryption, which forces the attack to implement online verification of the credentials in the vault. The authors introduce new tighter bounds for DTE security, the encoder and fundamental component of existing constructions of honey vaults. Finally, the authors introduce new attacks against specific DTE constructions, and evaluate their attack success against existing attacks, highlighting their relative effectiveness.

B.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

B.3. Reasons for Acceptance

1) The paper tackles three aspects of honey vault security: (1) proving new bounds on attacker capabilities in honey vault systems, (2) discussing limitations of prior honey vault system designs that leads to new attacks, and (3) characterizing the effectiveness of prior adaptive mechanisms. Overall, it provides a fairly substantial analysis of the theoretical and practical security of honey vault systems.
2) The paper makes a spectrum of contributions, including (1) new upper and lower bounds to successful attacks against honey vaults and (2) the introduction of encoding attacks against honey vaults.
3) The paper practically evaluates the new attacks attacks against two major honey vault systems, demonstrating an improved attack efficiency ranging from 1.15 to 4.35 times compared to existing methods.