How Password Managers can benefit from Honey Encryption

Seminar: Advances in Cryptography and IT-Security

Brian Sinkovec

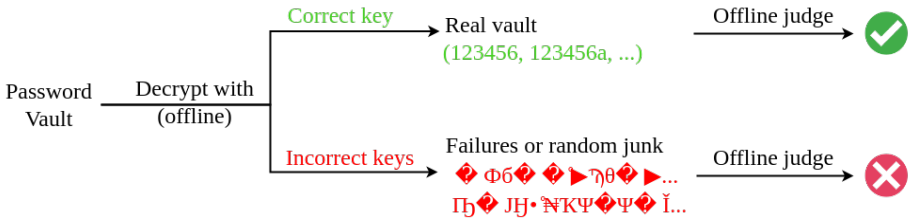
January 20, 2022
RWTH Aachen
Research Group IT-Security

Organizer: Ulrike Meyer
Supervisor: Andreas Klinger

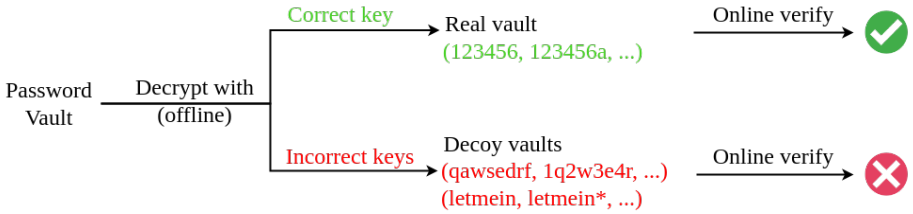
Abstract. Password-based authentication is still the pre-dominant authentication method for most online services we use today. Nevertheless, users rarely choose secure passwords, which enables attackers to mount dictionary or brute-force attacks. Therefore, password managers are considered a solid approach to securely manage user credentials. But they quickly become a valuable target for attackers, and poorly chosen master passwords may cause even more harm. Due to this, recent contributions tackle this problem, and *Honey Encryption* (HE) is one such approach. HE schemes deceive an attacker by providing plausible-looking password vaults when given a wrong master password. This forces the attacker to use online services for verification which drastically impairs the overall attack. In this seminar paper, we investigate state-of-the-art research on this topic, present the core idea of honey encryption, and also show some developed Distribution-Transforming Encoders, which is the heart of an HE scheme. We discuss the benefits of honey encryption in the context of password managers as well as its limitations and give an outlook for future work on this topic.

1 Introduction

Nowadays, most systems require authentication and authorization methods. With many methods around, password-based authentication is still the most common one. Long, randomly chosen passwords are considered to be sufficiently secure. However, if chosen poorly, passwords have a low entropy, which makes dictionary attacks or even brute-force attacks feasible within reasonable time. Several studies also show that poorly chosen passwords are not the exception [IS10,SB14]. This is due to the fact that humans usually choose passwords that they can memorize well, are shorter, have a similar structure, or are reused completely. Cracking such passwords can be done without much effort.



(a) Offline verification attack with traditional password vaults.



(b) Online verification attack with honey vaults.

Fig. 1: Comparison of traditional password vaults and honey vaults regarding an offline verification attack. [CLW+21]

Therefore, security experts often recommend to use *password managers* (PMs). Password managers, sometimes also known as *password vaults*, are applications that store user passwords in a secure way. They also help users create better passwords, e.g., by generating long, random ones or warning the user if the password was found in a password breach already. To access their password vault, users typically need to authenticate by a *master password*, which must be chosen carefully. The password vault is then encrypted using a traditional password-based encryption scheme (PBE).

Different types of PMs exist nowadays, e.g., separately installed PMs like Keepass [Kee21] and Lastpass [Las21], built-in PMs like the OSX Keychain, and browser-based ones, which are directly integrated into the browser, e.g., Chrome or Firefox. More and more modern PMs are cloud-based applications, and one of the key features is to synchronize the password vault between multiple devices. While this increases the usability and adoption of PMs, these types of features also introduce new threats that PMs have to face. For instance, the cloud-based PM Lastpass reported a security breach in which usernames and password vaults are leaked [Whi21].

In an *offline verification attack* (cf. Fig. 1a), an attacker obtained a local copy of a password vault and wants to decrypt it by mounting an offline dictionary or brute-force attack on the master password. When using traditional PBE algorithms, decrypting the password vault will only succeed if the encryption scheme is provided with the correct master password. Otherwise, decryption either fails or produces a randomly looking binary string. Therefore, an attacker is able to verify whether he supplied the correct master password or not in an

offline manner, i.e., this attack can be conducted without any server interaction. Thus, the attack is only limited by the attacker's computational capabilities.

In the last decade, a lot of research has been made to tackle this problem, and *Honey Encryption* is one such approach. In the context of password managers, password vaults encrypted with a honey encryption scheme are often referred to as *honey vaults*. Intuitively, when an attacker decrypts a honey vault with a wrong master password, a plausible looking *decoy vault* will be generated to deveice him. The decoy vault is supposed to be indistinguishable from the real vault. Thus, an attacker is required to shift the offline verification to an online verification attack (cf. Fig. 1b). He has to verify whether the decrypted password vault is the real one, e.g., by performing a login on a social media account. This drastically impairs the overall attack because such login forms often restrict the attack by allowing only a limited number of logins at a time.

The key aspects of how honey encryption works and which scientific contributions have been made in the past few years are presented in this seminar paper. The structure of this work is as follows: in the following section, password managers in general, their adoption, and which challenges they have to face are presented. To better understand the motivation behind honey encryption, we briefly discuss Kamouflage, an earlier approach that implements the idea of decoy vaults in a similar but different way. A general overview of honey encryption and how it works is shown afterward. To this extent, the core component of a honey encryption scheme, the Distribution Transforming Encoder, has been examined and new ones developed in recent contributions, which are presented shortly. Then, advantages and limitations of the honey encryption approach are discussed. Finally, we conclude and provide an overview for future work on this topic.

2 Background

2.1 Password Managers

At its heart, PMs store the passwords of a user in an encrypted password vault. Most of the time, encryption and decryption of the password vault is based on a password-based encryption scheme (PBE), e.g., AES-256 in CTR mode with a key derived from a password-based key derivation function. However, modern PMs implement a lot more features. Most PMs provide *synchronization* between devices, which is achieved by the cloud-based design of these PMs. The password vault of a user is encrypted locally and then stored on a cloud storage service like AWS or Microsoft Azure, which can be fetched by different devices of the same user. The usability of PMs is further increased by features like *autofill*, which automatically inserts the users' credentials into a login form, sometimes even without any user interaction. Also, PMs can typically store more than just passwords, e.g., domain names, additional notes, or any custom fields.

Furthermore, PMs help users to choose and manage more secure passwords. Almost all PMs provide features like *password generation* to create long, random passwords, and *password evaluation* to inform the user when any of his passwords have been found in a password breach or the password is in general insecure.

Although PMs are often recommended by security experts, their adoption is still in their beginning [FAKB17]. Several studies have shown that a lot of people simply don't know about PMs or refuse to use them [PZB+19,RWKA21]. Reasons for that are the lack of technical knowledge to use PMs or at least understand how they work, convenience, and past negative experiences.

2.2 Kamouflage

The goal of PMs is to provide secure password storage for its users, but also to suggest stronger passwords to use, e.g., by generating these. Therefore, the security of the users' online accounts solely depends on the strength of their master passwords. However, there is no evidence that users choose their master passwords more carefully. Weak master passwords allow an attacker, who captured an encrypted vault of a user, to mount an offline dictionary or even brute-force attacks against them. Thus, an attacker is only bounded by his computational power.

A first attempt to tackle this problem was provided by the authors of *Kamouflage* [BBBB10]. The idea of this approach was later the basis for honey encryption. Bojinov et al. presented the concept of decoy vaults which are stored alongside the real password vault. Kamouflage typically stores a large number of decoy vaults ($\sim 10^7$), and for each of these decoys, a master password different from the correct one is assigned. An attacker entering a wrong master password either receives one of these decoy vaults if the master password matches one of the associated decoy vaults or the program fails. The generation of decoy vaults is similar to the approach based on probabilistic context-free grammars presented in a later section. A potential drawback of this approach is the overhead of storing a huge number of decoy vaults additionally. Chatterjee et al. [CBJR15] broke the scheme by exploiting the fact that the generated master password of the decoys revealed the general structure of the original master password. This enables an attacker to significantly narrow down the search space, and thus Kamouflage degraded the overall security as opposed to traditional PBE-based PMs.

3 Honey Encryption

3.1 Honey Encryption Overview

In general, an HE scheme [JR14] consists of two algorithms, `HEnc` and `HDec` for encryption and decryption, respectively. The encryption algorithm `HEnc` first transforms a given plaintext M into a seed S and then encrypts the seed using a traditional PBE algorithm. Decryption works exactly the other way around. When provided with a wrong key, i.e., a wrong master password, the HE scheme will produce a "plausible-looking" *decoy message*. Therefore, an attacker is not able to verify whether he supplied the correct key to the HE scheme in an offline manner. In the context of PMs, the attacker is forced to shift to an online verification process, which drastically impairs the overall attack. The architecture of an HE scheme is depicted in Fig. 2.

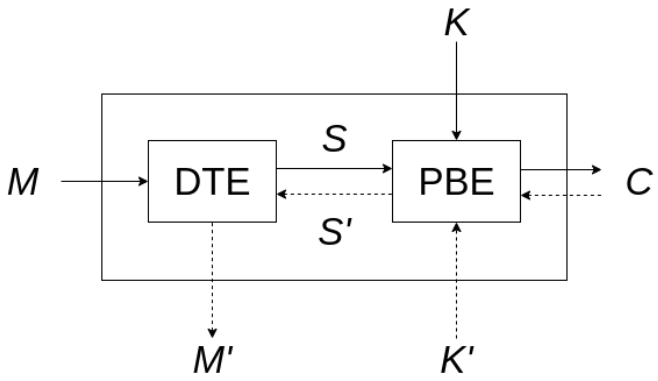


Fig. 2: High-level view of an HE scheme. The core components of such scheme are the Distribution-Transforming Encoder (DTE), and the traditional Password-based Encryption scheme (PBE).

Formally, an HE scheme is defined as follows. Let $\mathcal{M}$ be the message space, $\mathcal{K}$ the key space, and $\mathcal{C}$ the ciphertext space.

Definition 1. *Honey Encryption (HE) scheme* [JR14]. *An HE scheme is a pair of algorithms,* (`HEnc`, `HDec`)*, for encryption and decryption, respectively. For any given message $M \in \mathcal{M}$ and key $K \in \mathcal{K}$, the randomized algorithm $\texttt{HEnc}_K$ produces a ciphertext $C \in \mathcal{C}$. For any given ciphertext $C \in \mathcal{C}$ and key $K \in \mathcal{K}$, $\texttt{HDec}_K$ (deterministically) produces a message $M \in \mathcal{M}$. It is required that decryption always succeeds, i.e., for any $M \in \mathcal{M}$ and $K \in \mathcal{K}$, it holds:*

$$\Pr[\texttt{HDec}_K(\texttt{HEnc}_K(M)) = M] = 1$$

In its heart, a HE scheme consists of a *distribution-transforming encoder* (DTE), which is a message encoding scheme. Intuitively, a DTE transforms a message from some target distribution into another. When encoding a message, a DTE yields a "seed" from some seed space $\mathcal{S}$ and decoding such a seed yields the message again. Formally, a DTE is defined as follows:

Definition 2. *Distribution Transforming Encoder (DTE)* [JR14]. *A DTE is a pair of algorithms,* (`encode`, `decode`)*. The randomized algorithm* `encode` *takes a message $M \in \mathcal{M}$ and produces a seed $S \in \mathcal{S}$. The deterministic algorithm* `decode` *takes a seed $S \in \mathcal{S}$ and produces the corresponding message $M \in \mathcal{M}$. A DTE scheme is called* correct *iff for any $M \in \mathcal{M}$, the following holds:*

$$\Pr[\texttt{decode}(\texttt{encode}(M)) = M] = 1$$

Depending on the message distribution p_M, designing such a DTE is a challenging task. For uniform distributions, this is trivial and a general-purpose DTE is presented in the following section. However, password distributions are rather complex [MM12]. Therefore, newer contributions suggest and apply techniques from natural language processing, as these methods have been used successfully for password crackers [WAMG09].

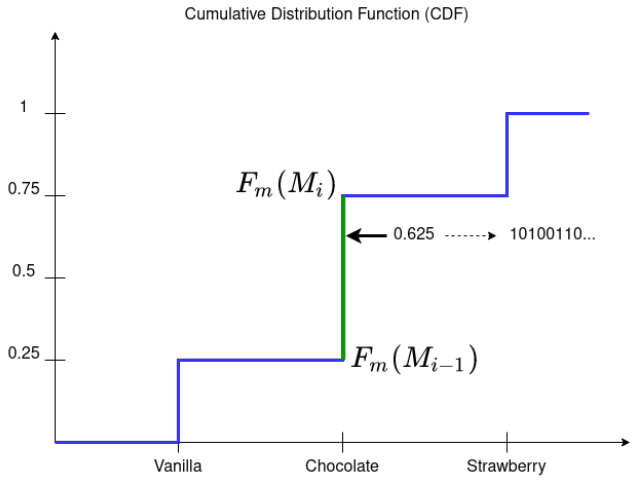


Fig. 3: Example cumulative distribution function (CDF) of the ordered message space $\mathcal{M} = \{\texttt{Vanilla, Chocolate, Strawberry}\}$.

3.2 Distribution Transforming Encoders

Inverse Sampling. Juels and Ristenpart [JR14] presented a general purpose DTE for uniform target distributions in their paper, which they called *inverse sampling* DTE, or *IS-DTE* for short. This DTE scheme works as follows: let p_m be a known message distribution of the ordered message space $\mathcal{M} = \{M_1, \ldots, M_{|\mathcal{M}|}\}$. Also, let F_m be the cumulative distribution function (CDF) over p_m, with $F_m(M_0) = 0$, and let the seed space be $\mathcal{S} = [0, 1)$. When encoding a message M_i, the IS-DTE scheme picks a random value $S \in \mathcal{S}$ uniformally s.t. $F_m(M_{i-1}) \leq S \leq F_m(M_i)$ holds. Decoding a given seed $S \in \mathcal{S}$ works by computing the inverse of the CDF, i.e., $M = F_m^{-1}(S) = \min_i\{F_m(M_i) > S\}$.

As an example, consider the following distribution of the finite, ordered message space $\mathcal{M} = \{\texttt{Vanilla, Chocolate, Strawberry}\}$ with $\texttt{Vanilla} < \texttt{Chocolate} < \texttt{Strawberry}$. Let p_m be the distribution with $p_m(\texttt{Vanilla}) = p_m(\texttt{Strawberry}) = 0.25$, and $p_m(\texttt{Chocolate}) = 0.5$, and the corresponding CDF F_m as depicted in Fig. 3. If we want to encode the message `Chocolate` using the IS-DTE, we would sample a value between $[F_m(\texttt{Vanilla}), F_m(\texttt{Chocolate})) = [0.25, 0.75)$, e.g., 0.625 which is then transformed into a binary string. Conversely, decoding first transforms the binary string into the respective decimal value and then determines the corresponding message according to F_m.

Natural Language Encoder. Chatterjee et al. [CBJR15] developed two DTE schemes with the complex, non-uniform distribution of passwords in mind. In particular, they presented the concept of *natural language encoders* (NLE). Their DTE schemes are based on n-gram Markov models and probabilistic context-free grammars (PCFG). The latter was described in more detail, whereas the former

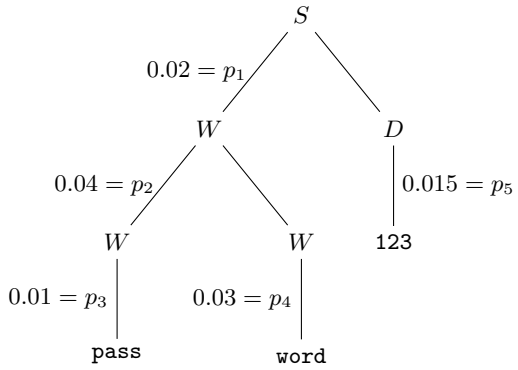


Fig. 4: Example parse tree of the word `password123` with the corresponding probabilities for each rule.

was developed further by Golla et al. [GBD16] which is explained in the following section. These techniques were chosen because they already have been used effectively to build password crackers, as they model the distribution of natural language texts, which passwords mostly are, quite well.

Definition 3. *Probabilistic Context-Free Grammar (PCFG). A PCFG is a five-tuple* $G = (N, \Sigma, P, S, p)$ *where*

- N *is a set of* non-terminal *symbols,*
- Σ *is a set of* terminal *symbols,*
- P *is a set of* productions*,*
- $S \in N$ *is the* start symbol*, and*
- $p : P \to [0, 1]$ *is a* function *denoting the* probability *for each rule.*

It is required that for each non-terminal $X \in N$*, the following holds:*

$$\sum_{X \to \beta \in P} p(X \to \beta) = 1$$

In a PCFG, a probability is assigned to each rule. Correspondingly, a parse tree, and thus a word of the PCFGs language, can be represented as the sequence of probabilities that were used for the left-derivation of that word. For instance, a parse tree of the word `password123` for a PCFG which contains (among other things) the following rules: $\{S \to WD, W \to WW \mid \texttt{pass} \mid \texttt{word}, D \to \texttt{123}\}$ is depicted in Fig. 4. The parse tree can be reconstructed by selecting a rule using p_1 from the rule set for the start symbol, then by using the probabilities p_2, p_3, etc. to construct the children and so forth. Encoding a single password using the PCFG-based DTE works as follows: one constructs the parse tree for the given password, which then yields the sequence of probabilities $p_1, \ldots, p_k$. Each probability can then be represented by an b-bit integer $X_1, \ldots, X_k$. The resulting seed is the vector of k integers, $\vec{X} = \langle X_1, \ldots, X_k \rangle$. Decoding such a vector works by reconstructing the corresponding parse tree, which then yields the password.

Extending the single-password DTE to a password-vault DTE can be done by applying the encoder to each password in the vault independently. While this works well for vaults consisting of computer-generated and human-chosen passwords, a vault that contains similar or related passwords may suffer from easier detection. Thus, the authors presented another technique to handle password vaults, which they called the *sub-grammar approach* (SG). Intuitively, the SG approach first encodes each password in the vault using the actual trained PCFG. Then, it generates a sub-grammar PCFG consisting of the rules used in parsing the passwords while keeping the probabilities and normalizing these. Finally, the sub-grammar is encoded and forms, together with each password encoded separately by the new sub-grammar, the output of the DTE.

The authors of the NLE approach also implemented their approach in an open source password management system, which they named `NoCrack` [Cha21].

Adaptive NLE based on Markov models. Golla et al. [GBD16] investigated the security of Chatterjee et al. NLE approach and Kamouflage further. First, they introduced the concept of *static* (like `NoCrack`) and *adaptive* NLE variants. The generated distribution of decoy vaults by a static NLE is *independent* from the actual vault. In contrast, adaptive NLEs generate distributions of decoy vaults that are *dependent* on the actual vault. This means that decoy vaults generated by an adaptive NLE contain passwords that are more likely to be similar to the passwords in the real vault. With static NLEs, the lack of this property allows an attacker to distinguish decoy vaults from the real one by performing a *Kullback-Leibler Divergence attack*.

In this attack, one first determines the distribution of the decoy vaults generated by the NLE. Then the Kullback-Leibler divergence (which computes the distance between two distributions) is used to rank the candidate vaults received from trial-decrypting several candidate master passwords. A larger distance between the distributions means that the candidate vault is more likely the real one. In their experiments, they discovered that this attack can distinguish decoy vaults from real vaults with a significantly higher probability.

Thus, the authors presented an adaptive NLE based on n-gram Markov models. Encoding and decoding using this approach work similarly to the previous approaches by mapping from the set of passwords to binary strings. The encoding of a password is done by first fixing the ordering of n-grams, e.g., lexicographically. For each transition probability $P(c_4|c_1, \ldots, c_3)$, the fixed order partitions the interval $[0, 1)$ into segments. Then, given a transition in the password, one determines the segment it falls into and samples a random value s uniformly. This is done for each transition, which yields a vector $\vec{S} = \langle s_1, \ldots, s_{|pwd|} \rangle$, which in turn can be transformed into a bitstring. Decoding works similarly by determining for each value in $\vec{S}$ the segment it falls into to retrieve the corresponding n-grams to reconstruct the password.

This construction is, however, also static. To transform it to an adaptive one, the authors suggest modifying the model s.t. n-grams of passwords appearing in the real vault get assigned higher probabilities. Otherwise, decoy vaults are,

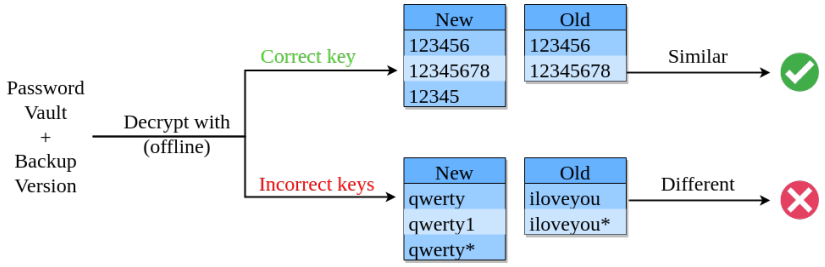


Fig. 5: In the multi-leakage case, an attacker possesses two versions of a honey vault. Thus, intersection attacks are possible.

again, distinguishable from the real one. They modified the model in two steps. First, they increase the probabilities single n-grams of passwords appearing in the vault, s.t. n-grams of these passwords are more likely to appear in the decoy vault as well. Then, they also boost the probability of random other n-grams (not appearing in the real vault) to reduce the risk of adding the actual passwords to the decoy vault. In the end, all probabilities are re-normalized. The analysis they provided showed that these adjustments improved the security of their model since their Kullback-Leibler divergence attack was less effective than on the static PCFG approach.

3.3 Incrementally Updatable HE Password Vaults

Cheng et al. [CLW+21] presented another threat that current HE schemes did not take into account yet. In the *multi-leakage case*, all exisitng HE schemes suffer from *intersection attacks*. In this scenario, an attacker was able to acquire two versions of a honey vault, e.g., the current version as well as a backup version. Fig. 5 depicts the attacker model that reveals the problem of intersection attacks. When decrypting both versions of the honey vault with the correct master password, the attacker receives two password lists that are similar. Otherwise, when providing a wrong master password, the HE scheme will produce a decoy vault for both versions, respectively. However, both decoy vaults will be different, i.e., they contain significantly more passwords that are not present in the respective other list. Thus, an attacker can differentiate between those two cases and is able to verify offline whether he guessed the correct master password.

This is due to the design of HE schemes developed so far: when a honey vault is updated, the whole vault, i.e., each password separately, is encoded and then concatenated. Due to the randomized `encode` algorithm, this leads to different seeds, and therefore to substantially different decoy vaults between two versions.

Cheng et al. [CLW+21] introduced a DTE based on the adaptive NLE model of Golla et al. [GBD16] with a prefix-keeping property, i.e., the generated seed of an old honey vault version is a prefix of the subsequent version. Then, if the PBE scheme is also prefix-keeping, the similarity between the decoy vaults of

two versions is ensured. Therefore, an attacker degenerates to an attacker in the single-leakage case.

4 Discussion

Using honey encryption in the context of PMs comes with several advantages and limitations. The key advantage of HE schemes is that they provide security beyond the brute-force bound [JR14]. This means, even if the HE scheme fails to provide plausible looking plaintexts on wrong keys, the amount of effort an attacker has to invest is never less than offline brute-force attacks on traditional PBE-only PMs. Also, integrating a HE scheme into a PM does not degrade usability, as the user does not have to perform any additional interactions to benefit from this feature. Other techniques to increase the security of PMs fail at this requirement. Converting a traditional PBE-only scheme into a HE scheme is also trivial if one possesses a good DTE. When considering simple or uniform password distributions, simple and effective DTEs can be used, like the IS-DTE.

However, in most circumstances, it is to be expected that password distributions are rather complex. Thus, developing DTEs for human-chosen passwords is a challenging task and still an open question. Recent contributions in this research topic have shown that existing DTE implementation may still suffer from unknown threats. For instance, Cheng et al. [CZL+19] presented another type of attack, which they called *encoding attacks*, from which all DTEs suffer presented in this seminar paper. In short, they discovered that these implementations do not consider the ambiguity to encode a password (e.g., multiple parse trees may exist for a single word of a PCFGs language), but choose those *encoding paths* deterministically. Due to this, encoding attacks can exclude some decoy seeds without knowledge about the password distribution. Further research into formal analysis techniques for HE schemes is therefore required.

Also, until now, HE schemes only consider the encryption and decryption of passwords stored in a password vault. However, encryption of additional information like domain names, credit card secrets, and other private information was not considered yet. Therefore, attackers get insights into the password behavior of a user, and HE security might not hold anymore [JR14]. Note that this is also challenging to implement because domain names and credit card numbers do not share in any way the same distribution as passwords.

Another problem for legitimate users of honey vaults occurs if the user had a typo when entering his master password. The user may not realize or even be unable to recognize that he received a decoy vault. This is a problem that might degrade usability, and therefore the adoption of HE schemes in PMs significantly. However, there are approaches like presenting the same image on each re-login [DT05]. The user can then verify whether he supplied the correct master password when the same image is shown again.

Finally, even though HE schemes do not require any additional user interaction and therefore do not degrade usability, they still may produce a significant computation and storage overhead.

5 Conclusion

To conclude, one can see that a lot of approaches have been proposed to accomplish better security for PMs. In this seminar paper, recent contributions following the honey encryption approach have been presented and summarized. A general overview of how HE schemes work and different DTE which were developed in the past were shown. Also, some attacks exploiting the flaws existing in these DTEs were presented briefly. Finally, advantages and limitations of HE in the context of PMs were discussed.

Although password-based authentication is still the most common authentication form today, people struggle to choose secure passwords. Password managers increase the security and usability of users, help them to choose more secure passwords, and typically are synchronized over multiple devices. While their adoption is still not as widespread as it should be, there are many attack vectors that need to be considered, especially due to the value of these targets. Honey encryption tackles the problem of offline verification attacks, in which an attacker is only bound by his computational power. The most important advantage of using HE schemes in PMs is that they do not degrade the security, and even if the DTE fails at its requirements, an offline brute-force attack is the best an attacker is able to perform. However, also mentioned by the authors of these DTEs, it is still an open task to formally analyze HE schemes and the corresponding DTEs. Only then, currently used PMs like LastPass or Keepass may integrate this technique into their products.

References

[BBBB10] Hristo Bojinov, Elie Bursztein, Xavier Boyen, and Dan Boneh. Kamouflage: Loss-Resistant Password Management. In *Computer Security – ESORICS 2010*, pages 286–302, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[CBJR15] Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart. Cracking-Resistant Password Vaults Using Natural Language Encoders. In *2015 IEEE Symposium on Security and Privacy*, pages 481–498, 2015.

[Cha21] Rahul Chatterjee. NoCrack. https://github.com/rchatterjee/nocrack/, 2021. Accessed: 2022-01-16.

[CLW+21] Haibo Cheng, Wenting Li, Ping Wang, Chao-Hsien Chu, and Kaitai Liang. Incrementally Updateable Honey Password Vaults. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 857–874. USENIX Association, August 2021.

[CZL+19] Haibo Cheng, Zhixiong Zheng, Wenting Li, Ping Wang, and Chao-Hsien Chu. Probability Model Transforming Encoders Against Encoding Attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1573–1590, Santa Clara, CA, August 2019. USENIX Association.

[DT05] Rachna Dhamija and J. Tygar. The Battle against Phishing: Dynamic Security Skins. In *Proceedings of the 1st Symposium on Usable Privacy and Security*, volume 06, pages 77–88, 01 2005.

[FAKB17] Michael Fagan, Yusuf Albayram, Mohammad Khan, and Ross Buck. An investigation into users' considerations towards using password managers. *Human-centric Computing and Information Sciences*, 7:12, 2017.

[GBD16] Maximilian Golla, Benedict Beuscher, and Markus Dürmuth. On the Security of Cracking-Resistant Password Vaults. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1230–1241, New York, NY, USA, 2016. Association for Computing Machinery.

[IS10] Philip Inglesant and M. Angela Sasse. The true cost of unusable password policies: password use in the wild. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010.

[JR14] Ari Juels and Thomas Ristenpart. Honey Encryption: Security Beyond the Brute-Force Bound. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 293–310, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Kee21] KeePass. KeePass Password Manager. https://keepass.info/index.html, 2021. Accessed: 2022-01-16.

[Las21] LastPass. LastPass Password Manager. https://www.lastpass.com/, 2021. Accessed: 2022-01-16.

[MM12] David Malone and Kevin Maher. Investigating the Distribution of Password Choices. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 301–310, New York, NY, USA, 2012. Association for Computing Machinery.

[PZB+19] Sarah Pearman, Shikun Aerin Zhang, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Why people (don't) use password managers effectively. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, pages 319–338, Santa Clara, CA, August 2019. USENIX Association.

[RWKA21] Hirak Ray, Flynn Wolf, Ravi Kuber, and Adam J. Aviv. Why Older Adults (Don't) Use Password Managers. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 73–90. USENIX Association, August 2021.

[SB14] Elizabeth Stobert and Robert Biddle. The Password Life Cycle: User Behaviour in Managing Passwords. In *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 243–255, Menlo Park, CA, July 2014. USENIX Association.

[WAMG09] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. Password Cracking Using Probabilistic Context-Free Grammars. In *2009 30th IEEE Symposium on Security and Privacy*, pages 391–405, 2009.

[Whi21] Lance Whitney. LastPass CEO reveals details on security breach. https://www.cnet.com/tech/services-and-software/lastpass-ceo-reveals-details-on-security-breach/, 2021. Accessed: 2022-01-16.