OTPPM: A One-time Pad-based Password Manager

Haozhe Min, Yang Li, Weijun Li, Yingpei Zeng*
School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China

***Abstract*—Password managers are now widely used to help manage passwords. However, password managers usually encrypt passwords with algorithms that are not information-theoretically secure, and such algorithms are increasingly vulnerable to attacks over time. Given the critical role of passwords in safeguarding information systems and the possibility that users may retain their passwords for extended periods, potentially up to a century, there exists a substantial risk that attackers could preserve retrieved password vaults and crack them later when the encryption algorithms are broken. In this paper, we first propose methods to estimate the lifetime of conventional encryption algorithms, which shows that they usually are only about $20 \sim 40$ years. Then, we present OTPPM, a novel password manager employing one-time pad (OTP) encryption technology. As the OTP encryption scheme is known to be information-theoretically secure, we develop algorithms to integrate OTP into password management. With OTPPM, passwords can be securely accessed remotely since the password vaults are stored on cloud-based servers. A prototype of OTPPM has been implemented, and evaluations demonstrate its practicality and efficiency for real-world use.**

***Index Terms*—Password Manager, Encryption Algorithm Lifetime, One-time Pad, OTP Encryption, Password Vault**

I. Introduction

Passwords play a crucial role in information systems. With the rapid evolution of the internet, individuals and enterprises have generated vast amounts of data in information systems that require effective management and protection under various online accounts. Password leaks may cause frequent data breach occurrences, and these breaches pose significant threats to personal privacy and corporate financial health [1]. In addition, it is known that people frequently use the same password across multiple accounts [2], which significantly increases the risk of a cascading security breach, where the compromise of one account could potentially jeopardize all associated accounts [3].

Password managers have emerged as a widely adopted solution for addressing the challenges of password management [4]. These tools enable users to securely store and manage credentials for multiple accounts while facilitating the generation of strong, unique passwords to enhance security. Numerous password managers have been developed within the industry [5], [6]. In the academic sphere, extensive research has also been conducted on this topic [7]–[12], with a primary focus on the security features and usability aspects of password managers. For example, the vulnerabilities related to the autofill feature in password managers are identified [8], and honey vault is proposed to protect the real password vault [13].

However, the encryption algorithms employed by current password managers lack information-theoretic security, posing a significant risk as such encryption algorithms inevitably degrade over time due to advances in computational power and cryptanalysis techniques [4], [14]. Both symmetric and asymmetric encryption algorithms are vulnerable to these developments. For instance, the Data Encryption Standard (DES), introduced in 1975, was successfully cracked by 1997 [15], while the RSA (Rivest–Shamir–Adleman) algorithm, first published in 1977, now faces escalating threats from quantum computing [16]–[18]. If password managers utilize such algorithms to encrypt passwords in vaults, attackers who gain access to these vaults could archive them and decrypt the contents once the algorithms become obsolete. A striking example occurred in 2019, when the passwords of UNIX pioneers were compromised due to the leakage of encrypted credentials on GitHub. The encryption methods, once considered secure in the 1970s, are now deemed ”dangerously obsolete” [19]. While this risk applies broadly to all data encrypted with aging algorithms, password vaults are particularly attractive targets due to the critical role passwords play in securing information systems and the tendency of users to retain the same passwords for extended periods, potentially spanning decades or even their entire lifetimes [2].

This study presents a novel password manager, OTPPM (One-time Pad Password Manager), which leverages the principles of one-time pad encryption [20]. We first propose two methods to estimate the lifetime of conventional encryption algorithms: a Kaplan-Meier estimator [21] based on historical data, and a theoretical estimator based on computational growth and cryptanalytic progress. The results show that their lifetime is usually only about $20 \sim 40$ years. In contrast, OTP encryption is always secure. The primary advantage of OTP encryption lies in its use of a unique key for each encryption operation, ensuring information-theoretic security. OTPPM incorporates carefully designed algorithms to effectively manage and utilize OTP keys. Notably, OTPPM enables secure remote access to passwords by storing encrypted vaults on a cloud-based server. A prototype implementation of OTPPM was developed and evaluated for usability and performance, with the results demonstrating its practicality and efficiency in real-world applications.

In summary, we make the following main contributions:

- We propose two methods to estimate the lifetime of encryption algorithms.
- We propose a new password manager OTPPM which could leverage the information-theoretically secure OTP encryption to protect passwords.
- We implement a prototype of OTPPM and test its usability and performance. We will publish our code for research usage at https://github.com/OTPPM.

The remainder of this paper is structured as follows. Section II introduces the background of the field. Section III presents the methods to estimate the encryption algorithm lifetime. Section IV discusses our threat model. Section V presents the design details of OTPPM. Results and analysis of the study are covered in Section VI and Section VII. Section VIII presents the related work before we conclude our paper in Section IX.

II. Background

A. Password Manager

A password manager is a computer program that allows users to store and manage their passwords for information systems like social media and mobile applications [4], [8]. Password managers can be in different forms like standalone password managers, and built-in password managers [8], [9]. Standalone password managers include popular password managers like LastPass [5] and 1Password [6]. Built-in password managers include built-in password managers in web browsers like Chrome and operation systems like iOS [9].

Password managers typically store user passwords in encrypted databases, commonly referred to as password vaults [7], [10]. Symmetric encryption algorithms, such as AES, are widely used for encrypting these databases [22], with encryption keys derived from user-selected master passwords using key derivation functions like PBKDF (Password-Based Key Derivation Function) [23], [24]. To facilitate remote access, password vaults are often stored on cloud servers, exposing them to potential leakage in the event of cloud data breaches [1]. The security of the encryption algorithms is therefore very important [22], as attackers may attempt to compromise these vaults. However, the algorithms employed by current password managers [7], [10] are increasingly vulnerable to evolving computational capabilities and advances in cryptographic attack methods [4], [14].

B. One-time Pad

The one-time pad (OTP) [20] is an encryption method known for its theoretical absolute security. First described by Frank Miller in 1882 [25], the OTP is proved as a perfectly secure encryption (i.e., also called absolute security, unconditional security, or information-theoretic security) technique [20]. The OTP is a random bit sequence and must be used only once. During the encryption process, the plaintext is XORed bit by bit with the corresponding bits from the pad, and decryption is achieved by XORing the ciphertext with the same bits from the pad. The simple XORing process also makes OTP encryption more efficient than other encryption algorithms that require multiple steps [26].

However, OTP is not without its limitations. The most critical of these is the requirement that the OTP length must match or exceed the length of the plaintext, restricting its applicability to scenarios involving only a limited number of bytes. Fortunately, this constraint aligns with the needs of our application in password managers since the size of password data usually is small, as demonstrated in Section VI-E. Another significant limitation is that each OTP bit must be used only once, and through careful design, our password manager adheres to this principle as well.

III. Estimation of Encryption Algorithm Lifetime

The encryption algorithms employed by conventional password managers will inevitably become obsolete [20], [27]. We aim to estimate the lifetime of encryption algorithms, defined by the National Institute of Standards and Technology (NIST) as the algorithm security lifetime [28]. Although we attempted to identify existing methods for this purpose, no suitable approaches were found. In this section, we propose two methods to estimate the lifetime of encryption algorithms.

A. Estimation Using the Kaplan-Meier Estimator

We first estimate the lifetime of encryption algorithms utilizing the Kaplan-Meier estimator based on historical data. The Kaplan-Meier estimator (also known as the product-limit estimator) [21] is one of the most widely used methods for estimating survival functions, particularly in medical and public health research, and has been applied in other domains, such as the time-to-failure analysis of industrial components [29], [30]. A notable advantage of this method is its ability to account for censored data, where the occurrence of an event (e.g., death) remains unobserved (e.g., loss to follow-up). In our analysis, the survival time of an encryption algorithm is defined as the interval between its introduction and its obsolescence. We determine the obsolescence time based on NIST recommendations [31]. If an algorithm remains approved for use as of 2025, we treat it as censored since its future status is unknown. The survival function is calculated as:

$$S(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}), \tag{1}$$

where t_i denotes the time at which at least one algorithm becomes obsolete (i.e., an event occurs), d_i is the number of algorithms that become obsolete at time t_i, and n_i is the number of algorithms still considered secure up to t_i.

The results are presented in Figure 1. The analysis includes 11 algorithms [31], encompassing symmetric encryption, asymmetric encryption, and hash functions, as these three categories represent the principal types of algorithms used to "encrypt" content [27]. The findings indicate that most algorithms remain secure for less than 30 years. The median survival time, when the survival probability drops to 0.5, is approximately 28 years. This result aligns with expectations, given the natural progression of cryptanalysis and the continual increase in computational capabilities.

B. Estimation With a Theoretical Model

We also develop a theoretical model to estimate the time required to compromise an encryption algorithm through brute-force attacks, incorporating the effects of both computational growth and cryptanalytic advancements.

Assume the current computational power is C_0 key searches per second for the algorithm under consideration. Computational power is assumed to double every y years. Meanwhile,

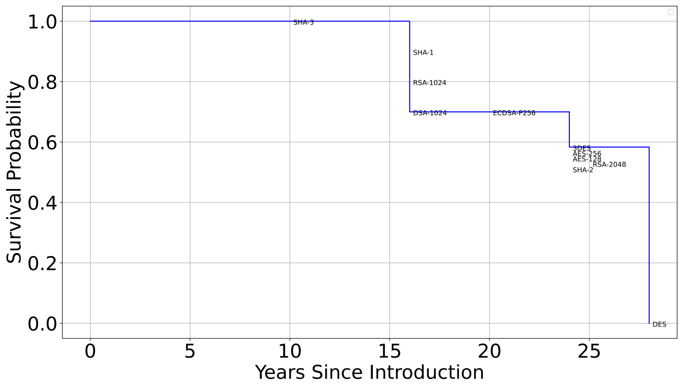

Fig. 1. Kaplan-Meier survival curve for encryption algorithms.

cryptanalytic progress is modeled such that the effective computational complexity of attacks decreases by a factor of 2^x every 10 years. The initial computational complexity of breaking the algorithm is $A_0 = 2^{a_0}$.

The computational power after t years (in keys per second) is given by:

$$C(t) = C_0 2^{yt}. \tag{2}$$

The effective computational complexity $A(t)$ required to break the algorithm at time t is:

$$A(t) = 2^{a_0} 2^{-xt/10}. \tag{3}$$

An attacker aiming to complete the brute-force search within a time window Δt must satisfy:

$$\int_t^{t+\Delta t} C(\tau) d\tau \geq A(t). \tag{4}$$

For simplicity, we assume Δt is one year. Noting that one year has approximately $2^{24.91}$ seconds ($3600 \times 24 \times 365$), the above inequality becomes:

$$2^{24.91} C_0 2^{yt} \geq 2^{a_0} 2^{-xt/10}, \tag{5}$$

which simplifies to:

$$24.91 + \log_2 C_0 + yt \geq a_0 - xt/10 \tag{6}$$

$$\implies t \geq \frac{a_0 - 24.91 - \log_2 C_0}{y + x/10}. \tag{7}$$

Equation 7 provides an estimate of the minimum time required for a successful brute-force attack. For instance, consider the widely adopted AES-128 algorithm [32]. Its initial computational complexity parameter a_0 is 128. In 2022, the search attack was approximately $9998 \times 10^6 \approx 2^{33.22}$ keys per second on an NVIDIA RTX 3080 GPU [33], [34]. For a distributed system with 20 million such GPUs, the effective computational rate would be about $2^{54.22}$ keys per second [34]. Assuming $y = 1$, consistent with Huang's Law that suggests GPU performance doubles annually [35], [36], and setting $x = 10$ to reflect potential quantum speedups like Grover's algorithm [18], we find:

$$t \geq \frac{128 - 24.91 - 54.22}{1 + 10/10} = 24.43. \tag{8}$$

Thus, under these assumptions, an attacker with access to 20 million commercial-off-the-shelf GPUs could potentially recover an AES-128 key within one year around 2046. If we set $x = 20$ to be more optimistic about the quantum computing, t would be ≥ 16.29.

IV. Threat Model

We briefly describe the threat model of OTPPM. We assume the password vaults stored in the cloud may be retrieved by attackers. For example, they may be leaked when data leakage incidents happen from cloud service providers. We also assume such attackers are *patient attacker*, which means they may store the obtained password vaults, waiting until either computational resources become sufficient or the encryption algorithms in use become vulnerable, enabling them to decrypt the vaults in the future.

However, we assume there is no patient attacker in the local environment. This means there is no patient attacker to intercept local communication, such as communications via LAN, Bluetooth, or USB, and there is no patient attacker to comprise local servers or devices. We need such an assumption because we need to manage OTPs in the local environment. The assumption is also plausible since local environments are under higher security protection (e.g., protected by firewalls). There may be other ordinary attackers in the local environment who want to get the passwords but will not save encrypted password vaults or other ciphertext and wait for cracking in the future.

V. Design of OTPPM

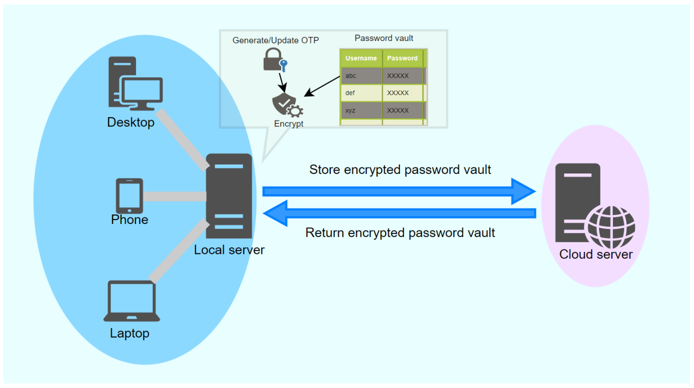

Fig. 2. The architecture of OTPPM.

Figure 2 illustrates the architecture and overall workflow of OTPPM. Similar to other password managers, OTPPM employs a password vault to store user credentials. However, OTPPM differentiates itself by utilizing one-time pad (OTP) encryption to secure the vault. The workflow of OTPPM involves three primary components: (1) **User Devices**, including desktops, phones, and laptops, which a user uses to initiate password operations such as retrieval and updates. (2) **Local Server**, which manages OTPs, accesses password vaults, and interacts with the cloud server on behalf of the user. The local server can operate on a standalone machine or reside on the user's devices, such as desktops or phones. Additionally, multiple devices may share a single local server to facilitate password management. (3) **Cloud Server**, which stores encrypted password vaults and ensures synchronization across devices, as user devices may not always connect to the same local server. In OTPPM, only OTP-encrypted password vaults are stored on the cloud server, and OTPs are never uploaded, even when encrypted with algorithms like AES. This design is critical under the threat model, which assumes that patient attackers may obtain the cloud server's data.

A password retrieval (reading) operation is used as an example to illustrate a typical workflow, with additional workflows discussed in Section V-C. When a user wants to access his or her passwords, the user communicates with the local server through their device, providing his or her username and password (which will be used as the master password as well). The local server authenticates the user with the cloud server using the credentials, and once authentication is successful, it downloads the encrypted password vault from the cloud server. The local server then decrypts the OTP using the master password and uses the decrypted OTP to decrypt the password vault, subsequently returning the passwords to the user.

A. *One-time Pad Management*

OTP management includes OTP generation and updates, OTP encryption and decryption, and OTP synchronization. Only user devices and local servers are involved in OTP management, and the cloud server is unaware of OTPs. OTP generation and updates occur under specific conditions, such as when no OTP exists, the current OTP is insufficient in length, or a regeneration is required. Initially, when a new user is added to the OTPPM system, the user must request the local server to generate a new OTP. If the existing OTP becomes inadequate, for instance, due to the addition of numerous passwords to the vault, two options are available: the system can either generate a new OTP and append it to the existing one or produce a longer OTP to completely replace the old one. Additionally, users may opt to regenerate their OTPs to reorganize the password vault. This is particularly useful in scenarios where a significant number of passwords have been deleted, leading to a substantial change in the vault's structure, as discussed later. OTPs can be generated using real random sources within the computer, such as mouse movements, network packet activity, and CPU noise, employing algorithms like Fortuna [37]. Alternatively, OTPs can be generated by dedicated hardware devices and securely transferred to the local server [38].

OTP encryption and decryption are employed to safeguard OTPs during transmission and storage. To protect OTPs, OTPPM utilizes the AES-CBC encryption method [39] before transmitting them to another local server or storing them locally, as the system assumes potential threats from ordinary attackers during local transmission. The AES encryption key is derived from the user's master password using the PBKDF2 algorithm [23]. Before being used to encrypt the user's password vault, the OTP is securely decrypted.

OTP synchronization is required when a user employs multiple local servers, ensuring that the OTP is consistently updated across servers whenever it is generated or updated. This synchronization is essential because local servers must have access to the user's OTP to decrypt the password vault. To achieve synchronization, the user employs local communication methods, such as LAN, Bluetooth, or USB, to manually transfer the encrypted OTP between local servers, exporting it from one server and importing it into another. Avoiding cloud-based synchronization prevents patient attackers from intercepting the OTP. Since synchronization is an infrequent operation, it is considered manageable in terms of user effort. To safeguard against potential OTP loss, users are advised to back up their OTPs in highly secure storage.

B. *Password Vault Management*

Password vault management is to manage the content of the password database, i.e., the password vault, which includes creating the password vault when the user first adds a password entry, updating the password vault when the user modifies his/her passwords, and reading the password vault when user need to read his/her passwords.

First, the format of the password vault is introduced. Similarly, the password vault of OTPPM is composed of password entries [7], [10]. Each entry includes a UUID to identify the entry in the password vault uniquely, a Label to identify the validity of the entry (will be explained later), and other fields like URL, Username, and Password for representing the specific information of the entry. The password vault of OTPPM is simply the concatenation of all the password entries one by one.

In OTPPM, updating the password vault, such as adding, modifying, or deleting entries, is handled in a unique manner. Rather than directly modifying the original vault, updates are appended to the end of the vault, as illustrated in Figure 3. This approach ensures that the original content remains unchanged, eliminating the need for re-encryption with a new OTP portion. Instead, only the appended section, which is typically much smaller, is encrypted using an unused segment of the OTP. The appended section may include multiple updated entries if the user modifies multiple passwords simultaneously and may consist of several appended parts if updates occur at different times. A Label field is employed to indicate the validity of each entry. For example, if the Label field of a new entry is marked as invalid, it signifies that the corresponding entry has been deleted. The handling of the password vault is further demonstrated through the reading algorithm described below.

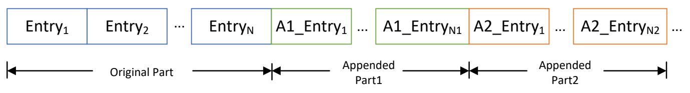

Fig. 3. An example password vault after being updated several times (in plaintext).

The steps for reading password entries from a password vault are outlined in Algorithm 1. A map data structure is employed to store the mapping between UUIDs and their corresponding password entries. Each entry in the vault is processed sequentially. For the currently processed entry, its UUID field is checked to determine whether the map already contains a corresponding entry. If an entry with the same UUID exists, the algorithm examines the Label field: if the label indicates invalid, the entry is removed from the map; if the label is valid, the algorithm calls COPYENTRY to overwrite the existing entry in the map with the current one. If the map does not already contain an entry with the same UUID, the current entry is directly added to the map. Finally, the password entries in the map are returned as the password entries in the vault.

Algorithm 1: Reading Entries from the Password Vault

```
Input: PasswordVault (in plaintext)
Output: PasswordEntries
1  ID2EntryMap ← $\emptyset$ ;   /* Initialize an empty
     mapping from a UUID (i.e., key) to its
     PasswordEntry (i.e., value) */
2  for Part in PasswordVault do
3      for Entry in Part do
4          if ID2EntryMap.CONTAINSKEY(Entry.UUID) then
5              if Entry.UUID == INVALID_ENTRY then
6                  ID2EntryMap.REMOVE(Entry.UUID)
7              else
8                  COPYENTRY(Entry,
                     ID2EntryMap[Entry.UUID])
9              end
10         else
11             ID2EntryMap[Entry.UUID] = Entry
12         end
13     end
14 end
15 PasswordEntries = ID2EntryMap.VALUES()
```

The update of the password vault is implemented by retaining the currently updated password entries. Whenever an entry is newly added, modified, or deleted, it is marked as updated using an in-memory-only field (i.e., not serialized into storage). If the entry is deleted, its Label field is further changed to INVALID_ENTRY. Once a predefined timer expires or the user manually triggers synchronization, the updated entries are appended as an additional part at the end of the current password vault. This appended part is then encrypted and the vault is synchronized with the cloud server (as described in Section V-C).

Over time, appended parts may accumulate as password entries are updated multiple times, resulting in a password vault with unnecessary size growth. For example, a modified entry may occupy two or more entries if it is modified multiple times. To address this, the password vault can be reorganized (retrofit) by consolidating all changes and re-encrypting the vault with a new OTP. This process involves merging all appended parts into the original section of the password vault, creating a new, streamlined vault that contains only the most up-to-date password data, and synchronizing the vault to the cloud server.

C. User Management and Password Vault Encryption

The cloud server in OTPPM is responsible for managing user accounts, functioning similarly to traditional password managers. New users register with the cloud server via their local server, after which they determine a username and password. This password also serves as the user's master password, reducing the need to remember multiple credentials. Simultaneously, the user's local server generates an OTP, as previously described, which is securely stored on the local server and never uploaded to the cloud server in any form. Upon successfully logging into the cloud server through their local server, users can download their encrypted password vaults to the local server for reading and updating. Once updated, the password vaults can be re-uploaded to the cloud server. Details on reading and updating the password vault are provided in Section V-B, while the processes of vault encryption and decryption are discussed as follows.

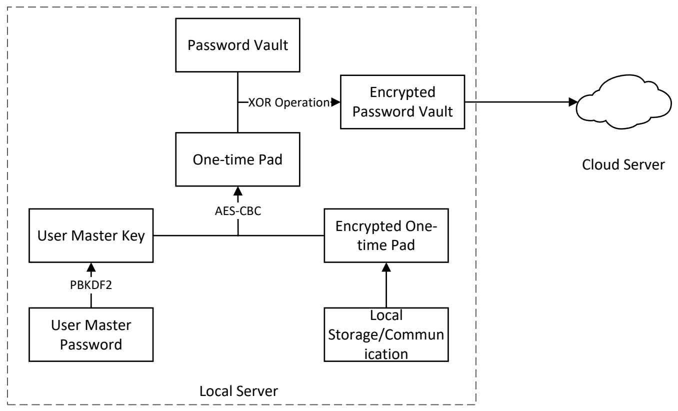

Fig. 4. The process of encrypting the password vault.

The password vault is encrypted with the OTP before being uploaded to the cloud server and must be decrypted at the local server before it can be read or updated. The complete encryption process is illustrated in Figures 4. During encryption, the user's master password (i.e., main password) is first used to derive an encryption key via PBKDF2 [23]. This key decrypts the encrypted OTP using AES-CBC [39], as described earlier in Section V-A. Once the OTP is retrieved, the password vault is encrypted through a byte-wise XOR operation with the OTP. After encryption, the vault is ready to be uploaded to the cloud server. The decryption process mirrors the encryption steps: the OTP is first decrypted using the key derived from the master password, and the same XOR operation is applied to decrypt the password vault using the OTP. The decrypted password vault can then be accessed and updated through the local server, as detailed in Section V-B.

VI. EVALUATION

A. Implementation

We developed a prototype of OTPPM primarily using the Go programming language. The implementation of the local server and cloud server comprises approximately 1,000 lines of Go code. Additionally, the local server includes a web-based user interface, designed with HTML5 and CSS3, enabling access from various user devices. Figure 5 illustrates the graphical user interface (GUI) for adding a new password entry as an example.

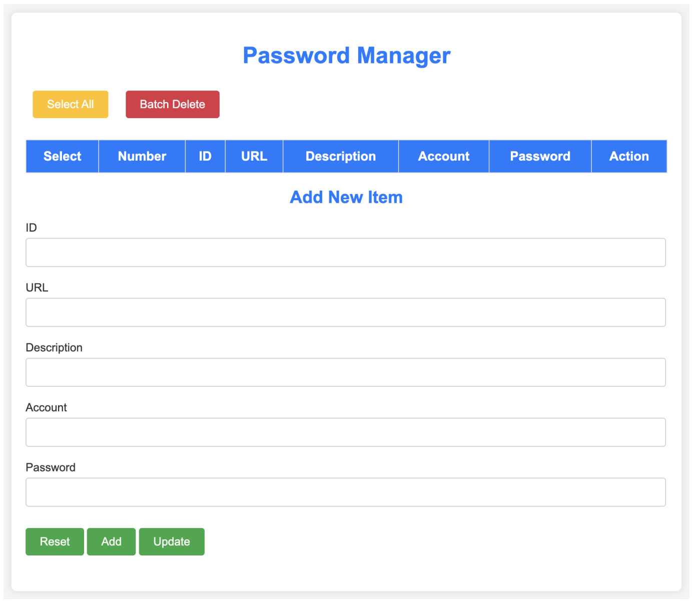

Fig. 5. The web interface of the OTPPM prototype.

B. Experiment Configuration

A computer with an 11th Gen Intel(R) Core(TM) i9-11950H @ 2.60GHz, 2.61 GHz, 32.0 GB RAM, a 64-bit operating system, and a 1024 GB SSD hard drive is used to run the local server and also as the user device. The computer is also used to run extra comparison tests. An ECS server in a public cloud is used as the cloud server.

C. Time Efficiency Assessment

To evaluate the performance of the OTPPM prototype in practical scenarios, we measured the time taken for several operations. Specifically, the login time to the local server refers to the duration from clicking the login button in the web UI to successfully logging in to both the local and cloud servers. The time of transmitting the vault denotes the time taken to transfer the password vault from the cloud server to the local server. The time for opening an entry reflects the duration required to access a selected entry, while the time for updating an entry represents the time needed to update an entry (performed only on the local server). The synchronization time with the remote server indicates the time spent updating the cloud server with changes to the password vault. Synchronization can either be triggered periodically or manually when changes are made to the vault. Each operation was performed ten times to ensure reliable measurements.

The data presented in Table I reveals that the login time, vault transmission time, and synchronization time with the cloud server exhibit slightly greater variability (i.e., standard deviations), and with higher mean values across different trials. These results are likely influenced by factors such as network conditions and server processing capabilities. In contrast, the time required to open an entry is shorter and more consistent, as it involves only interaction with the local server and needs less complex server processing. Most operations are completed within 1 to 2 seconds. An exception is the synchronization time with the cloud server, which may take 2 to 3 seconds, but this occurs only when the password vault is modified and can be handled in the background. These findings suggest that OTPPM offers relatively fast performance.

D. Encryption Performance

To evaluate the performance of OTP encryption in securing password vaults, a critical operation in OTPPM, we conducted a comparative analysis with several widely used encryption algorithms. The password vault size was assumed to range from 50 KB to 500 KB. The encryption algorithms selected for this study included RSA [40], AES [39], 3DES [41], and Blowfish [42]. For each combination of vault size and encryption algorithm, the encryption process was repeated 20 times to calculate the average encryption time, ensuring reliable performance metrics.

The results, presented in Figure 6, demonstrate that OTP encryption achieves the highest encryption speed across all tested data sizes. This outcome aligns with expectations, as OTP encryption relies solely on a simple XOR operation, with its execution time primarily dictated by data I/O overhead. AES encryption exhibits the second-fastest performance, benefiting from its efficient algorithmic design and the optimization support provided by modern hardware, including specialized instruction sets. In contrast, 3DES, which applies the DES encryption process three times, shows comparatively slower speeds than AES. Blowfish performs slightly slower than AES but outpaces 3DES. RSA encryption demonstrates reasonable efficiency with smaller data sizes but suffers a marked decline as data size increases, due to the computational complexity of operations involving large integers. Furthermore, RSA often requires segment-wise encryption for larger datasets, compounding its performance limitations. Overall, while OTP achieves the best performance, the other algorithms, excluding RSA, demonstrate acceptable efficiency. OTP's distinctiveness lies primarily in its absolute security.

E. One-time Pad Usage

Since OTP encryption requires a pad of sufficient length and the OTP must be regenerated or updated once exhausted, we conducted an experiment to analyze OTP consumption over time. In this study, we assume an average password entry size of 100 B and a password vault initially containing 100 entries, consistent with a survey indicating that most individuals have fewer than 100 password-protected accounts [4]. Thus, the initial size of the password vault is 10 KB. We further assume that users update their password vaults at varying frequencies and ratios. For instance, a monthly update frequency corresponds to updating the vault every month,

TABLE I
OPERATION TIME STATISTICS (IN SECONDS) OF 10 TRIALS.

	Login Local Server	Transmit the Vault	Open an Entry	Update an Entry	Synchronize with Cloud Server
Average	1.347	0.659	0.289	0.477	2.598
Standard Deviation	0.523	0.266	0.107	0.179	0.789

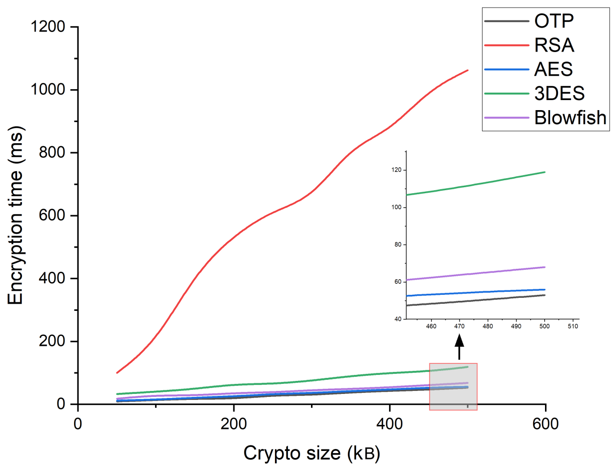

Fig. 6. The time used for different encryption algorithms.

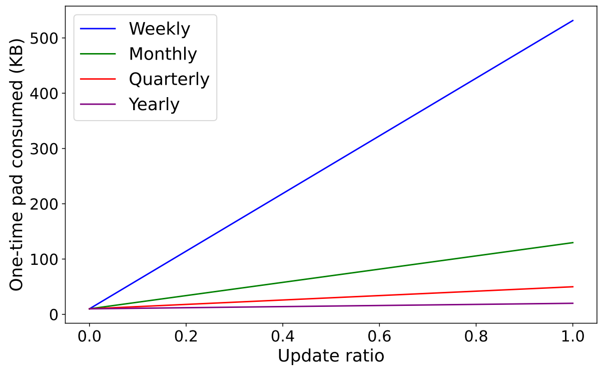

Fig. 7. The one-time pad usage of an initial 10KB password vault under different update frequencies and ratios after a year.

while an update ratio of 0.1 signifies that $0.1 \times 100 = 10$ entries are modified during each update. The cumulative OTP consumption, corresponding to the vault size after one year, is then calculated.

The results are presented in Figure 7. It is evident that higher update frequencies and larger update ratios result in greater OTP consumption. However, the annual OTP usage remains modest. For example, when 50 entries (i.e., an update ratio of 0.5) are updated monthly, the total OTP consumption after one year is only 70 KB, an increase of just 60 KB over the initial 10 KB required for the vault. Under these conditions, an initial OTP size of 1 MB would suffice for approximately 16 years before requiring regeneration or synchronization across all associated local servers. These findings suggest that the OTPPM password manager demands relatively modest OTP lengths for typical use cases, with a moderately sized OTP (e.g., 500 KB) lasting several years under ordinary usage patterns.

VII. SECURITY ANALYSIS

The security of the password vault in OTPPM is ensured during both transmission to and storage within the cloud. This is because the vault is encrypted before transmission to the cloud. We have carefully designed algorithms to ensure that the encryption scheme adheres to the principles of OTP, such as the requirement that each bit of the OTP is used only once. In cases where the OTP length is insufficient, it is incrementally updated or regenerated. Additionally, when users modify or delete entries, the original vault remains unaltered; instead, a new part is appended, ensuring that the updated data is encrypted using a new portion of the OTP.

While we assume that local communication channels (such as LAN and USB) contain no patient attackers, the OTP is encrypted both during local communication and when in local storage. This precaution aims to protect against ordinary attackers or ”honest but curious” users. The encryption is implemented using the standard AES algorithm, which we consider sufficient to guard against threats posed by such users. The local storage of the password vault is also secure since it is encrypted with the OTP.

VIII. RELATED WORK

Password managers, as a crucial security tool, have been the subject of increasingly in-depth research aimed at enhancing both their security and operational convenience [4], [7]–[9], [11], [12], [22], [43], [44]. The research could be roughly divided into below two categories.

The security of password managers In [7] and [10], the authors examine the vault formats and authentication mechanisms employed by existing password managers, revealing that widely recognized encryption algorithms, such as AES, are frequently utilized. Silver et al. [8] analyze vulnerabilities related to the autofill feature in password managers, identifying several successful attack vectors. Li et al. [22] investigate other web-based attacks, including CSRF and XSS, that target password managers. In [43], Wang et al. propose a method where secrets are stored on the user’s smartphone, allowing the device to participate in password generation to mitigate password leakage. However, this approach relies on the availability of a smartphone and assumes the security of not absolute secure cryptographic algorithms, such as SHA-256, which are not impervious to future threats.

Several researchers have explored the use of honey vaults to protect real password vaults. Bojinov et al. [13] propose Kamouflage, a scheme that stores $N-1$ decoy vaults alongside the real vault in the password storage. When an attacker attempts to decrypt the storage using an incorrect master password, they may access a decoy vault. Since the passwords in the decoy vault resemble human-memorable passwords, the attacker would need to attempt online logins to verify them. Websites can leverage these login attempts to alert users promptly. Chatterjee et al. [45] introduce a novel secure encoding scheme that utilizes a natural language encoder to generate more convincing decoys. Cheng et al. [46] propose an incremental update mechanism designed to prevent attackers from gleaning any information from outdated password storage. The honey vault concept is orthogonal to our method and could potentially be integrated into it.

The usability of password managers With the widespread adoption of password managers, numerous researchers have focused on their practical usage and how to enhance operational convenience. Pearman et al. [4] observe that users of standalone password managers tend to prioritize security more than those using built-in password managers. Ray et al. [9] explore password manager adoption among older adults, revealing that this demographic exhibits a heightened mistrust of cloud-based password storage. Our OTP-based password manager addresses this concern by storing the password vault in cloud storage with perfect encryption, potentially alleviating their mistrust. Mayer et al. [11] examine password manager usage within a large university, finding that 77% of participants already use such tools. Munyendo et al. [12] investigate factors that drive users to switch password managers, identifying usability and cost (e.g., subscription fees) as primary motivators. While existing research on the operational convenience of password managers may not directly pertain to security, it significantly informs the design of more user-friendly password management solutions.

IX. Conclusion and Future Work

In this study, we first proposed two methods to estimate the lifetime of conventional encryption algorithms. Then, we designed and implemented a password manager named OTPPM based on one-time pad (OTP) encryption technology, aiming to provide a highly secure and convenient password manager for users. With OTPPM each user's password vault is encrypted with a one-time pad, preventing hackers from accessing passwords even if remote cloud servers are compromised. Through a series of experiments and security analysis, OTPPM demonstrates good performance and clear security advantages. This makes it a compelling choice for users requiring stringent password protection.

Acknowledgments

This research was partially funded by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LY22F020022, the National Natural Science Foundation of China under Grant No. 61902098, and the 2023 Zhejiang Province Science and Technology Innovation Program for College Students (Zhejiang Xinmiao Talents Program), under the general grant number 2023R407 and project number 2023R407027. We also thanks all students which involved in the early development of the prototype, including Pan Zeng.

References

[1] L. Cheng, F. Liu, and D. Yao, "Enterprise data breach: causes, challenges, prevention, and future directions," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 5, p. e1211, 2017.

[2] K. Zurkus, "Google survey finds two in three users reuse passwords," 2019, [Online]. Available: https://www.infosecurity-magazine.com/news/google-survey-finds-two-users/ (current February 2019).

[3] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki *et al.*, "Data breaches, phishing, or malware? understanding the risks of stolen credentials," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1421–1434.

[4] S. Pearman, S. A. Zhang, L. Bauer, N. Christin, and L. F. Cranor, "Why people (don't) use password managers effectively," in *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, 2019, pp. 319–338.

[5] LastPass, "LastPass Password Manager," 2024, [Online]. Available: https://www.lastpass.com/ (current August 2024).

[6] 1Password, "1Password Password Manager," 2024, [Online]. Available: https://1password.com/ (current August 2024).

[7] P. Gasti and K. B. Rasmussen, "On the security of password manager database formats," in *Computer Security–ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings 17.* Springer, 2012, pp. 770–787.

[8] D. Silver, S. Jana, D. Boneh, E. Chen, and C. Jackson, "Password managers: Attacks and defenses," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 449–464.

[9] H. Ray, F. Wolf, R. Kuber, and A. J. Aviv, "Why older adults (don't) use password managers," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 73–90.

[10] F. Yu and H. Yin, "A security analysis of the authentication mechanism of password managers," in *2021 IEEE 21st International Conference on Communication Technology (ICCT).* IEEE, 2021, pp. 865–869.

[11] P. Mayer, C. W. Munyendo, M. L. Mazurek, and A. J. Aviv, "Why users (don't) use password managers at a large educational institution," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1849–1866.

[12] C. W. Munyendo, P. Mayer, and A. J. Aviv, "" i just stopped using one and started using the other": Motivations, techniques, and challenges when switching password managers," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 3123–3137.

[13] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, "Kamouflage: Loss-resistant password management," in *Computer Security–ESORICS 2010: 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings 15.* Springer, 2010, pp. 286–302.

[14] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 657–666.

[15] B. Rothke, "Brute force: Cracking the data encryption standard," 2010, [Online]. Available: https://www.rsaconference.com/library/blog/brute-force-cracking-the-data-encryption-standard (current January 2010).

[16] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science.* Ieee, 1994, pp. 124–134.

[17] D. J. Bernstein and T. Lange, “Post-quantum cryptography,” *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[18] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[19] D. Goodin. (2019) Forum cracks the vintage passwords of ken thompson and other unix pioneers. [Online]. Available: https://arstechnica.com/information-technology/2019/10/forum-cracks-the-vintage-passwords-of-ken-thompson-and-other-unix-pioneers/ (current October 2019).

[20] C. E. Shannon, “Communication theory of secrecy systems,” *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[21] E. L. Kaplan and P. Meier, “Nonparametric estimation from incomplete observations,” *Journal of the American statistical association*, vol. 53, no. 282, pp. 457–481, 1958.

[22] Z. Li, W. He, D. Akhawe, and D. Song, “The {Emperor’s} new password manager: Security analysis of web-based password managers,” in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 465–479.

[23] L. Ertaul, M. Kaur, and V. A. K. R. Gudise, “Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms,” in *Proceedings of the international conference on wireless networks (ICWN)*, 2016, p. 66.

[24] J. A. Halderman, B. Waters, and E. W. Felten, “A convenient method for securely managing passwords,” in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 471–479.

[25] S. M. Bellovin, “Frank miller: Inventor of the one-time pad,” *Cryptologia*, vol. 35, no. 3, pp. 203–222, 2011.

[26] I. Avdonin, M. Budko, M. Budko, V. Grozov, and A. Guirik, “A method of creating perfectly secure data transmission channel between unmanned aerial vehicle and ground control station based on one-time pads,” in *2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2017, pp. 410–413.

[27] G. C. Kessler, “An overview of cryptography,” 2025.

[28] E. Barker, “Recommendation for key management: Part 1 – general (nist special publication 800-57 part 1 revision 5),” National Institute of Standards and Technology, Tech. Rep., 2020.

[29] A. Widodo and B.-S. Yang, “Application of relevance vector machine and survival probability to machine degradation assessment,” *Expert Systems with Applications*, vol. 38, no. 3, pp. 2592–2599, 2011.

[30] F. Anderson, R. Dawid, D. McMillan, and D. García-Cava, “A bayesian reliability analysis exploring the effect of scheduled maintenance on wind turbine time to failure,” *Wind Energy*, vol. 26, no. 9, pp. 879–899, 2023.

[31] E. Barker and A. Roginsky, “Transitioning the use of cryptographic algorithms and key lengths (nist special publication 800-131a revision 2),” National Institute of Standards and Technology, Tech. Rep., 2019.

[32] J. Daemen and V. Rijmen, “Aes proposal: Rijndael,” 1999.

[33] W.-K. Lee, H. J. Seo, S. C. Seo, and S. O. Hwang, “Efficient implementation of aes-ctr and aes-ecb on gpus with applications for high-speed frodokem and exhaustive key search,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 6, pp. 2962–2966, 2022.

[34] C. Tezcan, “Key lengths revisited: Gpu-based brute force cryptanalysis of des, 3des, and present,” *Journal of Systems Architecture*, vol. 124, p. 102402, 2022.

[35] T. S. Perry, “Move over, moore’s law. make way for huang’s law [spectral lines],” *IEEE Spectrum*, vol. 55, no. 5, pp. 7–7, 2018.

[36] R. Merritt, “Heeding huang’s law: Video shows how engineers keep the speedups coming,” 2023, [Online]. Available: https://blogs.nvidia.com/blog/huangs-law-dally-hot-chips/ (current September 2023).

[37] B. Schneier, “Fortuna a secure pseudorandom number generator,” 2024, [Online]. Available: https://www.schneier.com/academic/fortuna/ (current August 2024).

[38] M. Borowski and M. Leśniewicz, “Modern usage of “old” one-time pad,” in *2012 Military Communications and Information Systems Conference (MCC)*. IEEE, 2012, pp. 1–5.

[39] M. Vaidehi and B. J. Rabi, “Design and analysis of aes-cbc mode for high security applications,” in *Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014*. IEEE, 2014, pp. 499–502.

[40] X. Zhou and X. Tang, “Research and implementation of rsa algorithm for encryption and decryption,” in *Proceedings of 2011 6th international forum on strategic technology*, vol. 2. IEEE, 2011, pp. 1118–1121.

[41] R. C. Merkle and M. E. Hellman, “On the security of multiple encryption,” *Communications of the ACM*, vol. 24, no. 7, pp. 465–467, 1981.

[42] B. Schneier, “Description of a new variable-length key, 64-bit block cipher (blowfish),” in *International Workshop on Fast Software Encryption*. Springer, 1993, pp. 191–204.

[43] L. Wang, Y. Li, and K. Sun, “Amnesia: A bilateral generative password manager,” in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 313–322.

[44] S. Zibaei, A. Salehi-Abari, and J. Thorpe, “Dissecting nudges in password managers: simple defaults are powerful,” in *Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023)*, 2023, pp. 211–225.

[45] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, “Cracking-resistant password vaults using natural language encoders,” in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 481–498.

[46] H. Cheng, W. Li, P. Wang, C.-H. Chu, and K. Liang, “Incrementally updateable honey password vaults,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 857–874.