HP-OTP: One-Time Password Scheme Based on Hardened Password

Zixuan Ding, Yihe Duan, *Member, IEEE*, and Ding Wang

***Abstract*—Mobile devices enable the widespread adoption of One-Time Passwords (OTPs) as a crucial component of Two-Factor Authentication (2FA). The impact of OTP leakage is often considered less severe than that of static passwords. However, existing OTP standards, such as S/key, HOTP, and TOTP are vulnerable to key-compromise impersonation attacks, desynchronization attacks, and "small n" attacks. These vulnerabilities allow adversaries to bypass 2FA by exploiting pre-shared symmetric keys once either the device or the server is compromised. Furthermore, asymmetric (chain-based) OTP schemes incur high computational overhead and require periodic initialization, which limits their usability. In this work, we propose HP-OTP, a challenge-response OTP scheme that achieves password hardening without modifying password-OTP implementation architectures. Password hardening on the device side allows the server to store only a non-reversible verification credential used to generate challenges. The device's response OTP integrates the candidate password, possession factor, and a random salt. By redefining OTPs in 2FA from a standalone possession factor to a joint function of the password and the device, HP-OTP prevents adversaries from bypassing the possession factor or exploiting compromises of either the device or the server. Comprehensive security and performance evaluation results demonstrate the security and efficiency of HP-OTP, with verification taking less than 5 milliseconds.**

***Index Terms*—One-time password, password authentication, provable security, two-factor authentication.**

I. Introduction

ONE-TIME passwords (OTPs) are designed for single login to minimize the risk of fraudulent attempts. Typically, OTPs are a series of characters or numbers received or generated by a user's device. The most common form is a six-digit number sent via Short Message Service (SMS) to the mobile device. However, relying solely on SMS for OTP-only authentication introduces significant security risks, as the SMS communication channel has repeatedly been shown to be insecure [1]. Attackers can redirect OTP messages [2] or intercept SMS-based OTPs through SS7 network attacks [3]. The National Institute of Standards and Technology (NIST) discourages transmitting OTPs using SMS since 2017 [4].

In addition to SMS-based distribution, the HMAC-based One-Time Password (HOTP) algorithm [5] and the Time-based One-Time Password (TOTP) algorithm [6] are widely used OTP-generation methods that typically rely on a pre-shared symmetric secret and a synchronized counter or timestamp. However, the parameters of these synchronization mechanisms (e.g., counters and timestamps) are often predictable or observable to an attacker. Once the pre-shared secret is compromised, adversaries can precompute future OTP values. Moreover, adversaries may deliberately desynchronize client and server counters, preventing successful verification.

Service providers typically integrate OTPs with other authentication factors to implement Two-Factor Authentication (2FA). For example, Google Authenticator [7] necessitates a separate additional authentication mechanism, such as password authentication. However, the verification processes of these authentication mechanisms are typically conducted independently, without cryptographic binding. An adversary who compromises the server can obtain the symmetric OTP seeds and perform offline password guessing. Similarly, compromise of the client device or the associated symmetric secret enables adversaries to generate valid OTPs. Consequently, OTP schemes relying on long-term symmetric keys are vulnerable to Key-Compromise Impersonation (KCI) attacks [8].

An approach to mitigating KCI attacks in OTP systems is to replace long-term symmetric secrets with public-key mechanisms or one-way hash chains. In these designs, the server stores a credential generated from the secret, ensuring that a server-side compromise does not allow an adversary to compute valid OTPs. A prominent example of an asymmetric OTP scheme is T/key [9]. In the T/key construction, the user's device pre-computes a hash chain derived from an initial secret seed and indexed by time, while the server stores only the tail nodes of this chain. During authentication, the server computes backward from the received current-time node to the stored tail node for verification. However, such chain-based OTP schemes have inherent limitations. They require strict clock/counter synchronization, impose substantial registration and verification overhead, and must periodically regenerate the entire chain due to its finite length, further increasing the computational burden.

Moreover, merely incorporating OTP as an additional authentication factor does not necessarily enhance the overall security if

Received 13 April 2024; revised 18 November 2025; accepted 21 November 2025. Date of publication 9 December 2025; date of current version 12 March 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62222208, the Tianjin Natural Science Foundation Project under Grant 25JCJQJC00230, and in part by the Fundamental Research Funds for the Central Universities, Nankai University under Grant 63253229. *(Corresponding author: Ding Wang.)*

The authors are with the College of Cryptology and Cyber Science, Nankai University, Tianjin 300350, China, also with the Key Laboratory of Data and Intelligent System Security, Ministry of Education, Nankai University, Tianjin 300350, China, and also with the Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China (e-mail: dingzixuan@mail.nankai.edu.cn; duanyihe@mail.nankai.edu.cn; wangding@nankai.edu.cn).

Digital Object Identifier 10.1109/TDSC.2025.3641892

the verification mechanisms of different factors operate independently. Such a disjointed authentication approach may introduce security gaps, as adversaries can exploit vulnerabilities in either the password or OTP verification process to bypass authentication. For example, adversaries can exploit symmetric keys to generate OTPs, thereby undermining the intended security guarantees of the possession factor and invalidating the OTP as a second factor [10]. To strengthen security, one idea is to integrate the verification of both factors in a manner that reinforces each other's resilience against attacks. Such integration ensures that compromising one factor alone does not significantly weaken the overall authentication security, thereby providing a more robust defense against unauthorized access attempts.

After identifying weaknesses in existing OTP schemes, including compromise risks and loose coupling between authentication factors, we consider an asymmetric OTP architecture based on the existing device-server interaction model used in Password Hardening Service (PHS). Hardened passwords can serve as asymmetric keys for generating OTPs when the device does not leak the salt. Furthermore, password-based mechanisms do not require additional hardware or clock synchronization. We transform the intrinsic logic of OTP in 2FA: existing OTP implies that the user possesses the device, and additionally, the user needs to authenticate another factor. In our scheme, OTP represents the user's possession of knowledge and the device. For 2FA, the authentication factors are not changed, but by constructing an inherent combination, adversaries who have compromised the server cannot directly impersonate the user by exploiting any single factor independently.

A. Motivations

Current OTP applications primarily include SMS OTP [1], HOTP [5], and TOTP [6]. Existing chain-based OTP schemes [9], [11] and challenge-response OTP schemes [12], [13] have seen limited adoption, in part due to design limitations such as high computational overhead and fragile synchronization requirements. SMS OTP is not recommended for continued use in transmitting OTPs because of the channel hijacking attacks [14]. Meanwhile, deployments of HOTP/TOTP may become insecure under device or server compromise, as their long-term symmetric keys can then be exposed.

Taking HOTP/TOTP-based Google Authenticator as a widely adopted representative example, OTP authentication and password authentication operate simultaneously but independently. An adversary who compromises the server can easily generate OTPs and guess passwords offline. Consequently, the device as a second factor is easily bypassed. Additionally, in asymmetric (chain-based) OTP schemes, the device incurs significant computational overhead during the registration phase to generate a hash chain, and this process must be periodically repeated due to the finite length of the hash chain. During authentication, both the device and server require substantial computational resources to repeatedly iterate hash operations.

In addition, existing OTP schemes [5], [6], [9], [11] share a common flaw: they cannot withstand device compromise attacks. We argue that a well-designed challenge–response OTP framework can effectively mitigate these weaknesses. This approach requires only one interaction rounds: the server issues a challenge, and the user device responds with an OTP. The randomness and freshness of challenge generation eliminate the need for synchronization mechanisms. Additionally, the server can verify the device's response without storing any long-term symmetric secret, thereby resisting server-side compromise attacks. The flexibility of the challenge–response OTP framework further allows the device to integrate other authentication factors to counter device compromise attacks.

In this work, we propose HP-OTP, a challenge-response OTP scheme that resolves limitations of existing OTP methods by mitigating SMS interceptability [1], reducing key-exposure risks in HOTP/TOTP [7], [10], avoiding the computational and synchronization burden of hash-chain designs [9], [11], and preserving security even under device or server compromise.

B. Contributions

This work investigates the shortcomings of current OTP schemes and implementations, highlighting the vulnerabilities and limitations that impede their security, particularly the high computational overhead of chain-based designs, the exposure risks of long-term symmetric keys, and the loose coupling that enables authentication-factor bypass. In response, we propose the construction of a novel challenge-response OTP scheme named HP-OTP. By adopting asymmetric password-hardening principles, HP-OTP introduces enhanced security properties and improved resilience against device and server compromise. Specifically, our key contributions are summarized as follows:

Identifying security issues in OTP implementations: We first identify the deficiencies of current OTP schemes, particularly emphasizing the security risks inherent in symmetric secret sharing. The deficiencies include susceptibility to KCI attacks and desynchronization attacks. *For the first time*, we show that several widely deployed OTP implementations still suffer from desynchronization vulnerabilities and scenarios in which the second factor (device) can be bypassed.

Constructing a challenge-response OTP scheme: Without altering the existing implementation architecture, we construct an challenge-response OTP scheme named HP-OTP based on hardened passwords. The device employs an Oblivious Pseudo-Random Function (OPRF) to harden the password into a one-way authentication credential.The credential stored on the server ensures that neither a server compromise nor a device compromise exposes the password or enables the computation of valid OTPs. The server and the device negotiate OTPs based on the hardened password and random numbers.

For the first time, we demonstrate that OTP verification can jointly attest to the user's knowledge of the password and possession of the device, thereby constituting two authentication factors within a unified construction. In addition, by storing an irreversible credential instead of a symmetric seed, HP-OTP prevents compromised servers from bypassing the user's possession factor (e.g., device).

Security proofs and practical evaluations: We enumerate a comprehensive evaluation metric as a reference for OTP

schemes. Furthermore, we offer a comprehensive comparison between current OTP implementations and popular OTP schemes. Our analysis shows that the proposed HP-OTP satisfies the stated security goals and application requirements. Additionally, we provide formal proofs to substantiate the security of OTP and passwords. In real-world implementation scenarios, we evaluate the computational overhead of HP-OTP and related schemes to assess practical feasibility. Security and experimental evaluations demonstrate that HP-OTP is secure and feasible.

C. Related Work

There are four main methods for generating OTPs: server-distributed (e.g., SMS-based OTP [1]), chain-based OTP (e.g., S/key [11], T/key [9]), time/event-synchronized OTP (e.g., TOTP [6], HOTP [5]), and challenge-response-based OTP (e.g., YubiKey challenge-response OTP [12]). We discuss them by category in this section.

Lamport first proposed the concept of using passwords and one-way functions to derive one-time passwords [15]. Subsequently, Haller proposed the first standard, S/key (RFC 1760 [11]) authentication system, for OTP [16]. Servers and clients pre-share seeds and dictionaries, and iterate and verify OTPs through one-way hashing. S/key authentication system utilizes a single secret "seed" to compute a finite OTP hash sequence. The scheme imposes a limit on authentication attempts, requiring a process restart after multiple attempts.

The S/key scheme has some inherent flaws. In the hash chain, OTPs remain valid for an extended period unless used, making them susceptible to exposure or prediction. If the server transmits the counter of each authentication attempt to the client, this exposure of the counter directly reveals the "position" of the current authentication in the hash chain, narrowing the adversary's attack scope and making it easier to target the remaining unused OTPs. Additionally, the S/key scheme is vulnerable to the "small n" attack [17], where adversaries impersonating the server may induce the client to disclose unused OTPs. Furthermore, the use of the same hash function in every iteration of the chain in S/key makes it easier to break through precomputation attacks.

Jin et al. [18] offered a formal demonstration of the Lamport's scheme within the standard model, and utilize TOTP to establish a novel concept termed "Proof of Aliveness (POA)". However, the practicality of the initial construction of PoA is limited by the finite nodes in the hash chain. Yang et al. [19] enhance the PoA construction by linking multiple chains together using a pseudo-random generator chain in the PoA scheme. The enhanced PoA scheme integrates one-time signature schemes into the structure of the second construction to achieve PoA's automatic supplementation.

Eldefrawy et al. [20] extend Lamport's scheme [15] to enhance infiniteness and forwardness, allowing the length of the hash chain to increase without restarting or bootstrapping. Multiple OTPs are generated by utilizing an initial seed and two nested hash chains: one dedicated to seed updates and the other for OTP generation. Registration and verification phases engage in 3-message interactions, based on challenge-response protocols. Users require the server to initialize the protocol and generate OTPs through interrogation.

Kogen et al. extend the Lamport's S/key schemes [15] by integrating time constraints into the passwords and propose the T/key scheme [9]. In T/key, the device generates a time seed and constructs a hash chain based on the seed. The device encodes the tail of the chain as a QR code, which is then scanned by a computer and sent to the authentication server for storage. Subsequently, the device steps backward from the tail to the head of the chain every 30 seconds. At each step, the device displays the current element of the chain, and the user logs in by scanning the code displayed on the computer screen. However, the lifespan of each chain is limited, and excessively long chains can impose a significant computational burden on the device.

Goyal et al. [21] propose partitioning a large value of N into $\lfloor N/R \rfloor$ subcycles of length R to share the cost with the host itself. The authentication process requires three messages of interaction, and users must know the seed s at each login, making encrypting s crucial. Similar to Lamport's scheme [15], reinitialization is still required after N authentications.

In the chain-based OTP schemes, the server only stores the tail node of the hash chain and performs verification by conducting one-way iterations on the received hash node. The advantage lies in its ability to resist server compromise attacks. However, due to the finiteness of the chain, user devices need to interact with the server regularly to regenerate new hash chains. Moreover, since hash iterations are involved, the overhead of resetting and verification is relatively high.

Lei et al. [1] identified various weaknesses in SMS OTP-based authentication mechanisms in modern mobile OS: While user interaction features (e.g., "ask-to-copy" and One-Tap) offer convenience, users struggle to distinguish apps. In addition, the permission prompts fail to fully explain risks, letting malicious applications bypass restrictions. Flaws in modern SMS APIs may enable interception, reading, or exploitation of OTPs, along with potential hash collisions. To address these threats, they proposed a secure API and clarified specifications for backend servers to obtain hash codes.

Peeters et al. proposed the SMS OTP Security (SOS) protocol [22] to tackle the vulnerability of SMS to rerouting attacks by enhancing SMS-based 2FA. The server and device share a symmetric authentication key. When authenticating, the server sends a random string to the device, the device then computes an OTP using this string and the shared key. SOS relies on the pre-shared key established during registration, with the security of this key sharing relying on the Gap Diffie-Hellman (GDH) assumption.

While SMS OTP provides convenience, substantial research has repeatedly demonstrated that the SMS pathway lacks security in most contexts. For instance, threats like SIM swapping exploits [2] and weaknesses in the SS7 protocol [3], [23] have persisted over time and continue to pose significant risks.

In 2005, Open Authentication (OATH) introduced the HMAC-based One-Time Password Algorithm (HOTP) [5] for USB devices and smart cards. The client and server share a counter and a key, and the OTP is generated and truncated by the HMAC-SHA-1 algorithm. Another OTP scheme that does not rely on maintaining counters is based on clock synchronization, known as the Time-based One-Time Password Algorithm (TOTP) [6]. In TOTP, the value T derived from a time reference

and time step replaces the counter used in HOTP. Both the client and server must synchronize on UNIX time to work properly.

Google Authenticator [7], [24] is a widely used software-based authenticator. It supports both HOTP [5] and TOTP [6], where OTPs can be generated even when the mobile device is offline. During registration, the server generates a secret key for the user, the key is then utilized by the user to scan the QR code using the Google Authenticator app. The server and the user device keep the key simultaneously. In HOTP [5], user device and server pre-share a counter C (initialized to 0) for calculating OTP as $HOTP(K, C) = Truncate\ (HMAC - SHA - 1(K, C))$, where K is the pre-shared key. Upon successful verification, the counters in the two parties are incremented accordingly. However, when the counters of the user device and server lose synchronization (exceed the look-ahead window, typically 5), the OTP cannot be verified unless the user resets it. TOTP [6] replaces the counter C with the timestamp. Furthermore, a common threat to both HOTP and TOTP schemes [7], [25], [26] is their vulnerability to device and server compromise, as both the user's device and the server hold symmetric keys.

Ding and Wang proposed a new scheme named HTOTP [10] based on TOTP. They designed a two-factor framework where passwords and OTPs are closely integrated, with OTP seeds derived from passwords and device-stored salts to boost the security of device ownership. By incorporating honeywords [27], [28], the server stores decoy OTP seeds to resist server breaches and online/offline password guessing attacks. This shift from loose to tight coupling between passwords and OTPs addresses device-side leaks and identity forgery, while enhancing the unforgeability and loss resistance of the second factor. In case of a server breach, attackers cannot perform offline password guessing verification since the server only holds decoy OTP seeds generated from honeywords and salted password hashes. Additionally, honeycheckers can detect online guessing attempts and trigger alerts.

Gong et al. [13] introduced a challenge-response OTP authentication scheme utilizing sub-passwords. During the registration phase, the user selects an ID and a static password, which are sent to the server. The server generates N random sub-passwords, a hash function key, and a random permutation function, then sends the sub-passwords and key to the user. In the authentication phase, the device generates a random integer sequence to compute the OTP and sends the ID, static password, and random integer sequence to the server. The server verifies the information, calculates the OTP, and generates a new challenge value to send back to the user. Upon successful login, both the user and the server update their respective sub-passwords based on the random permutation function and increment the counter value. A larger number of precomputed subkeys and rounds still imposes a significant overhead on user devices.

A well-known hardware OTP manufacturer is Yubico [29]. Yubico OTP is a 44-character string generated by a hardware device and verified by the central validation server, YubiCloud. The validation server stores OTP configuration secrets (private ID and AES key) and tracks the usage counters for previously valid OTPs and session usage. When a user submits a Yubico OTP during the authentication process, the application routes the OTP to the validation server, and the server uses the appropriate AES key to decrypt the OTP. After decryption, the validity of the OTP is checked using the private ID, usage counters, and session usage counters. YubiKey also supports OATH-HOTP/TOTP [12]. YubiKey can provide feedback for an HOTP token upon touch. However, since YubiKey does not have an internal clock, it needs to be used with an application (Yubico Authenticator) for TOTP authentication. Hardware implementations still have less-than-ideal characteristics. Carrying a separate device may not be practical for some individuals. Additionally, if the device lacks a proper user interface to input a PIN for OTP generation, it may be vulnerable if stolen.

YubiKey implements two distinct challenge-response mechanisms for authentication. The first, adhering to RFC 2104 [30], employs HMAC-SHA1 to process challenges, supporting up to 512-bit inputs with a 160-bit key, yielding a 48–80 bit response that remains consistent for identical inputs. The second, Yubico OTP, utilizes a fixed 48-bit challenge, combines it with an 80-bit unique device identifier as padding, and secures it with a 128-bit AES key. This approach may produce varying responses for the same challenge due to the device-specific identifier.

Existing representative challenge-response OTP authentication methods are not user-friendly due to excessive interaction rounds, high computational overhead, and additional hardware requirements. Therefore, this work proposes a challenge-response OTP scheme requiring at most one interaction round.

Additionally, there are other types of OTP schemes. Yang et al. introduced a new authentication scheme called Group TOTP (GTOTP) [31]. GTOTP allows users to prove that they are members of an authenticated group without revealing the identities. The proposed framework can transform an asymmetric TOTP scheme into GTOTP. However, it's worth noting that the initial GTOTP structure faces limitations, particularly in dynamic group management. Additionally, each member of the group may encounter substantial storage overhead, posing challenges for scalability and efficiency in large-scale deployments. Sun et al. proposed a dynamic password scheme called TrustOTP [32] based on ARM TrustZone. TrustOTP combines the flexibility of software tokens with the security of hardware tokens, enabling secure implementation of OTP schemes (e.g., S/key [11] HOTP [5], and TOTP [6]) on devices capable of resisting malicious mobile operating systems.

Cao et al. [33] further improved GTOTP's dynamic group management [31], DGTOTP offers an alternative approach by integrating a third-party Registration Authority (RA) to take over storage and computation tasks. The RA constructs a Merkle tree with virtual verification points as its leaves and dynamically distributes independent secret seeds to new group members. Each member's secret seed functions as the key for the chameleon hash, enabling the real verification points of TOTP instances to dynamically link with the Merkle tree leaves during the authentication process. That said, DGTOTP still retains relatively high overhead.

Erdem et al. proposed a cloud-based OTP service called OTPaaS [34], which introduces additional OTP cloud servers to assist authentication. Users first register with the OTP provider to complete key exchange, and then connect to the service

provider. The OTP provider stores a pairing table of users, service providers, and OTPs. When logging in, users input their username-password pair and OTP to the service provider. The service provider first verifies the username-password pair, and then requests OTP verification from the OTP server.

OTPaaS [34] transitions from a two-party interaction between the user and service provider to a three-party interaction involving the user, service provider, and OTP provider, entailing five messages of communication. However, OTP as a second factor may not fully address the risk of compromised servers accessing user passwords. Additionally, the service provider's dependence on the OTP provider for verification introduces an element of trust, which could potentially be exploited by an adversary if the OTP provider's systems are compromised or if there are vulnerabilities in the verification process.

II. Overview

A. Preliminaries

The design of HP-OTP builds upon the concepts and adapted variants of the following technologies.

Oblivious Pseudo-Random Functions (OPRF): The participants in OPRF are two parties: one holding the key k of the PRF $f()$, and the other holding the input x. In OPRF process, the party holding the input learns the result $f_k(x)$, while the other party does not learn anything.

Here, we provide an example constructed based on the principles of Diffie-Hellman problem [35]. Let $\mathbb{G}$ be a group of prime order q, and let H' and H be hash functions mapping to elements in $\mathbb{G}$ and l-bit strings, respectively. $F_k(x) = H(x, (H'(x))^k)$, where the key $k \leftarrow \$\mathbb{Z}_q$. Suppose this function is jointly computed by A and B, where B inputs the key k, A inputs pwd, then only A learns $rwd = F_k(pwd)$. The protocol is straightforward: A sends $a = (H'(pwd))^r$, where $r \leftarrow \$\mathbb{Z}_q$. B returns $b = a^k$, then A calculates $rwd = H(x, b^{1/r})$. Under the random oracle model assuming Gap One-More Diffie-Hellman (OM-DH), the protocol is universally composable [36].

Password Hardening Service (PHS): In the user-server authentication, despite the server providers adhering to the recommended password storage and verification strategy (e.g., NIST [14], OWASP [37]), the servers possess all the parameters needed for password verification, including salted-hashes, salts, secret keys, etc. Consequently, a server compromise allows adversaries to perform offline password guessing attacks.

By introducing an external Password Hardening (PH) server, the service provider interacts with the PH server to verify user passwords rather than monopolizing password verification locally. This separation prevents offline password guessing on the service provider's side.

Here, we illustrate PHS with the example of Everspaugh et al.'s Pythia [38]. Giving a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are of prime order q. Let $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$ be two hash functions. The server holds a secret key $k_w \leftarrow \$\mathbb{Z}_q$, and the function is $F_{k_w}(t, m) = e(H_1(t), H_2(m))^{k_w}$, where t is a tweak and m is a message (password). In registration phase, the client selects $r \leftarrow \$\mathbb{Z}_q$, calculates $e(H_1(t), H_2(m)^r) = e(H_1(t), H_2(m))^r$, and sends the calculation value to the server. The server calculates $e(H_1(t), H_2(m))^{rk_w}$ and sends it to the client. The client obtains $F_{k_w}(t, m) = e(H_1(t), H_2(m))^{k_w}$ by raising $1/r$. In verification phase, the client selects $r' \leftarrow \$\mathbb{Z}_q$, calculates $x \leftarrow H_2(m)^r$, and sends t, x to the server. The server calculates and returns $y \leftarrow e(H_1(t), x)^{k_w}$. The client verifies if $F_{k_w}(t, m) = y^{1/r'}$. Subsequently, numerous PH schemes (Phoenix [39], PO-COM [40], (t, m)-PHE [41], PW-Hero [42]) are proposed. The constraints on password verification based on the mandatory interaction between the server and external PH server remain unchanged.

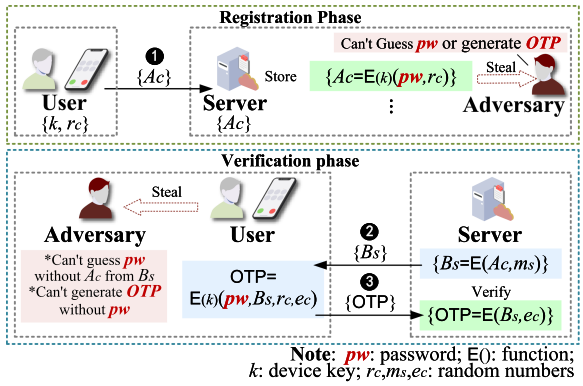

Fig. 1. System Model and Interaction Flow of HP-OTP. During the registration phase, the user device sends the one-way computation result A_c of the password pw, to the server as a verification credential. An adversary compromising the verification credential A_c cannot guess pw without the device key k and the random number r_c. In the verification phase, upon receiving an authentication request, the server generates a challenge B_s based on A_c and a random number m_s. The device generates a response OTP based on the candidate password and a random number e_c in response to the challenge B_s. The server verifies the OTP using the verification credential A_c and the random numbers m_s and e_c, rather than pw. An adversary compromising the user device cannot recover A_c from the challenge B_s to perform offline password guessing. Furthermore, the adversary cannot generate a valid OTP without pw.

B. System Model

The proposed HP-OTP involves two entities, the device D and the server S. The user U provides the device identity DID to the service provider and enters a password pw at D. The registration and verification are carried out through interactions between D and S. D does not store any parameters related to pw or user identity information. The server stores a one-way calculated value of pw, but lacks the necessary parameters to perform the reverse calculation and prevent offline guessing. Fig. 1 shows the system model and interaction flow.

The core objective of PHS [38] is to split password verification authority by introducing a third-party PH server, dividing the client-server password verification process into password hardening and credential verification. Its primary advantage is that the server only stores hardened credentials rather than the original password hash, preventing adversaries from directly performing offline password guessing using the credentials. HP-OTP does not introduce an independent PH server but inherits and reconfigures the core authority splitting logic of PHS, migrating the password hardening function of the PH server to the user device. The device hardens the password pw using a locally generated key k_c and random salt r_c, computing

$A_c = g^{H(pw\|r_c)\cdot k_c}$, which is then sent to the server for storage. If either the server or the device is compromised, adversaries cannot directly use A_c, or the key k_c and salt r_c, to perform offline password guessing, aligning with PHS's core security goal that neither party alone can conduct offline guessing.

OPRF [36] is a collaborative two-party computation function designed to allow the server to validate an input without perceiving it. HP-OTP adopts the core security goal of server obliviousness to the input, achieved through a lightweight cryptographic design. The challenge B_s sent by the server to the device conceals the verification credential A_c using a random number, the user generates the response OTP based on the random number e_c and the password in response to the challenge B_s without learning A_c. The server remains completely oblivious to the core input pw, only obtaining the verification result of the OTP, random number e_c, and challenge B_s through collaboration. HP-OTP's construction closely aligns with OPRF's goal of ensuring that one party does not perceive the input.

The HP-OTP protocol includes algorithms $\Pi : \{\mathsf{Setup}, < \mathcal{D}, \mathcal{S} >_{Reg}, < \mathcal{D}, \mathcal{S} >_{verf}, < \mathcal{D}, \mathcal{S} >_{PwC}\}$, where Π, Reg, $Verf$, and PwC represent the scheme HP-OTP, registration, verification, and password change, respectively. Specifically, HP-OTP consists of the following four stages:

Setup phase: S executes $\Pi.\mathsf{Setup}(1^\lambda)$ based on the security parameter λ and publishes the public parameter pp.

Registration phase: When U enters the device identity DID and password pw separately on S and D, S and D interact to perform $\Pi. < \mathcal{D}(pw), \mathcal{S}(DID) >_{Reg}$. $\Pi. < \mathcal{D}, \mathcal{S} >_{Reg}$ returns a random number $r_c \leftarrow \$\mathbb{Z}_q$ and parameters $\{DID, A_c\}$ to D and S, respectively, where r_c is the password salt, A_c is the password verification credential and is used for OTP negotiation.

Verification phase: When U enters the device identity DID and password pw separately on S and D, S and D interact to perform $\Pi. < \mathcal{D}(pw), \mathcal{S}(DID) >_{verf}$. $\Pi. < \mathcal{D}, \mathcal{S} >_{verf}$ prompts users to enter the OTP calculated by D, and S verifies the OTP.

Password change phase: U enters the old password pw and the new password pw' on D, and D interacts with S to perform $\Pi. < \mathcal{D}(pw, pw'), \mathcal{S} >_{PwC}$ to verify the old password. After verification, $\Pi. < \mathcal{D}, \mathcal{S} >_{PwC}$ returns A'_c and r_c^{new} to S and D respectively, indicating the replacement of the old password with the new one.

III. Security of HP-OTP

A. Security Goals

The evaluation metric proposed by Wang and Wang [43] for two-factor authentication is widely adopted. Erdem and Sandıkkaya [34] make adjustments to the evaluation metric for cloud server OTP. In this study, we generalize the evaluation metric of OTP to enhance its adaptability across a broader spectrum of OTP schemes. The essential security requirements for OTP are as follows:

S1) *Resistance to device loss (server compromise):* An adversary who compromises either the server or the device cannot threaten the security of the password and OTP, or impersonate user login. Only when an adversary simultaneously compromises both parties can she launch attacks against the password and OTP.

S2) *Resistance to pre-computation attacks:* Adversaries are unable to pre-compute password salted-hash or password-based one-way results using rainbow tables or password dictionaries. In the worst-case scenario of global compromise, adversaries should only be able to repetitively compute the process for each guess.

S3) *Resistance to message interception:* Even if adversaries control the channel to intercept transmitted messages during verification, they should not be able to obtain or compute the OTP and guess passwords.

S4) *Forward (backward) security:* The leakage of previous and current OTPs will not affect the subsequent OTP generation and verification. Adversaries cannot use the obtained OTPs to guess upcoming OTPs.

S5) *Resistance to replay attacks:* OTP is disposable, meaning adversaries cannot pass authentication by repeatedly attempting to use the same OTP.

S6) *Two (multi)-factor security:* Authentication is only successful for users who possess both the device and the password factors simultaneously.

S7) *Resistance to Key-Compromise Impersonation (KCI) attack:* We adaptively modify the original KCI attack definition [8] to suit OTP scenarios. Even if the server data is compromised by an adversary, she is unable to impersonate users to pass the server's authentication.

S8) *Resistance to desynchronization attack:* When the user device and server lose synchronization (e.g., counters in HOTP [5] or clocks in TOTP [6]), OTP authentication remains unaffected.

For practicality, ease of deployment, and user-friendliness, we further consider the following application requirements.

C1) *No additional verifier table:* Neither party stores the password or OTP used for verification.

C2) *No clock synchronization:* OTP generation and verification are independent of the clock, the devices do not require additional clock synchronization.

C3) *Password friendly:* The length and complexity of OTPs are user-friendly, facilitating manual input by users in special circumstances.

C4) *Resistance to guessing attacks:* The user's long-term password and OTP should resist guessing attacks, with the adversary's probability of guessing the OTP and password not higher than random guessing.

C5) *Resistance to keyboard/screen recording attacks:* OTP schemes should be able to counter potential threats posed by trojans embedded in login terminals. Screen or keyboard logging attacks will not leak long-term keys, such as long-term passwords, pre-shared keys, etc.

C6) *No additional device expenses:* Devices incur no additional expenses outside the registration and authentication phases, including computational costs and maintaining clock synchronization.

As shown in Table I, we compare the proposed HP-OTP with widely used OTP schemes (Google Authenticator [5], [6], Yubico OTP [29], and YubiKey [12]) and popular schemes (Kogan

TABLE I
COMPARISON OF SECURITY GOALS AND APPLICATION REQUIREMENTS OF OTP SCHEMES

Scheme	Year	Ref.	Round*	S1	S2	S3	S4	S5	S6	S7	S8	C1	C2	C3	C4	C5	C6
The Proposed HP-OTP	Current	\	2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HTOTP (Ding and Wang)	2025	[26]	1	✓	✓	✓	✓	×[1]	✓	✓	✓	✓	×	✓	✓	✓	✓
DGTOTP (Cao et al.)	2024	[33]	2	×	✓	✓	✓	✓	×	✓	×	✓	×	✓	✓	✓	×
SOS (Peeters et al.)	2022	[21]	2	×	×	✓	✓	✓	✓	×	✓	×	✓	✓	✓	✓	✓
NDSS21′ (Lei et al.)	2021	[1]	2	×	✓	×	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓
GTOTP (Yang et al.)	2021	[31]	2	×	✓	✓	✓	✓	×	✓	×	✓	×	✓	✓	✓	×
OTPaaS (Erdem et al.)	2019	[34]	5	×[5]	✓	×	✓	✓	✓	✓	×	✓	✓	✓	✓	×	✓
PoA (Jin et al.)	2019	[17]	1	×	×	✓	✓	✓	×	✓	×	✓	✓	✓	✓	✓	×
T/key (Kogan et al.)	2017	[9]	1	×	✓	✓	✓	✓	×	✓	×	✓	×	×	✓	✓	×
Yubico OTP	2010-	[29]	2	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	×	✓	×	✓
YubiKey (OATH-HOTP)	2011-	[11]	2	✓	×	✓	✓	✓	✓	⊗[3]	×	✓	✓	✓	✓	×	✓
YubiKey (OATH-TOTP)	2011-	[11]	2	✓	×	✓	✓	✓	✓	⊗[3]	×	✓	×	✓	✓	×	×
Google Authenticator (HOTP)	2010-	[5]	1	✓	×	✓	✓	✓	✓	⊗[2]	×	✓	✓	✓	△[4]	×	✓
Google Authenticator (TOTP)	2010-	[6]	1	✓	×	✓	✓	×[1]	✓	⊗[2]	×	✓	×	✓	△[4]	×	×

✓ : Achieving (Resisting) the corresponding goal (attack). ×: Not achieving (Resisting) the corresponding goal (attack). \: Not considered.
⊗ : Achieving (Resisting) with additional conditions. △: Depending on other implementation conditions.
*: Interaction rounds during the OTP verification. [1]: Valid within the time window.
[2]: Original [5] and [6] cannot achieving (resisting), Google Authenticator adds password authentication.
[3]: Yubico Authenticator adds password authentication. [4]: Depends on the password verification strategy of Google Authenticator.
[5]: Service provider is compromised, user passwords leak, and no need for OTP verification.
S1: Resistance to device loss (server compromise). **S2**: Resistance to pre-computation attacks. **S3**: Resistance to message interception.
S4: Forward (backward) security. **S5**: Resistance to replay attacks. **S6**: Two (multi)-factor security.
S7: Resistance to KCI attack. **S8**: Resistance to desynchronization attack. **C1**: No additional verifier table.
C2: No clock synchronization. **C3**: Password friendly. **C4**: Resistance to guessing attacks.
C5: Resistance to keyboard/screen recording attacks. **C6**: No additional device expenses.

et al.'s T/key [9], Erdem et al.'s OTPaaS [34], and Yang et al.'s GTOTP [31]). The comparison results indicate that HP-OTP has advantages in both security goals and application requirements.

B. Formal Security Definition

An adversary of OTP system has the following capabilities and limitations: (1) When an adversary breaches either the user device or the server, they can obtain the registration records and keys. However, simultaneous compromise of both the device and the server is not feasible[1]; (2) By breaching a device, the adversary can recover recently transmitted or received messages, leveraging cached TLS session keys and communication data [46], [47]; (3) Equipped with a password dictionary, the adversary can continuously attack password-based security mechanisms; (4) The adversary may collect used OTPs; (5) Referring to the Dolev-Yao model [48], the adversary can launch active attacks based on channel hijacking, such as Man-In-The-Middle (MITM) attacks.

We employ adversarial games to formally describe the capabilities of the adversary and the security aspects of HP-OTP. Including semantic security of OTPs, security of password storage, and security of password verification. The security and game definitions are outlined as follows:

OTP semantic security: In this scenario, suppose the adversary compromises the terminal or server to steal OTPs, and this theft may occur multiple times. We discuss the possibility of the adversary extracting passwords and identities from OTPs. Intuitively, we aim to demonstrate through a proof by contradiction whether the adversary can differentiate between OTPs and random strings based on existing information without performing verification protocol.

This game is labeled as OTP-SS and involves two stages, involving interactions between the device D and the server S. Initially, in the primary phase, D and S play the roles of challengers, denoted as $\mathcal{R}_D$ and $\mathcal{R}_S$, respectively. In the subsequent phase, the malicious server S is identified as the adversary $\mathcal{A}$. The mechanics of the game are outlined as follows:

- Setup: $\mathcal{R}_S$ can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi.<\mathcal{R}_D, \mathcal{R}_S>_X, X \in \{Reg, Verf\}\}$. $\mathcal{A}$ generates a password pw and sends it to $\mathcal{R}_D$.
- Challenge: $\mathcal{R}_D$ interacts with $\mathcal{R}_S$ to perform $\Pi.<\mathcal{R}_D, \mathcal{R}_S>_{Reg}$ using the password pw. Afterward, $\mathcal{A}$ inherits all parameters from $\mathcal{R}_S$ but is prohibited from verifying OTP. User U initiates a login request at $\mathcal{A}$, entering pw on $\mathcal{R}_D$. $\mathcal{R}_D$ interacts with $\mathcal{A}$ to perform $\Pi.<\mathcal{R}_D, \mathcal{A}>_{Verf}$, and generates OTP OTP_b according to the verification protocol, where $b \leftarrow \$\{0,1\}$. Simultaneously, $\mathcal{R}_D$ generates a random string OTP_{1-b} that conforms to OTP rules. The device sends $\{OTP_b, OTP_{1-b}\}$ to $\mathcal{A}$.
- Output: $\mathcal{A}$ possesses the password pw, registration record A_c, and device identity DID but cannot execute OTP verification. Based on the above information, $\mathcal{A}$ outputs a decision bit b', $b'=b$ represents $\mathcal{A}$ wins the game. The advantage is defined as

$$Adv^{\mathcal{A}}_{OTP-SS}(\lambda) = \left|\Pr[b'=b] - \frac{1}{2}\right|. \tag{1}$$

Definition 1 (OTP Semantic Security): If for an OTP scheme and a Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that $Adv^{\mathcal{A}}_{OTP-SS}(\lambda) \leq negl(\lambda)$, the scheme is said to be OTP semantic secure.

[1]This is because, as noted in studies [44], [45], password-based cryptographic protocols become insecure when both entities are compromised, leaving them susceptible to offline password guessing attacks. This represents a typical vulnerability in password-based cryptographic protocols.

Password storage security: This scenario simulates the real-life situation of storage leakage. Assuming the adversary compromises the server to obtain user registration records, we discuss the possibility of the adversary guessing the user's password based on the records.

This game is labeled as OTP-PSS, involving interactions between the device D and the malicious server S. D and S play the roles of challenger and adversary, denoted as $\mathcal{R}$ and $\mathcal{A}$, respectively. The mechanics of the game are outlined as follows:

- Setup: $\mathcal{R}$ can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi. < \mathcal{R}, \mathcal{A} >_X, X \in \{Reg, Verf\}\}$. $\mathcal{A}$ generates and sends two different passwords pw_0 and pw_1 to $\mathcal{R}$.
- Challenge: $\mathcal{R}$ selects $b \leftarrow \$\{0,1\}$ and executes $\Pi. < \mathcal{R}, \mathcal{A} >_{Reg}$ based on pw_b.
- Output: $\mathcal{A}$ possesses the passwords $\{pw_0, pw_1\}$, registration record A_c, device identity DID, and can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi. < \mathcal{R}, \mathcal{A} >_{Verf}\}$. Then, $\mathcal{A}$ outputs a decision bit b', $b' = b$ represents $\mathcal{A}$ wins the game. The advantage is defined as

$$Adv^{\mathcal{A}}_{OTP-PSS}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \tag{2}$$

Definition 2 (Password Storage Security): If for an OTP scheme and a Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that $Adv^{\mathcal{A}}_{OTP-PSS}(\lambda) \leq negl(\lambda)$, then this scheme can be considered as password storage security.

Password verification security: This scenario simulates the situation where either the device or the server is compromised, and the adversary executes the verification process to guess the password. We discuss the possibility of adversary guessing the user's password by interacting with the honest party. These two games are denoted as OTP-PVS1 and OTP-PVS2, respectively.

OTP-PVS1 involves two stages, the interactions between the device D and the server S. In the primary phase, D and S play the roles of challengers, denoted as $\mathcal{R}_D$ and $\mathcal{R}_S$, respectively. In the subsequent phase, the adversary $\mathcal{A}$ is assumed to control the device D. The mechanics of the game are outlined as follows:

- Setup: $\mathcal{R}_S$ can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi. < \mathcal{R}_D, \mathcal{R}_S >_X, X \in \{Reg, Verf\}\}$. $\mathcal{A}$ generates and sends two different passwords pw_0 and pw_1 to $\mathcal{R}_D$.
- Challenge: $\mathcal{R}_D$ selects $b \leftarrow \$\{0,1\}$ and executes $\Pi. < \mathcal{R}_D, \mathcal{R}_S >_{Reg}$ based on pw_b.
- Output: $\mathcal{A}$ possesses $\{pw_0, pw_1\}$, device identity DID, and can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi. < \mathcal{R}_S, \mathcal{A} >_{Verf}\}$. Based on the above information, $\mathcal{A}$ outputs a decision bit b', $b' = b$ represents $\mathcal{A}$ wins the game. The advantage is defined as

$$Adv^{\mathcal{A}}_{OTP-PVS1}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \tag{3}$$

OTP-PVS2 involves two stages, the interactions between the device D and the server S. In the primary phase, D and S play the roles of challengers, denoted as $\mathcal{R}_D$ and $\mathcal{R}_S$, respectively. In the subsequent phase, the malicious server S is identified as the adversary $\mathcal{A}$. The mechanics of the game are outlined as follows:

TABLE II
SYMBOL TABLE

Symbol	Definition
D	User device
S	Server
U	User
DID	Unique device identifier
pw/pw'	Registered password/Candidate password
k_c	Device secret key
r_c	Random salt generated by D
q_c	Salted hash of the password: $q_c = H(pw\|\|r_c)$
A_c	Password verification credential stored in S: $A_c = g^{q_c \cdot k_c}$
A'_c	Device-side generated verification credential: $A'_c = g^{q'_c \cdot k_c}$
B_s	Encrypted parameter sent by S to D: $B_s = A_c^{m_s}$
C_s	Device-side computed parameter: $C_s = B_s^{t_c \cdot d'_c}$
C'_s	Server-side computed parameter: $C'_s = g^{t'_c \cdot m_s}$
u	OTP component: $u = F_{OTP}(e_c, C_s)$
OTP	One-time password: $OTP = [e_c\|\|u]$ (concatenation of e_c and u)
$H()$	Hash function
$F_{OTP}()$	OTP generation function (hash + truncation)

- Setup: $\mathcal{R}_S$ can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi. < \mathcal{R}_D, \mathcal{R}_S >_X, X \in \{Reg, Verf\}\}$. $\mathcal{A}$ generates and sends two different passwords pw_0 and pw_1 to $\mathcal{R}_D$.
- Challenge: $\mathcal{R}_D$ selects $b \leftarrow \$\{0,1\}$ and executes $\Pi. < \mathcal{R}_D, \mathcal{R}_S >_{Reg}$ based on pw_b.
- Output: $\mathcal{A}$ possesses $\{pw_0, pw_1\}$, registration record A_c, device identity DID, and can access the oracle $\mathbb{O} \leftarrow \{\Pi.\mathsf{Setup}, \Pi. < \mathcal{R}_D, \mathcal{A} >_{Verf}\}$. Based on the above information, $\mathcal{A}$ outputs a decision bit b', $b' = b$ represents $\mathcal{A}$ wins. The advantage is defined as

$$Adv^{\mathcal{A}}_{OTP-PVS2}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \tag{4}$$

Definition 3 (Password Verification Security): If for an OTP scheme and a Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that $Adv^{\mathcal{A}}_{OTP-PVS1}(\lambda) \leq negl(\lambda)$ and $Adv^{\mathcal{A}}_{OTP-PVS2}(\lambda) \leq negl(\lambda)$, then this scheme can be considered as password verification secure.

IV. THE PROPOSED HP-OTP

A. Construction Overview

In this section, we briefly outline the approach to constructing a secure scheme, providing a framework for understanding the rationale behind the design details discussed later. Table II consolidates key symbols used in the protocol.

The construction of password storage and verification follows the principles of distribution and OPRF, ensuring that the password salted-value is pseudo-random for both service providers and device storage. In the construction of password storage during registration phase, the device D first sends $A_c \leftarrow g^{q_c k_c}$ to the server S, where k_c is the device's secret key, $q_c \leftarrow H(pw \parallel r_c)$, pw is the password, and $r_c \leftarrow \$\mathbb{Z}_q$ is a random number. According to the Elliptic Curve Discrete Logarithm Problem (ECDLP), it is infeasible for S to recover $q_c k_c$ from A_c. Moreover, since r_c is unknown and H is one-way, S cannot mount an offline dictionary attack to recover pw from q_c. Therefore, although S stores the password verification credential A_c, S cannot learn any password-related information.

During the password verification phase, the server S generates a new random number $m_s \leftarrow \$\mathbb{Z}_q$ and computes $B_s \leftarrow A_c^{m_s}$. Here, m_s can be considered analogous to the role of k in OPRF, encrypting parameter A_c to prevent the device from obtaining. The encryption prevents adversaries from impersonating users to obtain the password verification credential A_c for offline password guessing in the event of compromising the device. On the device side, the device calculates $C_s \leftarrow B_s^{(q_c' k_c)^{-1}}$ to eliminate $q_c k_c$ from the exponent of B_s, the elimination operation is analogous to cancelling r by multiplying with $1/r$ in OPRF constructions. This means that when the user inputs the correct password pw' during password verification, both the device-side and server-side can simultaneously know the parameter g^{m_s}. Without disclosing the password to each other, the simultaneous possession of the same one-time secret parameter implies implicit mutual identity verification, serving as the foundation for the subsequent successful authentication of the password.

The HP-OTP interactive verification architecture adheres to Password-Hardened Services (PHS) principles by leveraging the user device as an external password hardening component. This design distributes the responsibility for password verification between the device and the server, ensuring that neither party solely controls the authentication process. The interactive verification mitigates the risk of offline password guessing associated with compromising either the server S or the device D.

In the registration phase, D sends $A_c \leftarrow g^{q_c k_c}$ to S, where k_c is the device's secret key, $q_c \leftarrow H(pw \parallel r_c)$, pw is the password, and D stores the random number $r_c \leftarrow \$\mathbb{Z}_q$. Correspondingly, D provides the random number r_c and the key k_c for subsequent password verification. In the verification phase, S sends the password salted-value $B_s \leftarrow A_c^{m_s}$ processed with a random number m_s to D (acted as an external password hardening server). D generates new parameters $A_c' \leftarrow g^{H(pw' \| r_c) k_c}$ based on the user's newly entered candidate password pw'. However, the device cannot recover A_c from B_s to determine whether the candidate password pw' is correct. D generates an OTP based on locally generated A_c' and B_s sent by S. Then S compares A_c' on the device side with A_c stored during registration, thereby verifying the correctness of the user's login password pw'. Throughout, S never acquires the original password pw or the candidate password pw', and S lacks the advantage of offline password guessing without knowing the salt r_c.

In terms of user usability, HP-OTP explicitly optimizes OTP format to align with user familiarity while maintaining security, addressing the deviation from user-friendly codes. The OTP generation function F_{OTP} is designed to produce 6–8 digit OTPs, which are easy for users to input and consistent with mainstream systems like Google Authenticator, by adjusting the truncation step in its hash-based construction, or to support longer OTPs sent directly by the device. In addition, the OTP $[e_c \| u]$ (where e_c is a random number and u is the hash-derived value) is internally structured but presented to users as a single contiguous string. This avoids complicating input with segmented formats.

B. Formal Description

Setup phase: Consider a cyclic group $\mathbb{G}$ of prime order q, and let g be a generator of $\mathbb{G}$. $H : \{0,1\}^* \rightarrow \{0,1\}^l \in \mathbb{Z}_q$ represents a one-way hash function, $F_{OTP} : (\{0,1\}^l, \{0,1\}^*) \rightarrow \{0,1\}^{l_{OTP}}$ represents a one-way OTP generation function. The user device holds the secret key k_c and a unique identifier DID. The devices and servers are denoted as D and S, respectively. $\{H(), F_{OTP}(), \mathbb{G}, g\}$ are public parameters pp.

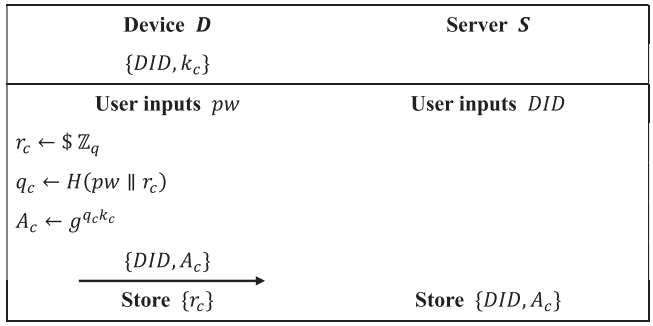

Fig. 2. The main operations of registration phase.

Registration phase: The registration mechanism of HP-OTP aims to ensure that the server verifies the identity in authentication processes. The difference from Password-Authenticated Key Exchange (PAKE) lies in the registration mechanism of HP-OTP, HP-OTP aligns with the current user habits of OTP. Typically, users register online when using a network service for the first time, making it challenging to achieve the ideally secure registration channel required by PAKE protocols. In this scenario, we make adaptive adjustments to the registration of OTP authentication, ensuring password security without relying on additional security assumptions and eliminating multiple interaction rounds. The registration phase of the proposed OTP authentication protocol is highly efficient and concise. The user inputs the device identity and password separately into the service provider and personal device. The device processes the password through salted-hashing and encryption, and the server stores the one-way value. The specific registration process is as follows:

The user inputs the device identity DID and the password pw on the server S and the device D, respectively. Subsequently, D and S interact to register. S establishes a communication channel with D. D randomly selects a number r_c from the set $\mathbb{Z}_q$, then computes the hash value $q_c \leftarrow H(pw \parallel r_c)$. D maps the hashed value into the group using the secret key k_c, obtaining a one-way encryption result $A_c \leftarrow g^{q_c k_c}$. Finally, D sends $\{DID, A_c\}$ to the server and locally stores the random number r_c. The server receives and stores $\{DID, A_c\}$. The steps are shown in Fig. 2.

After registration, both the device and the server are unable to offline guess the password. The server stores $A_c \leftarrow g^{H(pw \| r_c) k_c}$, where the random number r_c and the device key k_c are unavailable, so the server cannot independently reproduce the calculation of A_c to verify guesses. On the device side, even if the device is compromised, only the salt value r_c can be obtained. In addition, online guessing by an adversary interacting with the server is restricted by the server interaction frequency. After the registration phase, the device D and the server S store $\{DID, r_c, k_c\}$ and $\{DID, A_c\}$, respectively.

Verification phase. STEP 1: On logging in, the user enters the device identity DID and a candidate password pw' on the

server S and the device D, respectively. When receiving the user's login request, S first generates a random number m_s and encrypts A_c using m_s as $B_s \leftarrow A_c^{m_s}$, where $B_s = g^{m_s q_c k_c}$, k_c is the device secret key, $q_c = H(pw \parallel r_c)$ is the password salted-hash. S sends B_s to D. Based on the difficulty of ECDLP, it is impossible for D to recover A_c from B_s without knowing the random number m_s. Therefore, even if D is captured by the adversary, she cannot obtain the password salted-hash from S for offline guessing.

STEP 2: On the device side, the device D first calculates salted-hash $q_c' \leftarrow H(pw' \parallel r_c)$ based on the inputted candidate password pw', where r_c is the random number generated in registration phase. Then D computes the inverse of $q_c' k_c$ as $d_c' \leftarrow (q_c' k_c)^{-1}$ to eliminate the corresponding parameters in the exponent of B_s. D calculates $A_c' \leftarrow g^{q_c' k_c}$. Then D generates an l-bit random number e_c and combines it with q_c' to calculate the hash value $t_c \leftarrow H(A_c' \parallel e_c)$ and t_c is used as an exponent to calculate $C_s \leftarrow B_s^{t_c d_c'}$, where $C_s = g^{m_s t_c q_c k_c (q_c' k_c)^{-1}}$. If the user-entered candidate password pw' matches the registered password pw, then $C_s = g^{m_s t_c}$. But D cannot determine whether $q_c'^{-1}$ and q_c are eliminated from $B_s^{t_c d_c'}$ because the random number m_s is unknown. Therefore, the device D cannot determine whether the candidate password pw' is correct. D executes the OTP generation function to obtain $u \leftarrow F_{OTP}(e_c, C_s)$. After that, the OTP $[e_c \parallel u]$ is returned to the server either via device transmission or user input.

The key step $d_c' \leftarrow (q_c' k_c)^{-1}$ is designed to enable implicit verification of password while maintaining the asymmetry of the scheme, preventing identity forgery even if the server or device is compromised. This eliminates the password-related term q_c and device key k_c from the exponent, leaving only the server's random number m_s and the hash value t_c.

The server cannot compute d_c' because it lacks r_c (stored in the device) and k_c (device secret key), so it cannot forge C_s even if compromised. Only when $pw' = pw$ (i.e., $q_c' = q_c$) will C_s (device-side) match $C_s' = g^{t_c' \cdot m_s}$ (server-side, where $t_c' = H(A_c \parallel e_c)$), ensuring the OTP generated by $F_{OTP}(e_c, C_s)$ is valid. If an adversary compromises the server, they cannot obtain r_c or k_c to compute d_c', so they cannot generate a valid C_s to forge the OTP. Similarly, a compromised device cannot obtain m_s (server's random number) to reverse-engineer the server's verification logic.

STEP 3: The server S converts the OTP to a binary number $[e_c \parallel u]_2 = OTP$, extracting e_c. S calculates the hash value $t_c' \leftarrow H(A_c \parallel e_c)$ and uses t_c' as an exponent to calculate $C_s' = g^{t_c' m_s}$. When the candidate password pw' is equal to the registered password pw, according to the Diffie-Hellman theorem, C_s on the device side is equal to C_s' on the server side. In other words, the result u' obtained by the server executing $u' \leftarrow F_{OTP}(e_c, C_s')$ should be equal to u in the inputted OTP. Equality implies that the user password is correct, resulting in a successful login. The verification steps are shown in Fig. 3.

Password change phase: After successfully logging in, the user can send a request to the server to update the password. User inputs the original password pw' and the updated password pw^{new} on the device D. D first calculates the salted-hash $q_c' \leftarrow H(pw' \parallel r_c)$ of the original password pw' and the encrypted value $A_c' \leftarrow g^{q_c' k_c}$. Afterward, D generates a completely new random number $r_c^{new} \leftarrow \$\mathbb{Z}_q$ and repeats the above calculation steps for the new password pw^{new}, obtaining $q_c^{new} \leftarrow H(pw^{new} \parallel r_c^{new})$ and $A_c^{new} \leftarrow g^{q_c^{new} k_c}$. D sends $\{DID, A_c', A_c^{new}\}$ to the server S.

Upon receiving $\{DID, A_c', A_c^{new}\}$, S first verifies whether $A_c' = A_c$. If this condition holds, S replaces A_c with A_c^{new} and sends a confirmation to D. D replaces r_c with r_c^{new}. The steps are shown in Fig. 4.

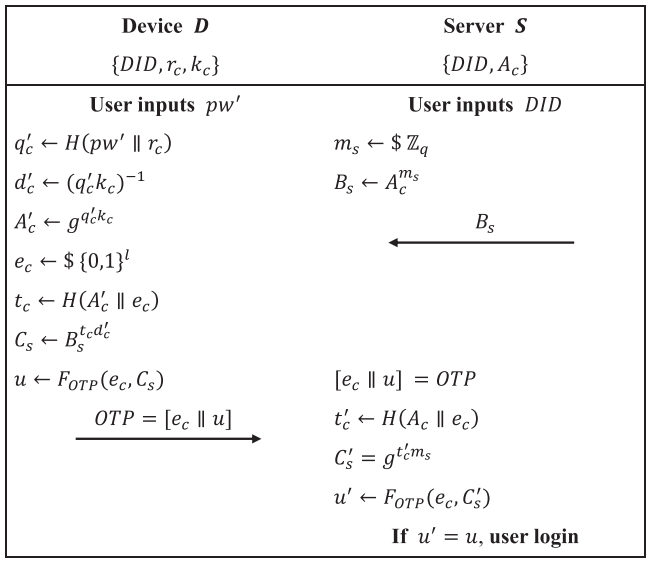

Fig. 3. The main operations of verification phase.

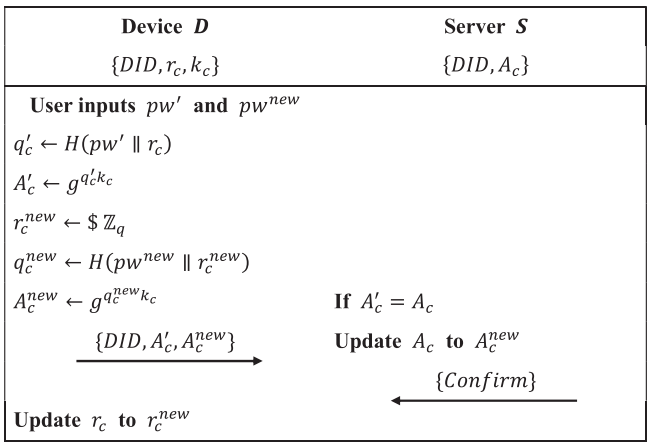

Fig. 4. The main operations of password change phase.

C. Informal Security Analysis

Here, we employ an informal, heuristic approach to discuss the security goals of HP-OTP. The definitions of these security goals can be found in Section III-A. We explore various aspects of security, including resistance to known attacks, resilience to compromise scenarios, and the ability to ensure confidentiality and authenticity in OTP verification processes.

Fig. 5. Typical device compromise attacks. The attacks typically span hardware, system, and software levels and proceed via entry points such as rooted OS, malicious apps, or physical theft, ultimately allowing local extraction or generation of valid OTPs.

Resistance to device compromise: Device compromise attacks have long been regarded as an inherent limitation of OTP schemes, schemes cannot effectively prevent OTP or key leakage once the device is compromised. As illustrated in Fig. 5, the adversary's capabilities typically span hardware, system, and software levels. A compromised device serves as a breach point for the adversary to achieve OTPs through diverse means.

In the event of device compromise (where k_c and r_c are extracted), HP-OTP prevents offline password guessing through the interactive verification. An adversary with k_c and r_c can generate candidate salted hashes $q_c' = H(pw'\|r_c)$ and compute $A_c' = g^{q_c' \cdot k_c}$, but cannot verify these candidates offline. To validate a guessed password pw', the adversary must interact with the server to obtain $B_s = A_c^{m_s}$, where m_s is a server-generated nonce. The adversary cannot perform offline verification of the guessed password without knowing m_s.

Assuming the adversary obtains the user's device with the intention of guessing the user's password online or forging user to login. Here we discuss the probability of guessing the password or logging in on a single attempt. The device stores $\{DID, r_c, k_c\}$, where DID, r_c, and k_c respectively represent the device identity, random salt, and device secret key. The process of guessing the password online is as follows: First, the adversary inputs the candidate password pw'. Then, the device and the server interact to perform the verification protocol of HP-OTP. Finally, the adversary enters the OTP generated by the device on the login terminal.

Under a single random guess, the probability that a forged OTP is accepted is $1/2^{l_{OTP}}$, where l_{OTP} is the bit length of the OTP, and the probability of guessing the password is $C' \cdot q^{s'}$, where C' and s' are the linear regression parameters of the password distribution $\mathbb{D}$ [43], q is the number of guessing. Adversaries who compromise the device have no advantage in guessing OTPs compared to random guessing.

Resistance to server compromise: Assuming the adversary compromises the server to obtain the stored data $\{DID, A_c\}$, where $A_c \leftarrow g^{H(pw\|r_c)k_c}$, the random number r_c and the device key k_c are unavailable for the adversary. Therefore, the adversary cannot verify the guessed password pw' using the available information. HP-OTP's resistance to server-side offline guessing is comparable to Password Hardening Services (PHS) like Pythia [38], but with distinct trust trade-offs: Pythia relies on a third-party PH server to split password verification, preventing the service provider from performing offline guessing. However, it introduces trust in the PH server and increases latency due to multi-party interactions. HP-OTP eliminates third-party dependence by leveraging the user's device as a distributed hardening component. The server stores only $A_c = g^{H(pw\|r_c)\cdot k_c}$ and lacks r_c and k_c, making offline password recovery impossible even if the server is compromised. This matches Pythia's security guarantee against server-side offline guessing but reduces infrastructure complexity. The security boundary of HP-OTP is consistent with that of PHS: an adversary cannot compromise the PH server and the device simultaneously.

Resistance to pre-computation attacks: Assuming the adversary possesses a password dictionary and pre-computes the hash values of passwords for the comparisons after compromise. The salted hash of the password is $q_c \leftarrow H(pw \parallel r_c)$, processed by the device as $A_c \leftarrow g^{q_c k_c}$ and stored by the server, where r_c and k_c represent the random hash salt and the device secret key, respectively. In this case, the adversary must capture the device in advance to pre-computate A_c', and then interact or compromise the server to verify, precomputation of password candidates becomes infeasible.

Resistance to message interception: Message interception poses a serious threat to one-factor SMS-based OTP security. In the proposed HP-OTP, OTPs are not distributed by the server or generated from a pre-shared seed. Instead, they are negotiated between the device and the server based on hardened asymmetric password and random numbers. In the verification protocol, only message $B_s \leftarrow A_c^{m_s} = g^{m_s q_c k_c}$ is transmitted from the server to the device. Assuming the adversary intercepts this message, where k_c is the device secret key, $q_c \leftarrow H(pw \parallel r_c)$ is the password salted-hash. It is difficult for the adversary to recover q_c from B_s due to ECDLP and without knowing the device secret key k_c, random salt r_c, and random number m_s. On the device side, OTP is formed by the combination of e_c and u, where e_c is randomly generated by the device, and $u \leftarrow F_{OTP}(e_c, C_s)$, $C_s \leftarrow B_s^{t_c d_c}$, $d_c \leftarrow (q_c k_c)^{-1}$, and $t_c \leftarrow H(A_c \parallel e_c)$, where F_{OTP} is the OTP generation function. Similarly, even if the adversary intercepts B_s, they cannot generate OTP without knowing k_c and r_c.

Forward (backward) security: The generation of OTP during verification is based on the hardened password and the principles of Diffie–Hellman key exchange. For the adversary, the random numbers are unpredictable. Even if the adversary obtain the current and previous OTPs and long-term keys, she cannot infer the subsequent OTPs. Therefore, HP-OTP possesses both forward and backward security.

Resistance to replay attacks: In each round of verification, the device and server generate unique random numbers for OTP generation, ensuring the pseudorandomness of OTPs. The probability of having the same OTP in different rounds is $1/2^{l_{OTP}}$, where $l_{OTP} = 24$.

Two-factor security: OTP is formed by the combination of e_c and u, where $u \leftarrow F_{OTP}(e_c, C_s), C_s \leftarrow B_s^{t_c d_c}, d_c \leftarrow (q_c k_c)^{-1}$, and $t_c \leftarrow H(A_c \parallel e_c)$. F_{OTP} is the OTP generation function, k_c is the device secret key, $q_c = H(pw \parallel r_c)$ is the password salted-hash, $A_c \leftarrow g^{q_c k_c}$, $B_s \leftarrow A_c^{m_s} = g^{m_s q_c k_c}$, e_c and m_s are

randomly selected by the device and the server, respectively. Therefore, the adversary must simultaneously possess both the device and password pw to log in.

Resistance to Key-Compromise Impersonation (KCI) attack: Assuming the adversary compromises the server to obtain the stored $\{DID, A_c\}$ and attempts to forge users to log in, where $A_c = g^{q_c k_c}$, k_c is the device secret key, $q_c = H(pw \parallel r_c)$ is the password salted-hash. During the verification phase, the server generates $B_s \leftarrow A_c^{m_s}$, and sends it to the user's device, where m_s is a random number. If the adversary does not possess the device or hijack the channel to obtain B_s, she cannot calculate the OTP. Here, we enhance the adversary's capability to enable them to eavesdrop on the channel. For the adversary, the value $t_c \leftarrow H(A'_c \parallel e_c)$ in the verification protocol is computable, where e_c is randomly selected. However, when e_c is determined, the adversary cannot compute $C_s \leftarrow B_s^{t_c d'_c} = g^{m_s t_c q_c k_c (q'_c k_c)^{-1}} = g^{m_s t_c}$ without knowing the correct password salted-hash q'_c and device secret key k_c. Therefore, the adversary cannot generate the correct OTP, and HP-OTP resists KCI attacks.

Resistance to desynchronization attack: The proposed HP-OTP does not rely on counter or clock synchronization, thus resistant to desynchronization attacks.

Resistance to active attacks: HP-OTP's design reduces the risk of active attacks (e.g., MITM attack, replay attack, message modification, forgery attack, DOS attack) by leveraging its challenge-response mechanism and asymmetric cryptographic key:

Tamper Resistance of B_s: During verification, the server sends $B_s \leftarrow A_c^{m_s}$ (where m_s is a fresh random number) to the device. Even if an adversary tampers with B_s, the device generates $C_s \leftarrow B_s^{t_c \cdot d'_c}$ based on the user's password pw' and stored r_c, k_c. A forged B_s would result in an invalid C_s, and thus an incorrect OTP $u \leftarrow F_{OTP}(e_c, C_s)$, which the server would reject during verification (since $u' \neq u$).

Unforgeability of OTP: The OTP $[e_c \| u]$ is derived from e_c (device-generated randomness) and C_s (a function of B_s, pw', r_c, and k_c). An adversary lacks r_c and k_c to compute valid C_s or u, even if they intercept B_s or e_c.

Implicit Authentication: The consistency between C_s (device-side) and C'_s (server-side) relies on the Diffie-Hellman-like key agreement, which is only satisfied if $pw' = pw$ and messages remain unaltered. This ensures MITM attackers cannot forge valid OTPs without compromising both parties' secrets.

Mitigating DoS attacks: Each authentication round uses new nonce. Blocking or delaying one round does not affect subsequent attempts, as fresh randomness renders old messages obsolete, preventing persistent disruption via replay.

V. Security Proof

A. OTP Semantic Security

Theorem 1: Let $Adv_{F_{OTP}}$ and $Adv^{\mathcal{A}}_{ECDLP}$ represent the advantage of $\mathcal{A}$ breaking the one-way hash function F_{OTP} and hard problem ECDLP, respectively. The advantage of $\mathcal{A}$ breaking OTP semantic security can be denoted as:

$$Adv^{\mathcal{A}}_{OTP-SS} \leq Adv^{\mathcal{A}}_{F_{OTP}} \cdot Adv^{\mathcal{A}}_{ECDLP} \tag{5}$$

Proof: The theorem's validity is established through the formalization of games, where the advantage lies in discerning the bit b from the set (0,1). Consequently, the notion of OTP semantic security arises when $\mathcal{A}$ cannot differentiate between b and the existing information (e.g., passwords, record).

- $Game_0$: Identical to game OTP-SS, refer to Section III-B.
- $Game_1$: The adversary $\mathcal{A}$ randomly selects OTP_b. This game is identical to $\mathbf{Game_0}$, unless $\mathcal{A}$ can break the one-way hash function F_{OTP} and recover e_c and C_s from $u \leftarrow F_{OTP}(e_c, C_s)$, the probability of the adversary breaking F_{OTP} is $Adv^{\mathcal{A}}_{F_{OTP}}$.
- $Game_2$: This game is identical to $\mathbf{Game_1}$, unless the adversary breaks ECDLP, and recovers m_s and t_c from $C_s = g^{m_s t_c}$. Thus, the relationship between $t_c = H(A_c \parallel e_c)$ and A_c is easily discernible.
- $Game_3$: This game is identical to $\mathbf{Game_2}$, except it models $\mathcal{A}$'s attempt to distinguish the OTP from a random string (equivalent to inferring registration records from the OTP). For $\mathcal{A}$ to succeed in $\mathbf{Game_3}$, it must simultaneously overcome the challenges in $\mathbf{Game_1}$ (breaking F_{OTP}) and $\mathbf{Game_2}$ (solving ECDLP). Since these two hard problems are computationally independent, the advantage of $\mathcal{A}$ in distinguishing the OTP (i.e., breaking OTP semantic security) is bounded by the product of their individual advantages: $Adv^{\mathcal{A}}_{OTP-SS} \leq Adv^{\mathcal{A}}_{F_{OTP}} \cdot Adv^{\mathcal{A}}_{ECDLP}$. □

B. Password Storage Security

Theorem 2: Let $Adv^{\mathcal{A}}_{ECDLP}$ represents the advantage of $\mathcal{A}$ breaking the hard problem ECDLP. l_{r_c} and l_{k_c} respectively represent the bit lengths of the random number r_c and the device secret key k_c. The advantage of $\mathcal{A}$ breaking password storage security can be denoted as:

$$Adv^{\mathcal{A}}_{OTP-PSS} \leq Adv^{\mathcal{A}}_{ECDLP} / 2^{l_{r_c} + l_{k_c}} \tag{6}$$

Proof: The theorem's validity is established through the formalization of games, where the advantage lies in discerning the bit b from the set (0,1). Consequently, the notion of password storage security arises when $\mathcal{A}$ cannot differentiate between b using the existing information (e.g., passwords, identities, registration record).

- $Game_0$: Identical to game OTP-PSS in Section III-B.
- $Game_1$: The adversary $\mathcal{A}$ randomly selects pw_b. This game is identical to $\mathbf{Game_0}$, unless $\mathcal{A}$ can break ECDLP to recover $q_c k_c$ from the registration record A_c, the advantage of a Probabilistic Polynomial-Time (PPT) adversary breaking ECDLP is $Adv^{\mathcal{A}}_{ECDLP}$.
- $Game_2$: This game is identical to $\mathbf{Game_1}$, unless the adversary guesses the device secret key k_c to recover the salted-hash $q_c = H(pw \parallel r_c)$ from $q_c k_c$. The advantage of the adversary guessing k_c is $1/2^{l_{k_c}}$, where l_{k_c} is the bit length of the device secret key k_c.
- $Game_3$: This game is identical to $\mathbf{Game_2}$, unless the adversary guesses the random number r_c. The advantage of the adversary guessing r_c is $1/2^{l_{r_c}}$, where l_{r_c} is the bit length of the random number r_c. In conclusion, the

adversary's advantage to distinguish pw_0 and pw_1 is no greater than $Adv^{\mathcal{A}}_{ECDLP}/2^{l_{r_c}+l_{k_c}}$. □

C. Password Verification Security

Here, we discuss games OTP-PVS1 and OTP-PVS2, respectively, to demonstrate password verification security.

Theorem 3: Let $Adv^{\mathcal{A}}_{ECDLP}$ represents the advantage of $\mathcal{A}$ breaking the hard problem ECDLP. l_{m_s} represents the bit length of the random number m_s. The advantage of $\mathcal{A}$ breaking password verification security after capturing the device can be denoted as:

$$Adv^{\mathcal{A}}_{OTP-PVS1} \leq Adv^{\mathcal{A}}_{ECDLP}/2^{l_{m_s}} \tag{7}$$

Proof: The theorem's validity is established through the formalization of games, where the advantage lies in discerning the bit b from the set (0,1). Consequently, the notion of password storage security arises when $\mathcal{A}$ cannot differentiate between b based on capturing the device.

- $Game_0$: Identical to game OTP-PVS1 in Section III-B.
- $Game_1$: The adversary $\mathcal{A}$ randomly selects pw_b. This game is identical to $\mathbf{Game_0}$, unless $\mathcal{A}$ can break ECDLP to recover $q_c k_c m_s$ from the parameter B_s sent from the server, the advantage of the adversary breaking ECDLP is $Adv^{\mathcal{A}}_{ECDLP}$.
- $Game_2$: This game is identical to $\mathbf{Game_1}$, unless the adversary guesses the random number m_s to recover the salted-hash $q_c = H(pw \parallel r_c)$ from $q_c k_c m_s$, where the device secret key k_c is available for $\mathcal{A}$. The advantage of the adversary guessing m_s is $1/2^{l_{m_s}}$, where l_{m_s} is the bit length of the random number m_s.
- $Game_3$: This game is identical to $\mathbf{Game_2}$, focusing on $\mathcal{A}$'s ultimate goal: distinguishing between pw_0 and pw_1. For $\mathcal{A}$ to succeed in this task, it must first overcome the challenges in $\mathbf{Game_1}$ (solving ECDLP to recover $q_c k_c m_s$) and then in $\mathbf{Game_2}$ (guessing m_s to extract q_c). Since these two steps are computationally independent, the total advantage of $\mathcal{A}$ in distinguishing pw_0 and pw_1 is bounded by the product of their individual advantages: $Adv^{\mathcal{A}}_{OTP-PVS1} \leq Adv^{\mathcal{A}}_{ECDLP}/2^{l_{m_s}}$. □

Theorem 4: Let $Adv^{\mathcal{A}}_{ECDLP}$ represents the advantage of $\mathcal{A}$ breaking the hard problem ECDLP. l_{r_c} and l_{k_c} respectively represent the bit lengths of the random number r_c and the device secret key k_c. The advantage of $\mathcal{A}$ breaking password verification security after compromising the server can be denoted as:

$$Adv^{\mathcal{A}}_{OTP-PVS2} \leq Adv^{\mathcal{A}}_{ECDLP}/2^{l_{r_c}+l_{k_c}} \tag{8}$$

Proof: The theorem's validity is proven by formalizing games, where the advantage is distinguishing bit b from $\{0, 1\}$. Password storage security holds when $\mathcal{A}$ cannot discern b despite compromising the server.

- $Game_0$: Identical to game OTP-PVS2 in Section III-B.
- $Game_1$: The adversary $\mathcal{A}$ randomly selects pw_b. This game is identical to $\mathbf{Game_0}$, unless $\mathcal{A}$ can break ECDLP to recover $q_c k_c$ from the parameter A_c sent from the server, the advantage of the adversary breaking ECDLP is $Adv^{\mathcal{A}}_{ECDLP}$.
- $Game_2$: This game is identical to $\mathbf{Game_1}$, unless the adversary guesses the device secret key k_c to recover the salted-hash $q_c = H(pw \parallel r_c)$ from $q_c k_c$. The advantage of the adversary guessing k_c is $1/2^{l_{k_c}}$, where l_{k_c} is the bit length of the device secret key k_c.
- $Game_3$: This game is identical to $\mathbf{Game_2}$, unless the adversary guesses the random number r_c. The advantage of the adversary guessing r_c is $1/2^{l_{r_c}}$, where l_{r_c} is the bit length of the random number r_c. In conclusion, the adversary's advantage to distinguish pw_0 and pw_1 is no greater than $Adv^{\mathcal{A}}_{ECDLP}/2^{l_{r_c}+l_{k_c}}$. □

VI. Evaluation

This section examines the computational costs of the proposed HP-OTP, with source code available at https://github.com/dingzixuan8899/. Additionally, we broaden the analysis by comparing expenses with related schemes within the same environment. By employing this comparative methodology, we aim to conduct a thorough assessment that illuminates the efficiency of HP-OTP in practical contexts.

A. Evaluation Environment

During assessment, the server S operates on Ubuntu (Version: 22.04.3 LTS) and is equipped with an Intel Xeon Gold 6320R @2.10 GHz processor, 256 GB of RAM, and an NVIDIA RTX 3090 GPU. The device D runs on an Android device (Version: Android 13) with a MediaTek Dimensity 8100 processor, 8 GB of RAM, and a Mali-G610 MC6 GPU. We encode the server-side code using Python and utilize WebSocket for registration. WebSocket is a protocol allowing full-duplex communication over TCP. During verification, the key distinction from related OTP schemes [5], [6], [9], [12], [29], [31], [34] is that HP-OTP eliminates the need for a secure channel.

In the aforementioned experimental setup, we measured the time costs incurred by both the device and the server for basic cryptographic operations, serving as practical benchmarks for readers. The server-side large integer operations such as inversion and exponentiation are based on the Python library gmpy2. Cryptographic operations such as symmetric encryption/decryption, hashing, and random number generation are based on the Python library pycrypto. The script on the device side is developed in Python and utilizes libraries (gmpy2, pycrypto, and websockets) for cryptographic and interaction operations. For the OTP generation function F_{OTP} involved in the proposed HP-OTP, we construct it by referencing the hash combined with truncation operation in HOTP [5]. We perform 1000 iterations for the cryptographic operations involved and obtain the average values, as shown in Table IV.

Table III presents a comparison of the proposed HP-OTP and related schemes in terms of interaction message and overhead. In contrast to Table I, we selected OTP schemes that provide specific algorithms. Please note that here, HOTP [5] and TOTP [6] differ from Google and YubiKey in Table I. The OTP algorithms provided by RFC standards are the original constructions, which have flaws such as pre-computation attacks, KCI attack, and

TABLE III
COMPARISON OF OVERHEADS OF OTP SCHEMES

Scheme	Registration			Verification			Storage	
	Round	Device	Server	Round	Device	Server	Device	Server
HOTP [5]	1	\	\	1	$H_{ZQ}(149.96\mu s)$	$H_{ZQ}(1.26\mu s)$	$L_C + L_H$	$L_C + L_H$
TOTP [6]	1	\	\	1	$H_{ZQ}(149.96\mu s)$	$H_{ZQ}(1.26\mu s)$	$L_C + L_T$	$L_C + L_T$
T/key [9]	1	$kH_{ZQ}(149.96k\ \mu s)$	\	1	$t_1 H_{ZQ}(149.96t_1\ \mu s)$	$(k-t_1)H_{ZQ}(1.26t_2\ \mu s)$	$L_{ID} + L_{ZQ} + L_T$	$L_{ID} + L_H + L_T$
GTOTP [31]	2	$> 2H_{ZQ} + 2E_G(2.03ms)$	$> Mt + Enc + 2H_{ZQ} + 2E_G(1.87ms)$	2	\	$> Mt + H_{ZQ} + Dec(2.10ms)$	$> L_{En} + 2H_{ZQ}$	L_{Mt}
DGTOTP [33]	2	$8H_{ZQ} + Enc(3.57ms)$	$Mt + 7dH_{ZQ}(1.45{+}1.05d\ ms)$	2	\	$> Mt + 4H_{ZQ} + Dec(2.70ms)$	L_{En}	$L_{Mt} + 2dL_H$
SOS [21]	6	$2E_G + H_{ZQ} + F_{OTP}(2.09ms)$	$2E_G + 3H_{ZQ}(0.07ms)$	3	$H_{ZQ} + F_{OTP}(301.56\mu s)$	$H_{ZQ} + F_{OTP}(2.6\mu s)$	$L_H + 2L_{ID}$	$L_H + 2L_{ID}$
HTOTP [26]	1	$wH_{ZQ} + 2E_G(1.73{+}0.15w\ ms)$	$2E_G(64.3\mu s)$	1	$H_{ZQ} + F_{OTP}(301.56\mu s)$	$wF_{OTP}(1.34w\mu s)$	$L_{HZ} + L_T$	$wL_{HZ} + L_T$
HP-OTP	1	$H_{ZQ} + E_G(1.01ms)$	\	2	$2H_{ZQ} + 2E_G + F_{OTP}(2.18ms)$	$H_{ZQ} + E_G + F_{OTP}(34.75\mu s)$	L_{ZQ}	$L_{ID} + L_G$

H_{ZQ}: Hash from $\{0,1\}^*$ to $\mathbb{Z}_q$. L_H : Length of H_{ZQ}. L_G: Length of the element in $\mathbb{G}$. F_{OTP}: OTP function from $\{0,1\}^{(8\|*)}$ to $\{0,1\}^{16}$.
E_G : Exponentiation in $\mathbb{G}$. L_{ID}: Length of identity. L_{ZQ}: Length of the element in $\mathbb{Z}_q$. \: Not considered.
Enc : Symmetric encryption. Dec: Symmetric decryption. Mt: Merkle tree generation. L_{Mt}: Length of Merkle tree.
L_{En} : Length of encryption block. L_T : Length of timestamp. L_C: Length of counter. w: The size of honeywords [26].

TABLE IV
EXECUTION TIME OF BASIC OPERATIONS

Server		Device	
Operation	Time (μs)	Operation	Time (μs)
Hash ($\mathbb{Z}_q$)	1.26	*Hash* ($\mathbb{Z}_q$)	149.96
MUL ($\mathbb{Z}_q$)	0.41	*MUL* ($\mathbb{Z}_q$)	25.32
INV ($\mathbb{G}$)	5.41	*INV* ($\mathbb{G}$)	433.12
EXP ($\mathbb{G}$)	32.15	*EXP* ($\mathbb{G}$)	863.21
$F_{OTP}()$	1.34	$F_{OTP}()$	151.60
$Enc()$	354.29	$Enc()$	2371.94
$Dec()$	643.85	$Dec()$	4596.64
$Mtree^*$	1451.54		

Hash ($\mathbb{Z}_q$): Hash from $\{0,1\}^*$ to $\mathbb{Z}_q$. *MUL* ($\mathbb{Z}_q$): Multiplication in $\mathbb{Z}_q$.
INV ($\mathbb{G}$) : Invert in $\mathbb{G}$. *EXP* ($\mathbb{G}$) : Exponentiation in $\mathbb{G}$.
$F_{OTP}()$: OTP function. $Mtree$: Merkle tree generation [31].

desynchronization attack, and therefore cannot be directly applied. Google and YubiKey enhance the original OTP architecture by introducing additional authentication factors (e.g., passwords and USB hardware), thus achieving a more secure and practical 2FA.

B. Comparative Evaluation

As shown in Table III, in the T/key scheme [9], during the registration phase, the device constructs a hash chain based on timestamps and sends the tail node to the server. The device's overhead is related to the length (nodes) of the chain, denoted as k nodes. During the verification phase, the device needs to calculate the interval from the registration time based on the current time and divide it to determine the current corresponding hash chain node. Similarly, the server calculates the chain between the current time and the tail node. T/Key reduces real-time computation via hash chains but requires periodic resets and only supports single-device-factor authentication.

The GTOTP scheme [31] requires 2 messages for registration and authentication. To enable OTP authentication within the group, the server needs to maintain a Merkle tree and validate the OTP of leaf nodes (representing users) based on the Merkle tree. Neither of the related schemes constitutes two/multi-factor authentication, thus additional authentication strategies are needed. In addition, T/key [9] and GTOTP [31] are vulnerable to desynchronization attacks and require time synchronization, along with additional computational overhead for preparation. DGTOTP [33] is an optimized version of GTOTP for dynamic group management, introducing a third-party RA to share storage and computation tasks. However, it retains 2.70 ms for device verification and requires group-level key management, limiting its applicability to individual user scenarios.

HTOTP [10] is a TOTP-based improved scheme that integrates passwords and OTPs through honeywords, storing decoy OTP seeds on the server to resist server compromise attacks. However, it relies on clock synchronization, failing if clock drift exceeds the time window.

We list the main computational overheads of OTP schemes and provide an approximate calculation of the computational time overhead in the same environment. We omit low overhead operations, so the actual execution overhead is slightly higher than the theoretical values in Table III. The proposed HP-OTP does not rely on additional time synchronization. Instead, HP-OTP is based on the interaction between the server and the device, and the user's hardened asymmetric password to achieve OTP authentication. When the user sends a request from the login terminal, the server sends an encrypted random parameter to the user's device. The device and the server negotiate the OTP based on the random values and the password, making the OTP one-time and unpredictable.

The device overhead of HP-OTP in the verification phase (2.18 ms) is higher than that of symmetric architecture schemes such as HOTP/TOTP (0.15 ms). The core reason lies in the necessary computational overhead brought by security enhancement. The specific trade-off relationships are as follows:

- *Security cost of asymmetric architecture:* HOTP/ TOTP rely on pre-shared symmetric keys and require one hash operation to generate OTPs, resulting in simple operation logic but flaws: the server stores symmetric keys, and once the server is compromised, adversaries can generate OTPs (S7. KCI attack in Section III-A). In contrast, HP-OTP adopts a challenge-response asymmetric design, which requires an exponentiation operation and one hash operation. This design aims to integrate password hardening logic into the device side, enabling the server to only store a non-reversible verification credential, thereby resisting server and device compromise attacks (S1. Resistance to device loss (server compromise) in Section III-A).
- *Overhead of integration of password and device:* In other OTP schemes (e.g., HOTP/TOTP [5], [6], T/key [9], SOS [22]), password verification and OTP verification are independent processes, allowing adversaries to crack them separately. HP-OTP implicitly embeds password correctness verification into the OTP generation process through

TABLE V
PERFORMANCE EVALUATION FOR RESOURCE-CONSTRAINED DEVICES AND CROSS-OS PLATFORMS

Device Type	Hardware/OS Configuration	Registration Time (μs)	Verification Time (μs)
Mid-range Android (Baseline)	MediaTek Dimensity 8100, Android 13, 8GB RAM	1012	2187
Low-end Android	Snapdragon 450, Android 11, 2GB RAM	1896	4173
Mid-range iOS	Apple A15 Bionic, iOS 17, 4GB RAM	935	1720
Low-end iOS	Apple A13 Bionic, iOS 14, 3GB RAM	1457	2915
Resource-Constrained IoT	ESP32-C3 (RISC-V, 160MHz), FreeRTOS	4782	9463

the inverse operation—only when the correct password is entered will the device-side C_s and server-side C'_s satisfy Diffie-Hellman consistency ($C_s = g^{m_s \cdot t_c}$). This integration requires an additional hash operation and one group inverse operation, but its benefit is to achieve 2FA security where "cracking a single factor is ineffective" (S6. Two (multi)-factor security in Section III-A).

- *Efficiency compensation for clock real-time randomness independence schemes:* T/Key [9] and HTOTP [10] rely on clock synchronization, which reduces real-time computation but requires periodic hash chain reset (T/Key) or clock calibration (HTOTP), resulting in hidden overhead (e.g., T/Key takes approximately 150 ms to generate a 1000-node hash chain). By generating random numbers m_s (server) and e_c (device) in real time, HP-OTP eliminates synchronization dependence (C2. No clock synchronization in Section III-A), replacing periodic high-overhead operations with minimal real-time computation (2.18 ms), leading to better long-term deployment efficiency.

The above analysis and experimental data show that the total device-side overhead of HP-OTP (registration 1.01 ms + verification 2.18 ms) results in no noticeable latency for users. Compared to T/Key's periodic hash chain reset [9], OTPaaS's 5-message interaction [34], and the security threats of symmetric OTP schemes [5], [6], HP-OTP maintains lower overhead while resisting to device/server compromises and forgery attacks.

C. Diversity and Real-World Feasibility

In addition to the analysis of theoretical overhead in Section VI-B, we supplement multi-dimensional tests covering diverse devices, network conditions, attack simulations, and usability metrics. We also present a mobile prototype implementation to verify real-world deployability.

We extend tests beyond the initial MediaTek Dimensity 8100 to include resource-constrained devices and cross-OS platforms. Experimental settings and results are presented in Table V. HP-OTP maintains feasibility even on low-end devices. A low-end Android device (Snapdragon 450) completes verification in 4.17 ms, approximately 50% slower than the baseline but still within the millisecond range. IoT devices (ESP32-C3) exhibit higher overhead but remain practical for scenarios with relaxed latency requirements, with verification taking about 3 ms. iOS devices leverage hardware acceleration, achieving slightly better performance than the Android baseline, demonstrating strong cross-OS adaptability.

We use tool combinations to simulate network scenarios (4 G/5 G/WiFi) and measure end-to-end latency, ensuring consistency with real-world network characteristics. We customize a timestamp logging module in both server-side and device-side code to record three time points: T_1: Server sends the challenge parameter B_s. T_2: Device receives B_s and starts OTP generation. End-to-end latency is calculated as $T_2 - T_1$, and we collect 1000 valid samples per scenario to eliminate random errors.

In addition, we test attack-like perturbations to simulate real-world instability. Given that OTP authentication relies on the integrity and timeliness of critical message transmission, we focus on message drop attacks—a common real-world threat induced by poor network conditions or adversarial jamming. The design of our attack simulation is tightly aligned with HP-OTP's protocol logic in Section IV-B and real-world network characteristics, ensuring the validity of performance evaluations. HP-OTP's verification phase involves only two core messages: the challenge $B_s = A_c^{m_s}$ and the OTP $[e_c \| u]$. To implement controlled message drops, we use the Linux tc utility to inject random packet losses into the communication channel. On the device side, we synchronize network constraints via ADB to ensure consistent drop rates across both ends of the link.

We select a 20–30% average packet loss rate [49] to mimic bursty drops. This range reflects the upper bound of packet loss in harsh but common network scenarios [50]. The tests show that T/Key [9] (relying on time-synchronized hash chains) experiences a 24% failure rate when packet loss exceeds 15%, as clock drift breaks hash chain node matching. HOTP/TOTP [5], [6] suffer desynchronization when counter/clock updates are lost, leading to failure at 10–15% loss. By testing at 20–30% loss, the tests explicitly demonstrate HP-OTP's advantage in resisting desynchronization, a core security goal (*S8.Resistance to desynchronization attack*) in Section III-A.

To handle message drops, HP-OTP implements a limited retry mechanism with an exponential backoff schedule (100 ms, 200 ms, 400 ms, 3 maximum attempts). This design is chosen to balance user experience and network efficiency, with justifications rooted in protocol characteristics and usability requirements in Section III-A (*C2.No clock synchronization*). Even with three retries, the wait time remains acceptable for real-time login scenarios. Exponential growth reduces the risk of retry storms, which are a common issue with fixed backoff, where repeated immediate retries exacerbate network congestion and increase packet loss. This aligns with HP-OTP's goal of minimal infrastructure dependence.

Unlike HOTP/TOTP [5], [6], which risk counter desynchronization with retries, HP-OTP's stateless verification means each retry generates a fresh m_s generated by server and e_c generated by device. The exponential backoff thus only affects timing, not protocol correctness, ensuring retries do not introduce new security risks (e.g., desynchronization). This attack simulation

TABLE VI
AUTHENTICATION PERFORMANCE METRICS OF HP-OTP UNDER VARIABLE NETWORK LATENCY AND MESSAGE DROP ATTACK SCENARIOS

Network Scenario	Latency (ms)	Packet Loss Rate	Authentication Failure Rate	Retry Success Rate	Attack Simulation (AFR/RSR)
5G	17.25	0-2%	0.1%	100%	7.1%/95.3%
WiFi (LAN)	1.45	0-2%	0%	100%	10.1%/94.7%
WiFi (WAN)	86.31	0-4%	0.7%	99.8%	12.3%/87.9%
4G	53.17	0-2%	0.4%	99.9%	11.6%/88.2%

design incorporates targeted message drops, realistic loss rates, and user-centric backoff to ensures accurate evaluation of HP-OTP's resilience in real-world unstable networks. The results (87.9%-95.3% retry success rate) confirm that HP-OTP maintains practical usability even under adversarial or harsh network conditions, outperforming existing schemes like T/Key [9], HOTP, and OTPaaS [34].

Key metrics are shown in Table VI. In a normal network environment, the Authentication Failure Rate (AFR) is positively correlated with the packet loss rate, while the Retry Success Rate (RSR) exceeds 99.8%. Under simulated message hijacking attack scenarios, the AFR ranges from 7.1% to 12.3%, with the RSR between 88.2% and 95.3%.

VII. CONCLUSION

We propose an asymmetric OTP scheme named HP-OTP to address the deficiencies in existing OTP implementations, including reliance on symmetric seeds, vulnerability under device or server compromise, and loose coupling between authentication factors. Meanwhile, we list comprehensive security goals and application requirements as a reference for OTP implementations. HP-OTP alters the fundamental logic of OTP in 2FA, shifting from OTP merely signifying the device to encompassing both password and device. Devices and servers negotiate OTP based on hardened passwords and random numbers. We analyze the security properties of HP-OTP and formally prove the OTP security, password storage and verification security. In real-world testing, the overall time overhead for HP-OTP verification remains below 5 milliseconds.

REFERENCES

[1] Z. Lei, Y. Nan, Y. Fratantonio, and A. Bianchi, "On the insecurity of SMS one-time password messages against local attackers in modern mobile devices," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[2] J. Zhao, B. Ding, Y. Guo, Z. Tan, and S. Lu, "Securesim: Rethinking authentication and access control for SIM/eSIM," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*,2021, pp. 451–464.

[3] S. Khandelwal, "Real-world SS7 attack - hackers are stealing money from bank accounts," Apr. 2017. [Online]. Available: https://thehackernews.com/2017/05/ss7-vulnerability-bank-hacking.html

[4] NIST, "NIST special publication 800-63B: Digital identity guidelines," Jun. 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-63b.pdf

[5] D. M'Raihi, D. M'Raihi, F. Hoornaert, D. Naccache, M. Bellare, and O. Ranen, "HOTP: An HMAC-Based one-time password algorithm," RFC 4226, Dec. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4226

[6] D. M'Raihi, J. Rydell, M. Pei, and S. Machani, "TOTP: Time-based one-time password algorithm," RFC 6238, May 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6238

[7] Google, "Google authenticator opensource," Apr. 2021. [Online]. Available: https://github.com/google/google-authenticator

[8] D. Abbasinezhad-Mood, S. M. Mazinani, M. Nikooghadam, and A. Ostad-Sharif, "Efficient provably-secure dynamic ID-based authenticated key agreement scheme with enhanced security provision," *IEEE Trans. Depend. Secur. Comput.*, vol. 19, no. 2, pp. 1227–1238, Mar./Apr. 2022.

[9] D. Kogan, N. Manohar, and D. Boneh, "T/key: Second-factor authentication from secure hash chains," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017, pp. 983–999.

[10] Z. Ding and D. Wang, "HTOTP: Honey time-based one-time passwords," *IEEE Trans. Inform. Foren. Secur.*, vol. 20, pp. 4438–4453, 2025.

[11] N. M. Haller, "The S/KEY one-time password system," RFC 1760, Feb. 1995. [Online]. Available: https://www.rfc-editor.org/info/rfc1760

[12] Yubico, "Oath walk-through," Apr. 2025. [Online]. Available: https://developers.yubico.com/OATH/OATH_Walk-Through.html

[13] L. Gong, J. Pan, B. Liu, and S. Zhao, "A novel one-time password mutual authentication scheme on sharing renewed finite random sub-passwords," *J. Comput. System Sci.*, vol. 79, no. 1, pp. 122–130, 2013.

[14] P. Grassi et al., "Digital identity guidelines: Authentication and lifecycle management," Tech. Rep. 800-63B, Jun. 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-63b

[15] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[16] P. J. N. II, M. Straw, C. Metz, and N. M. Haller, "A one-time password system," RFC 2289, Feb. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2289

[17] C. J. Mitchell and L. Chen, "Comments on the S/KEY user authentication scheme," *ACM SIGOPS Operating Syst. Rev.*, vol. 30, no. 4, pp. 12–16, 1996.

[18] C. Jin, Z. Yang, M. van Dijk, and J. Zhou, "Proof of aliveness," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 1–16.

[19] Z. Yang, C. Jin, X. Cao, M. v. Dijk, and J. Zhou, "Optimizing proof of aliveness in cyber-physical systems," *IEEE Trans. Depend. Secur. Comput.*, vol. 21, no. 4, pp. 3610–3628, Jul./Aug. 2024, doi: 10.1109/TDSC.2023.3335188.

[20] M. H. Eldefrawy, M. K. Khan, K. Alghathbar, T.-H. Kim, and H. Elkamchouchi, "Mobile one-time passwords: Two-factor authentication using mobile phones," *Secur. Commun. Netw.*, vol. 5, no. 5, pp. 508–516, 2012.

[21] V. Goyal, A. Abraham, S. Sanyal, and S. Y. Han, "The N/R one time password system," in *Proc. IEEE Int. Conf. Inf. Technol.: Coding Comput.*, 2005, vol. 1, pp. 733–738.

[22] C. Peeters, C. Patton, I. N. S. Munyaka, D. Olszewski, T. Shrimpton, and P. Traynor, "SMS OTP security (SOS): Hardening sms-based two factor authentication," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2022, pp. 2–16.

[23] K. Ullah, I. Rashid, H. Afzal, M. M. W. Iqbal, Y. A. Bangash, and H. Abbas, "SS7 vulnerabilities—A survey and implementation of machine learning vs rule based filtering for detection of SS7 network attacks," *IEEE Commun. Surveys Tut.*, vol. 22, no. 2, pp. 1337–1371, Secondquarter 2020.

[24] Google, "Google authenticator for android (open source version)," Apr. 2021. [Online]. Available: https://github.com/google/google-authenticator-android

[25] Microsoft, "Learn more about microsoft authenticator," Apr. 2025. [Online]. Available: https://www.microsoft.com/en-us/security/mobile-authenticator-app

[26] Lastpass, "One-time passwords," Apr. 2025. [Online]. Available: https://lastpass.com/otp.php

[27] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2013, pp. 145–160.

[28] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 966–983.

[29] Yubico, "Yubico OTP," Apr. 2025. [Online]. Available: https://docs.yubico.com/yesdk/users-manual/application-otp/yubico-otp.html

[30] D. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," RFC 2104, Feb. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2104
[31] Z. Yang, C. Jin, J. Ning, Z. Li, A. Dinh, and J. Zhou, "Group time-based one-time passwords and its application to efficient privacy-preserving proof of location," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2021, pp. 497–512.
[32] H. Sun, K. Sun, Y. Wang, and J. Jing, "Trustotp: Transforming smartphones into secure one-time password tokens," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2015, pp. 976–988.
[33] X. Cao et al., "Dynamic group time-based one-time passwords," *IEEE Trans. Inform. Foren. Secur.*, vol. 19, pp. 4897–4913, 2024.
[34] E. Erdem and M. T. Sandıkkaya, "OTPaaS—one time password as a service," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 743–756, Mar. 2019.
[35] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.
[36] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online)," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2016, pp. 276–291.
[37] O. C. S. Team, "Password storage cheat sheet," 2021. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
[38] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart, "The Pythia PRF service," in *Proc. USENIX Secur. Symp.*, 2015, pp. 547–562.
[39] R. W. Lai, C. Egger, D. Schröder, and S. S. Chow, "Phoenix: Rebirth of a cryptographic password-hardening service," in *Proc. USENIX Secur. Symp.*, 2017, pp. 899–916.
[40] J. Schneider, N. Fleischhacker, D. Schröder, and M. Backes, "Efficient cryptographic password hardening services from partially oblivious commitments," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2016, pp. 1192–1203.
[41] J. Brost, C. Egger, R. W. F. Lai, F. Schmid, D. Schröder, and M. Zoppelt, "Threshold password-hardened encryption services," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2020, pp. 409–424.
[42] C. Jia, S. Wu, and D. Wang, "Reliable password hardening service with opt-out," in *Proc. Int. Symp. Reliable Distrib. Syst.*, 2022, pp. 250–261.
[43] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Trans. Depend. Secur. Comput.*, vol. 15, no. 4, pp. 708–722, Jul./Aug. 2018.
[44] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2018, pp. 456–486.
[45] Z. Zhang, Y. Wang, and K. Yang, "Strong authentication without temper-resistant hardware and application to federated identities," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–15.
[46] Microsoft, "Transport layer security (TLS) registry settings," Apr. 2025. [Online]. Available: [Online]. Available: https://learn.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings?tabs=diffie-hellman
[47] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446
[48] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.
[49] A. M. Abdelmoniem and B. Bensaou, "T-RACKs: A faster recovery mechanism for TCP in data center networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 1074–1087, Jun. 2021.
[50] A. Phanishayee et al., "Measurement and analysis of TCP throughput collapse in cluster-based storage systems," in *Proc. USENIX Secur. Symp.*, 2008, vol. 8, pp. 1–14.

Zixuan Ding is currently working toward the PhD degree with the College of Cryptology and Cyber Science, Nankai University, Tianjin, China. He has authored or coauthored papers with venues like *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Internet of Things Journal*, and *IEEE Transactions on Vehicular Technology*. His research interests include applied cryptography, authentication protocol, and password-based authentication.

Yihe Duan (Member, IEEE) is currently working toward the PhD degree with Nankai University, Tianjin, China. He has published papers at *IEEE Symposium on Security and Privacy* and *IEEE Transactions on Information Forensics and Security*. His research interests include applied cryptography and password-based authentication.

Ding Wang received the PhD degree in information security with Peking University, in 2017, and was supported by the "Boya Postdoctoral Fellowship" in Peking University from 2017 to 2019. He is currently a full professor with Nankai University. He has authored or coauthored more than 100 papers with venues like *IEEE Symposium on Security and Privacy*, ACM CCS, NDSS, Usenix Security, *IEEE Transactions on Dependable and Secure Computing* and *IEEE Transactions on Information Forensics and Security*. His research has been reported by over 200 media like The Wall Street Journal, Daily Mail, Forbes, IEEE Spectrum, and Communications of the ACM, and resulted in the revision of the authentication guideline NIST SP800-63-2. His research interests focus on passwords, authentication, and provable security.