PROTECT: Efficient Password-Based Threshold Single-Sign-On Authentication for Mobile Users against Perpetual Leakage

Yuan Zhang, *Member, IEEE*, Chunxiang Xu, *Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, Kan Yang, *Member, IEEE*, Nan Cheng, *Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

Abstract—Password-based single-sign-on authentication has been widely applied in mobile environments. It enables an identity server to issue authentication tokens to mobile users holding correct passwords. With an authentication token, one can request mobile services from related service providers without multiple registrations. However, if an adversary compromises the identity server, he can retrieve users' passwords by performing dictionary guessing attacks (DGA) and can overissue authentication tokens to break the security. In this paper, we propose a password-based threshold single-sign-on authentication scheme dubbed PROTECT that thwarts adversaries who can compromise identity server(s), where multiple identity servers are introduced to authenticate mobile users and issue authentication tokens in a threshold way. PROTECT supports key renewal that periodically updates the secret on each identity server to resist perpetual leakage of the secret. Furthermore, PROTECT is secure against off-line DGA: a credential used to authenticate a user is computed from the password and a server-side key. PROTECT is also resistant to online DGA and password testing attacks in an efficient way. We conduct a comprehensive performance evaluation of PROTECT, which demonstrates the high efficiency on the user side in terms of computation and communication and proves that it can be easily deployed on mobile devices.

Index Terms—Mobile users, password-based single-sign-on authentication, dictionary guessing attacks, secret renewal, online password testing attacks

1 Introduction

With the booming development of smartphones and mobile devices, users utilizing such devices can access a rich set of featured mobile services in a convenient way [1], [2], [3], [4], [5]. Typical applications include mobile payment systems, where a user requests different services and makes payments only using a smartphone without carrying cash. While users enjoy great conveniences from the mobile services, they are confronted with critical threats: adversaries always attempt to breach users' devices for personal gains. For example, an adversary may impersonate a valid user to conduct a fraudulent transaction in a mobile payment system. Once he succeeds, he can access the service without payment [6], [7]. As such, a secure authentication method is often required by the mobile service providers to identify their users.

Human-memorable passwords remain the most prevalent form of user authentication in mobile systems [8]. Such an authentication mechanism provides users an efficient and convenient way to identify themselves [9], [10]. However, it would be painful for users to register an individual account from each mobile service provider and remember all the usernames and passwords. In reality, in order to free users from multiple registrations, several service providers may employ an identity server to authenticate users on behalf of themselves, where a user provides her/his identity and password to the identity server, obtains an authentication token if the authentication passes, and requests services from the providers with the token [11]. Such a paradigm is called password-based single-sign-on authentication and has been used in various industries for mobile users, such as Amazon OpenID,[1] Facebook login,[2] and Google Identity Platform.[3]

Despite the desirable benefits from the password-based single-sign-on authentication, critical security concerns have been raised. Existing password-based single-sign-on authentication schemes (e.g., Kerberos [11], Json Web Token,[4] and SAML[5])

• *Y. Zhang and H. Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. E-mail: ZY_LoYe@126.com, hongweili@uestc.edu.cn.*
• *C. Xu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Key Laboratory of Financial Mathematics (Putian University), Fujian Province University, Putian, Fujian 351100, China. E-mail: chxxu@uestc.edu.cn.*
• *K. Yang is with the Department of Computer Science, The University of Memphis, Memphis, TN 38152 USA. E-mail: kan.yang@memphis.edu.*
• *N. Cheng is with the School of Telecommunication, Xidian University, Xi'an 710126, China. E-mail: dr.nan.cheng@ieee.org.*
• *X. Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 17 Jan. 2019; revised 14 Jan. 2020; accepted 18 Feb. 2020. Date of publication 24 Feb. 2020; date of current version 5 May 2021. (Corresponding author: Chunxiang Xu and Yuan Zhang.) Digital Object Identifier no. 10.1109/TMC.2020.2975792

1. https://docs.aws.amazon.com/cognito/latest/developerguide/open-id.html
2. https://developers.facebook.com/docs/facebook-login
3. https://developers.google.com/identity/protocols/OpenIDConnect
4. https://jwt.io/
5. https://developers.onelogin.com/saml

are mainly confronted with the single-point-of-failure problem. Specifically, once the identity server is compromised, adversaries can forge and overissue valid tokens without being authenticated [12]. Worse still, users' passwords would be leaked from the database of compromised identity server.[6] The leaked passwords have significant value to adversaries, since mobile users always use sister passwords (tweaked or reused passwords) in different systems. The recent work [13] has demonstrated that given a user's one sister password, the advantage of an adversary to break the user's password in other systems can be improved significantly. Traditional method [14] requires the identity server to only store hashed passwords, which serve as credentials for user authentication. This method, however, cannot ensure the security if the hashed passwords are leaked, since human-memorable passwords are inherently low-entropy and are vulnerable to off-line dictionary guessing attacks (DGA).

Recently, a password-based threshold single-sign-on authentication scheme, called PASTA, was proposed [15] to address the single-point-of-failure problem by employing multiple identity servers to authenticate users. In particular, the generation of tokens is distributed among multiple identity servers, where a master secret key used to generate tokens is shared among the identity servers in a threshold way [16]. Each identity server has a master secret share, such that compromising any less than a threshold number of identity servers would not break the security. To resist off-line DGA, the authentication credential depends on the password and a server-side key that is shared among all identity servers in a threshold way. As long as the server-side key remains inaccessible to the adversary, the password cannot be retrieved by performing DGA. During an authentication, the user derives a new secret based on her/his password from identity servers in an oblivious manner such that she/he can compute the credential using the secret without leaking any information about the password to them. We call this secret servers-derived password.

In spite of the advantages of PASTA, it also bears a quit strong assumption that adversaries cannot break a threshold number of identity servers over a long period of time. Nevertheless, fixed bounds on adversary's corruption limit are unrealistic for long-lived mobile systems using single-sign-on authentication, and the protection provided by secret sharing could be insufficient [17], [18], [19], [20]: a sophisticated adversary can perpetually attempt to break into the identity servers to corrupt all of them given enough time. Once he collects a threshold number of master secret shares, he can forge and overissue tokens to increase his profits in the system. A straightforward way to resist such the adversary is to let identity servers periodically re-share a new master secret key. However, this requires all related service providers to update the system parameters in order to invalidate all existing tokens and verify the newly generated tokens. Consequently, each user has to request a new authenticated token from identity servers, which brings prohibitive costs for the user who only equips a mobile device. It also causes a heavy computation and communication burden for service providers. Moreover, in PASTA, to resist malicious users who access identity servers to obtain enough servers-derived passwords and perform DGA (such attack is called online DGA and will be detailed in Section 5), different users should generate different server-side keys and distribute them over all identity servers, which requires mobile users to bear high computation and communication costs and also causes heavy storage costs for identity servers. We notice that existing schemes (e.g., PASTA) are vulnerable to a type of attack, since the identity servers are considered as an oracle to respond to users' authentication requests without knowing the authentication results.

In this paper, we first generalize the type of attack existing in actual authentication systems, which is called online passwords testing attack (OPTA). Then, we propose a password-based threshold single-sign-on authentication against perpetual leakage for mobile users, dubbed PROTECT. PROTECT employs multiple identity servers to authenticate mobile users and issue authentication tokens. Each identity server stores a credential that is built on a user's servers-derived password for authenticating the user, such that adversaries cannot retrieve the password from the credential by performing off-line DGA. Identity servers renew their master secret shares in a fixed time interval (called an epoch). The renewal of master secret shares would not change the master secret key, and thereby would not "break down" the scheme. The adversary, who collects a threshold number of master secret shares from different epochs, cannot forge a valid token. Furthermore, PROTECT utilizes a hybrid mechanism to protect online DGA, where multiple users form a group to use the same server-side key, and the number of authentication token requests for each user in an epoch is limited by identity servers. This improves the storage efficiency on identity servers significantly with resistance against online DGA. In PROTECT, identity servers would not be considered as oracles, since they can extract the authentication results during the interaction with users. Hence, PROTECT is secure against OPTA. Specifically, the contributions of this paper are summarized as follows.

- We propose a secure and efficient password-based threshold single-sign-on authentication scheme (PROTECT) for mobile users, where multiple identity servers are employed to authenticate mobile users and issue authentication tokens in a threshold way. Identity servers in PROTECT periodically renew their master secret shares so as to thwart adversaries who can perpetually compromise master secret shares.
- We propose a hybrid mechanism and integrate it into PROTECT to resist online DGA: all users are grouped, and each group of users utilizes one server-side key; the number of token requests made by each user during an epoch is limited by identity servers. It enables mobile users to free from substantial computational costs and allows identity servers to avoid huge storage costs.
- We present security proofs to demonstrate that PROTECT can be secure and robust from various attacks. We also conduct a comprehensive performance analysis, which shows the high efficiency of PROTECT. Compared with existing schemes (e.g., PASTA), PROTECT provides a stronger security guarantee and is more efficient in terms of storage overhead.

6. "Csdn leaked passwords summary," https://gist.github.com/cpylua/2708012#file-csdn-passwords

The remainder of this paper is organized as follows. We review the related works in Section 2 and present the preliminaries in Section 3. We define the system model, threat model, and design goals in Section 4. In Sections 5 and 6, we overview and propose PROTECT, respectively. We analyze the security of PROTECT in Section 7 and evaluate the performance in Section 8. Finally, we draw the conclusions and outlook the future work in Section 9.

2 Related Work

Password-based single-sign-on authentication is one of the most important methods that is applied in mobile environments. Compared with other authentication paradigms (which utilize other authentication factors, e.g., physiological features, secret keys shared between users and the authenticator, distances between users and the authenticator, and certificates issued by a Certificate Authority, rather than passwords [3], [21], [22], [23], [24], [25]), it only requires mobile users to memorize their passwords without other investments in equipments. In such the authentication paradigm, an identity server, which is employed by the service provider(s), authenticates the user by checking whether she/he provides the correct password and issues an authentication token if the authentication passes. With the authentication token, the user can request multiple services from other mobile service providers.

Existing schemes [11], [22] mainly focus on the security from two aspects: authentication token security and password security. In particular, adversaries (including malicious insiders working at the provider) may forge an authentication token to request services from application servers without being authenticated by the identity server. Furthermore, adversaries may extract users' passwords from the identity server's database.

To protect the scheme from authentication token forgery, simply requiring the identity server to well maintain the secret used to generate tokens could be insufficient, since it suffers from the single-point-of-failure problem. A natural approach to resolve this tension is to distribute the token generation to multiple identity servers in a threshold way [26], [27], [28], [29], [30], [31], [32], such that the security of token is ensured even if the adversary can compromise multiple identity servers (less than the threshold number ones).

Integrating this mechanism into the password-based single-sign-on authentication scheme is still confronted with the password leakage [33]. To prevent the user's password from being leaked to the identity servers or service provider(s), traditional approaches require the identity servers to only store the hash value of the password [34], [35] which serves as a credential to authenticate the user. However, human-memorable passwords are inherently low-entropy, this sufficiently gives rise to dictionary guessing attacks [13], [36], where an adversary may calculate the hash values of all password candidates and identify the hash value which matches the targeted one. This allows the adversary to retrieve the target password in an off-line manner. Prior works on resisting off-line DGA can mainly be classified into three categories.

- Password-based authentication with salted password hashing [37]. The authentication credential is the hash value of a password and a long-term random number. As long as the number (i.e., salt) remains inaccessible to adversaries, the authentication credential is secure against off-line DGA.
- Password-based authentication built on emerging cryptographic primitives. The authentication credential is computed using emerging hash functions, e.g., memory-hard functions [38], [39], [40], [41], which increases the costs of performing off-line DGA significantly.
- Password-based authentication utilizing an independent server. An independent server is introduced to harden the users' passwords [42], [43], [44]. Specifically, the authentication credential is computed on the password and a secret that is possessed by the server. As long as the secret is inaccessible to adversaries, the security of the scheme against DGA is ensured.

Although these works make the adversary's task harder, they bear a quite strong assumption that the identity servers would be honest and reliable over a long period of time. Once the identity server is compromised, the security of these schemes would be broken.

To ensure the security of passwords and authentication tokens, Agrawal *et al.* [15] introduced a concept of password-based threshold token-based authentication (PASTA), where the token generation is distributed to multiple identity servers and the authentication credential stored in each identity server is computed from a servers-derived password which depends on the password itself and a server-side secret. In the authentication phase, the user first interacts with the identity servers to retrieve the servers-derived password in an oblivious way such that the user can obtain it without leaking any information about her/his password to the identity servers. *If and only if the user has the correct password, she/he can compute the valid servers-derived password*. Then the user computes the corresponding credential for authentication. However, PASTA is vulnerable to the online password testing attacks (which will be shown in Section 6.3) and perpetual leakage of secret shares on identity servers. In this paper, we propose PROTECT, a secure and efficient password-based threshold single-sign-on authentication scheme for mobile users. PROTECT is secure against online password testing attacks and can resist external adversaries who are able to perpetually break into some identity servers over a long period of time. Compared with PASTA, PROTECT provides a stronger security guarantee with high efficiency in terms of storage overhead.

3 Preliminaries

3.1 Notations and Conventions

We use ℓ to denote the security parameter. For two integers $i, n \in Z_p, (i \leq n)$, we denote by $[1, n]$ the set $\{1, 2, \ldots, n\}$ and denote by $i \ll n$ that i is far less than n. For a finite set T, $|T|$ denotes the number of components in T. For two bit-strings x and y, we denote by $x||y$ their concatenation.

3.2 Basic Theory

Threshold Cryptography. In a (t, n)-threshold cryptosystem, a secret is shared among some set of n players, and each of them has a secret *share*. Any t players are able to pool their shares and perform certain cryptographic operations (e.g., signing a message, encryption/decryption, reconstructing the secret), but no coalition of fewer than t players can get *any* information about the secret from their collective shares.

Bilinear Map. Let G be an additive group and G_T be a multiplicative group. G and G_T have the same prime order p. A bilinear map $e : G \times G \rightarrow G_T$ has the following properties:

- Bilinearity. $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G$, $a, b \in Z_p^*$.
- Non-degeneracy. $\forall P, Q \in G$ and $P \neq Q, e(P, Q) \neq 1$.
- Computability. There is an efficient algorithm for computing e.

Computational Diffe-Hellman (CDH) problem in G. Let P be a generator of G, for any unknown $a, b \in Z_p^*$, given aP, bP to compute abP.

Pseudorandom Function. Let $F : \{0,1\}^{\ell} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a pseudorandom function if for all probability polynomial-time distinguishers $\mathcal{D}$, there is a negligible function $negl$ such that

$$|\Pr[\mathcal{D}^{F(k,\cdot)}(1^{\ell}) = 1] - \Pr[\mathcal{D}^{\mathcal{F}(\cdot)}(1^{\ell}) = 1]| \leq negl(\ell),$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^{\ell}$ and the randomness of $\mathcal{D}$, and the second probability is taken over uniform choice of $\mathcal{F} \in Func_{\ell}$ and the randomness of $\mathcal{D}$, $Func_{\ell}$ is the set of all functions mapping ℓ-bit strings to ℓ-bit strings.

4 Problem Statement

4.1 Modelling Password-Based Threshold Single-Sign-On Authentication in Mobile Environments

There are three entities in a password-based threshold single-sign-on authentication scheme.

- *Mobile users*. A set of mobile users is involved in the scheme. Different users are independent, and thereby we would not preset the number of users in the scheme. Each user equips with a mobile device, has an identity and a password, and needs to request an authentication token from the identity servers. In the scheme, the only secret a user possesses is her/his password, and the mobile device the user uses for access stores the authentication token on her/his behalf.
- *Identity servers*. A set of identity servers $\{\mathcal{IS}_1, \mathcal{IS}_2, \ldots, \mathcal{IS}_n\}$ is involved in the scheme. Each identity server authenticates the user by checking that whether she/he possesses the correct password.
- *Application servers*. The application servers are subject to mobile service providers, they employ the identity servers to authenticate users. If a user passes the authentication, she/he can request mobile services from the application servers.

We illustrate the procedure of a password-based threshold single-sign-on authentication for mobile users in Fig. 1. This procedure is extended from plain password-based single-sign-on authentication schemes [11] and consists of two phases: *Registration* and *Authentication*.

In the *Registration* phase, all identity servers share a master secret key in a (t, n)-threshold way and each of them has a master secret share. A mobile user $\mathcal{U}$ generates an identity

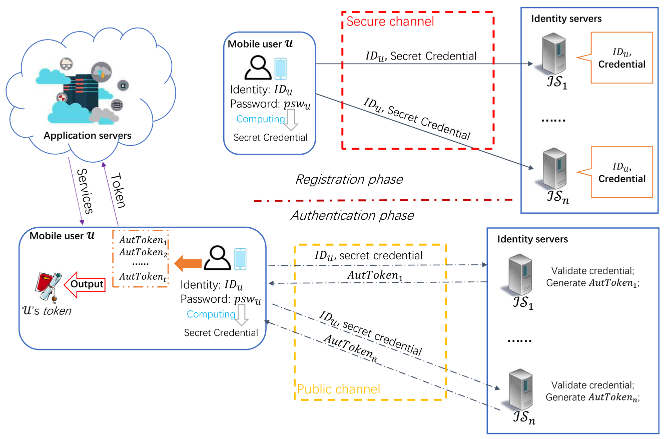

Application servers
Services
Token
Secure channel
Mobile user $\mathcal{U}$
Identity: $ID_{\mathcal{U}}$
Password: $psw_{\mathcal{U}}$
Computing
Secret Credential
$ID_{\mathcal{U}}$, Secret Credential
$ID_{\mathcal{U}}$, Secret Credential
Identity servers
$ID_{\mathcal{U}}$, Credential
$\mathcal{IS}_1$
......
$ID_{\mathcal{U}}$, Credential
$\mathcal{IS}_n$
Registration phase
Authentication phase
Mobile user $\mathcal{U}$
$AutToken_1$
$AutToken_2$
......
$AutToken_t$
Output
$\mathcal{U}$'s token
Identity: $ID_{\mathcal{U}}$
Password: $psw_{\mathcal{U}}$
Computing
Secret Credential
$ID_{\mathcal{U}}$, secret credential
$AutToken_1$
$ID_{\mathcal{U}}$, secret credential
$AutToken_n$
Public channel
Identity servers
Validate credential; Generate $AutToken_1$;
$\mathcal{IS}_1$
......
Validate credential; Generate $AutToken_n$;
$\mathcal{IS}_n$

Fig. 1. Password-based threshold single-sign-on authentication.

$ID_{\mathcal{U}}$ and a human-memorable password $psw_{\mathcal{U}}$ for registration. Based on $psw_{\mathcal{U}}$, $\mathcal{U}$ computes a secret credential for subsequential authentications. After the registration, each identity server maintains $ID_{\mathcal{U}}$ and the corresponding credential and the mobile user only needs to memorize her/his password for subsequent authentications. In this phase, a secure channel is established between the user's mobile device and each identity server via a protocol, e.g., TLS [45].

In the *Authentication* phase, $\mathcal{U}$ first computes a credential and sends the identity and credential to identity servers using a mobile device. Each identity server checks the validity of the credential. If the checking passes, the identity server generates an authentication token share using its master secret share and sends it to $\mathcal{U}$, where the token is an authenticated message that contains necessary information for authentication, e.g., $\mathcal{U}$'s information and attributes, expiration time, and other auxiliary information. After receiving a threshold number t of authentication token shares, $\mathcal{U}$ can reconstruct a valid authentication token. With the token, $\mathcal{U}$ can request mobile services from mobile application servers who hold the public parameters.

4.2 Threat Model

In reality, the entropy of human-memorable passwords is low. Therefore, passwords suffer from dictionary guessing attacks. In the threat model, we assume that adversaries can perform DGA to compromise users' passwords. Furthermore, the password-based threshold single-sign-on authentication is confronted with threats from two different angles: external adversaries and internal adversaries.

External Adversaries. An external adversary can perform the following attacks to break the security.

- *Eavesdropping attacks*. During the authentication between a mobile user and the identity servers, the adversary may intercept the interaction messages and perform off-line dictionary guessing attacks to retrieve the user's password.
- *Perpetual leakage attacks*. The adversary may perpetually attempt to break into the identity servers over a long period of time, which enables him to collect more than t master secret shares from compromised identity servers to forge valid authentication tokens.

- *Online passwords testing attacks.* The adversary has a set of password candidates $\overrightarrow{PC} = \{pswc_1, pswc_2, \ldots, pswc_\Delta\}$ and all users' identities $\overrightarrow{ID} = \{ID_{\mathcal{U}_1}, ID_{\mathcal{U}_2}, \ldots, ID_{\mathcal{U}_\Theta}\}$, where $|\overrightarrow{PC}| \ll |\overrightarrow{ID}|$ (i.e., $\Theta \ll \Delta$). In OPTA, the adversary first fixes a password chosen from $\overrightarrow{PC}$. Then, he tries different identities in $\overrightarrow{ID}$ to request an authentication token from identity servers. If all the requests are rejected, he chooses another password from $\overrightarrow{PC}$ and repeats the above process. If at least one user utilizes a password existing in $\overrightarrow{PC}$, this can be identified. OPTA can be considered as a variant of DGA: In a DGA, the adversary fixes an account (ID) and tries different passwords from the dictionary until one succeeds; In an OPTA, the adversary fixes a password from the dictionary, and try different accounts until one succeeds (if the adversary fails to pass the authentication by using all accounts, he can fix another password from the dictionary and repeat this process until one succeeds).

We stress that collecting password candidates to perform OPTA is feasible and practical. Specifically, in reality users would always use weak passwords (e.g., dictionary words, repeating characters, and personal identifiable information[7]) for convenient authentication, and for a specific user, the number of her/his passwords is quit small, which means that she/he may use the sister passwords in different systems [13]. In recent years, passwords leakage from compromised/vulnerable systems have become commonplace, an adversary may collect passwords from leaked databases and then perform OPTA to break the security of existing password-based authentication schemes. Moreover, the adversary may calculate out some common-use passwords from the leaked databases to perform OPTA Therefore, OPTA should be well considered in password-based authentication schemes.

Internal Adversaries. There are two types of internal adversaries.

- *Malicious identity servers.* They can perform the following attacks to break the security of the scheme.
  - *Off-line DGA.* A malicious identity server may utilize the locally stored secret and perform offline DGA to retrieve the users' passwords. It would cause serious effects.[8] For example, if users' passwords are retrieved, the malicious identity server can break the security of other authentication systems [13] with a high advantage and also can impersonate a user to access the related service providers without registration.
  - *Forgery of authentication tokens.* A malicious identity server may collude with others to forge and overissue authentication tokens on behalf of all identity servers. This enables the malicious identity server to increase its profits in the system significantly, at the expense of the related service providers' profits.
- *Malicious mobile users.* A malicious user may impersonate other valid users (i.e., victims) to pass the identity servers' authentications and request victims' authentication tokens without having the victims' passwords. Such an attack is called *impersonation attack*. Once he succeeds, he can access the related service providers without payment. Typically, a user may request an authentication token using a mobile device provided by others. Later, an adversary who can access the same device may impersonate the user to request an authentication token.

To prove the security of PROTECT under the adversary model, we follow the security notions of *unpredictability* and *obliviousness* [15]. Details will be provided in Section 7. In the threat model, we would not consider the denial of service (DoS) attacks.

4.3 Challenges and Design Goals

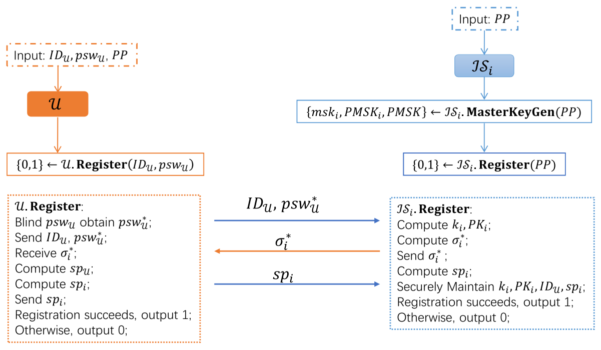
In this paper, we target to construct a threshold password-based authentication scheme for mobile users, in which there exist two challenges:

1) *Resistance against perpetual leakage of master secret shares.* Since an external adversary may break into identity servers multiple times over a long period of time, master secret shares would be perpetually leaked to the adversary. Consequently, the adversary can forge and overissue valid authentication tokens to break the security of the system. Periodically resharing a new master secret key among all identity servers can resist the external adversary, whereas it incurs a heavy computation and communication burden in practice. As such, how to resist external adversaries without introducing heavy costs is a challenging problem.
2) *Resistance against OPTA.* An adversary may collect password candidates and retrieve the passwords by performing OPTA. Since password candidates can be extracted in many ways, OPTA should be resisted to ensure the security of systems.

To address the challenges under the aforementioned model, the following objects should be achieved.

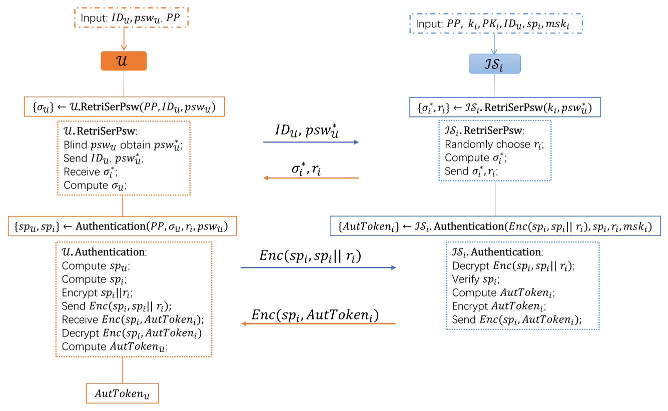
- *Functionality.* The identity servers can authenticate mobile users. For a mobile user, she/he *only* needs to possess her/his password and use her/his mobile device to pass the identity servers' authentication without additional investments in equipments.
- *Security.* The proposed scheme should achieve resistance against eavesdropping attacks, perpetual leakage attacks, OPTA, off-line DGA, forgery of authentication tokens, and impersonation attacks.
- *Efficiency.* The communication and computation overhead on mobile users should be as efficient as possible. The storage overhead on the identity servers should be highly efficient, since the identity servers have to maintain the secret shares for a long period of time. The renewal of secret shares should not introduce heavy computation and communication costs on identity servers and should not require users and related service providers to participate.

7. "Top 10 signs your password is weak," https://authanvil.com/blog/top-10-signs-your-password-is-weak
8. "1.4 billion clear text credentials discovered in a single database," https://medium.com/4iqdelvedeep/1-4-billion-clear-text-credentials-discovered-in-a-single-database-3131d0a1ae14



Input: $ID_{\mathcal{U}}, psw_{\mathcal{U}}, PP$
$\mathcal{U}$
$\{0,1\} \leftarrow \mathcal{U}.\mathbf{Register}(ID_{\mathcal{U}}, psw_{\mathcal{U}})$
$\mathcal{U}$.Register:
Blind $psw_{\mathcal{U}}$ obtain $psw_{\mathcal{U}}^*$;
Send $ID_{\mathcal{U}}$, $psw_{\mathcal{U}}^*$;
Receive σ_i^*;
Compute $sp_{\mathcal{U}}$;
Compute sp_i;
Send sp_i;
Registration succeeds, output 1;
Otherwise, output 0;
$ID_{\mathcal{U}}, psw_{\mathcal{U}}^*$
σ_i^*
sp_i
Input: PP
$\mathcal{IS}_i$
$\{msk_i, PMSK_i, PMSK\} \leftarrow \mathcal{IS}_i.\mathbf{MasterKeyGen}(PP)$
$\{0,1\} \leftarrow \mathcal{IS}_i.\mathbf{Register}(PP)$
$\mathcal{IS}_i$.Register:
Compute k_i, PK_i;
Compute σ_i^*;
Send σ_i^*;
Compute sp_i;
Securely Maintain $k_i, PK_i, ID_{\mathcal{U}}, sp_i$;
Registration succeeds, output 1;
Otherwise, output 0;

Fig. 2. Overview of registration.



Input: $ID_{\mathcal{U}}, psw_{\mathcal{U}}, PP$
$\mathcal{U}$
$\{\sigma_{\mathcal{U}}\} \leftarrow \mathcal{U}.\mathrm{RetriSerPsw}(PP, ID_{\mathcal{U}}, psw_{\mathcal{U}})$
$\mathcal{U}$.RetriSerPsw:
Blind $psw_{\mathcal{U}}$ obtain $psw_{\mathcal{U}}^*$;
Send $ID_{\mathcal{U}}$, $psw_{\mathcal{U}}^*$;
Receive σ_i^*;
Compute $\sigma_{\mathcal{U}}$;
$ID_{\mathcal{U}}, psw_{\mathcal{U}}^*$
σ_i^*, r_i
$\{sp_{\mathcal{U}}, sp_i\} \leftarrow \mathrm{Authentication}(PP, \sigma_{\mathcal{U}}, r_i, psw_{\mathcal{U}})$
$\mathcal{U}$.Authentication:
Compute $sp_{\mathcal{U}}$;
Compute sp_i;
Encrypt $sp_i || r_i$;
Send $Enc(sp_i, sp_i || r_i)$;
Receive $Enc(sp_i, AutToken_i)$;
Decrypt $Enc(sp_i, AutToken_i)$
Compute $AutToken_{\mathcal{U}}$;
$Enc(sp_i, sp_i || r_i)$
$Enc(sp_i, AutToken_i)$
$AutToken_{\mathcal{U}}$
Input: $PP, k_i, PK_i, ID_{\mathcal{U}}, sp_i, msk_i$
$\mathcal{IS}_i$
$\{\sigma_i^*, r_i\} \leftarrow \mathcal{IS}_i.\mathrm{RetriSerPsw}(k_i, psw_{\mathcal{U}}^*)$
$\mathcal{IS}_i$.RetriSerPsw:
Randomly choose r_i;
Compute σ_i^*;
Send σ_i^*, r_i;
$\{AutToken_i\} \leftarrow \mathcal{IS}_i.\mathrm{Authentication}(Enc(sp_i, sp_i || r_i), sp_i, r_i, msk_i)$
$\mathcal{IS}_i$.Authentication:
Decrypt $Enc(sp_i, sp_i || r_i)$;
Verify sp_i;
Compute $AutToken_i$;
Encrypt $AutToken_i$;
Send $Enc(sp_i, AutToken_i)$;

Fig. 3. Overview of authentication.

5 Overview of PROTECT

In PROTECT, a set of mobile users and a group of identity servers $\{\mathcal{IS}_1, \mathcal{IS}_2, \ldots, \mathcal{IS}_n\}$ are involved. A formal definition of PROTECT is given in the following.

Definition 1. *PROTECT consists of six algorithms,* Setup, MasterKeyGen, Register, RetriSerPsw, Authentication, *and* RenewShare.

- *Setup*. In this algorithm, the system parameter is generated and sent to users and identity servers.
- *MasterKeyGen*. This algorithm enables a master secret key to be shared among all identity servers.
- *Register*. This algorithm enables a user to register with all identity servers.
- *RetriSerPsw*. This algorithm enables a user to retrieve a servers-derived password from the identity servers.
- *Authentication*. This algorithm enables a user to request an authentication token from the identity servers.
- *RenewShare*. This algorithm enables each identity server to renew its master secret share.

Specifically, there are three phases in PROTECT: *Registration*, *Sign-on*, and *Key Renewal*. Hereinafter, we overview the authentication between a user $\mathcal{U}$ and n identity servers. All algorithms takes the security parameter ℓ as input by default and we would not explicit show it in the following.

Registration phase consists of three algorithms: *Setup*, *MasterKeyGen*, and *Register*. We show the execution of algorithms on the ith identity server $\mathcal{IS}_i (i \in [1, n])$ and $\mathcal{U}$ in Fig. 2 and describe below, where *Setup* is executed by the system and is not shown in the figure.

In *Setup*, the system is initiated, the system parameter PP is generated, and PP is distributed all entities in the scheme.

MasterKeyGen. This algorithm takes the public parameter as input to share a master secret key msk among all identity servers, where $\mathcal{IS}_i$'s master secret share msk_i. The corresponding master public key and master public share are $PMSK$ and $PMSK_i$, respectively.

Register. This algorithm is interactively executed by $\mathcal{U}$ and all identity servers. The identity servers generate and share a server-side key k (PK is the corresponding public key) for $\mathcal{U}$ and $\mathcal{IS}_i$ has a server-side key share k_i with the corresponding public share PK_i. $\mathcal{U}$ creates an identity $ID_{\mathcal{U}}$ and a password $psw_{\mathcal{U}}$, she/he blinds $psw_{\mathcal{U}}$ to obtain $psw_{\mathcal{U}}^*$ and sends it to the identity servers. $\mathcal{IS}_i$ signs $psw_{\mathcal{U}}^*$ using k_i to get σ_i^* and sends it to $\mathcal{U}$. After receiving t signatures, $\mathcal{U}$ combines them to get a signature on $psw_{\mathcal{U}}$ under k and computes a servers-derived password $sp_{\mathcal{U}}$. Finally, $\mathcal{U}$ computes an authentication credential sp_i using $sp_{\mathcal{U}}$ and $psw_{\mathcal{U}}$ for $\mathcal{IS}_i$. In PROTECT, both msk and k are generated using a distributed (t, n)-threshold secret sharing technique without a trusted dealer, which ensures that any identity server cannot extract msk and k, and compromising less than t identity servers would not break the security.

Sign-on phase consists of two algorithms: *RetriSerPsw* and *Authentication*, which is shown in Fig. 3.

RetriSerPsw. This algorithm is interactively executed by $\mathcal{U}$ and all identity servers. $\mathcal{U}$ derives $\sigma_{\mathcal{U}}$ from identity servers, where $\sigma_{\mathcal{U}}$ is a signature on $psw_{\mathcal{U}}$ under k and is used to compute the servers-derived password.

Authentication. This algorithm enables a user to be authenticated by identity servers and to receive an authentication token for future accesses to related services. With $\sigma_{\mathcal{U}}$ and $psw_{\mathcal{U}}$, $\mathcal{U}$ computes $sp_{\mathcal{U}}$ and further computes the authentication credentials. Each identity server checks the validity of the credential, and issues an authentication token share using the master key share if the checking succeeds. $\mathcal{U}$ can retrieve a valid authentication token $AutToken_{\mathcal{U}}$ from t authentication token shares. The random element r_i is used to resist relay attacks. The interaction messages are encrypted by the credential to thwart eavesdropping attacks. With $AutToken_{\mathcal{U}}$, $\mathcal{U}$ requests services from related service providers. The service providers can verify $AutToken_{\mathcal{U}}$ using the public parameters of identity servers and provide $\mathcal{U}$ the corresponding servers if the verification succeeds.

Key renewal phase consists of an algorithm *RenewShare*, where each identity server renews its master secret share msk_i to prevent PROTECT from compromising master secret key shares. To ensure that renewal of master secret shares would not "break down" PROTECT, the master secret key msk would not be changed after execution of *RenewShare*. The key technique is to require all identity servers to share a "0" in a distribute way. For each identity server, the renewed master secret share is computed by adding the previous master secret share and the share of "0". Note that the shares of "0" are not equal to 0, which enables all master secret shares to be changed after renewal without changing msk. After execution of *RenewShare*, each identity server deletes the old master secret share and only maintain the renewed one. The renewal of master secret shares would only be executed once in an epoch.

There is still a subtle security issue. When a large number of users exist in the system, where an adversary compromises at least one identity server and controls a part of registered users. The adversary can utilize different user identities[9] and try different passwords by repeating *RetriSerPsw*. Eventually, the adversary can obtain all the servers-derived password candidates and can perform off-line DGA to recover the target password. Such an attack is called online DGA. PROTECT utilizes a hybrid mechanism to resist the online DGA: the number of requests on servers-derived password and authentication token made by one user during an epoch is limited by identity servers, all users are grouped according to the registration time, a group of users shares a server-side key, and different groups of users have different server-side keys.

6 Proposed PROTECT

6.1 Construction of PROTECT

For the sake of brevity, we describe the authentication between a mobile user $\mathcal{U}$ and the identity servers and do not show the process that mobile users request services from related service providers.

Algorithm 1. Distributed Secret Sharing Protocol

```
Require: Security parameter $\ell$, identity servers' indexes $\{1,\ldots,n\}$, threshold number $t$, the index $i$ of identity server who executes this algorithm.
Ensure: All identity servers share a secret $s$ and the corresponding public key $PS$;
    For $i=1,2,\ldots,n$, $\mathcal{IS}_i$ computes a secret share $s_i$ and the corresponding public share $PS_i$.
1: $\mathcal{IS}_i$ randomly chooses $a_{i,0}\in Z_p^*$ and a polynomial $f_i(x)=a_{i,0}+a_{i,1}x+\cdots+a_{i,t-1}x^{t-1}$ over $Z_p$ with degree at most $t-1$ such that $f_i(0)=a_{i,0}$;
2: For $\epsilon=1,2,\ldots,t-1$, $\mathcal{IS}_i$ sends $a_{i,0}P$ and $a_{i,\epsilon}P$ to *all other* identity servers. $\mathcal{IS}_i$ sends $f_i(j)$ *secretly* to $\mathcal{IS}_j$ for $j=1,2,\ldots,n; j\neq i$;
3: After receiving $f_j(i)$ from $\mathcal{IS}_j$ $(j\in[1,n], j\neq i)$, $\mathcal{IS}_i$ verifies $f_j(i)P=\sum_{\epsilon=0}^{t-1} i^{\epsilon}\cdot a_{j,\epsilon}P$. If the verification fails, $\mathcal{IS}_i$ rejects $f_j(i)$;
4: $\mathcal{IS}_i$ computes the secret share $s_i=\sum_{\gamma=1}^{n} f_\gamma(i)$, the public share $PS_i=s_iP$;
5: $\mathcal{IS}_i$ computes public key $PS=\sum_{i=1}^{n} a_{i,0}P$, securely stores $s_i$, maintains $\{PS_1,PS_2,\ldots,PS_n\}$, and deletes other values generated above.
6: Here, the secret key $s=\sum_{i=1}^{n} a_{i,0}$ is distributed to all identity servers but does not appear explicitly in the scheme.
```

Phase 1. Registration. In this phase, system parameters are generated, and mobile users register with identity servers.

Setup. With the security parameter ℓ, system parameter $PP=\{p,P,G,G_T,e,h,\hbar,H,F,E,\rho,\phi,t,n\}$ is determined, in which G is an additive group whose generator is P and order is prime p, G_T is a multiplicative group, $e: G\times G\rightarrow G_T$ is a bilinear pairing, $h,\hbar:\{0,1\}^*\rightarrow Z_p$, $H:\{0,1\}^*\rightarrow G$ are secure hash functions, $F: Z_p\times Z_p\rightarrow Z_p$ is a pseudorandom function, E is a secure symmetric-key encryption algorithm (e.g., AES), ρ is an upper limit of authentication token requests made by a user in an epoch, ϕ is an upper limit that a user fails to pass identity servers' authentication, t denotes the threshold, and n denotes the total number of identity servers.

MasterKeyGen. With the public parameters, an identity server $\mathcal{IS}_i$ generates its master secret share msk_i by performing the distributed secret sharing protocol which is shown in Algorithm 1 [20], [46], [47], [48], where the master secret $msk=s$, the correspond public master key $PMSK=PS$, $\mathcal{IS}_i$'s master secret share $msk_i=s_i$ and the corresponding public master share $PMSK_i=PS_i$.

Register. A mobile user $\mathcal{U}$ registers with identity servers as follows.

- $\mathcal{U}$ generates a user identity $ID_{\mathcal{U}}$ and a human-memorable password $psw_{\mathcal{U}}$. $\mathcal{U}$ randomly chooses $r\in Z_p^*$, computes $psw_{\mathcal{U}}^*=rH(psw_{\mathcal{U}})$, and sends $(ID_{\mathcal{U}},psw_{\mathcal{U}}^*)$ to all identity servers.
- For $i\in[1,n]$, $\mathcal{IS}_i$ first checks whether $ID_{\mathcal{U}}$ exists in its local storage, if yes, $\mathcal{IS}_i$ informs $\mathcal{U}$ that the user identity is duplicate; Otherwise, $\mathcal{IS}_i$ stores $ID_{\mathcal{U}}$ and determines the group number of $\mathcal{U}$. Here, we assume $\mathcal{U}$ is the first member of her/his group, and identity servers need to generate a new server-side key for this group of users.
- $\mathcal{IS}_i$ generates the server-side key share by performing the distributed secret sharing protocol which is shown in Algorithm 1, where $a_{i,0},a_{i,1},\ldots,a_{i,t-1}$ are newly chosen and are different from those in generating the master secret share, the server-side key $k=s$, the corresponding public server-side key $PK=PS$, IS_i's server-side key share $k_i=s_i$ and the corresponding public server-side share $PK_i=PS_i$.
- $\mathcal{IS}_i$ computes $\sigma_i^*=k_i\cdot psw_{\mathcal{U}}^*$ and sends it to $\mathcal{U}$.
- $\mathcal{U}$ checks the validity of σ_i^* by checking

$$e(\sigma_i^*,P)=e(psw_{\mathcal{U}}^*,PK_i).$$

After receiving t valid signatures (denoted by $\{\sigma_1^*, \sigma_2^*,\ldots,\sigma_t^*\}$), $\mathcal{U}$ computes

$$\omega_\eta=\prod_{1\leq\iota\leq t,\iota\neq\eta}\frac{\iota}{\iota-\eta}, \tag{1}$$

$$\sigma_{\mathcal{U}}=r^{-1}\sum_{\eta=1}^{t}\omega_\eta\sigma_\eta^*. \tag{2}$$

$\mathcal{U}$ verifies the correctness of $\sigma_{\mathcal{U}}$ by checking

$$e(\sigma_{\mathcal{U}},P)=e(H(psw_{\mathcal{U}}),PK).$$

Here, due to Lagrange interpolation, $\sigma_{\mathcal{U}}=kH(psw_{\mathcal{U}})$ is a signature of $psw_{\mathcal{U}}$ under the server-side key k and is generated using the BLS signature [49].
- $\mathcal{U}$ computes a servers-derived password $sp_{\mathcal{U}}$ as

$$sp_{\mathcal{U}}=F(h(\sigma_{\mathcal{U}}),psw_{\mathcal{U}}).$$

- For $i=1,2,\cdots,n$, $\mathcal{U}$ computes $sp_i=\hbar(sp_{\mathcal{U}}||i)$ and sends sp_i to $\mathcal{IS}_i$.
- $\mathcal{IS}_i$ stores sp_i and initiates two counter $\rho_{\mathcal{U}}=0$ and $\phi_{\mathcal{U}}=0$.
- $\mathcal{U}$ deletes $\sigma_{\mathcal{U}}$, r, $sp_{\mathcal{U}}$, and sp_i; $\mathcal{IS}_i$ deletes σ_i^*, and securely stores $ID_{\mathcal{U}}$, sp_i, k_i, $\rho_{\mathcal{U}}$, and $\phi_{\mathcal{U}}$ for subsequently authenticating $\mathcal{U}$.

9. If the adversary only uses a certain identity, the identity servers would realize the misbehavior that the user is trying to sign-on very frequently within a short period of time.

- For the other users in the same group, $\mathcal{IS}_i$ utilizes k_i to generate the signatures for them. For the first user in a different group, all identity servers repeat the above steps to generate new server-side key shares for her/him.

After the *Registration* phase, a mobile user only needs to memorize her/his password for authentication.

Phase 2. Sign-On. In this phase, a mobile user requests an authentication token from the identity servers. Note that $\mathcal{U}$ cannot compute the servers-derived password $sp_{\mathcal{U}}$ without the aid of identity servers. Without $sp_{\mathcal{U}}$, $\mathcal{U}$ cannot compute the authentication credentials $sp_1, sp_2, \ldots, sp_n$ to pass the identity servers' authentications. Therefore, in the *Sign-on* phase, $\mathcal{U}$ first interacts with identity servers to retrieve $sp_{\mathcal{U}}$. Then, $\mathcal{U}$ computes the credentials to identify herself/himself. Once the authentication passes, identity servers sends $\mathcal{U}$ an authentication token share.

Step 1. Retrieving the servers-derived password $sp_{\mathcal{U}}$.

RetriSerPsw.

- $\mathcal{U}$ inputs the password $psw_{\mathcal{U}}$, randomly chooses $\alpha \in Z_p^*$, and computes $psw_{\mathcal{U}}^* = \alpha \cdot H(psw_{\mathcal{U}})$. $\mathcal{U}$ sends $\{ID_{\mathcal{U}}, psw_{\mathcal{U}}^*\}$ to $\mathcal{IS}_i$ for $i = 1, 2, \ldots, n$.
- After receiving $ID_{\mathcal{U}}$, $\mathcal{IS}_i$ checks $\rho_{\mathcal{U}} \leq \rho$ and $\phi_{\mathcal{U}} \leq \phi$, if the checking fails, it aborts; Otherwise, it retrieves the corresponding server-side key share k_i, computes $\sigma_i^* = k_i \cdot psw_{\mathcal{U}}^*$, increments $\rho_{\mathcal{U}}$ by 1, randomly chooses $r_i \in Z_p$, and sends $\{\sigma_i^*, r_i\}$ to $\mathcal{U}$.
- After receiving $\{\sigma_i^*, r_i\}$, $\mathcal{U}$ verifies its validity by checking
$$e(\sigma_i^*, P) = e(psw_{\mathcal{U}}^*, PK_i).$$
If the checking fails, $\mathcal{U}$ rejects.
- After receiving t valid signatures (for the sake of brevity, we denote these signatures by $\{\sigma_1^*, \sigma_2^*, \ldots, \sigma_t^*\}$), $\mathcal{U}$ computes $\xi_\kappa = \prod_{1 \leq \iota \leq t, \iota \neq \kappa} \frac{\iota}{\iota - \kappa}$ and $\sigma_{\mathcal{U}} = \alpha^{-1} \sum_{\kappa=1}^{t} \xi_\kappa \sigma_\kappa^*$. $\mathcal{U}$ verifies the correctness of $\sigma_{\mathcal{U}}$ by checking $e(\sigma_{\mathcal{U}}, P) = e(H(psw_{\mathcal{U}}), PK)$.
- $\mathcal{U}$ deletes $psw_{\mathcal{U}}^*$, α, σ_i^*; $\mathcal{IS}_i$ deletes $psw_{\mathcal{U}}^*$.

Step 2. Authentication and authentication token generation.

Authentication. $\mathcal{U}$ is authenticated by identity servers and requests an authentication token.

- $\mathcal{U}$ computes $sp_{\mathcal{U}} = F(h(\sigma_{\mathcal{U}}), psw_{\mathcal{U}})$ and $sp_i = \hbar(sp_{\mathcal{U}}||i)$, and sends $ESP_i = E(sp_i, r_i)$ to $\mathcal{IS}_i$.
- After receiving ESP_i, $\mathcal{IS}_i$ decrypts it and obtains r_i. If the decryption fails or r_i is not the one it sent before, $\mathcal{IS}_i$ aborts and increments $\phi_{\mathcal{U}}$ by 1; Otherwise, $\mathcal{IS}_i$ computes $Aut_i = msk_i \cdot H(Token)$, where $Token$ denotes a message that may contain $\mathcal{U}$'s information/attributes, expiration time, a policy that would control the nature of access, and other auxiliary information. $\mathcal{IS}_i$ computes $EAutToken_i = E(sp_i, Aut_i||Token)$, and sends $EAutToken_i$ to $\mathcal{U}$.
- After receiving $EAutToken_i$, $\mathcal{U}$ decrypts it and obtains Aut_i and $Token$. $\mathcal{U}$ verifies the validity of Aut_i by checking $e(Aut_i, P) = e(H(Token), PMSK_i)$. If the checking fails, she/he rejects.
- After receiving t valid signatures (for the sake of brevity, we denote these signatures by $\{Aut_1, Aut_2, \ldots, Aut_t\}$), $\mathcal{U}$ computes $\chi_\nu = \prod_{1 \leq \varepsilon \leq t, \varepsilon \neq \nu} \frac{\varepsilon}{\varepsilon - \nu}$ and $Aut_{\mathcal{U}} = \sum_{\nu=1}^{t} \chi_\nu Aut_\nu$. $\mathcal{U}$ verifies the correctness of $Aut_{\mathcal{U}}$ by checking $e(Aut_{\mathcal{U}}, P) = e(H(Token), PMSK)$.
- $\mathcal{U}$ deletes $sp_{\mathcal{U}}$, sp_i, ESP_i, r_i, Aut_i, and $EAutToken_i$; $\mathcal{IS}_i$ deletes ESP_i, r_i, Aut_i, $EAutToken_i$.
- $AutToken_{\mathcal{U}} = \{Token, Aut_{\mathcal{U}}\}$ serves as the $\mathcal{U}$'s authentication token.

Phase 3. Key Renewal. In this phase, each identity server renews its master secret share to resist the leakage of shares. It renews security protection and thwarts the adversary who can perpetually compromise identity servers over a long period of time. The renewal of master secret shares is executed only once in an epoch.

RenewShare. For each identity server $\mathcal{IS}_i$, $i = 1, 2, \ldots, n$, at the end of an epoch it renews its master secret share msk_i as follows.

- $\mathcal{IS}_i$ randomly selects a polynomial $l_i(x) = b_{i,1}x + b_{i,2}x^2 + \cdots + b_{i,t-1}x^{t-1}$ over Z_p with degree at most $t-1$.
- For $\epsilon = 1, 2, \ldots, t-1$, $\mathcal{IS}_i$ sends $b_{i,\epsilon}P$ to all other identity servers. $\mathcal{IS}_i$ sends $l_i(j) \bmod p$ secretly to $\mathcal{IS}_j$ for $j = 1, 2, \ldots, n; j \neq i$.
- After receiving $l_j(i)$, $\mathcal{IS}_i$ checks
$$l_j(i)P = \sum_{\gamma=1}^{t-1} i^\gamma b_{j\gamma}P. \tag{3}$$
If the checking fails, it aborts.
- $\mathcal{IS}_i$ computes a new master secret key share $msk_i^{'}$ as
$$msk_i^{'} = msk_i + \sum_{j=1}^{n} l_j(i). \tag{4}$$
The corresponding public master secret share is $PMSK_i^{'} = msk_i^{'}P$.
- $\mathcal{IS}_i$ deletes $l_i(x)$, $b_{i,\epsilon}$, $l_i(j)$, and msk_i.

Finally, $\mathcal{IS}_i$ resets $\rho_{\mathcal{U}}$ and $\phi_{\mathcal{U}}$. A new epoch begins.

6.2 Correctness Proof

In *Regiser*, the authentication credential $sp_{\mathcal{U}}$ has the form of $sp_{\mathcal{U}} = F(h(\sigma_{\mathcal{U}}), psw_{\mathcal{U}})$, where $\sigma_{\mathcal{U}} = kH(psw_{\mathcal{U}})$ is essentially the BLS signature [49], which is a deterministic signature algorithm. Therefore, if $\mathcal{U}$ takes the correct $psw_{\mathcal{U}}$ as the input in *RetriSerPsw*, she/he can compute $sp_{\mathcal{U}}$ that is the same as the one computed in *Register*. With $sp_{\mathcal{U}}$, $\mathcal{U}$ is able to compute sp_i and to pass $\mathcal{IS}_i$'s authentication.

Note that $msk = \sum_{i=1}^{n} a_{i,0} = \sum_{i=1}^{n} f_i(0)$. Assume that the renewed master secret is $msk^{'} = \sum_{i=1}^{n} f_i^{'}(0)$. Since $f_i^{'}(x) = f_i(x) + l_i(x)$, we have $msk^{'} = \sum_{i=1}^{n} f_i^{'}(0) = \sum_{i=1}^{n} f_i(0) + l_i(0)$. As described above, $l_i(0) = 0$, we further obtain $msk^{'} = \sum_{i=1}^{n} f_i(0) + l_i(0) = \sum_{i=1}^{n} f_i(0) = msk$. Therefore, after executing *RenewShare*, each identity server renews its master secret share without changing the master secret key msk.

6.3 Further Discussion

In PROTECT, the *Sign-on* phase requires four rounds of communications between a mobile user and an identity server, which is shown in Fig. 4.

Inspired by PASTA [15], we may reduce the communication overhead from four rounds to two rounds, as shown in Fig. 5: identity servers do not have to check the validity of

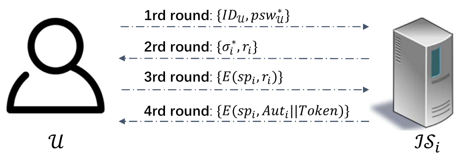

1rd round: $\{ID_{\mathcal{U}}, psw^*_{\mathcal{U}}\}$
2rd round: $\{\sigma^*_i, r_i\}$
3rd round: $\{E(sp_i, r_i)\}$
4rd round: $\{E(sp_i, Aut_i||Token)\}$
$\mathcal{U}$
$\mathcal{IS}_i$

Fig. 4. Four rounds of communications in PROTECT.

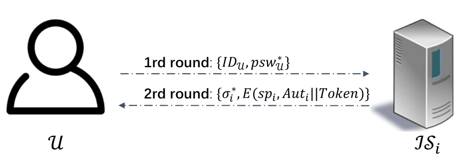

1rd round: $\{ID_{\mathcal{U}}, psw^*_{\mathcal{U}}\}$
2rd round: $\{\sigma^*_i, E(sp_i, Aut_i||Token)\}$
$\mathcal{U}$
$\mathcal{IS}_i$

Fig. 5. Two rounds of communications.

the user $\mathcal{U}$'s password and an identity server $\mathcal{IS}_i$ only encrypts a token by using the authentication credential $sp_i = \hbar(sp_{\mathcal{U}}||i)$ which is received in the *Registration* phase and used to verify the validity of $psw_{\mathcal{U}}$. The key observation here is that only if $\mathcal{U}$ has used the correctness password in the first round, can she/he compute the correct $sp_{\mathcal{U}}$ and decrypt the ciphertexts to collect t token shares, and finally recover the authentication token.

However, as part of our contribution, we point out that such an improvement in communication efficiency would introduce new security issues such that it is vulnerable to *online password testing attacks* defined in Section 4.2. Specifically, assuming there are lots of registered users with identity $\overrightarrow{ID} = \{ID_{\mathcal{U}_1}, ID_{\mathcal{U}_2}, \ldots, ID_{\mathcal{U}_\Theta}\}$. The adversary calculates a set of password candidates $\overrightarrow{PC} = \{pswc_1, pswc_2, \ldots, pswc_\Delta\}$. In reality, the number of password candidates is far less than the number of mobile users in the system, i.e., $|\overrightarrow{PC}| \ll |\overrightarrow{ID}|$. The adversary can perform OPTA to retrieve the users' identities and the corresponding passwords, if at least one user utilizes a password existing in the set. It is hard to detect OPTA for identity servers in the 2-round scheme [15], since identity servers cannot determine whether a password is repeated and also cannot get the authentication result, they cannot distinguish $\mathcal{A}$ from a valid user. By comparison, such attack can be easily resisted in PROTECT (i.e., 4-round scheme), since identity servers can obtain the authentication result after the 3rd round communication. With the authentication results, identity servers can lock an account if multiple authentications fail such that $\mathcal{A}$ cannot test all passwords in the list for all users.

Essentially, the reason the 2-round scheme described above is vulnerable to OPTA is that the identity servers are considered as oracles and cannot forbid a mobile user to request multiple authentication tokens. If the identity servers cannot obtain the authentication result, they cannot detect the attacks. By comparison, the 4-round scheme is secure against OPTA: the identity servers can reject a user if she/he has failed to sign in multiple times, i.e., the number of failures for login request made by a user can be strictly restricted.

In PROTECT, if $\phi_{\mathcal{U}}$ reaches the upper limit ϕ, further authentication queries by $\mathcal{U}$ would be ignored by the identity servers, until the end of the epoch. This can prevent PROTECT against attacks from an external adversary that impersonates $\mathcal{U}$ to increase $\rho_{\mathcal{U}}$ and stops $\mathcal{U}$ from requesting authentication tokens from identity servers. We also consider an alternative mechanism for alleviating the above attack while ensuring the availability of single-sign-on authentication for users. Instead of directly locking $\mathcal{U}$'s account after ϕ reaches, identity servers utilize an exponential delay mechanism as follows: each time identity servers receive a query from $\mathcal{U}$ (after $\phi_{\mathcal{U}} > \phi$), an artificial delay t_D is introduced before key servers respond to $\mathcal{U}$'s query. Initially, t_D could be small and doubles this quantity after each query. This delay would be removed in the next epoch.

7 Security Analysis

We would not consider adversaries who may interfere with the *Registration* phase and *Key renewal* phase in PROTECT. In these two phases, all communications are over secure channels, and the interaction messages are well protected. However, the *Sign-on* phase is completely under the control of adversaries, where the interaction messages between a mobile user and identity servers can be arbitrarily intercepted and modified by adversaries. We analyze the security of PROTECT from three aspects.

7.1 External Adversaries

An external adversary may intercept the interaction messages between a user and identity servers to retrieve a user's password by performing the dictionary guessing attack. An external adversary may also collect the server-side key shares and master secret shares by compromising identity servers to retrieve a user's password or forge an authentication token. Other external adversaries can be considered as either a malicious user or a misbehavior identity server. In the following, we analyze the security of PROTECT against external adversaries with respect to the two strategies.

First, assuming an external adversary targets to retrieve a victim $\mathcal{U}^*$'s password $psw_{\mathcal{U}^*}$. Recall Fig. 4, in the *Sign-on* phase, the adversary can collect $ID_{\mathcal{U}^*}$, $psw^*_{\mathcal{U}^*}$, $\{\sigma^*_i, E(sp_i, sp_i), E(sp_i, Aut_i||Token)\}_{i=1,2,\ldots,n}$ from the communication channels between $\mathcal{U}^*$ and the identity servers. The security against external adversaries relies on the security of generation of servers-derived password $sp_{\mathcal{U}^*}$. In Section 7.2, we will prove the security of generation of $sp_{\mathcal{U}^*}$ against a stronger adversary (has more information than the external adversary) who targets to compromise $psw_{\mathcal{U}^*}$.

Second, assume that the external adversary targets to retrieve the master secret key msk by collecting any t shares of msk. However, the shares of msk are periodically updated. We prove that the adversary, who collects t shares of msk generated in two different epochs which are denoted by $\{msk_1, msk_2, \ldots, msk_{t'}, msk^\star_{t'+1}, msk^\star_{t'+2}, \ldots, msk^\star_t\}$, $(1 < t' < t < n)$, cannot retrieve msk. The master secret has the form

$$msk = \sum_{i=1}^{t} \omega_i msk_i = \sum_{i=1}^{t} \omega_i msk^\star_i \tag{5}$$

$$= \sum_{i=1}^{t} \left(\prod_{\substack{1\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} f_j(i) \right) \tag{6}$$

$$= \sum_{i=1}^{t} \left(\prod_{\substack{1\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} (f_j(i) + l_j(i)) \right). \tag{7}$$

Here, the only way the adversary retrieves msk is to compute

$$\sum_{i=1}^{t'} \omega_i msk_i + \sum_{i=t'+1}^{t} \omega_i msk_i^\star \tag{8}$$

$$\begin{aligned} &= \sum_{i=1}^{t'} \left(\prod_{\substack{1\le\eta\le t' \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} f_j(i) \right) \\ &+ \sum_{i=t'+1}^{t} \left(\prod_{\substack{t'\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} f_j(i) + \sum_{j=1}^{n} l_j(i) \right) \end{aligned} \tag{9}$$

$$\begin{aligned} &= \sum_{i=1}^{t} \left(\prod_{\substack{1\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} f_j(i) \right) \\ &+ \sum_{i=t'+1}^{t} \left(\prod_{\substack{t'\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} l_j(i) \right) \end{aligned} \tag{10}$$

$$= msk + \sum_{i=t'+1}^{t} \left(\prod_{\substack{t'\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} l_j(i) \right). \tag{11}$$

Since the adversary cannot collect $l_j(i)$ for $j = 1, 2, \ldots, n$ and $i = t' + 1, \ldots, t$, he cannot compute

$$\begin{aligned} msk &= \sum_{i=1}^{t'} \omega_i msk_i + \sum_{i=t'+1}^{t} \omega_i msk_i^\star \\ &- \sum_{i=t'+1}^{t} \left(\prod_{\substack{t'\le\eta\le t \\ \eta\ne i}} \frac{\eta}{\eta - i} \right) \left(\sum_{j=1}^{n} l_j(i) \right). \end{aligned} \tag{12}$$

Therefore, PROTECT is secure against external adversaries.

7.2 Compromised Identity Server(s)

We assume that an adversary $\mathcal{A}$ can compromise any t' identity servers to extract the master secret shares, servers-side key shares, and server-side credentials, where $t' \le t$ and $\mathcal{A}$ can be a malicious identity server itself. $\mathcal{A}$'s target is to retrieve a victim $\mathcal{U}^*$'s password. Note that the server-side credential on the ith identity server is $sp_i = \hbar(sp_{\mathcal{U}^*}||i)$ which is computed on the servers-derived password $sp_{\mathcal{U}^*}$. Due to the preimage resistance of hash function $\hbar$, it is computationally infeasible to compute $sp_{\mathcal{U}^*}$ given sp_i for $i \in [1, n]$. The most effective way to compromise $sp_{\mathcal{U}^*}$ for $\mathcal{A}$ is to compromise the generation of $sp_{\mathcal{U}^*}$. The security of $sp_{\mathcal{U}^*}$ is captured by the Lemmas 1 and 2, which ensures that PROTECT can resist $\mathcal{A}$.

Lemma 1. *(Unpredictability) In PROTECT, a servers-derived password is unpredictable, which means that given a user $\mathcal{U}^*$'s password $psw_{\mathcal{U}^*}$, an adversary cannot predict the corresponding servers-derived password $sp_{\mathcal{U}^*}$, even if he can compromise up to t' identity servers.*

Proof. We define an unpredictability game.

Unpredictability Game.

1) An Environment $\mathcal{E}$ initiates PROTECT, generates a server-side key k, generates n shares $\{k_1, k_2, \ldots, k_n\}$ of k using the (t, n)-threshold secret sharing protocol described in Section 6, and generates the corresponding public parameter PP. $\mathcal{E}$ sends PP to a simulator $\mathcal{S}$. $\mathcal{S}$ then forwards PP to an adversary $\mathcal{A}$.
2) $\mathcal{A}$ chooses t' indexes $T' = \{i_1, i_2, \ldots, i_{t'}\}$, and sends T' to $\mathcal{S}$. $\mathcal{S}$ forwards T' to $\mathcal{E}$. $\mathcal{E}$ responses with $\{k_{i_1}, k_{i_2}, \ldots, k_{i_{t'}}\}$. $\mathcal{S}$ forwards it to $\mathcal{A}$ and sets $q_1, q_2, \ldots, q_n = 0$.
3) $\mathcal{A}$ randomly chooses $\tilde{x} \in Z_p^*, r \in Z_p^*$, and computes $\tilde{c} = r \cdot H(\tilde{x})$.
4) For $i \in [1, n]/T'$, $\mathcal{A}$ sends a query of $\sigma_i = k_i \tilde{c}$ to $\mathcal{S}$.
5) Upon receiving the query, $\mathcal{S}$ sets $q_i = q_i + 1$, and forwards the query to $\mathcal{E}$.
6) $\mathcal{E}$ computes σ_i and sends it to $\mathcal{S}$. $\mathcal{S}$ forwards σ_i to $\mathcal{A}$.
7) $\mathcal{A}$ repeats the above query on different i. Finally, $\mathcal{A}$ outputs $\tilde{\sigma}$.
8) $\mathcal{A}$ wins, iff
   - $\sum_{i=1}^{n} q_i < t - t'$, and
   - $\tilde{\sigma}$ is a valid signature on $\tilde{c}$.

We prove that if $\mathcal{A}$ can win the unpredictability game with a non-negligible probability, $\mathcal{S}$ can break the existential unforgeability of BLS signature [49] with the same probability. Specifically, during the interactions, $\mathcal{S}$ can collect $\{k_\imath, \sigma_\imath\}_{\imath=T'}$, $\{\sigma_i\}_{i\in[1,n]/T'}$ and $\tilde{\sigma}$. Note that no matter what strategy $\mathcal{A}$ takes, it can effectively do no better than the strategy that $\sum_{i=1}^{n} q_i + t' = t - 1$. As such, $\mathcal{S}$ can collect $t - 1$ signatures under $t - 1$ server-side key shares which are denoted by $\sigma_{i_1'}, \sigma_{i_2'}, \ldots, \sigma_{i_{t-1}'}$ and a signature under the server-side key (which is generated by $\mathcal{A}$). Assuming $\mathcal{A}$ targets to forge a signature on $\tilde{c}$ under the tth server-side key share, he outputs

$$\sigma_t = \left(\tilde{\sigma} - \left(\sum_{\eta=i_1'}^{i_{t-1}'} \omega_\eta \sigma_\eta \right) \right) \cdot \frac{1}{\omega_t}, \tag{13}$$

as the forged BLS signature, where $\omega_\eta = \prod_{i_1' \le \iota \le i_{t-1}', \iota \ne \eta} \frac{\iota}{\iota - \eta}$ and $\omega_t = \prod_{i_1' \le \iota \le i_{t-1}'} \frac{\iota}{\iota - t}$. □

Lemma 2. *(Obliviousness) In PROTECT, the generation of a servers-derived password is oblivious, which means that given a user $\mathcal{U}^*$'s servers-derived password $sp_{\mathcal{U}^*}$, an adversary cannot learn anything about the corresponding password $psw_{\mathcal{U}^*}$ (e.g., the hash value of $psw_{\mathcal{U}^*}$).*

Proof. Intuitively, in PROTECT, an adversary can only collect some blinded messages on different passwords which are random elements in G due to the random choice of r. Therefore, the adversary cannot learn

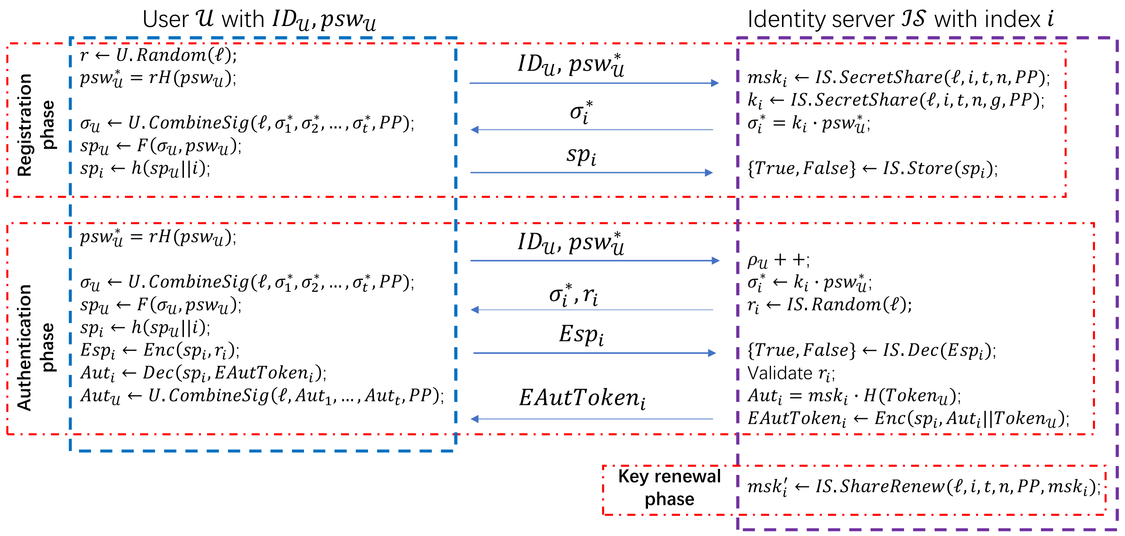

User $\mathcal{U}$ with $ID_{\mathcal{U}}, psw_{\mathcal{U}}$
Identity server $\mathcal{IS}$ with index i
Registration phase
$r \leftarrow U.Random(\ell)$;
$psw_{\mathcal{U}}^* = rH(psw_{\mathcal{U}})$;
$\sigma_{\mathcal{U}} \leftarrow U.CombineSig(\ell, \sigma_1^*, \sigma_2^*, \dots, \sigma_t^*, PP)$;
$sp_{\mathcal{U}} \leftarrow F(\sigma_{\mathcal{U}}, psw_{\mathcal{U}})$;
$sp_i \leftarrow h(sp_{\mathcal{U}}||i)$;
$ID_{\mathcal{U}}, psw_{\mathcal{U}}^*$
σ_i^*
sp_i
$msk_i \leftarrow IS.SecretShare(\ell, i, t, n, PP)$;
$k_i \leftarrow IS.SecretShare(\ell, i, t, n, g, PP)$;
$\sigma_i^* = k_i \cdot psw_{\mathcal{U}}^*$;
$\{True, False\} \leftarrow IS.Store(sp_i)$;
Authentication phase
$psw_{\mathcal{U}}^* = rH(psw_{\mathcal{U}})$;
$\sigma_{\mathcal{U}} \leftarrow U.CombineSig(\ell, \sigma_1^*, \sigma_2^*, \dots, \sigma_t^*, PP)$;
$sp_{\mathcal{U}} \leftarrow F(\sigma_{\mathcal{U}}, psw_{\mathcal{U}})$;
$sp_i \leftarrow h(sp_{\mathcal{U}}||i)$;
$Esp_i \leftarrow Enc(sp_i, r_i)$;
$Aut_i \leftarrow Dec(sp_i, EAutToken_i)$;
$Aut_{\mathcal{U}} \leftarrow U.CombineSig(\ell, Aut_1, \dots, Aut_t, PP)$;
$ID_{\mathcal{U}}, psw_{\mathcal{U}}^*$
σ_i^*, r_i
Esp_i
$EAutToken_i$
$\rho_{\mathcal{U}}++$;
$\sigma_i^* \leftarrow k_i \cdot psw_{\mathcal{U}}^*$;
$r_i \leftarrow IS.Random(\ell)$;
$\{True, False\} \leftarrow IS.Dec(Esp_i)$;
Validate r_i;
$Aut_i = msk_i \cdot H(Token_{\mathcal{U}})$;
$EAutToken_i \leftarrow Enc(sp_i, Aut_i||Token_{\mathcal{U}})$;
Key renewal phase
$msk_i' \leftarrow IS.ShareRenew(\ell, i, t, n, PP, msk_i)$;

Fig. 6. Implementation of PROTECT.

anything on psw (e.g., $H(psw)$). Note that this holds, even if the adversary knows the server-side key k [27].

We define an obliviousness game in the following.

Obliviousness Game.

1) An Environment $\mathcal{E}$ initiates PROTECT, generates a server-side key k, generates n shares $\{k_1, k_2, \dots, k_n\}$ of k using the (t, n)-threshold secret sharing protocol described in Section 6, and generates the corresponding public parameter PP. $\mathcal{E}$ sends PP to a simulator $\mathcal{S}$. $\mathcal{S}$ then forwards PP to an adversary $\mathcal{A}$.
2) $\mathcal{A}$ randomly chooses $\tilde{c} \in Z_p$ and sends $\tilde{c}$ to $\mathcal{S}$. $\mathcal{S}$ forwards $\tilde{c}$ to $\mathcal{E}$.
3) $\mathcal{E}$ randomly chooses $\tilde{r}$, computes $\sigma_{\tilde{c}} = \tilde{r}kH(\tilde{c})$, and sends $sp_{\tilde{c}} = F(h(\sigma_{\tilde{c}}), \tilde{c})$ to $\mathcal{S}$. $\mathcal{S}$ forwards $sp_{\tilde{c}}$ to $\mathcal{A}$.
4) $\mathcal{A}$ can repeat steps 2) and 3) at most $poly(\ell)$ times, where ℓ is the security parameter.
5) $\mathcal{A}$ randomly chooses $c_0 \in Z_p$ and $c_1 \in Z_p$, and sends c_1 and c_2 to $\mathcal{S}$. $\mathcal{S}$ forwards them to $\mathcal{E}$.
6) $\mathcal{E}$ randomly chooses $b \in \{0, 1\}$ and $r \in Z_p^*$. Then
   - if $b = 0$, $\mathcal{E}$ computes $sp_{c_b} = F(h(krH(c_b)), c_b)$,
   - else, $\mathcal{E}$ randomly chooses a function $\mathcal{F} \in Func_{Z_p}$ and computes $sp_{c_b} = \mathcal{F}(c_b)$, where $Func_{Z_p}$ is the set of all functions mapping Z_p to Z_p.

   $\mathcal{E}$ sends sp_{c_b} to $\mathcal{S}$. $\mathcal{S}$ forwards sp_{c_b} to $\mathcal{A}$.
7) $\mathcal{A}$ outputs b^*.
8) $\mathcal{A}$ wins, iff $b^* = b$.

The probability of $\mathcal{A}$ that wins the obliviousness game is defined by $\Pr_{\mathcal{A}}^{Obli,\ell}$. We prove that if $|\Pr_{\mathcal{A}}^{Obli,\ell} - 1/2|$ is non-negligible, the simulator $\mathcal{S}$ can break the security of pseudo-random function F with $|\Pr_{\mathcal{A}}^{Obli,\ell} - 1/2|$. In particular, if $\mathcal{A}$ outputs b^* and wins the obliviousness game, $\mathcal{S}$ can output the same bit to distinguish $F(h(krH(c_b)), c_b)$ from $\mathcal{F}(c_b)$. □

7.3 Malicious Mobile User

A malicious user may perform two types of attacks to break the security of PROTECT.

1) He can perform the online DGA to retrieve a victim's password.
2) He can perform impersonation attacks to pass the identity servers' authentication and obtain an authentication token.

In PROTECT, we utilize a rate-limiting mechanism to limit the number of the requests of authentication token for a user in an epoch. Therefore, a malicious user cannot retrieve the victim's password by performing the online dictionary guessing attack. Moreover, identity servers also need to verify the user's credentials (i.e., sp_i for $i = 1, 2, \dots, n$). Without the correct password, it is computationally infeasible to compute the corresponding credentials. Therefore, PROTECT is able to resist impersonation attacks.

8 Implementation and Evaluation

8.1 Implementation of PROTECT

We implement PROTECT by using C++ language and MIRACL library 5.6.1 version.[10] For the algorithms executed by mobile users, all the experiments are conducted on a smartphone (HUAWEI MT2-L01) with Android 4.2.2 system, a Kirin 910 CPU with memory 1250 MB. For the algorithms executed by identity servers, all the experiments are conducted on a laptop with macOS system, an Intel Core i7 CPU, and 16 GB DDR3 of RAM. We select $\ell = 80$ bits as the security level (i.e., the security guarantee is equivalent to AES with an 80-bit key). The pairing-friendly curve is MNT curve.[11]

A schematic of PROTECT implementation is shown in Fig. 6. We elaborate on the implementation using functions. For clarity, we prefix calls of functions with U when they are made by the user and with IS when they are made by the identity server. With the security level, the system parameters PP are determined and we do not show the function that is used to generate PP in Fig. 6.

10. "Crypto library overview," https://libraries.docs.miracl.com

11. Actually, MNT curve is used to construct Type-3 pairings, i.e., $e : G_2 \times G_1 \rightarrow G_T$, where G_1 and G_2 are additive groups. Since Type-3 pairings are more efficient than Type-1 pairings when we choose the same security level, in the implementation of PROTECT, we utilize Type-3 pairing to analyze the performance. This would not affect the feasibility and security of PROTECT.

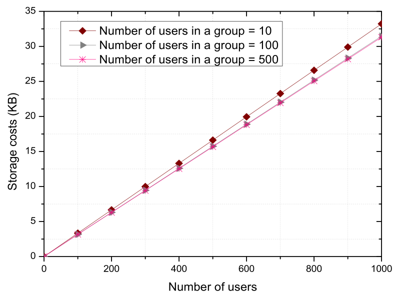


Fig. 7. Storage costs on the identity server of PROTECT.

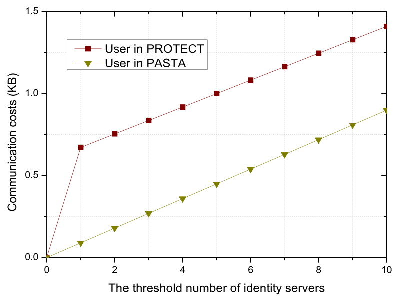


Fig. 9. Communication costs on the mobile user.

Registration phase. With the system parameters, a master secret key is shared among n identity servers, and each of them has a master secret share msk_i. Each identity server calls function $SecretShare()$ shown in Algorithm. 1 to complete this process. A user $\mathcal{U}$ calls $Random()$ to generate a random number r and blinds the password to obtain $psw_{\mathcal{U}}$. Then the user sends $psw^*_{\mathcal{U}}$ as well as her/his identity $ID_{\mathcal{U}}$ to all identity servers, which initiates a registration. Each identity server first calls $SecretShare()$ to generate a server-side key for a group of users and generates a server-side key share k_i. Then each identity server signs $psw^*_{\mathcal{U}}$ using the server-side key share k_i and responses the user with the signature. With a threshold number t of valid signatures, $\mathcal{U}$ calls $CombineSig()$ to combine these signatures and retrieve a signature of its password under the server-side key. Then $\mathcal{U}$ computes the corresponding servers-derived password $sp_{\mathcal{U}}$. $\mathcal{U}$ also computes a (SHA-256) hash value of $sp_{\mathcal{U}}$ and the index of each identity server, it serves as the server-side credential. After each identity server securely stores the credential, the registration is completed. The registration only needs to be executed once for each user.

Authentication phase. Anytime $\mathcal{U}$ needs to request an authentication token $Aut_{\mathcal{U}}$ from identity servers, she/he first retrieves the servers-derived password $psw_{\mathcal{U}}$ from identity servers as she/he did in the *Registration* phase. Each identity server calls $Random()$ to generate a random number which is used to resist relay attacks. Each identity server also records the number of $\mathcal{U}$'s requests and stop responding if the bound is reached. With $psw_{\mathcal{U}}$, $\mathcal{U}$ computes server-side credentials for identity servers, encrypts each credential using the credential itself, and sends each ciphertext to the corresponding identity server. We stress that $\mathcal{U}$ needs to send n credentials to n identity servers, since all identity servers have to check whether $\mathcal{U}$ passes the authentication. Each identity server decrypts the received ciphertext and generates a signature on the authentication token using msk_i. Then each identity server encrypts the signature as well as the token using the credential and sends the ciphertext to $\mathcal{U}$. Finally, $\mathcal{U}$ decrypts the received ciphertexts and retrieve the authentication token. Authentication between identity servers and a user may be executed multiple times (up to bound times for a user) in an epoch.

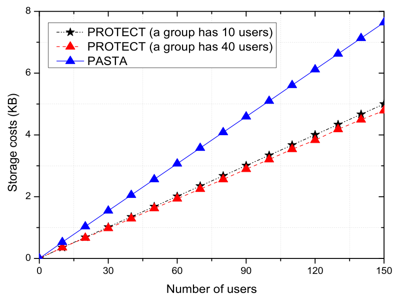


Fig. 8. Comparison of storage costs on the identity server.

Key renewal phase. At the end of an epoch, each identity server calls $ShareRenew()$ to renew the master secret share and server-side key share. This process only needs to be executed once in an epoch.

8.2 Storage Overhead

PROTECT does not require mobile users to store any digital message for the authentication. Mobile users only need to memorize their identity and passwords for authentication and requesting authentication tokens from identity servers. Therefore, PROTECT can be easily deployed by mobile users.

In this section, we focus on the storage overhead on identity servers. In PROTECT, apart from the system parameters, an identity server needs to store the master secret share, server-side key share, the number of authentication requests made by each user. We show the storage costs on the identity server of PROTECT in Fig. 7. By comparison, in Fig. 8 we show the storage costs of identity server of PASTA [15] with the same setting. Particularly, compared with PASTA, PROTECT is able to reduce around 40 percent storage costs on identity servers. Since in PASTA, different users have different server-side keys, this incurs heavy storage costs. In PROTECT, we resolve this tension by utilizing the hybrid mechanism, where a group of users uses the same server-side key but the number of token requests made by each user is limited. Therefore, identity servers only need to store a server-side key for each group of users, which reduces the storage overhead significantly.

8.3 Communication Overhead

PROTECT requires 4 rounds of communications between a user and identity servers. By comparison, PASTA [15] only requires 2 rounds of communications. However, the additional 2 rounds of communications in PROTECT ensure that identity servers can obtain the authentication result, which enables PROTECT to resist password testing attacks.

In the registration, the communication costs on the user side of PROTECT are the same as those of PASTA. The registration is a one-time operation, here we would not analyze the communication efficiency.

In Fig. 9, we show the communication costs of the user in the *Authentication* phase of PROTECT and PASTA, where

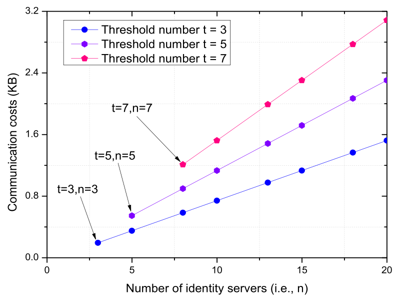


Fig. 10. Communication costs on the identity server side.

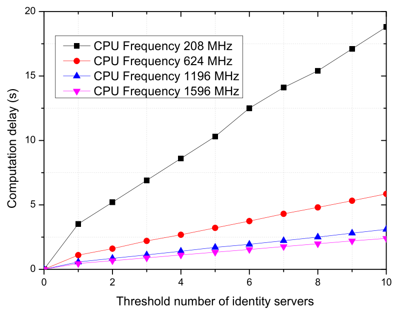


Fig. 11. Computation delay on the mobile user w.r.t. threshold number t.

the size of $Token_{\mathcal{U}}$ is set to 256 bits, and the total number of identity servers is set to 10. Compared with PASTA, PROTECT introduces more communication costs on the user side, these additional costs are caused by two parts. The first one is that the user needs to communicate with n identity servers in the first two rounds of communications to ensure the synchronization of all identity servers such that the total numbers of authentication requests made by the user on each identity server are consistent. By comparison, in PASTA, such synchronization is not required and the user only needs to communicate with t identity servers, which requires identity servers in PASTA bear a heavy storage costs to ensure the security. With the booming development of smartphones and mobile devices, the additional communication costs can be acceptable and would not become a bottleneck for mobile users in practice. In the *Authentication* phase of PROTECT, identity servers do not need to interact with each other. The communication costs on each identity server for an authentication are constant (i.e., divide the user's communication costs in PROTECT by the number of identity servers n).

In the *Key renewal* phase of PROTECT, all identity servers interact with each other to renew their master secret shares. Note that PROTECT *does not* introduce a trusted *dealer* to assist identity servers in renewing the master secret shares. The communication costs of an identity server to renew its master secret share are shown in Fig. 10.

In reality, the length of each epoch (i.e., the frequency of performing key renewal) depends on the system requirements, it can be pre-fixed according to the security parameter or dynamically adjusted with the state of systems protected by PROTECT. Since identity servers could be well equipped and keep online in PROTECT, the communication costs on the identity server side can be accepted and would not become a bottleneck for applications.

8.4 Computation Overhead

We first estimate the computation costs in terms of basic cryptographic operations. The corresponding notations are given in Table 1.

On the mobile user side, in the *Authentication* phase, a servers-derived password is retrieved, credentials used to pass the identity servers' authentication are calculated, and an authentication token is computed. The corresponding computation costs are $(4t+4)\cdot Pair_{G_T} + (2t+3)\cdot Mul_G + \ 2t\cdot Add_G + (t+3)\cdot Hash_G + (2t-2)\cdot Mul_{Z_p} + \ (4t-4)\cdot Add_{Z_p} + (t+1)\cdot Hash_{Z_p} + 2t\cdot Enc/Dec + C_F$, where t is the threshold number and n is the total number of identity servers. Note that the computation costs on the user side solely depend on the *threshold number* of identity servers, rather than the total number of identity servers. Fig. 11 shows the computation delay of a mobile user in the *Authentication* phase on different CPU frequencies of a smartphone. As shown in Fig. 11, the computational delay on the not-so-powerful smartphone is very short. Specifically, it only takes within 6s to complete an authentication and obtain a token for the user equipped with the smartphone whose frequency is 624 MHz, when the threshold number of identity servers $t = 10$. We stress that the computation capacity of current mainstream smartphones is much more powerful than that of the smartphone on which we conduct the experiments. Therefore, PROTECT can be well applied to mobile users.

The computation costs on the identity server side in the *Authentication* phase are constant, i.e., $Mul_G + Enc/Dec$. In the *Key renewal* phase, each identity server needs to compute a new master secret share to renew the security protection. The corresponding computation costs are $t(n-1)\cdot Mul_G + (n-1)(t-1)\cdot Add_G + \ (3n-2)(t-1)\cdot Mul_{Z_p} + (n(t-1)+1)\cdot Add_{Z_p}$. Figs. 12 and 13 show the computation delay of an identity server to renew its master secret share, we can see that renewing the master secret share on the identity server side takes within 50 ms when there are 20 identity servers and the threshold number $t = 10$.

In summary, according to the efficiency analysis and performance evaluation, we can observe that PROTECT is more efficient than PASTA [15] in terms of storage overhead. Furthermore, compared with PASTA, the authentication between the user and an identity server in PROTECT needs additional two rounds of communications, which

TABLE 1
Notation of Cryptographic Operations

Symbol	Operation	Symbol	Operation	Symbol	Operation
$Pair_{G_T}$	Computation of pairing e	Add_G	Group operation in G	$Hash_{Z_p}$	Hash a value into Z_p
Mul_G	Multiplication in G	$Hash_G$	Hash a value into G	Add_{Z_p}	Addition in Z_p
Enc/Dec	Symmetric-key encryption/decryption	Mul_{Z_p}	Multiplication in Z_p	C_F	Computing a PRF $F(\cdot)$

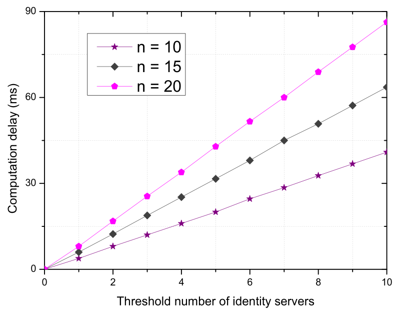


Fig. 12. Computation delay on the identity server w.r.t. the threshold number t.

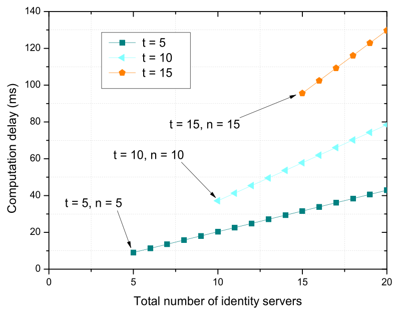


Fig. 13. Computation delay on the identity server w.r.t. the total number n.

also brings slight communication costs on the user side (about 0.5 KB). However, these extra costs are mainly for protecting PROTECT from password testing attacks. Furthermore, these costs would not introduce a heavy burden for current mobile devices. Therefore, these communication costs on mobile users can be, to a large extent, tolerated in mobile environments. The security analysis and performance evaluation demonstrate that renewing master secret share on the identity server side surely enhances the security guarantee without introducing heavy communication and computation costs.

9 Conclusion and Future Work

In this paper, we have proposed PROTECT, a password-based threshold single-sign-on authentication scheme for mobile users to resist off-line DGA and perpetual leakage of secrets on identity servers. We have proposed a hybrid mechanism and integrated it into PROTECT to efficiently thwart online DGA. We also have generalized and defined the online password testing attack and shown how it breaks the security of existing schemes. We have proved the security of PROTECT and conducted a comprehensive performance evaluation, which has demonstrated that PROTECT provides a stronger security guarantee and is more efficient in terms of storage overhead than existing schemes.

For the future work, we will investigate how to renew the authentication credentials on identity servers without requiring users' participation. We also notice that one can use the key components of PROTECT in other scenarios to enhance the security. In particular, a notion of "message-locked encryption" (MLE) was formalized [50] for the problem of saving storage costs in an encrypted data system (e.g., cloud storage systems [51], [52], [53]) by combining duplicate data across multiple users. Applying the key components of PROTECT in MLE may resist the brute-force attacks [48], [54], [55], [56] without the single-point-of-failure problem, which is also an interesting future direction to enhance the security of MLE.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant 2017YFB0802000, the National Natural Science Foundation of China under Grants 61872060, 61370203, Key Laboratory of Financial Mathematics of Fujian Province University (Putian University) under Grant JR201901, and the China Scholarship Council. The first author would like to thank the colleagues from BBCR lab, University of Waterloo, for their valuable discussions.

References

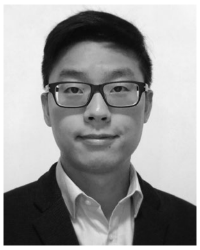
[1] H. Zhu, J. Hu, S. Chang, and L. Lu, "Shakein: Secure user authentication of smartphones with single-handed shakes," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2901–2912, Oct. 2017.
[2] P. Zhao *et al.*, "Understanding smartphone sensor and app data for enhancing the security of secret questions," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 552–565, Feb. 2017.
[3] W. Tang, K. Zhang, J. Ren, Y. Zhang, and X. Shen, "Flexible and efficient authenticated key agreement scheme for bans based on physiological features," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 845–856, Apr. 2019.
[4] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.
[5] Y. Zhang, C. Xu, N. Cheng, H. Li, H. Yang, and X. Shen, "Chronos +: An accurate blockchain-based time-stamping scheme for cloud storage," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2019.2947476.
[6] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Čapkun, "Smartphones as practical and secure location verification tokens for payments," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–15.
[7] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, Aug. 2018.
[8] L. Wu, J. Wang, K. R. Choo, and D. He, "Secure key agreement and key protection for mobile device user authentication," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 319–330, Feb. 2019.
[9] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.
[10] R. Chatterjee, A. Athayle, D. Akhawe, A. Juels, and T. Ristenpart, "pASSWORD tYPOS and how to correct them securely," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 799–818.
[11] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 33–38, Sep. 1994.
[12] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, 2008, pp. 75–88.
[13] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1242–1254.
[14] Y. Zhang, X. Lin, and C. Xu, "Blockchain-based secure data provenance for cloud storage," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2018, pp. 3–19.
[15] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "PASTA: PASsword-based threshold authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 2042–2059.
[16] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[17] J. Baron, K. E. Defrawy, J. Lampkins, and R. Ostrovsky, "How to withstand mobile virus attacks, revisited," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2014, pp. 293–302.
[18] K. E. Defrawy and J. Lampkins, "Founding digital currency on secure computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1–14.

[19] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *Proc. Annu. Int. Cryptology Conf.*, 1995, pp. 339–352.
[20] Y. Zhang, C. Xu, J. Ni, H. Li, and S. Xie, "Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2923222.
[21] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-based authentication for cloud computing," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2009, pp. 157–166.
[22] Q. Jiang, J. Ni, J. Ma, L. Yang, and X. Shen, "Integrated authentication and key agreement framework for vehicular cloud computing," *IEEE Netw.*, vol. 32, no. 3, pp. 28–35, May/Jun. 2018.
[23] A. Yang, E. Pagnin, A. Mitrokotsa, G. P. Hancke, and D. S. Wong, "Two-hop distance-bounding protocols: Keep your friends close," *IEEE Trans. Mobile Comput.*, vol. 17, no. 7, pp. 1723–1736, Jul. 2018.
[24] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 169–11 180, Nov. 2018.
[25] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Apr. 2019.
[26] M. Abdalla, S. Miner, and C. Namprempre, "Forward-secure threshold signature schemes," in *Proc. Cryptographers Track RSA Conf.*, 2001, pp. 441–456.
[27] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," in *Proc. Int. Workshop Public Key Cryptography*, 2003, pp. 31–46.
[28] I. Damgård and M. Koprowski, "Practical threshold RSA signatures without a trusted dealer," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2001, pp. 152–165.
[29] R. Gennaro, S. Goldfeder, and A. Narayanan, "Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2016, pp. 156–174.
[30] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ECDSA with fast trustless setup," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 1179–1194.
[31] Y. Lindell and A. Nof, "Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 1837–1854.
[32] D. L. Vo, F. Zhang, and K. Kim, "A new threshold blind signature scheme from pairings," in *Proc. Symp. Cryptography Inf. Secur.*, 2003.
[33] C. Diomedous and E. Athanasopoulos, "Practical password hardening based on TLS," in *Proc. Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2019, pp. 441–460.
[34] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, 2001, pp. 136–145.
[35] D. Wang and P. Wang, "Offline dictionary attack on password authentication schemes using smart cards," in *Proc. Inf. Secur.*, 2013, pp. 221–237.
[36] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. 12th ACM Conf. Comput. Commun. Secur.*, 2005, pp. 364–372.
[37] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proc. USENIX Secur. Workshop*, 1990, pp. 5–14.
[38] J. Alwen and V. Serbinenko, "High parallel complexity graphs and memory-hard functions," in *Proc. 47th Annu. ACM Symp. Theory Comput.*, 2015, pp. 595–603.
[39] J. Alwen, B. Chen, C. Kamath, V. Kolmogorov, K. Pietrzak, and S. Tessaro, "On the complexity of scrypt and proofs of space in the parallel random oracle model," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2016, pp. 358–387.
[40] J. Blocki and A. Datta, "CASH: A cost asymmetric secure hash algorithm for optimal password protection," in *Proc. IEEE 29th Comput. Secur. Found. Symp.*, 2016, pp. 371–386.
[41] D. Boneh, H. Corrigan-Gibbs, and S. Schechter, "Balloon hashing: A memory-hard function providing provable protection against sequential attacks," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2016, pp. 220–248.
[42] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart, "The pythia PRF service," in *Proc. 24th USENIX Conf. Secur. Symp.*, 2015, pp. 547–562.
[43] J. Schneider, N. Fleischhacker, D. Schröder, and M. Backes, "Efficient cryptographic password hardening services from partially oblivious commitments," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1192–1203.
[44] R. W. Lai, C. Egger, D. Schröder, and S. S. Chow, "Phoenix: Rebirth of a cryptographic password-hardening service," in *Proc. 26th USENIX Conf. Secur. Symp.*, 2017, pp. 899–916.
[45] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," Tech. Rep., 2008, Accessed: Mar. 2020. [Online]. Available: https://datatracker.ietf.org/doc/rfc5246/
[46] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 28th Annu. Symp. Found. Comput. Sci.*, 1987, pp. 427–438.
[47] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptology Conf.*, 1991, pp. 129–140.
[48] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.
[49] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2001, pp. 514–532.
[50] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2013, pp. 296–312.
[51] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 676–688, Mar. 2017.
[52] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2922958.
[53] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2908400.
[54] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. 22nd USENIX Conf. Secur.*, 2013, pp. 179–194.
[55] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, and X. Gui, "Toward encrypted cloud media center with secure deduplication," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 251–265, Feb. 2017.
[56] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2018.2791432.

Yuan Zhang (Member, IEEE) received the BSc and PhD degrees from the University of Electronic Science Technology of China (UESTC), Chengdu, China, in 2013 and 2019, respectively. He was a visiting PhD student from 2017 to 2019 in BBCR Lab, Department of ECE, the University of Waterloo, Canada. He is currently an associate professor with the School of Computer Science and Engineering, UESTC. His research interests are applied cryptography, data security, and blockchain technology.

Chunxiang Xu (Member, IEEE) received the BSc and MSc degrees in applied mathematics from Xidian University, Xi'an, China, in 1985 and 2004, respectively, and the PhD degree in cryptography from Xidian University, Xi'an, China, in 2004. She is currently a professor of the School of Computer Science and Engineering, UESTC. Her research interests include information security, cloud computing security, and cryptography.

Hongwei Li (Senior Member, IEEE) received the PhD degree in computer software and theory from the University of Electronic Science Technology of China, China, in 2008. He is a professor with the School of Computer Science and Engineering, UESTC, China. He is also with the Science and Technology on Communication Security Laboratory, Chengdu, China. He worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada for one year until Oct. 2012. His research interests include network security, applied cryptography, and trusted computing. He serves as an associate editor of the *Peer-to-Peer Networking and Applications*. He is a member of the China Computer Federation, and a member of the China Association for Cryptologic Research.



Kan Yang (Member, IEEE) received the BEng degree in information security from the University of Science and Technology of China, China, in 2008, and the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2013. He is an assistant professor with the Department of Computer Science, The University of Memphis, Memphis, Tennessee. He worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. His research interests include security and privacy in cloud computing, big data, and Internet of Things.

Nan Cheng (Member, IEEE) received the BE and MS degrees from Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the PhD degree from the University of Waterloo, Waterloo, ON, Canada, in 2016. He is currently a professor with the School of Telecommunication, Xidian University, Xi'an, China. His current research interests include big data in vehicular networks and self-driving system.

Xuemin Shen (Fellow, IEEE) received the PhD degrees from Rutgers University, Camden, New Jersey, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He served as the technical program committee chair/co-chair for IEEE Globecom16, INFOCOM14, IEEE VTC10 Fall, and Globecom07, the Symposia chair for IEEE ICC10. He also serves as the editor-in-chief for the *IEEE Internet of Things Journal*, the *Peer-to-Peer Networking and Application*, and the *IET Communications*; a founding area editor for the *IEEE Transactions on Wireless Communications*. He received the Excellent Graduate Supervision Award, in 2006, and the Outstanding Performance Award, in 2004, 2007, 2010, and 2014 from the University of Waterloo, Waterloo, Canada, the Premiers Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada, the Distinguished Performance Award, in 2002 and 2007 from the Faculty of Engineering, University of Waterloo, Waterloo, Canada, the Joseph LoCicero Award and the Education Award 2017 from the IEEE Communications Society. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, and a distinguished lecturer of IEEE Vehicular Technology Society and Communications Society.