Practical Anonymous Password Authentication and TLS with Anonymous Client Authentication *

Zhenfeng Zhang
Trusted Computing and Information Assurance Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences
zfzhang@tca.iscas.ac.cn

Kang Yang†
Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences
yangkang@tca.iscas.ac.cn

Xuexian Hu
Institute of Software, Chinese Academy of Sciences & State Key Lab of Mathematical Engineering and Advanced Computing
xuexian_hu@hotmail.com

Yuchen Wang
Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences
wangyuchen@tca.iscas.ac.cn

ABSTRACT

Anonymous authentication allows one to authenticate herself without revealing her identity, and becomes an important technique for constructing privacy-preserving Internet connections. Anonymous password authentication is highly desirable as it enables a client to authenticate herself by a human-memorable password while preserving her privacy. In this paper, we introduce a novel approach for designing *anonymous password-authenticated key exchange* (APAKE) protocols using algebraic message authentication codes (MACs), where an algebraic MAC wrapped by a password is used by a client for anonymous authentication, and a server issues algebraic MACs to clients and acts as the verifier of login protocols. Our APAKE construction is secure provided that the algebraic MAC is *strongly existentially unforgeable under random message and chosen verification queries attack* (suf-rmva), weak pseudorandom and tag-randomization simulatable, and has *simulation-sound extractable non-interactive zero-knowledge proofs* (SE-NIZKs).

To design practical APAKE protocols, we instantiate an algebraic MAC based on the q-SDH assumption which satisfies all the required properties, and construct credential presentation algorithms for the MAC which have optimal efficiency for a randomize-then-prove paradigm. Based on the algebraic MAC, we instantiate a highly practical APAKE protocol and denote it by `APAKE`, which is much more efficient than the mechanisms specified by ISO/IEC 20009-4. An efficient revocation mechanism for `APAKE` is also proposed.

We integrate `APAKE` into TLS to present an anonymous client authentication mode where clients holding passwords can authenticate themselves to a server anonymously. Our implementation with 128-bit security shows that the average connection time of `APAKE`-based ciphersuite is 2.8 ms. With `APAKE` integrated into the OpenSSL library and using an Apache web server on a 2-core desktop computer, we could serve 953 `ECDHE-ECDSA-AES128-GCM-SHA256` HTTPS connections per second for a 10 KB payload. Compared to ECDSA-signed elliptic curve Diffie-Hellman ciphersuite with mutual authentication, this means a 0.27 KB increased handshake size and a 13% reduction in throughput.

1. INTRODUCTION

Privacy protection has become a major concern with the rapid growth of cloud computing, big data and internet of things. For example, contact details of 1.5 million customers of Verizon Enterprise were put up for sale on a Dark Web forum recently [1]. Most people typically associate the loss of privacy with a feeling of invasion or loss of control [45]. The importance of user privacy in authentication systems has been emphasized by the European privacy standard [31] and by the US government in the National Strategy for Trusted Identities in Cyberspace [52]. NIST has developed three privacy engineering objectives - predictability, manageability, and disassociability [47], where disassociability captures one of the essential elements of privacy-enhancing systems that the system actively protects or "blinds" an individual's identity or associated activities from unnecessary exposure.

Authentication of participants is usually required in computer systems-based applications to establish trust relations. An effective approach to protect users' privacy in authentication systems is anonymous authentication which achieves *secure authentication* and *anonymity* simultaneously [45], i.e., no unauthorized user can fool a server into granting it access, and the server should not know which user it is interacting with. As stated in [21], privacy-conscious service providers (SPs) have a strong incentive to adopt anonymous authentication, and it is in their best interest to keep client information private on technical unavailability grounds.

*The work is supported by National Basic Research Program of China (No.2013CB338003) and National Natural Science Foundation of China (No.U1536205, 61572485, 61502527).

†Corresponding author

CCS'16, October 24-28, 2016, Vienna, Austria

 ISBN 978-1-4503-4139-4/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2976749.2978354

Password-based authentication systems have been widely deployed in information systems to guarantee authorized access to desktop, mobile and web applications, as passwords have advantages of being memorable, avoiding comprehensive public key infrastructure for distributing client certificates and dedicated hardware for storing secret keys. Password-based authentication with key exchange protocols have been extensively explored [7, 4, 14, 44, 49, 8] and widely standardized [39, 57, 40]. The TLS ciphersuite using secure remote password protocol [57] is provided in OpenSSL.

In traditional password-based authentication systems, a user keeps a password confidential but pays no attention to privacy protection of herself [2], since the identity information is usually transmitted explicitly so that the server can determine which password should be used. Therefore, anonymous authentication that can work with password-based technology are highly desirable [45].

Anonymous password authentication protocols have been proposed by Viet et al. [58] via integrating an oblivious transfer protocol within a two-party password-authenticated key exchange protocol, and later improved in [60, 54]. Such protocols allow a client holding a password to authenticate herself to the server, while preserving her privacy. However, these protocols have an inherent limitation for computation efficiency, i.e., a server performs $O(N)$ computations per protocol execution, where N is the total number of users.

Another approach for anonymous password authentication was proposed in [61, 62], where a user obtains a (CL [17] or BBS [12]) signature from a server, wraps the signature with her password and stores it on some extra storage, such as a smartphone, a tablet (e.g., iPad), a USB flash memory, or even in a public directory (e.g., cloud). The extra storage is only needed to be integrity-protected, which is weaker than a dedicated hardware. To login the server, the user recovers the signature from a password-wrapped credential using her password, and then proves possession of the signature. In these schemes, the server's cost is independent of the scale of user set. However, homomorphic encryption is needed in [61, 62] to resist off-line dictionary attacks. For 80-bit security level, a user with a 2.53GHz notebook costs 385 ms and a server with a 3.0GHz desktop computer costs 430 ms per login protocol run.

Anonymous password authentication also attracts the interest of industry standard organizations. In a standard for anonymous entity authentication, the mechanisms based on weak secrets are named as ISO/IEC 20009-4 and developed by ISO/IEC JTC 1, SC 27, IT Security techniques. Three mechanisms have been included in ISO/IEC 20009-4 [43].

For the Transport Layer Security (TLS) protocol [28], there are three modes supported: mutual authentication, server authentication (with no client authentication), and total anonymity. The first mode needs client certificates to offer authentication, and thus provides no client-anonymity. The last two modes do not provide any authentication of clients, and the mode of total anonymity is inherently vulnerable to man-in-the-middle attacks and strongly discouraged. A TLS mode of *anonymous client authentication* is of great interest, where clients can authenticate to a server without revealing their identities, the server is assured that only authorized clients can provide secure authentication.

In CCS 2015, Cassola et al. [21] consider a practical scenario of anonymous authentication for Wi-Fi connectivity using open hotspots hosted on untrusted Access Points (APs). A dishonest ISP may track which APs a client connects to and when, revealing clients' mobility patterns and other sensitive information. A protocol was proposed in [21] that allows SPs to authenticate their clients, but hides clients' identities from APs and SPs at the time of authentication.

1.1 Our Contributions

In this paper, we propose a novel approach for designing anonymous password authentication protocols by using *algebraic MACs* which are constructed using only group operations rather than block ciphers or hash functions [24]. Specifically, an algebraic MAC is issued by a server to a user, and then used as a credential for authentication. The algebraic MAC is protected by a user's password and stored on some extra storage with integrity-protection. The underlying algebraic MAC is required to be weak pseudorandom in order to resist off-line dictionary attacks, and admits efficient zero-knowledge proofs so that a user can prove possession of a credential. Thus, only registered users owning algebraic MACs can authenticate themselves to the server while preserving their privacy. This approach sufficiently incorporates the symmetric feature of algebraic MACs with that of anonymous password authentication, eliminates the dependence of homomorphic encryption, and yields conceptually simple and provably secure constructions.

To construct practical APAKE protocols, we instantiate an algebraic MAC scheme based on the q-SDH assumption [11], and show that it is suf-rmva secure, weak pseudorandom and tag-randomization simulatable, and allows labeled SE-NIZKs. For credential presentation, the Show algorithm costs one exponentiation and one multi-exponentiation to generate a presentation proof and the ShowVerify algorithm costs one multi-exponentiation for verification, which are optimal for the randomize-then-prove paradigm.

Based on the instantiated algebraic MAC scheme, we obtain a highly practical APAKE protocol, which is denoted by `APAKE`. Compared with the mechanisms specified by ISO/IEC 20009-4 [43], `APAKE` provides significant performance advantages, and may invoke interest of the standard community. An efficient revocation mechanism for `APAKE` is proposed, and the resulting protocol is denoted by `APAKEr`.

We integrate `APAKE` and `APAKEr` into the TLS protocol to provide a mode of anonymous client authentication. For an ECDSA-signed elliptic curve Diffie-Hellman ciphersuite, we denote the ciphersuite with anonymous client authentication by `ECDHE3`, and denote the ciphersuite with only server authentication (resp., mutual authentication) by `ECDHE1` (resp., `ECDHE2`). Let `ECDHE4` be the `ECDHE3` supporting revocation. Based on OpenSSL library, we implemented in C the `ECDHEi` ciphersuite at a 128-bit security level for $i = 1, \cdots 4$. HTTPS connections per second supported by the server are reported in Figure 1. When using the `secp256r1` elliptic curve and an Apache web server on a 2-core desktop computer, the server can handle 953 `ECDHE-ECDSA-AES128-GCM-SHA256` HTTPS connections with anonymous client authentication per second for a 10 KB payload, and a factor 1.13x fewer than `ECDHE2`. While the average connection time of `ECDHE3` ciphersuite is 2.8 ms, that of `ECDHE4` ciphersuite is 3.4 ms.

1.2 Related Work

To enhance users' privacy, anonymous signature schemes, such as group signatures [27], blind signatures [25] and Direct Anonymous Attestation (DAA) [15] have been exten-

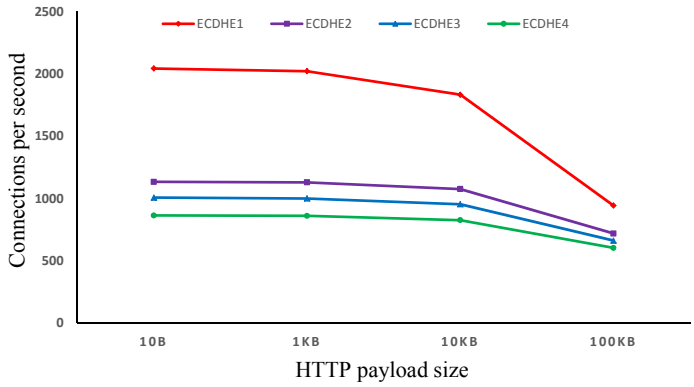

Figure 1: HTTPS connections per second supported by the server at a 128-bit security level

sively investigated. DAA has been adopted by the Trusted Computing Group and standardized by ISO/IEC [41, 42].

Anonymous credentials are introduced by Chaum [26], and a series of schemes [19, 18, 16] have been proposed. Several privacy-enhancing attribute-based credential systems [51] have been developed, including IBM's Idemix system [38], and Microsoft's U-Prove system [48].

In CCS'14, Chase et al. [24] constructed keyed-verification anonymous credentials based on two algebraic MACs, where one is uf-cmva secure [29] in the generic group model [55], and the other is uf-cmva secure under the DDH assumption.

Cesena et al. [22] proposed a solution for anonymous authentication via integrating DAA into TLS, and obtained a ciphersuite which has a factor about 25x fewer HTTPS connections per second than ECDHE2, even if the computations of TPM are performed on a PC. Walker and Li [59] presented a key exchange protocol with anonymous authentication by combining DAA and the SIGMA family of key exchange protocols from IPsec and IKE.

In CCS'15, Fett et al. [32] proposed the first privacy-respecting Single Sign-On system (SPRESSO) for users to login web sites, and prove that it enjoys strong authentication and privacy properties. SPRESSO is a new system built from scratch and involves a forwarder (FWD) to forward messages from Identity Providers to Relying Parties.

2. BUILDING BLOCKS

In this section, we present the building blocks used in our APAKE construction. Firstly, we describe the notation and the assumptions used in this paper.

Notation. Throughout this paper, λ denotes the security parameter, $x \xleftarrow{\$} S$ denotes x is sampled uniformly at random from a set S, $[n]$ denotes the set $\{1, \ldots, n\}$. For an algorithm A, $(y_1, y_2, \ldots) \leftarrow A(x_1, x_2, \ldots)$ denotes the process of running A on input $(x_1, x_2, \ldots)$ and getting $(y_1, y_2, \ldots)$ as output. A function $f : \mathbb{N} \to [0, 1]$ is negligible if for any positive c, we have $f(\lambda) < 1/\lambda^c$ for sufficient large λ.

Let $\mathbb{G}$ be a multiplicative group of prime order p generated by g, 1 be the identity element of $\mathbb{G}$ and $\mathbb{G}^*$ denote $\mathbb{G}\backslash\{1\}$.

2.1 Assumptions

q-SDH Assumption [11]. Given $(g, g^x, \ldots, g^{x^q})$ for $x \xleftarrow{\$} \mathbb{Z}_p^*$, it is hard to output a pair $\left(c, g^{1/(x+c)}\right)$ for $c \in \mathbb{Z}_p\backslash\{-x\}$.

q-DDHI Assumption [10]. Given $(g, g^x, \ldots, g^{x^q})$ for $x \xleftarrow{\$} \mathbb{Z}_p^*$, it is hard to distinguish $g^{1/x}$ from a random element.

DDH Assumption. Given (g, g^x, g^y) for $x, y \xleftarrow{\$} \mathbb{Z}_p^*$, it is hard to distinguish g^{xy} from a random element in $\mathbb{G}^*$.

2.2 Non-Interactive Zero-Knowledge Proofs

Non-interactive zero-knowledge proofs (NIZKs) enable a prover to prove in zero-knowledge that a statement $\mathtt{x}$ is in a given language $\mathcal{L}$ defined by an NP-relation $\mathcal{R}$, i.e., $\mathcal{L} = \{\mathtt{x} | \exists \mathtt{w}\ s.t.\ \mathcal{R}(\mathtt{x}, \mathtt{w}) = 1\}$. An NIZK could also be extended to support (optional) labels, meaning that both a prover and a verifier are given a label ℓ as input.

A labeled NIZK should satisfy soundness and unbounded zero-knowledge, where the former requires that no adversary can prove any false statement, and the latter means that there exists a simulator which is able to simulate any proof for any statement $\mathtt{x}$ and any label ℓ without knowing the witness $\mathtt{w}$. If a labeled NIZK also provides simulation-sound extractability, then it is called a labeled *simulation-sound extractable non-interactive zero-knowledge proof* (SE-NIZK) [36], where there exists an online extractor [34] works even if the adversary sees simulated proofs and information about previously extracted values. In this paper, we consider labeled SE-NIZKs in the *random oracle model* (ROM) [6], and refer to [9] for a formal definition.

We adopt the notations of [20] to abstract labeled NIZKs. Let $\Sigma \leftarrow \mathsf{SPK}\{(\mathtt{witness}) : \mathtt{statement}\}(\ell)$ denote a labeled SE-NIZK on a label ℓ, $\pi \leftarrow \mathsf{NIZK}\{(\mathtt{witness}) : \mathtt{statement}\}$ be an NIZK. We write $\mathsf{Verify}_{\mathsf{SPK}}(\mathtt{statement}, \Sigma, \ell)$ for the procedure that verifies a labeled SE-NIZK proof Σ, and write $\mathsf{Verify}_{\mathsf{NIZK}}(\mathtt{statement}, \pi)$ for verifying an NIZK proof π.

2.3 Algebraic MAC

Following [24], an algebraic MAC scheme $\mathcal{MAC}$ is defined as a triple of algorithms $\mathcal{MAC} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ with associated message space $\mathcal{M}_c$ and tag space $\mathcal{T}$.

- $\mathsf{KeyGen}(1^\lambda)$: On input a security parameter 1^λ, the key generation algorithm outputs a secret key sk and a set of parameters $\mathsf{par}_{\mathsf{mac}}$ which is an implicit input in the following algorithms.
- $\mathsf{MAC}(\mathsf{sk}, m)$: On input the secret key sk and a message m, the MAC algorithm outputs an authentication tag σ.
- $\mathsf{Verify}(\mathsf{sk}, m, \sigma)$: On input the secret key sk, a message m and a tag σ, the *deterministic* verification algorithm outputs 1 if σ is valid on m under sk and 0 otherwise.

We assume that the key generation algorithm KeyGen satisfies the key-parameter consistency [24], meaning that there does not exist two keys sk and sk' such that $(\mathsf{par}_{\mathsf{mac}}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)$, $(\mathsf{par}_{\mathsf{mac}}, \mathsf{sk}') \in \mathsf{KeyGen}(1^\lambda)$ and $\mathsf{sk} \neq \mathsf{sk}'$.

Given the parameters $\mathsf{par}_{\mathsf{mac}}$ and a message-tag pair (m, σ), we assume that there exists a proof system $\mathsf{NIZK}\{(\mathsf{sk}) : \mathsf{Verify}(\mathsf{sk}, m, \sigma) = 1 \wedge (\mathsf{par}_{\mathsf{mac}}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)\}$ proving that σ is a valid tag on m under sk associated with $\mathsf{par}_{\mathsf{mac}}$.

Unforgeability. Based on the security notions of algebraic MACs [29, 24], we define a security notion of algebraic MACs, i.e., *strongly existentially unforgeable under random message and chosen verification queries attack* (suf-rmva).

Definition 1. An algebraic MAC scheme $\mathcal{MAC}$ is said to be suf-rmva secure if for any probabilistic polynomial time

(PPT) adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\text{suf-rmva}}_{\mathcal{MAC}}(\mathcal{A}) \stackrel{\text{def}}{=}$

$$\Pr\left[\begin{array}{l}(\mathsf{par_{mac}},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda);\\ (m^*,\sigma^*) \leftarrow \mathcal{A}^{\text{MAC}(\mathsf{sk}),\text{VERIFY}(\mathsf{sk},\cdot,\cdot)}(\mathsf{par_{mac}}) :\\ (m^*,\sigma^*) \notin Q \wedge \mathsf{Verify}(\mathsf{sk},m^*,\sigma^*)=1\end{array}\right] \leq \nu(\lambda),$$

where for each query MAC returns $m \stackrel{\$}{\leftarrow} \mathcal{M}_c$ and $\sigma \leftarrow \mathsf{MAC}(\mathsf{sk},m)$ and adds (m,σ) to the set Q which is initially empty, for each query (m,σ) VERIFY returns $\mathsf{Verify}(\mathsf{sk},m,\sigma)$.

Weak pseudorandomness. Based on the definition of *weak pseudorandom functions* (wPRFs) [46], we define a notion of weak pseudorandomness of algebraic MACs. For simplicity, we assume that the size of $\mathcal{M}_c$ is super-polynomial.

Definition 2. An algebraic MAC scheme $\mathcal{MAC}$ is said to be weak pseudorandom if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν, such that $\mathsf{Adv}^{\text{wpr}}_{\mathcal{MAC}}(\mathcal{A}) \stackrel{\text{def}}{=}$

$$2\Pr\left[b=b' \left| \begin{array}{c}(\mathsf{par_{mac}},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda); b \stackrel{\$}{\leftarrow} \{0,1\};\\ m \stackrel{\$}{\leftarrow} \mathcal{M}_c; \sigma_0 \leftarrow \mathsf{MAC}(\mathsf{sk},m); \sigma_1 \stackrel{\$}{\leftarrow} \mathcal{T};\\ b' \leftarrow \mathcal{A}^{\text{MAC}(\mathsf{sk})}(\mathsf{par_{mac}},m,\sigma_b)\end{array}\right.\right] - 1$$

$\leq \nu(\lambda)$, where for each query, the MAC oracle returns a random $m \in \mathcal{M}_c$ and $\sigma \leftarrow \mathsf{MAC}(\mathsf{sk},m)$.

Credential Presentation. A tag σ is used as a credential in this paper. The credential presentation consisting of $(\mathsf{Show},\mathsf{ShowVerify})$ algorithms, is a procedure of proving possession of a valid message-tag pair (m,σ), and is generally constructed via the randomize-then-prove paradigm.

In the randomize-stage, there are two algorithms Rerand and Derand such that $\mathsf{Rerand}(\sigma)$ returns a randomized credential T and a randomness a, and $\mathsf{Derand}(T,a)$ returns σ. For algebraic MACs, both a prover and a verifier can compute the same value $V = f_{\text{P}}(\mathsf{par_{mac}},T,m,a) = f_{\text{V}}(T,\mathsf{sk})$ using (m,a) and sk respectively, where f_{P} and f_{V} are efficiently computable functions specified by a concrete mechanism.

In the prove-stage, the prover proves knowledge of (m,a) such that $f_{\text{P}}(\mathsf{par_{mac}},T,m,a) = V$ using a labeled SE-NIZK.

- $\mathsf{Show}(\mathsf{par_{mac}},m,\sigma,\ell)$: On input $\mathsf{par_{mac}}$, a message-tag pair (m,σ) and a label $\ell \in \{0,1\}^*$, the Show algorithm runs $(T,a) \leftarrow \mathsf{Rerand}(\sigma)$, then computes $V \leftarrow f_{\text{P}}(\mathsf{par_{mac}},T,m,a)$, and executes $\Sigma \leftarrow \mathsf{SPK}\{(m,a) : f_{\text{P}}(\mathsf{par_{mac}},T,m,a) = V\}(\ell)$. Finally, it outputs a presentation proof $\sigma_C \leftarrow (T,V,\Sigma)$.
- $\mathsf{ShowVerify}(\mathsf{par_{mac}},\sigma_C,\ell,\mathsf{sk})$: On input $\mathsf{par_{mac}}$, a presentation proof $\sigma_C = (T,V,\Sigma)$, a label ℓ and the secret key sk, algorithm ShowVerify computes $\tilde{V} \leftarrow f_{\text{V}}(T,\mathsf{sk})$. If T is correctly formed and $\mathsf{Verify}_{\mathsf{SPK}}((\mathsf{par_{mac}},T,V),\Sigma,\ell) = 1$ and $V = \tilde{V}$, then ShowVerify returns 1, else it returns 0.

We say that the tag-randomization is *simulatable*, if there exists an efficient algorithm TVSim that takes as input $\mathsf{par_{mac}}$, and returns a pair (T',V') such that $V' = f_{\text{V}}(T',\mathsf{sk})$ and T' has the same distribution as T produced by $\mathsf{Rerand}(\sigma)$.

2.4 Password-based Encryption

Let $\mathcal{M}_e$ be a message space of super-polynomial size, $\mathcal{C}$ be a ciphertext space. A password-based encryption scheme $\mathcal{PE}$ with a password pw drawn uniformly at random from a dictionary $\mathcal{D}$, is defined as $\mathcal{PE} = (\mathsf{Enc},\mathsf{Dec})$.

- $\mathsf{Enc}_{pw}(M)$: On input a password $pw \in \mathcal{D}$ and a message $M \in \mathcal{M}_e$, the encryption algorithm outputs a ciphertext $C \in \mathcal{C}$, which is also denoted by $[M]_{pw}$.
- $\mathsf{Dec}_{pw}(C)$: On input pw and a ciphertext $C \in \mathcal{C}$, the decryption algorithm outputs a plaintext M for C.

We define a security notion of password-based encryption called *indistinguishability under equality test* (IND-ET), where an equality test oracle is used to model an adversary's ability deciding whether an online password guess is correct.

Definition 3. A password-based encryption scheme $\mathcal{PE}$ is said to be IND-ET secure, if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν, we have $\mathsf{Adv}^{\text{IND-ET}}_{\mathcal{PE},\mathcal{D}}(\mathcal{A}) \stackrel{\text{def}}{=}$

$$2\Pr\left[b=b' \left| \begin{array}{c}b \stackrel{\$}{\leftarrow} \{0,1\}; pw \stackrel{\$}{\leftarrow} \mathcal{D}; M \stackrel{\$}{\leftarrow} \mathcal{M}_e; C_0 \stackrel{\$}{\leftarrow} \mathcal{C};\\ C_1 \leftarrow \mathsf{Enc}_{pw}(M); b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ET}}(M,\cdot)}(C_b)\end{array}\right.\right] - 1$$

$\leq q_{\text{et}}/|\mathcal{D}| + \nu(\lambda)$,

where $\mathcal{O}_{\text{ET}}(M,\cdot)$ takes as input a $M' \in \mathcal{M}_e$ and outputs 1 if $M = M'$ and 0 otherwise, q_{et} is the number of oracle queries.

2.5 Digital Signature

A digital signature scheme $\mathcal{DS}$ is defined as a triple of algorithms $\mathcal{DS} = (\mathsf{Gen},\mathsf{Sign},\mathsf{Ver})$. The key generation algorithm $\mathsf{Gen}(1^\lambda)$ returns the public and secret keys (PK, SK). The signing algorithm $\mathsf{Sign}(\text{SK},M)$ returns a signature σ on a message M. The verification algorithm $\mathsf{Ver}(\text{PK},M,\sigma)$ returns 1 if σ is valid on M under PK and 0 otherwise.

The security notion of digital signatures is *existential unforgeability under adaptive chosen message attacks* (EUF-CMA) [35], which states that for any PPT adversary $\mathcal{A}$, there exists a negligible function ν, such that $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{DS}}(\mathcal{A})$

$$\stackrel{\text{def}}{=} \Pr\left[\begin{array}{c}(\text{PK},\text{SK}) \leftarrow \mathsf{Gen}(1^\lambda); (M^*,\sigma^*) \leftarrow \mathcal{A}^{\text{SIGN}(\text{SK},\cdot)}(\text{PK}) :\\ \mathsf{Ver}(\text{PK},M^*,\sigma^*) = 1 \wedge M^* \notin Q\end{array}\right]$$

$\leq \nu(\lambda)$, where for each query M, SIGN returns a signature σ on M and adds M to the set Q which is initially empty.

3. SECURITY MODEL

We formalise a security model for APAKE protocols in the *extra-storage setting*, meaning that a client needs to memorize a password and store a password-wrapped credential on some extra storage. The definitions of authenticated key exchange (AKE) security and client authentication combine the security model for PAKE protocols by Bellare, Pointcheval and Rogaway [4] and the model for anonymous authentication by Lindell [45]. The security model for anonymity is based on indistinguishability-based definition of anonymity for group signatures [3].

3.1 AKE Security and Client Authentication

Protocol participants. The participants of an APAKE protocol $\mathcal{P}$ involve a set of clients $\mathbf{C} = \{C_1,\cdots,C_N\}$ and a set of servers $\mathbf{S}$. For simplicity, we assume that $\mathbf{C}$ is fixed and $\mathbf{S}$ contains only one server S, i.e., $\mathbf{S} = \{S\}$.

Long-lived keys. The server S holds a long-term secret SK for issuing credentials and authenticating himself and publishes the system parameters params which are publicly available for all parties. Each client $C_i \in \mathbf{C}$ holds a password pw_i that is drawn independently and uniformly from

a dictionary $\mathcal{D}$, and a password-protected credential $cred_{C_i}$ generated by wrapping a credential issued by S with pw_i.

Protocol execution. Each participant $U \in \mathbf{C} \cup \mathbf{S}$ is modeled as a PPT Turing machine, and the δ-th instance of U is denoted by U^δ. An adversary $\mathcal{A}$ is given a set of password-protected credentials $\mathbf{Cred} = \{cred_i\}_{i \in \mathbf{C}}$ and params, and is assumed to have full control of the communication network. Adversary $\mathcal{A}$ is a PPT algorithm with a distinguished query tape. Queries written on this tape are answered according to the description of $\mathcal{P}$. The allowed queries are as below:

- $\mathtt{Send}(U, \delta, M)$: causes message M to be sent to instance U^δ for $U \in \mathbf{C} \cup \mathbf{S}$. The instance U^δ computes what the protocol says to, and sends back the computation result to $\mathcal{A}$. If this query causes U^δ to accept or terminate, it will also be made visible to $\mathcal{A}$.
- $\mathtt{Execute}(C_i, \rho, S, \delta)$: carries out an honest execution of $\mathcal{P}$ between a client instance C_i^ρ and a server instance S^δ and outputs the transcript of the execution. Although this queries could be simulated with `Send` queries, separate `Execute` queries are essential for dealing with off-line dictionary attacks [4].
- $\mathtt{Reveal}(U, \delta)$: returns the session key held by instance U^δ.
- $\mathtt{Test}(U, \delta)$: If instance U^δ for $U \in \mathbf{C} \cup \mathbf{S}$ has accepted and holds a session key sk_U^δ, the following happens. A bit $b \in \{0, 1\}$ is picked uniformly at random. If $b = 1$, sk_U^δ is returned to $\mathcal{A}$. Otherwise, a string picked at random from the space of session keys is returned. Adversary $\mathcal{A}$ is allowed to ask the `Test` query only once.

Partnering. Since the anonymity property implies that the server can only know that a client is a legitimate entity from a group $\mathbf{C}$, a partner identifier of the server is the group $\mathbf{C}$. An instance U^δ that accepts, holds a partner identifier pid_U^δ, a session identifier sid_U^δ (which is the transcript of the whole protocol) and a session key sk_U^δ. A client instance C_i^ρ and a server instance S^δ are said to be *partnered* if both accept, they hold $(pid_{C_i}^\rho, sid_{C_i}^\rho, sk_{C_i}^\rho)$ and $(pid_S^\delta, sid_S^\delta, sk_S^\delta)$ respectively, with $sid_{C_i}^\rho = sid_S^\delta$, $sk_{C_i}^\rho = sk_S^\delta$, $pid_{C_i}^\rho = S$, $pid_S^\delta = \mathbf{C}$ and $C_i \in \mathbf{C}$, and no other instance accepts with session identifier equal to $sid_{C_i}^\rho$.

Freshness. An instance U^δ is said to be fresh unless either a $\mathtt{Reveal}(U, \delta)$ query occurs or a $\mathtt{Reveal}(V, \rho)$ query occurs, where V^ρ is the partner of U^δ (if exists).

AKE security. Let $\mathsf{Succ}^{\mathrm{AKE}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ be the event that $\mathcal{A}$ makes a single $\mathtt{Test}(U, \delta)$ query such that the instance U^δ has terminated and is fresh, and eventually outputs a bit b' such that $b = b'$, where b is chosen in the `Test` query. A protocol $\mathcal{P}$ is said to be AKE secure, if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\mathsf{Adv}^{\mathrm{AKE}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} 2\Pr[\mathsf{Succ}^{\mathrm{AKE}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})] - 1 \leq q_{\mathrm{se}}/|\mathcal{D}| + \nu(\lambda),$$

where q_{se} is the number of $\mathtt{Send}(S, \cdot, \cdot)$ queries.

Client authentication. Due to the anonymity requirement, an adversary against client authentication is declared *successful* if it impersonates any client in the group $\mathbf{C}$ to the server while the server fails to detect.

To capture the security of client authentication, $\mathcal{A}$ is provided the same information and abilities as that in the AKE experiment, except that the `Test` query is ignored. Let $\mathsf{Succ}^{\mathrm{C2S}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ be the event that some server instance S^δ accepts but has no partner instance. We say that a protocol $\mathcal{P}$ achieves clients-to-server authentication if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\mathsf{Adv}^{\mathrm{C2S}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Succ}^{\mathrm{C2S}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})] \leq q_{\mathrm{se}}/|\mathcal{D}| + \nu(\lambda),$$

where q_{se} is the number of $\mathtt{Send}(S, \cdot, \cdot)$ queries.

3.2 Anonymity

The anonymity property requires that the server cannot determine whether two key exchange transactions are made by the same client. To model anonymity against the server, an adversary is given the server's secret key as in [24].[1] A `Reg` oracle is also provided to model the registration protocol.

Protocol participants. The participants of a protocol $\mathcal{P}$ consist of a set of clients $\mathbf{C} = \{C_1, \cdots, C_N\}$ and a server S.

Long-lived keys. An adversary $\mathcal{A}$ impersonating as the server S is given the system parameters params and the secret key SK. Each client $C_i \in \mathbf{C}$ holds a password pw_i. Her password-protected credential $cred_{C_i}$ is initiated as $\perp$ and will be generated in the execution of the protocol.

Protocol execution. At the beginning of the protocol, a random bit b is chosen. $\mathcal{A}$ is provided the following queries:

- $\mathtt{Reg}(U, M)$: If $U \in \mathbf{C}$ and $cred_U = \perp$, the following happens. Message M is sent to client U, and the client computes what the registration protocol says to and sends back the computation result to $\mathcal{A}$. If U accepts, a password-protected credential is generated by U and assigned to $cred_U$ which is sent to $\mathcal{A}$.
- $\mathtt{Send}(U, \delta, M)$: causes message M to be sent to instance U^δ for $U \in \mathbf{C}$ and $cred_U \neq \perp$. U^δ computes what the protocol says to, and the computation result is sent to $\mathcal{A}$.
- $\mathtt{CH}(i_0, i_1, \delta, M)$: If $i_0, i_1 \in \mathbf{C}$, $cred_{i_0} \neq \perp$, $cred_{i_1} \neq \perp$ and neither ${i_0}^\delta$ nor ${i_1}^\delta$ is used, the following happens. The instance ${i_b}^\delta$ with password pw_{i_b} and $cred_{i_b}$ computes what the protocol says to, and sends back the output of the computation to $\mathcal{A}$.

For any above query, if this query causes the client or the instance to accept or terminate, this will also be shown to $\mathcal{A}$. $\mathcal{A}$ is allowed to make arbitrary number of queries to `CH`.

Anonymity. Let $\mathsf{Succ}^{\mathrm{anon}}_{\mathcal{P}}(\mathcal{A})$ be the event that $\mathcal{A}$ outputs a bit b' such that $b = b'$, where the bit b was picked at the beginning of the protocol. An APAKE protocol $\mathcal{P}$ is said to be anonymous if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\mathsf{Adv}^{\mathrm{anon}}_{\mathcal{P}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} 2\Pr[\mathsf{Succ}^{\mathrm{anon}}_{\mathcal{P}}(\mathcal{A})] - 1 \leq \nu(\lambda).$$

4. OUR APAKE PROTOCOL

In this section, we present a new APAKE protocol in the extra-storage setting. It employs an algebraic MAC scheme $\mathcal{MAC} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ with credential presentation

[1] A model allowing $\mathcal{A}$ to generate the server's public key is applicable if the domain parameters are selected from a standard. It does not undermine anonymity if a proof of knowledge of SK is published.

algorithm $(\mathsf{Show}, \mathsf{ShowVerify})$, a password-based encryption scheme $\mathcal{PE} = (\mathsf{Enc}, \mathsf{Dec})$ and a digital signature scheme $\mathcal{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, which are described in Section 2. We prove that our construction achieves AKE security, client authentication and anonymity in the random oracle model.

4.1 Our Construction

Our construction consists of the following phases.

Setup. Given a security parameter λ, a server chooses a set of domain parameters $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a group of prime order p and generated by g. The server runs $(\mathsf{par_{mac}}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$. Let $H_1 : \{0,1\}^* \to \mathcal{M}_c$ and $H_2 : \{0,1\}^* \to \{0,1\}^\kappa$ be cryptographic hash functions, where κ is the length of session keys. The server publishes $\mathsf{params} \leftarrow (\mathbb{G}, p, g, \mathsf{par_{mac}}, \mathrm{PK})$ as the set of system parameters and sets $SK = (\mathsf{sk}, \mathrm{SK})$ as his secret key.

Registration. Each client needs to register to the server in advance. The registration phase is executed over a secure channel, which can be established, e.g., using TLS with the server's public key PK. The registration protocol is shown in Figure 2, and details are described as follows.

1. A client sends her identity ID to the server and authenticates herself to the server according to the server's policy.
2. If the server accepts the registration request from client ID, he does the following. The server computes a message $m \leftarrow H_1(\mathsf{ID})$, and generates an authentication tag σ on message m using sk. Then he generates an NIZK proof π proving knowledge of sk such that $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 1$ and $(\mathsf{par_{mac}}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)$ hold. Finally, the server sends a credential σ and its proof π to the client.
3. When receiving a pair (σ, π), the client computes $m \leftarrow H_1(\mathsf{ID})$, and verifies if π is valid on statement $(\mathsf{par_{mac}}, m, \sigma)$. If π is valid, the client encrypts the credential σ with her password pw into a ciphertext $[\sigma]_{pw}$, and puts the password-protected credential $cred \leftarrow (\mathsf{ID}, [\sigma]_{pw})$ to her preferred storage with integrity-protection.

Login. To login the server, a client authenticates herself to the server and establishes a session key with the server. Suppose that the client has already obtained her password-protected credential $cred = (\mathsf{ID}, [\sigma]_{pw})$. The login protocol is shown in Figure 3, and details are described as follows.

1. Upon a login request, the server picks $y \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $Y \leftarrow g^y$. Then, he generates a signature σ_S on message Y using SK, and sends (Y, σ_S) to the client.
2. When receiving a pair (Y, σ_S), the client verifies whether σ_S is valid on message Y under PK. If she accepts σ_S, she computes $m \leftarrow H_1(\mathsf{ID})$, and decrypts ciphertext $[\sigma]_{pw}$ with her password pw to recover credential σ. Then she chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and calculates $X \leftarrow g^x$. Next, the client runs algorithm Show on input $\mathsf{par_{mac}}$, (m, σ) and a label $\ell = (X, Y, \sigma_S)$ to generate a presentation proof $\sigma_C = (T, V, \Sigma)$. Finally, the client sends (X, σ_C) to the server.
3. Upon receiving a pair (X, σ_C), the server executes the $\mathsf{ShowVerify}$ algorithm on input $\mathsf{par_{mac}}$, a presentation proof σ_C, a label $\ell = (X, Y, \sigma_S)$ and sk to verify if σ_C is valid.
4. If they both do not abort, then the client and the server can compute the same session key K via $H_2(Y, \sigma_S, X, \sigma_C, Y^x)$ and $H_2(Y, \sigma_S, X, \sigma_C, X^y)$ respectively.

4.2 Security Proofs

We prove the security of the proposed APAKE protocol (denoted by $\mathcal{P}$) in Theorems 1, 2, and 3 respectively.

THEOREM 1 (AKE SECURITY). *If $\mathcal{MAC}$ is suf-rmva secure and weak pseudorandom, the tag-randomization is simulatable, SPK is a labeled SE-NIZK, $\mathcal{PE}$ is IND-ET secure, $\mathcal{DS}$ is EUF-CMA secure, the DDH assumption holds in $\mathbb{G}$, and both H_1 and H_2 are random oracles, then our APAKE protocol guarantees the AKE security. In particular, we have*

$$\mathsf{Adv}^{\mathrm{AKE}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \le q_{\mathrm{se}}/|\mathcal{D}| + O\left(\mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{SPK}}(\mathcal{B}_1) + \mathsf{Adv}^{\text{ss-ext}}_{\mathsf{SPK}}(\mathcal{B}_2)\right)$$
$$+ \mathsf{Adv}^{\text{suf-rmva}}_{\mathcal{MAC}}(\mathcal{B}_3) + N^2/|\mathcal{M}_c| + q_s^2/p + N\mathsf{Adv}^{\mathrm{wpr}}_{\mathcal{MAC}}(\mathcal{B}_4) +$$
$$O\left(N|\mathcal{D}|/|\mathcal{M}_e|\right) + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{DS}}(\mathcal{B}_6) + O\left(Nq_cq_s\mathsf{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\mathcal{B}_7)\right),$$

where q_c is the maximum number of sessions per client, q_s is the maximum number of server sessions, $\mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{SPK}}$ (resp., $\mathsf{Adv}^{\text{ss-ext}}_{\mathsf{SPK}}$) is the advantage for the unbounded zero-knowledge (resp., simulation-sound extractability) of SPK, and $\mathsf{Adv}^{\mathrm{DDH}}_{\mathbb{G}}$ is the advantage for the DDH assumption.

PROOF. Firstly, we construct an algorithm $\mathsf{CredSim}$ who can generate presentation proofs without knowledge of sk and any credential using a zero-knowledge simulator Sim for SPK. We also construct an algorithm $\mathsf{CredExt}$ who can extract a message-tag pair from any presentation proof produced by $\mathcal{A}$ using an online extractor Ext for SPK.

$\mathsf{CredSim}(\mathsf{par_{mac}}, \ell)$: Given $\mathsf{par_{mac}}$ and a label ℓ, $\mathsf{CredSim}$ runs $(T', V') \leftarrow \mathsf{TVSim}(\mathsf{par_{mac}})$ and $\Sigma' \leftarrow \mathsf{Sim}((\mathsf{par_{mac}}, T', V'), \ell)$ and outputs a presentation proof $\sigma_C \leftarrow (T', V', \Sigma')$, where TVSim is defined in Section 2.3.

$\mathsf{CredExt}(\mathsf{par_{mac}}, \sigma_C, \ell)$: Given $\mathsf{par_{mac}}$, $\sigma_C = (T, V, \Sigma)$ and a label ℓ, $\mathsf{CredExt}$ runs $\mathsf{Ext}((\mathsf{par_{mac}}, T, V), \Sigma, \ell)$. If T is not correctly formed or Ext returns `invalid`,[2] $\mathsf{CredExt}$ outputs `invalid`. Otherwise (Ext returns witness (m, a)), $\mathsf{CredExt}$ computes $\sigma \leftarrow \mathsf{Derand}(T, a)$ and outputs (m, σ).

Let $\mathcal{A}$ be an adversary who aims at breaking the AKE security of our APAKE protocol $\mathcal{P}$. This proof will proceed via a sequence of games $G_0, G_1, \ldots, G_7$. We will bound the decrease in $\mathcal{A}$'s advantage between two successive games, and use $\mathsf{Adv}_i(\mathcal{A})$ to denote the advantage of $\mathcal{A}$ in game G_i.

Game G_0. This is the real game. Recall that $\mathcal{A}$ is given access to $\mathbf{Cred} = \{cred_i = (\mathsf{ID}_i, [\sigma_i]_{pw_i})\}_{i=1}^N$, params, and all the oracles specified in the security model, where σ_i is a credential on $m_i = H_1(\mathsf{ID}_i)$. We have $\mathsf{Adv}^{\mathrm{AKE}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) = \mathsf{Adv}_0(\mathcal{A})$.

Game G_1 (Simulate and Extract). This game is the same as game G_0, except that using $\mathsf{CredSim}$ to generate presentation proofs for client instances, for each presentation proof σ_C created by $\mathcal{A}$, rejecting σ_C if $\mathsf{CredExt}$ outputs `invalid` or $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 0$ for $(m, \sigma) \leftarrow \mathsf{CredExt}(\mathsf{par_{mac}}, \sigma_C, \ell)$, and accepting σ_C otherwise, where $\ell = (X, Y, \sigma_S)$.

Analysis. When $\mathsf{CredExt}(\mathsf{par_{mac}}, (T, V, \Sigma), \ell)$ outputs (m, σ), $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 1$ if and only if $V = f_{\mathrm{V}}(T, \mathsf{sk})$. Moreover, simulated T' has the same distribution as real T. Thus, G_1 has the same distribution as G_0, except that the proofs of SPK are simulated by Sim and Ext fails for extraction. Let $\mathcal{B}_1$ (resp., $\mathcal{B}_2$) be an algorithm that breaks the unbounded zero-knowledge (resp., simulation-sound extractability) of SPK by interacting with $\mathcal{A}$. Then, we have

[2] If $\mathsf{Verify}_{\mathsf{SPK}}((\mathsf{par_{mac}}, T, V), \Sigma, \ell) = 0$, Ext returns `invalid`.

Client (params, pw)	**secure channel**	**Server** (params, SK)
	$\xrightarrow{\mathsf{ID}}$	$m \leftarrow H_1(\mathsf{ID})$, $\sigma \leftarrow \mathsf{MAC}(\mathsf{sk}, m)$
$m \leftarrow H_1(\mathsf{ID})$		$\pi \leftarrow \mathsf{NIZK}\{(\mathsf{sk}) : \mathsf{Verify}(\mathsf{sk}, m, \sigma) = 1 \wedge$
Abort if $\mathsf{Verify}_{\mathsf{NIZK}}((\mathsf{par}_{\mathsf{mac}}, m, \sigma), \pi) = 0$	$\xleftarrow{(\sigma, \pi)}$	$(\mathsf{par}_{\mathsf{mac}}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)\}$
$[\sigma]_{pw} \leftarrow \mathsf{Enc}_{pw}(\sigma)$		
Store $cred \leftarrow (\mathsf{ID}, [\sigma]_{pw})$		

Figure 2: Registration Protocol of Our APAKE Protocol

Client $(\mathsf{params}, pw, cred = (\mathsf{ID}, [\sigma]_{pw}))$		**Server** (params, SK)
Abort if $\mathsf{Ver}(\mathrm{PK}, Y, \sigma_S) = 0$		$y \xleftarrow{\$} \mathbb{Z}_p^*$, $Y \leftarrow g^y$
$m \leftarrow H_1(\mathsf{ID})$, $\sigma \leftarrow \mathsf{Dec}_{pw}([\sigma]_{pw})$	$\xleftarrow{(Y, \sigma_S)}$	$\sigma_S \leftarrow \mathsf{Sign}(\mathrm{SK}, Y)$
$x \xleftarrow{\$} \mathbb{Z}_p^*$, $X \leftarrow g^x$		
$\sigma_C \leftarrow \mathsf{Show}(\mathsf{par}_{\mathsf{mac}}, m, \sigma, (X, Y, \sigma_S))$	$\xrightarrow{(X, \sigma_C)}$	Abort if $\mathsf{ShowVerify}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, (X, Y, \sigma_S), \mathsf{sk}) = 0$
$K \leftarrow H_2(Y, \sigma_S, X, \sigma_C, Y^x)$		$K \leftarrow H_2(Y, \sigma_S, X, \sigma_C, X^y)$

Figure 3: Login Protocol of Our APAKE Protocol

$$\mathsf{Adv}_0(\mathcal{A}) = \mathsf{Adv}_1(\mathcal{A}) + O\left(\mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{SPK}}(\mathcal{B}_1) + \mathsf{Adv}^{\text{ss-ext}}_{\mathsf{SPK}}(\mathcal{B}_2)\right).$$

Game G_2 (MAC Forgery). This game is the same as G_1, except that rejecting any presentation proof σ_C produced by $\mathcal{A}$ such that $\mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, \ell)$ outputs (m, σ) and $(m, \sigma) \notin \{(m_i, \sigma_i)\}_{i=1}^N$.

Analysis. We bound the decrease in $\mathcal{A}$'s advantage from G_1 to G_2 using a reduction from the suf-rmva security of $\mathcal{MAC}$. Let $\mathcal{B}_3$ be an algorithm that has access to $\mathsf{par}_{\mathsf{mac}}$, an oracle MAC and a verification oracle VERIFY. $\mathcal{B}_3$ executes just as in G_1 and interacts with $\mathcal{A}$, with the following exceptions:

- $\mathcal{B}_3$ generates the system parameters params using $\mathsf{par}_{\mathsf{mac}}$. Then $\mathcal{B}_3$ makes N queries to oracle MAC and obtains $\{(m_i, \sigma_i)\}_{i=1}^N$ for $m_i \xleftarrow{\$} \mathcal{M}_c$. For each $i \in [N]$, $\mathcal{B}_3$ programs random oracle H_1 such that $H_1(\mathsf{ID}_i) = m_i$.
- For each $(m, \sigma) \leftarrow \mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, \ell)$, $\mathcal{B}_3$ accepts σ_C if and only if oracle $\mathrm{VERIFY}(\mathsf{sk}, m, \sigma)$ returns 1.

If $\mathcal{A}$ behaves differently between G_1 and G_2, there exists a pair $(m^*, \sigma^*) \leftarrow \mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C^*, \ell^*)$ such that oracle $\mathrm{VERIFY}(\mathsf{sk}, m^*, \sigma^*)$ returns 1 and $(m^*, \sigma^*) \notin \{(m_i, \sigma_i)\}_{i=1}^N$. Then $\mathcal{B}_3$ can output (m^*, σ^*) as its forgery. Thus, we have

$$\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\text{suf-rmva}}_{\mathcal{MAC}}(\mathcal{B}_3).$$

Game G_3 (Exclude Collisions). This game is the same as G_2, except that aborting if the event $\mathtt{abort}_1$ that $m_i = m_j$ for some $i, j \in [N]$ and $i \neq j$ occurs, for each $(m, \sigma) \leftarrow \mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, \ell)$ such that $m = m_i$ for some $i \in [N]$, accepting σ_C iff $\sigma = \sigma_i$, aborting if the event $\mathtt{abort}_2$ that the messages from server instances encounter a collision occurs.

Analysis. Since H_1 is a random oracle, $\Pr[\mathtt{abort}_1] \leq N^2 / |\mathcal{M}_c|$. Clearly, $\Pr[\mathtt{abort}_2] \leq q_s^2/p$. So we have

$$\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + N^2 / |\mathcal{M}_c| + q_s^2/p.$$

Game G_4 (Randomize Credentials). This game is the same as G_3, except that replacing $[\sigma_i]_{pw_i}$ with $[R_i]_{pw_i}$ for each $i \in [N]$ where $R_i \xleftarrow{\$} \mathcal{T}$, for each $(m, \sigma) \leftarrow \mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, \ell)$ such that $m = m_i$ for some $i \in [N]$, accepting σ_C if and only if $\sigma = R_i$.

Analysis. We bound the decrease in $\mathcal{A}$'s advantage from G_3 to G_4 using a reduction from the weak pseudorandomness of $\mathcal{MAC}$. We use a hybrid argument to complete the reduction. For each $j \in [N] \cup \{0\}$, let game $G_{3,j}$ be the same as G_3, except that setting $cred_i \leftarrow (\mathsf{ID}_i, [R_i]_{pw_i})$ for each $i \in [j]$ where $R_i \xleftarrow{\$} \mathcal{T}$, and using $\{R_i\}_{i \in [j]}$ to verify the presentation proofs produced by $\mathcal{A}$. It is clear that $G_{3,0}$ and $G_{3,N}$ are the same as G_3 and G_4 respectively. If $\mathcal{A}$ behaves differently between G_3 and G_4 with probability ϵ, then $\mathcal{A}$ must behave differently between $G_{3,j-1}$ and $G_{3,j}$ for some $j \in [N]$ with probability at least ϵ/N. We can construct an algorithm $\mathcal{B}_4$ which breaks the weak pseudorandomness of $\mathcal{MAC}$ with probability at least ϵ/N via interacting with $\mathcal{A}$. $\mathcal{B}_4$ is given $\mathsf{par}_{\mathsf{mac}}$, a challenge (m^*, σ^*), and an oracle MAC. $\mathcal{B}_4$ makes $N - j$ MAC queries and obtains $N - j$ message-tag pairs $\{(m_i, \sigma_i)\}_{i=j+1}^N$. Then, $\mathcal{B}_4$ programs random oracle H_1 such that $H_1(\mathsf{ID}_j) = m^*$ and $H_1(\mathsf{ID}_i) = m_i$ for each $i \in \{j+1, \ldots, N\}$. $\mathcal{B}_4$ also picks $j - 1$ random values $\{R_i\}_{i=1}^{j-1}$ in $\mathcal{T}$. Next, $\mathcal{B}_4$ executes as in G_3, except that $\mathcal{B}_4$ sets $cred_i \leftarrow (\mathsf{ID}_i, [R_i]_{pw_i})$ for each $i \in [j-1]$, $cred_j \leftarrow (\mathsf{ID}_j, [\sigma^*]_{pw_j})$ and $cred_i \leftarrow (\mathsf{ID}_i, [\sigma_i]_{pw_i})$ for each $i \in [N] \backslash [j]$, and $\mathcal{B}_4$ uses $\left(\{R_i\}_{i=1}^{j-1}, \sigma^*, \{\sigma_i\}_{i=j+1}^N\right)$ to verify the presentation proofs produced by $\mathcal{A}$. If $\sigma^* = \mathsf{MAC}(\mathsf{sk}, m^*)$, $\mathcal{B}_4$ behaves exactly as in $G_{3,j-1}$. Otherwise (i.e., σ^* is uniformly random in $\mathcal{T}$), $\mathcal{B}_4$ behaves exactly as in $G_{3,j}$. Then, $\mathsf{Adv}_{3,j-1}(\mathcal{A}) = \mathsf{Adv}_{3,j}(\mathcal{A}) + \mathsf{Adv}^{\mathrm{wpr}}_{\mathcal{MAC}}(\mathcal{B}_4)$. Thus, we have $\mathsf{Adv}_3(\mathcal{A}) \leq \mathsf{Adv}_4(\mathcal{A}) + N\mathsf{Adv}^{\mathrm{wpr}}_{\mathcal{MAC}}(\mathcal{B}_4)$.

Game G_5 (Randomize Ciphertexts). This game is the same as G_4, except that replacing $[R_i]_{pw_i}$ with a random ciphertext $C_i \in \mathcal{C}$ for each $i \in [N]$, and rejecting all client messages created by $\mathcal{A}$.

Analysis. In game G_4, since R_i is uniform at random in $\mathcal{M}_e = \mathcal{T}$ for each $i \in [N]$ and presentation proofs for client instances are simulated by $\mathsf{CredSim}$, the only way for guessing passwords is to amount on-line dictionary attacks by making $\mathtt{Send}$ queries to the server. Thus, we can bound the decrease in $\mathcal{A}$'s advantage from G_4 to G_5 using a re-

duction from the IND-ET security of $\mathcal{PE}$. The reduction is completed via a hybrid argument. For each $j \in [N] \cup \{0\}$, let game $G_{4,j}$ be the same as G_4, except that replacing $[R_i]_{pw_i}$ with a random ciphertext $C_i \in \mathcal{C}$ for each $i \in [j]$, and rejecting any client message (X, σ_C) produced by $\mathcal{A}$ if $(m, \sigma) \leftarrow \mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, \ell)$ and $m \in \{m_i\}_{i=1}^{j}$, where $\ell = (X, Y, \sigma_S)$. Clearly, $G_{4,0}$ and $G_{4,N}$ are the same as G_4 and G_5 respectively. If $\mathcal{A}$ behaves differently between $G_{4,j-1}$ and $G_{4,j}$ for some $j \in [N]$ with probability ϵ_j, then we can construct an algorithm $\mathcal{B}_{5,j}$ that breaks the IND-ET security of $\mathcal{PE}$ via interacting with $\mathcal{A}$ with almost the same probability. Specifically, $\mathcal{B}_{5,j}$ is given a challenge ciphertext $\mathfrak{C}$ and an equality test oracle $\mathcal{O}_{\text{ET}}(M, \cdot)$, where M is a random message in $\mathcal{M}_e$. Then $\mathcal{B}_{5,j}$ picks $C_i \xleftarrow{\$} \mathcal{C}$ for each $i \in [j-1]$ and $R_i \xleftarrow{\$} \mathcal{M}_e$ for each $i \in [N] \backslash [j]$. $\mathcal{B}_{5,j}$ executes just as in game G_4, with the following exceptions:

- For each $i \in [N]$, $\mathcal{B}_{5,j}$ sets the ciphertext in $cred_i$ as C_i if $i \in [j-1]$, $\mathfrak{C}$ if $i = j$, and $[R_i]_{pw_i}$ if $i \in [N] \backslash [j]$.
- For each $(m, \sigma) \leftarrow \mathsf{CredExt}(\mathsf{par}_{\mathsf{mac}}, \sigma_C, \ell)$ such that $m = m_i$ for some $i \in [N]$ where σ_C is produced by $\mathcal{A}$, $\mathcal{B}_{5,j}$ rejects σ_C if $i \in [j-1]$, or $i = j \wedge \mathcal{O}_{\text{ET}}(M, \sigma) = 0$, or $i \in [N] \backslash [j] \wedge \sigma \neq R_i$, and accepts σ_C otherwise.

If $\mathfrak{C} = \mathsf{Enc}_{pw}(M)$ where pw acts as the password pw_j of client ID_j, then $\mathcal{B}_{5,j}$ behaves exactly as in game $G_{4,j-1}$. Otherwise (i.e., $\mathfrak{C}$ is uniform at random in $\mathcal{C}$), $\mathcal{O}_{\text{ET}}(M, \cdot)$ returns 0 for all queries with probability at least $1 - (|\mathcal{D}| + 1)/|\mathcal{M}_e|$, since $\mathfrak{C}$ is independent from M and the probability that there exists a $pw' \in \mathcal{D}$ such that $\mathsf{Dec}_{pw'}(\mathfrak{C}) = M$ is at most $|\mathcal{D}|/|\mathcal{M}_e|$. Thus, $\mathcal{B}_{5,j}$ behaves exactly as in game $G_{4,j}$ with probability at least $1 - (|\mathcal{D}| + 1)/|\mathcal{M}_e|$. As a result, we have

$$\mathsf{Adv}_4(\mathcal{A}) - \mathsf{Adv}_5(\mathcal{A}) = \sum_{j=1}^{N} \left(\mathsf{Adv}_{4,j-1}(\mathcal{A}) - \mathsf{Adv}_{4,j}(\mathcal{A})\right) \leq$$

$$\sum_{j=1}^{N} \left(\mathsf{Adv}^{\text{IND-ET}}_{\mathcal{PE},\mathcal{D}}(\mathcal{B}_{5,j}) + \frac{|\mathcal{D}| + 1}{|\mathcal{M}_e|} \right) \leq \frac{q_{\text{se}}}{|\mathcal{D}|} + O\left(\frac{N|\mathcal{D}|}{|\mathcal{M}_e|}\right),$$

where the total number of queries to $\mathcal{O}_{\text{ET}}$ is bounded by q_{se}.

Game G_6 (Signature Forgery). This game is the same as G_5, except for aborting if the event $\mathtt{abort}_3$ that the first time some client instance accepts after receiving a signature σ_S^* on Y^* that was not output by a server instance occurs.

Analysis. Game G_6 has the same distribution as G_5 if $\mathtt{abort}_3$ does not occur. Thus, the difference between G_5 and G_6 can be bounded by a reduction from the EUF-CMA security of $\mathcal{DS}$. Let $\mathcal{B}_6$ be an algorithm that breaks the EUF-CMA security of $\mathcal{DS}$ via interacting with $\mathcal{A}$. $\mathcal{B}_6$ is given a public key PK^*, and sets PK^* as the server's public key. $\mathcal{B}_6$ simulates the protocol execution as in G_5, except that $\mathcal{B}_6$ generates signatures for all server instances by querying the signing oracle. If $\mathtt{abort}_3$ occurs, $\mathcal{B}_6$ outputs (Y^*, σ_S^*) as its forgery for $\mathcal{DS}$. Thus, we have $\mathsf{Adv}_5(\mathcal{A}) = \mathsf{Adv}_6(\mathcal{A}) + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{DS}}(\mathcal{B}_6)$.

Game G_7 (Randomize Session Key). This game is the same as G_6, except for replacing session key in test session $\tilde{U}^{\tilde{\delta}}$ and its partner $\tilde{V}^{\tilde{\rho}}$ with an independently random string.

Analysis. In this game, all client messages created by $\mathcal{A}$ are rejected, and $\mathcal{A}$ cannot replay presentation proofs from client instances since a message (Y, σ_S) from any server instance could act as a nonce and collision of the nonces is excluded in game G_3. Thus, we can bound the decrease in $\mathcal{A}$'s advantage from G_6 to G_7 using a reduction from the DDH assumption as follows. We construct an algorithm $\mathcal{B}_7$ that breaks the DDH assumption by interacting with $\mathcal{A}$. $\mathcal{B}_7$ is given an instance (g, g^u, g^v, W) of DDH and aims to distinguish $W = g^{uv}$ from a random element $W \in \mathbb{G}^*$. $\mathcal{B}_7$ simulates the protocol execution as in G_6 with the following exceptions. $\mathcal{B}_7$ distinguishes two cases. For case 1 that the test session is a server session, $\mathcal{B}_7$ picks $i^* \xleftarrow{\$} [q_s]$ and aborts if $i^* \neq \tilde{\delta}$. If $\mathcal{B}_7$ does not abort, it sets $Y = g^v$ and $X = g^u$ as the ephemeral Diffie-Hellman (DH) values for $\tilde{U}^{\tilde{\delta}}$ and its partner $\tilde{V}^{\tilde{\rho}}$ respectively. For case 2 that the test session is a client session, $\mathcal{B}_7$ picks $(\hat{U}, \hat{\delta}, j^*) \xleftarrow{\$} [N] \times [q_c] \times [q_s]$, and aborts if $(\hat{U}, \hat{\delta}) \neq (\tilde{U}, \tilde{\delta})$ or client instance $\tilde{U}^{\tilde{\delta}}$ receives the message that is not the output of the j^*-th session of the server. If $\mathcal{B}_7$ does not abort, it sets $X = g^u$ as the ephemeral DH value for $\hat{U}^{\hat{\delta}}$ and $Y = g^v$ as the one for the j^*-th server session. In both cases, $\mathcal{B}_7$ can respond all $\mathtt{Reveal}$ queries (including the sessions for $Y = g^v$) by computing the session keys using ephemeral DH exponents at the client side, since all accepted messages from clients are generated by $\mathcal{B}_7$. For both cases, $\mathcal{B}_7$ returns $H_2(Y, \sigma_S, X, \sigma_C, W)$ as the session key of the test session. If $\mathcal{B}_7$ does not abort, then $\mathcal{B}_7$ behaves exactly as in G_6 if $W = g^{uv}$, and behaves exactly as in G_7 if W is uniformly random in $\mathbb{G}^*$ since H_2 is a random oracle. Thus,

$$\mathsf{Adv}_6(\mathcal{A}) = \mathsf{Adv}_7(\mathcal{A}) + O\left(Nq_cq_s\mathsf{Adv}^{\text{DDH}}_{\mathbb{G}}(\mathcal{B}_7)\right).$$

Overall, we obtain the bound claimed in the theorem. □

THEOREM 2 (CLIENT AUTHENTICATION). *Our APAKE protocol $\mathcal{P}$ obtains clients-to-server authentication, provided that $\mathcal{MAC}$ is suf-rmva secure and weak pseudorandom, the tag-randomization is simulatable,* SPK *is a labeled SE-NIZK, $\mathcal{PE}$ is IND-ET secure, and H_1 is modeled as a random oracle. In particular, we have*

$$\begin{aligned}\mathsf{Adv}^{\text{C2S}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \leq\ & q_{\text{se}}/|\mathcal{D}| + O\left(\mathsf{Adv}^{\text{uzk}}_{\mathsf{SPK}}(\mathcal{B}_1) + \mathsf{Adv}^{\text{ss-ext}}_{\mathsf{SPK}}(\mathcal{B}_2)\right) \\ & + \mathsf{Adv}^{\text{suf-rmva}}_{\mathcal{MAC}}(\mathcal{B}_3) + N^2/\left|\mathcal{M}_c\right| + q_s^2/p + N\mathsf{Adv}^{\text{wpr}}_{\mathcal{MAC}}(\mathcal{B}_4) \\ & + O\left(N|\mathcal{D}|/|\mathcal{M}_e|\right).\end{aligned}$$

PROOF. (*Sketch*) This proof is proceeded by a series of games $G_0, G_1, \ldots, G_5$, where G_0 is the real game for client authentication, and $G_1, \ldots, G_5$ are essentially the same as that in the proof of Theorem 1. In game G_5, all client messages produced by $\mathcal{A}$ are rejected, and $\mathcal{A}$ cannot replay presentation proofs from client instances since a message (Y, σ_S) from any server instance could act as a nonce and collision of the nonces is excluded in game G_3. Thus, $\mathcal{A}$ cannot authenticate itself to the server in game G_5. □

THEOREM 3 (ANONYMITY). *If* NIZK *is sound,* SPK *is unbounded zero-knowledge, the tag-randomization is simulatable, and* KeyGen *satisfies key-parameter consistency, then our protocol $\mathcal{P}$ is anonymous. In particular, we have*

$$\mathsf{Adv}^{\text{anon}}_{\mathcal{P}}(\mathcal{A}) = \mathsf{Adv}^{\text{sound}}_{\mathsf{NIZK}}(\mathcal{B}_1) + \mathsf{Adv}^{\text{uzk}}_{\mathsf{SPK}}(\mathcal{B}_2).$$

where $\mathsf{Adv}^{\text{sound}}_{\mathsf{NIZK}}$ *is the advantage for the soundness of* NIZK.

PROOF. Let $\mathcal{A}$ be an adversary that breaks anonymity of $\mathcal{P}$. This proof will proceed via a series of games G_0, G_1, G_2, G_3, where G_0 is the real game. By $\mathsf{Adv}_i(\mathcal{A})$ we denote $\mathcal{A}$'s advantage in G_i, and $\mathsf{Adv}^{\text{anon}}_{\mathcal{P}}(\mathcal{A}) = \mathsf{Adv}_0(\mathcal{A})$.

Game G_1 (Soundness). This game is the same as G_0, except for aborting if the first time some client accepts a proof π on a statement $(\mathsf{par_{mac}}, m, \sigma)$ such that $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 0$.

Analysis. Since KeyGen satisfies key-parameter consistency, there exists the unique secret key sk such that $(\mathsf{par_{mac}}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)$. Thus, if $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 0$, then $(\mathsf{par_{mac}}, m, \sigma)$ is a false statement on NIZK. Then, we can construct an algorithm $\mathcal{B}_1$ that breaks the soundness of NIZK by interacting with $\mathcal{A}$. If $\mathcal{A}$ behaves differently between G_0 and G_1, then $\mathcal{B}_1$ can find a valid proof π on a false statement $(\mathsf{par_{mac}}, m, \sigma)$ such that $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 0$. Thus, we have $\mathsf{Adv}_0(\mathcal{A}) = \mathsf{Adv}_1(\mathcal{A}) + \mathsf{Adv}^{\mathrm{sound}}_{\mathsf{NIZK}}(\mathcal{B}_1)$.

Game G_2 (Simulate Proofs). This game is the same as G_1, except for using the zero-knowledge simulator Sim to generate a proof w.r.t. SPK for each challenge query.

Analysis. G_2 behaves exactly like G_1, except for the simulation of the proofs for SPK. From the unbounded zero-knowledge property of SPK, we have $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{SPK}}(\mathcal{B}_2)$, where $\mathcal{B}_2$ is an algorithm that breaks the unbounded zero-knowledge of SPK by interacting with $\mathcal{A}$.

Game G_3 (Simulate Tag-Randomization). This game is the same as G_2, except that for each challenge query replacing (T, V) with $(T', V') \leftarrow \mathsf{TVSim}(\mathsf{par_{mac}})$.

Analysis. In G_3, any message-tag pair (m, σ) accepted by any client ID satisfies $\mathsf{Verify}(\mathsf{sk}, m, \sigma) = 1$ where $m = H_1(\mathsf{ID})$. Thus, (T', V') simulated by TVSim has the same distribution as real (T, V). Then, we have $\mathsf{Adv}_2(\mathcal{A}) = \mathsf{Adv}_3(\mathcal{A})$.

Overall, we obtain the bound claimed in the theorem. □

5. INSTANTIATION OF APAKE

In this section, we instantiate the building blocks used in our APAKE construction. In particular, we give an example of suf-rmva secure and weak pseudorandom algebraic MACs with efficient labeled SE-NIZKs, which is a pairing-free variant of the weak Boneh-Boyen signature scheme [11]. We denote the algebraic MAC scheme by $\mathsf{MAC_{SDH}}$. The password-based encryption scheme $\mathcal{PE}$ is instantiated with an example recommended by Bellare and Rogaway [5] for the AuthA mechanism of IEEE P1363.2 standard [39], which satisfies the IND-ET security in the ROM. We use ECDSA to instantiate digital signature scheme $\mathcal{DS}$. Our instantiation assumes that $\mathsf{MAC_{SDH}}$, $\mathcal{PE}$, $\mathcal{DS}$ and key exchange use the same domain parameters $\mathsf{par} = (\mathbb{G}, p, g)$.

When applying the instantiations of the building blocks to the APAKE construction described in Section 4, we obtain a highly-efficient APAKE protocol, which is denoted by `APAKE`.

5.1 An SDH-based Algebraic MAC

The construction of $\mathsf{MAC_{SDH}}$ is described as follows.

$\mathsf{KeyGen}(1^\lambda)$ takes as input a security parameter 1^λ, chooses the group parameters $(\mathbb{G}, p, g)$ such that p is a 2λ-bit prime, picks $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$, computes $w \leftarrow g^\gamma$, and outputs $\mathsf{sk} = \gamma$ and $\mathsf{par_{mac}} = (\mathbb{G}, p, g, w)$. We assume that $(\mathbb{G}, p, g)$ is an implicit input in all the following algorithms.

$\mathsf{MAC}(\mathsf{sk}, m)$ takes as input $\mathsf{sk} = \gamma$ and a message $m \in \mathbb{Z}_p \backslash \{-\gamma\}$, computes $A \leftarrow g^{1/(\gamma+m)}$, and outputs $\sigma \leftarrow A$.

$\mathsf{Verify}(\mathsf{sk}, m, \sigma)$ takes as input $\mathsf{sk} = \gamma$, a message m and a tag $\sigma = A$, and outputs 1 if $A^{\gamma+m} = g$ and 0 otherwise.

Using the techniques in [11], we can prove that $\mathsf{MAC_{SDH}}$ is suf-rmva secure under the q-SDH assumption. Using the techniques in [11, 30], we can also prove that $\mathsf{MAC_{SDH}}$ is weak pseudorandom under the q-DDHI assumption. The detailed proofs can be found in Appendix A.

It is easy to see that KeyGen satisfies the key-parameter consistency, as (g, w) uniquely determines the secret key γ.

The $\mathsf{MAC_{SDH}}$ allows an efficient proof system $\mathsf{NIZK}\{(\gamma) : A^\gamma = A^{-m} g \wedge g^\gamma = w\}$, which is constructed using the Fiat-Shamir heuristic [33] to transform the corresponding Sigma protocol, and is shown as follows.

- On input a statement (g, w, m, A) and a witness γ, the prover picks $r \xleftarrow{\$} \mathbb{Z}_p$, computes $R_1 \leftarrow A^r$, $R_2 \leftarrow g^r$, $c \leftarrow H_3(g, w, m, A, R_1, R_2)$ and $s \leftarrow r + c\gamma \mod p$. Finally, it outputs a proof $\pi = (c, s)$.
- The verification algorithm $\mathsf{Verify_{NIZK}}$ takes as input a statement (g, w, m, A) and a proof $\pi = (c, s)$, and calculates $c' \leftarrow H_3(g, w, m, A, A^{s+cm} g^{-c}, g^s w^{-c})$. Then it outputs 1 if $c = c'$ and 0 otherwise.

Using the techniques in [50], one can prove that the NIZK is sound and unbounded zero-knowledge in the ROM.

Credential Presentation. The credential presentation algorithms Show and $\mathsf{ShowVerify}$ for $\mathsf{MAC_{SDH}}$ are constructed by $\mathsf{SPK}\{(m, a) : T^{-m} g^a = V\}(\ell)$ for $T = A^a$, and the SPK is instantiated by Fiat-Shamir transformed Sigma protocol.

- $\mathsf{Show}(m, \sigma, \ell)$: On input a message m, a credential $\sigma = A$ and a label ℓ, Show picks $a \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $T \leftarrow A^a$. Then it picks $r_m, r_a \xleftarrow{\$} \mathbb{Z}_p$, computes $R \leftarrow T^{-r_m} g^{r_a}$, $c \leftarrow H_4(g, T, R, \ell)$, $s_m \leftarrow r_m + cm \mod p$ and $s_a \leftarrow r_a + ca \mod p$, and sets $\Sigma = (c, s_m, s_a)$. Finally, Show outputs a presentation proof $\sigma_C = (T, \Sigma)$.
- $\mathsf{ShowVerify}(\sigma_C, \ell, \mathsf{sk})$: On input a presentation proof $\sigma_C = (T, (c, s_m, s_a))$, a label ℓ and $\mathsf{sk} = \gamma$, $\mathsf{ShowVerify}$ computes $V \leftarrow T^\gamma$, $R' \leftarrow T^{-s_m} g^{s_a} V^{-c}$ and $c' \leftarrow H_4(g, T, R', \ell)$. $\mathsf{ShowVerify}$ returns 1 if $T \neq 1$ and $c = c'$, and 0 otherwise.

It is easy to see that Rerand, Derand, f_{P} and f_{V} are specified as follows: $\mathsf{Rerand}(A)$ picks $a \xleftarrow{\$} \mathbb{Z}_p^*$, computes $T \leftarrow A^a$, and outputs (T, a); $\mathsf{Derand}(T, a)$ outputs $A \leftarrow T^{1/a}$; $f_{\mathrm{P}}(\mathsf{par_{mac}}, T, m, a) = T^{-m} g^a$ and $f_{\mathrm{V}}(T, \mathsf{sk}) = T^\gamma$.

The simulator TVSim is constructed as follows: given $\mathsf{par_{mac}} = (\mathbb{G}, p, g, w)$, TVSim chooses $t \xleftarrow{\$} \mathbb{Z}_p^*$, computes $T' \leftarrow g^t$ and $V' \leftarrow w^t$, and outputs (T', V'). Note that $V' = w^t = (T')^\gamma$ and T' has the same distribution as T generated by $\mathsf{Rerand}(A)$.

To obtain better efficiency, we do not involve V to σ_C following [24]. The reason behind this is that an online extractor Ext for SPK without knowing V can be obtained in the combined random oracle and generic group model following along the lines of [53, 56, 24]. The zero-knowledge simulator Sim for SPK can be constructed as follows: given a statement (g, T, V) and a label ℓ, Sim randomly picks $c, s_m, s_a \leftarrow \mathbb{Z}_p$ and programs the random oracle H_4 such that $H_4(g, T, R, \ell) = c$ with $R = T^{-s_m} g^{s_a} V^{-c}$, and outputs $\Sigma \leftarrow (c, s_m, s_a)$. Overall, the SPK is a labeled SE-NIZK.

Efficiency of Credential Presentation. For anonymous credentials, a presentation proof is usually generated using

Table 1: Comparison of APAKE Protocols

APAKE	Computation overhead		Comm.
	Client	Server	
[60]	$4E_{\mathbb{G}}$	$(N+3)E_{\mathbb{G}}$	$N+4$
[54]	$5E_{\mathbb{G}} + 1\mathtt{Dec}$	$(N+5)E_{\mathbb{G}} + N\mathtt{Enc}$	$2N+8$
[62]	$8E_{\mathbb{G}_1} + 1E^2_{\mathbb{G}_1} + 1E^5_{\mathbb{G}_1} + 2P$	$3E_{\mathbb{G}_1} + 3E^2_{\mathbb{G}_1} + 1E^5_{\mathbb{G}_1} + 2P$	18
APAKE	$3E_{\mathbb{G}} + 2E^2_{\mathbb{G}}$	$3E_{\mathbb{G}} + 1E^2_{\mathbb{G}}$	8

Legend: N is the total number of users, $E_{\mathbb{G}}$ denotes one exponentiation in $\mathbb{G}$, $E^n_{\mathbb{G}}$ denotes a multi-exponentiation of n values in $\mathbb{G}$, P represents a bilinear pairing operation, Enc (resp., Dec) denotes a symmetric-key encryption (resp., decryption) operation, and **Comm.** denotes the communication overhead.

the randomize-then-prove paradigm, where a credential is first randomized, and the randomized credential is then presented with a zero-knowledge proof. In general, the first step costs at least one exponentiation and the second step costs at least one multi-exponentiation, and the verification of presentation proofs costs at least one multi-exponentiation. Note that R' can be computed via $T^{-(s_m+c\gamma)}g^{s_a}$ in an execution of $\mathsf{ShowVerify}$. Therefore, the efficiency of $\mathsf{MAC}_{\mathsf{SDH}}$-based credential presentation is optimal.

5.2 An Example Password-Based Encryption

Let $H : \mathcal{D} \to \mathbb{G}^*$ be a cryptographic hash function modeled as a random oracle. An example of password-based encryption $\mathcal{PE}$ is described as follows.

$\mathsf{Enc}_{pw}(M)$ takes as input pw and a message $M \in \mathbb{G}^*$, computes $C \leftarrow M \cdot H(pw)$, and outputs C.

$\mathsf{Dec}_{pw}(C)$ takes as input pw and a ciphertext $C \in \mathbb{G}$, computes $M \leftarrow C/H(pw)$, and outputs M.

One can easily prove this example of $\mathcal{PE}$ is IND-ET secure, since the outputs of H are uniformly random.

5.3 Comparison of APAKE Protocols

We compare our protocol APAKE with the mechanisms specified by ISO/IEC 20009-4 in Table 1. For computation overhead, we only list the most time-consuming operations, and count the number of group elements in the login protocol. The output size of a hash function or a MAC algorithm is counted as a group element.

Table 1 shows that APAKE is much more efficient than the mechanisms [60, 54] in the password-only setting in terms of computation cost at the server side and communication overhead, and APAKE is also much more efficient than the mechanism [62] in the extra-storage setting.

6. SUPPORT OF REVOCATION

We present an accumulator-based revocation mechanism for APAKE, which enables non-membership proofs to be verified by a server with a secret key $\tilde{\gamma}$.

A server picks a random $\tilde{\gamma} \in \mathbb{Z}^*_p$, computes $\tilde{w} \leftarrow g^{\tilde{\gamma}}$, and adds $\tilde{w}$ to params and $\tilde{\gamma}$ to SK. The server maintains a revocation list $RL = \{(m_i, \tilde{V}_i)\}_{i=1}^{r}$, where $m_i = H_1(\mathsf{ID}_i)$ for a revoked user ID_i and $\tilde{V}_i = g^{1/\left(\prod_{j=1}^{i}(\tilde{\gamma}+m_j)\right)}$ for each $i \in [r]$. Let $\tilde{V}_0 = g$. For a registration request from a client ID, the server issues a witness $W_r \leftarrow \tilde{V}_r^{1/(\tilde{\gamma}+m)}$ along with $\tilde{\pi} \leftarrow \mathsf{NIZK}\{(\tilde{\gamma}) : W_r^{\tilde{\gamma}} = \tilde{V}_r W_r^{-m} \wedge g^{\tilde{\gamma}} = \tilde{w}\}$, and the client stores $(\mathsf{ID}, [A]_{pw}, W_r)$, where $m = H_1(\mathsf{ID})$.

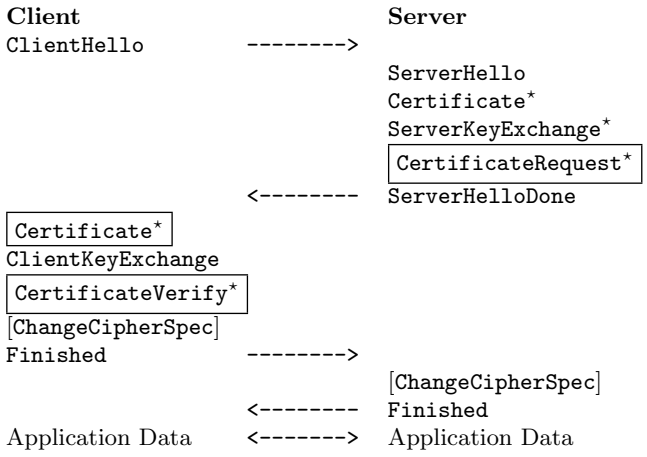

Figure 4: TLS Message Flows

Witness Update. Given the current revocation list $RL = \{(m_i, \tilde{V}_i)\}_{i=1}^{r'}$, an un-revoked user ID can update off-line her witness W_r iteratively with $m = H_1(\mathsf{ID})$ as follows:

$$W_{j+1} \leftarrow (W_j / \tilde{V}_{j+1})^{1/(m_{j+1}-m)}, \text{ for each } r \le j \le r' - 1.$$

Non-membership Proof. To prove that she has not been revoked, the client computes $m \leftarrow H_1(\mathsf{ID})$, randomizes the witness as $\tilde{T} \leftarrow W_r^z$ with a random $z \in \mathbb{Z}^*_p$, and generates $\tilde{\Sigma} \leftarrow \mathsf{SPK}\{(m, z) : \tilde{T}^{-m}\tilde{V}_r^z = \hat{V}\}$, where $\hat{V} = \tilde{T}^{\tilde{\gamma}}$.

Note that a revoked user ID_{j+1} $(r \le j < r')$ cannot update her witness, as the denominator $m_{j+1} - m = 0$. Using the technique in [11, Lemma 3.2], we can prove that no revoked user can forge a witness under the q-SDH assumption.

To support revocation, we extend the Show and $\mathsf{ShowVerify}$ algorithms in Section 5.1 by replacing the underlying SPK with $\mathsf{SPK}'\{(m, a, z) : T^{-m}g^a = V \wedge \tilde{T}^{-m}\tilde{V}_r^z = \hat{V}\}(\ell)$. By APAKEr we denote the APAKE protocol with above revocation.

7. APPLICATION OF APAKE TO TLS

We integrate APAKE into the TLS protocol to provide a TLS mode of server authentication with anonymous client authentication, where a client holding a password-wrapped credential and the password can authenticate herself to the server without revealing her identity, and denote the ECDSA-signed elliptic curve Diffie-Hellman ciphersuite with anonymous client authentication by ECDHE3.

Since a labeled SE-NIZK is also a signature of knowledge [23] where a label ℓ is the message to be signed, we consider Show and $\mathsf{ShowVerify}$ for $\mathsf{MAC}_{\mathsf{SDH}}$ as a signing algorithm and a verification algorithm respectively, where (m, A) is the secret key for generating signatures, and a signature σ_C on message ℓ is verified using sk. We refer to the "signature algorithm" as SigMAC, where $\mathsf{SigMAC} = (\mathsf{Show}, \mathsf{ShowVerify})$.

We also integrate APAKEr into TLS in the same way as APAKE except for using the extended $(\mathsf{Show}, \mathsf{ShowVerify})$ to support revocation and additionally publishing $\mathsf{Sign}(\text{SK}, \tilde{w})$. We denote the resulting ciphersuite by ECDHE4.

7.1 Integration into TLS

Assume that the server holds a certificate $cert_S$ on his ECDSA public key PK. The set of domain parameters $\mathsf{par} =$

Table 2: Performance of HTTPS using Apache with OpenSSL

Ciphersuite	Connections / second				Connection time (ms)	Handshake (bytes)	Client Auth.
	10 B payload	1 KB payload	10 KB payload	100 KB payload			
ECDHE1	2043.678 (1.06)	2022.282 (1.61)	1833.658 (1.80)	943.266 (1.09)	1.54 (0.05)	2200	None
ECDHE2	1133.08 (1.50)	1129.442 (1.69)	1075.69 (0.48)	718.736 (0.21)	2.39 (0.05)	3806	plain sigs
ECDHE3	1007.308 (2.25)	999.994 (1.74)	953.698 (1.28)	661.652 (1.02)	2.80 (0.01)	4078	anon. sigs
ECDHE4	863.712 (1.49)	860.364 (1.32)	826.032 (0.92)	602.928 (0.24)	3.40 (0.02)	4179	anon. sigs w/revoc.

Legend: mean, (std. dev.) in columns 2-6; **Client Auth.** represents the type of signatures used to provide client authentication.

$(\mathbb{G}, p, g)$ is selected from a trusted published source such as a standard, and its identifier is denoted by `parid`. The server acts as a Certification Authority to generate an ECDSA signed certificate $cert_{mac}$ on a dummy entity "apake" and public key w. Then $cert_{mac}$ is published and used by all clients who provide anonymous authentication with SigMAC. In the registration phase, each client should first check the validity of $cert_S$ and $cert_{mac}$, and then stores $cert_{mac}$ along with a password-wrapped credential $(\mathsf{ID}, [A]_{pw})$.

When using the X.509 certificate with ASN.1 data type, the certificate $cert_{mac}$ is defined as follows:

```
Certificate ::= SEQUENCE {
toBeSigned TBSCertificate,
algorithmIdentifier {ECDSA},
encrypted certsig,
... }
TBSCertificate ::= SEQUENCE { ...
subjectPublicKeyInfo SubjectPublicKeyInfo,
... }
SubjectPublicKeyInfo ::= SEQUENCE {
algorithm {SigMAC},
subjectPublicKey parid||w,
... },
```

where `SigMAC` specifies the $\mathsf{SigMAC} = (\mathsf{Show}, \mathsf{ShowVerify})$ algorithm, `parid||w` denotes the bit-string of $(\mathtt{parid}, w)$, $\mathtt{certsig} = \mathsf{Sign}(\mathrm{SK}, \mathtt{TBSCertificate})$, and the omitted fields are specified following the X.509 specification.

Below, we show how to integrate the `APAKE` into TLS [28]. We assume that a client has already recovered the "secret key" $(m = H_1(\mathsf{ID}), A)$ from a password-protected credential $(\mathsf{ID}, [A]_{pw})$ with her password pw, before initiating the TLS protocol. The ciphersuite `ECDHE3` is the same as the ECDSA-signed elliptic curve Diffie-Hellman ciphersuite with client authentication using ECDSA signatures, except that the messages marked with rectangles in Figure 4 are different. The server adds `apake` and (`SigMAC`, `SHA256`) to the fields of `CertificateRequest`, which is showed as follows:

```
struct {
ClientCertificateType {...,apake};
SignatureAndHashAlgorithm {...,(SigMAC, SHA256)};
... } CertificateRequest;
```

A client then utilizes $cert_{mac}$ in `Certificate`, and invokes the `SigMAC` to sign the handshake messages with the "secret key" (m, A) and generates `CertificateVerify`. The server can check the validity of $cert_{mac}$, and verify the validity of a "signature" σ_C using `SigMAC` and his secret key `sk`.

7.2 Implementation

We implemented in C the `ECDHE3` and `ECDHE4`, based on the OpenSSL v1.0.2g. For comparison, we also implemented the ECDSA-signed elliptic curve Diffie-Hellman ciphersuite with only server authentication (resp., mutual authentication), which is denoted by `ECDHE1` (resp., `ECDHE2`) and is included in TLSv1.2. Our implementation take places at a 128-bit security level, and uses the `secp256r1` curve. Apart from digital signature algorithms for client authentication, the ciphersuites `ECDHE1`, `ECDHE2`, `ECDHE3` and `ECDHE4` share the same ingredients, i.e., `ECDHE-ECDSA-AES128-GCM-SHA256`, where `AES128-GCM` denotes authenticated encryption (with associated data) using AES-128 in GCM (Galois Counter Mode). In the implementation of `ECDHE4`, we assume that the client has already updated off-line her witness W_r.

Experiment environment. Our experimental results are obtained in two desktop computers. The "client" computer has an Intel i5-3470 processor with 4 cores running at 3.2 GHz each. The "server" computer has an Intel Core2 Duo E7300 processor with 2 cores running at 2.66 GHz each. Both computers run the operating system of Ubuntu 15.04. Our softwares were both complied for the x86_64 architecture with -O2 optimizations using g++ 4.9.2.

7.3 Performance Evaluation

The performance of `ECDHE1`, `ECDHE2`, `ECDHE3` and `ECDHE4` within the context of an HTTPS connection is shown in Table 2. The approach for analyzing the performance in TLS/HTTPS follows that of Gupta et al. [37]. Besides, we follow the method of [13] to achieve the experimental data. The client and server computation platforms were connected over an isolated local area network with less than 1 ms ping time. The server was running Apache `httpd` 2.4.20 with the prefork multi-threading module.

The first section of Table 2 shows the number of simultaneous connections supported by the server. The client computer was running `siege` 4.0.1 tool[3] to create many HTTP connections in parallel for TLS. We did separate tests in the different HTTP payloads (10 bytes, 1 KB=1024 bytes, 10 KB, and 100 KB), so that simulating a variety of web page sizes. Each test was run for 100 seconds, and the results reported in Table 2 are the average of 5 runs with standard deviation listed in parentheses. During all tests, the client computer and network configuration was enough to ensure that the server's processor had at least 95% utilization. Note that session resumption was disabled. The second section of this table reports the average time which is required for a client to establish a connection, and is measured by using Wireshark from when the client opens the TCP connection to the server's IP address to when the client starts to receive the first packet of application data. The third section of Table 2 reports the size of the handshakes.

[3] `http://download.joedog.org/siege/siege-4.0.1.tar.gz`

Table 2 shows that ECDHE3 obtains between a factor 1.43-2.03x fewer HTTPS connections per second than ECDHE1, and between a factor 1.09-1.13x fewer HTTPS connections per second than ECDHE2. The ECDHE4 obtains between a factor 1.10-1.17x fewer HTTPS connections per second than ECDHE3. The average connection time and the size of the handshakes for ECDHE3 and ECDHE4 are attractive.

8. REFERENCES

[1] https://krebsonsecurity.com/2016/03/crooks-steal-sell-verizon-enterprise-customer-data/.

[2] M. Abdalla. Password-based authenticated key exchange: An overview. In *Provable Security 2014*, volume 8782 of *LNCS*, pages 1–9. Springer, 2014.

[3] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, pages 614–629.

[4] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, pages 139–155. Springer.

[5] M. Bellare and P. Rogaway. The AuthA protocol for password-based authenticated key exchange. Contribution to IEEE P1363.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pages 62–73. ACM Press, 1993.

[7] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[8] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO 2013*, volume 8042 of *LNCS*, pages 449–475. Springer, 2013.

[9] D. Bernhard, M. Fischlin, and B. Warinschi. Adaptive proofs of knowledge in the random oracle model. In *Public-Key Cryptography - PKC 2015*, pages 629–649.

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *EUROCRYPT 2004*, pages 223–238. Springer-Verlag.

[11] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, pages 56–73.

[12] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, 2004.

[13] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, 2015.

[14] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *EUROCRYPT 2000*, pages 156–171.

[15] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM CCS 2004*, pages 132–145. ACM Press.

[16] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *ASIACRYPT 2015*, pages 262–288. Springer.

[17] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, pages 268–289. Springer.

[18] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer-Verlag.

[19] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer-Verlag, 2001.

[20] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, pages 410–424.

[21] A. Cassola, E.-O. Blass, and G. Noubir. Authenticating privately over public Wi-Fi hotspots. In *ACM CCS*, pages 1346–1357. ACM, 2015.

[22] E. Cesena, H. Löhr, G. Ramunno, A.-R. Sadeghi, and D. Vernizzi. Anonymous authentication with TLS and DAA. In *TRUST 2010*, pages 47–62. Springer.

[23] M. Chase and A. Lysyanskaya. On signatures of knowledge. In *CRYPTO 2006*, pages 78–96. Springer-Verlag.

[24] M. Chase, S. Meiklejohn, and G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In *ACM CCS 2014*, pages 1205–1216. ACM Press. Full version is available at http://eprint.iacr.org/2013/516.

[25] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO 1982*, pages 199–203, 1982.

[26] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, pages 1030–1044, 1985.

[27] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, pages 257–265. Springer-Verlag.

[28] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246 (Proposed Standard), August 2008.

[29] Y. Dodis, E. Kiltz, K. Pietrzak, and D. Wichs. Message authentication, revisited. In *EUROCRYPT'12*, volume 7237 of *LNCS*, pages 355–374. Springer-Verlag, 2012.

[30] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography – PKC 2005*, pages 416–431. Springer-Verlag, 2005.

[31] European Parliament and Council of the European Union. Directive 2009/136/EC. Official Journal of the European Union, 2009.

[32] D. Fett, R. Küsters, and G. Schmitz. SPRESSO: A secure, privacy-respecting single sign-on system for the web. In *ACM CCS 2015*, pages 1358–1369.

[33] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, pages 186–194. Springer-Verlag.

[34] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer-Verlag.

[35] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, pages 281–308, 1988.

[36] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, pages 444–459. Springer-Verlag.

[37] V. Gupta, D. Stebila, S. Fung, S. C. Shantz, N. Gura, and H. Eberle. Speeding up secure web transactions using elliptic curve cryptography. In *NDSS*, 2004.

[38] IBM. Specification of the Identity Mixer Cryptographic Library. IBM Research Report RZ 3730, 2010.

[39] IEEE 1363.2. IEEE standard specifications for password based public-key cryptographic techniques. *IEEE Std 1363.2-2008*, pages 1–127, 2009.

[40] ISO/IEC 11770-4. Information technology – Security techniques – Key management – Part 4: Mechanisms based on weak secrets, 2006.

[41] ISO/IEC 11889:2015. Information technology - Trusted Platform Module Library, 2015.

[42] ISO/IEC 20008. Information technology - Security techniques - Anonymous digital signatures, 2013.

[43] ISO/IEC DIS 20009-4. Information technology – Security techniques – Anonymous entity authentication – Part 4: Mechanisms based on weak secrets, 2015.

[44] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human memorable passwords. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer-Verlag.

[45] Y. Lindell. Anonymous authentication. *Journal of Privacy and Confidentiality*, 2(2):4, 2007.

[46] M. Naor and O. Reingold. Synthesizers and their

application to the parallel construction of pseudo-random functions. *Computer and Systems Sciences*, 58(2):336–375, April 1999.
[47] NISTIR 8062. Privacy risk management for federal information systems, May 2015.
[48] C. Paquin and G. Zaverucha. U-Prove Cryptographic Specification V1.1 (Revision 3). Microsoft, 2013.
[49] D. Pointcheval. Password-based authenticated key exchange. In *PKC 2012*, pages 390–397. Springer.
[50] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
[51] K. Rannenberg, J. Camenisch, and A. Sabouri. *Attribute-based Credentials for Trust - Identity in the Information Society*. Springer, 2015.
[52] H. A. Schmidt. National strategy for trusted identities in cyberspace. Cyberwar Resources Guide, Item 163, 2010.
[53] C. Schnorr. Security of blind discrete log signatures against interactive attacks. In *Information and Communications Security*, volume 2229 of *LNCS*, pages 1–12. 2001.
[54] S. Shin and K. Kobara. Anonymous password authenticated key exchange: New construction and its extensions. *IEICE*, 93(1):102–115, 2010.
[55] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT 1997*, pages 256–266.
[56] N. P. Smart. The exact security of ECIES in the generic group model. In *Cryptography and Coding*, pages 73–84. Springer, 2001.
[57] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin. Using the secure remote password (SRP) protocol for TLS authentication. *RFC 5054*, 2007.
[58] D. Q. Viet, A. Yamamura, and H. Tanaka. Anonymous password-based authenticated key exchange. In *INDOCRYPT 2005*, volume 3797 of *LNCS*, pages 244–257.
[59] J. Walker and J. Li. Key exchange with anonymous authentication using DAA-SIGMA protocol. In *Trusted Systems*, volume 6802 of *LNCS*, pages 108–127. 2011.
[60] J. Yang and Z. Zhang. A new anonymous password-based authenticated key exchange protocol. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 200–212. Springer.
[61] Y. Yang, J. Zhou, J. Weng, and F. Bao. A new approach for anonymous password authentication. In *ACSAC 2009*, pages 199–208. IEEE.
[62] Y. Yang, J. Zhou, J. W. Wong, and F. Bao. Towards practical anonymous password authentication. In *ACSAC 2010*, pages 59–68. ACM.

APPENDIX

A. SECURITY PROOFS OF $\mathsf{MAC}_{\mathsf{SDH}}$

In this section, we prove that $\mathsf{MAC}_{\mathsf{SDH}}$ is suf-rmva secure in Theorem 4 and weak pseudorandom in Theorem 5.

THEOREM 4. *If the q-SDH assumption holds in $\mathbb{G}$, then the $\mathsf{MAC}_{\mathsf{SDH}}$ scheme is suf-rmva secure.*

PROOF. If there exists an adversary $\mathcal{A}$ that makes q_m queries to oracle MAC and q_v queries to oracle VERIFY, and breaks the suf-rmva security of $\mathsf{MAC}_{\mathsf{SDH}}$ with probability ϵ, then we can construct an algorithm $\mathcal{B}$ that breaks the q-SDH assumption with probability $\epsilon/(q_v+1)$ by interacting with $\mathcal{A}$, where $q = q_m + 1$.

Given a q-SDH instance $(g, g^{\gamma}, \ldots, g^{\gamma^q}) \in (\mathbb{G}^*)^{q+1}$ for some unknown $\gamma \in \mathbb{Z}_p^*$, $\mathcal{B}$ aims to output a solution $(c, g^{1/(\gamma+c)})$ for some $c \in \mathbb{Z}_p \backslash \{-\gamma\}$. $\mathcal{B}$ picks $i^* \xleftarrow{\$} [q_v+1]$ as the guess that the first fresh and valid forgery appears in the i^*-th verification query if $1 \le i^* \le q_v$ or the final output of $\mathcal{A}$ if $i^* = q_v + 1$. Then $\mathcal{B}$ chooses $m_1, m_2, \ldots, m_{q-1} \xleftarrow{\$} \mathbb{Z}_p$. Let $f(x) = \prod_{j=1}^{q-1}(x+m_j) = \sum_{j=0}^{q-1}\alpha_j x^j$ and $f_i(x) = f(x)/(x+m_i) = \prod_{j=1,j\neq i}^{q-1}(x+m_j) = \sum_{j=0}^{q-2}\beta_{i,j}x^j$ for each $i \in [q-1]$. Using the tuple $(g, g^{\gamma}, \ldots, g^{\gamma^q})$ and the technique of Boneh-Boyen [11, Lemma 3.2], $\mathcal{B}$ can compute $g' = \prod_{j=0}^{q-1}(g^{\gamma^j})^{\alpha_j} = g^{f(\gamma)}$, $w = \prod_{j=1}^{q}(g^{\gamma^j})^{\alpha_{j-1}} = g^{\gamma f(\gamma)} = (g')^{\gamma}$, and $A_i = \prod_{j=0}^{q-2}(g^{\gamma^j})^{\beta_{i,j}} = g^{f_i(\gamma)} = g^{f(\gamma)/(\gamma+m_i)} = (g')^{1/(\gamma+m_i)}$ for each $i \in [q-1]$. Next, $\mathcal{B}$ returns $\mathsf{par}_{\mathsf{mac}} = (\mathbb{G}, p, g', w)$ to $\mathcal{A}$ and responds the queries made by $\mathcal{A}$ as below:

For i-th MAC query, $\mathcal{B}$ returns (m_i, A_i) to $\mathcal{A}$.

For i-th verification query (m_i', A_i'), $\mathcal{B}$ responds as follows:

- If $i = i^*$, $\mathcal{B}$ sets $(m, A) = (m_i', A_i')$ and aborts.
- Otherwise (i.e., $i < i^*$), $\mathcal{B}$ returns 1 if $(m_i', A_i') = (m_j, A_j)$ for some $1 \le j \le q-1$ and 0 otherwise.

Finally, if $\mathcal{B}$ does not abort, $\mathcal{A}$ outputs a forgery (m^*, A^*). In this case $(i^* = q_v + 1)$, $\mathcal{B}$ sets $(m, A) = (m^*, A^*)$.

If $\mathcal{B}$ guesses correctly with probability $1/(q_v+1)$, $\mathcal{B}$'s simulation is perfect, and (m, A) is a fresh and valid forgery, i.e., $(m, A) \notin \{(m_i, A_i)\}_{i=1}^{q-1}$ and $A^{\gamma+m} = g'$. Thus, we have $m \notin \{m_1, \ldots, m_{q-1}\}$, since for any valid message-tag pair $(\tilde{m}, \tilde{A})$ under g' and γ, $\tilde{A}$ is uniquely determined by $\tilde{m}$. Let $f(x) = f'(x)(x+m) + \eta$ for some $\eta \in \mathbb{Z}_p^*$, and write $f'(x) = \sum_{j=0}^{q-2}\delta_j x^j$. Note that $A = (g')^{1/(\gamma+m)} = g^{f(\gamma)/(\gamma+m)} = g^{f'(\gamma)+\eta/(\gamma+m)}$. $\mathcal{B}$ computes $g^{1/(\gamma+m)} = \left(A/g^{f'(\gamma)}\right)^{1/\eta}$ with $g^{f'(\gamma)} = \prod_{j=0}^{q-2}(g^{\gamma^j})^{\delta_j}$, and outputs $(m, g^{1/(\gamma+m)})$ as a solution of the q-SDH problem. □

THEOREM 5. *If the q-DDHI assumption holds in group $\mathbb{G}$, then $\mathsf{MAC}_{\mathsf{SDH}}$ is weak pseudorandom.*

PROOF. If there exists an adversary $\mathcal{A}$ that makes q_m queries to oracle MAC and breaks the weak pseudorandomness of $\mathsf{MAC}_{\mathsf{SDH}}$ with probability ϵ, then we can construct an algorithm $\mathcal{B}$ that breaks the q-DDHI assumption with probability at least $\epsilon - (q-1)/p$ via interacting with $\mathcal{A}$, where $q = q_m + 1$.

Given $(g, g^{\alpha}, \ldots, g^{\alpha^q}, T) \in (\mathbb{G}^*)^{q+2}$ for some unknown $\alpha \in \mathbb{Z}_p^*$, $\mathcal{B}$ aims to distinguish $T = g^{1/\alpha}$ from a random element T. Firstly, $\mathcal{B}$ picks $m_1, \ldots, m_{q-1}, m \xleftarrow{\$} \mathbb{Z}_p$. Using the Binomial Theorem, $\mathcal{B}$ can compute the tuple $(g, g^{\gamma}, \ldots, g^{\gamma^q})$ where $\gamma = \alpha - m$. Then, as in the proof of Theorem 4, $\mathcal{B}$ generates $g' = g^{f(\gamma)}$, $w = (g')^{\gamma}$, and $A_i = (g')^{1/(\gamma+m_i)}$ for each $i \in [q-1]$, where $f(x) = \prod_{j=1}^{q-1}(x+m_j)$. Next, $\mathcal{B}$ returns $\mathsf{par}_{\mathsf{mac}} = (\mathbb{G}, p, g', w)$ to $\mathcal{A}$. For i-th query to oracle MAC, $\mathcal{B}$ returns a pair (m_i, A_i) to $\mathcal{A}$. If m is equal to one of $m_1, \ldots, m_{q-1}$, $\mathcal{B}$ aborts. Otherwise, the polynomial $f(x)$ is not divisible by $(x+m)$. In this case, we have $f(x) = q(x)(x+m) + \eta$ for some $\eta \neq 0$, and write $q(x) = \sum_{j=0}^{q-2}\delta_j x^j$. Then, $\mathcal{B}$ computes $g^{q(\gamma)} = \prod_{j=0}^{q-2}(g^{\gamma^j})^{\delta_j}$. Next, $\mathcal{B}$ returns $(m, \sigma = g^{q(\gamma)} \cdot T^{\eta})$ to $\mathcal{A}$ as the challenge. Finally, $\mathcal{A}$ outputs a guess b', and $\mathcal{B}$ outputs b'.

It is easy to see that the simulation of oracle MAC is perfect. If $T = g^{1/\alpha}$, then

$$\sigma = g^{q(\gamma)}T^{\eta} = g^{q(\gamma)}g^{\eta/(\gamma+m)} = g^{f(\gamma)/(\gamma+m)} = (g')^{1/(\gamma+m)}.$$

If T is uniformly distributed in $\mathbb{G}^*$, then so is σ. If $\mathcal{B}$ does not abort, $\mathcal{B}$ succeeds if $\mathcal{A}$ wins. As $\Pr\left[\exists i \in [q-1]\ s.t.\ m = m_i\right] \le (q-1)/p$, we have the claimed bound. □