Round-Efficient Anonymous Password-Authenticated Key Exchange Protocol in the Standard Model

Qihui Zhang[1], Wenfen Liu[2(✉)], Kang Yang[3], Xuexian Hu[1], and Ying Mei[4]

[1] PLA SSF Information Engineering University, Zhengzhou 450001, China
[2] Guilin University of Electronic Technology, Guilin 541004, China
liuwenfen@guet.edu.cn
[3] State Key Laboratory of Cryptology, Beijing 100878, China
[4] Automobile NCO School, Army Military Transportation University, Bengbu 233000, China

Abstract. Anonymous password-authenticated key exchange (APAKE) protocols allow for authenticating legitimate users via low-entropy passwords while keeping their actual identities private. They are important cryptographic primitives for privacy protection, which have attracted much attention recently and have been standardized in the international standard ISO/IEC 20009-4. However, most of the existing APAKE schemes (especially including all the APAKE schemes in the storage-extra setting) are developed in the random oracle model. In this paper, we present the first storage-extra APAKE protocol in the standard model by combing the technique of algebraic MAC with oblivious designated-verifier non-interactive zero-knowledge (DVNIZK) proof. Toward our aim, we first give out a new construction of the oblivious DVNIZK proof system, which is compatible with a new class of algebraic MAC schemes. As a consequence, our APAKE protocol needs only 2 flows of messages in the authentication phase, which is very efficient in terms of rounds. Moreover, we show that this protocol enjoys stronger security guarantees while achieves considerably computational performance.

1 Introduction

Among numerous mechanisms for user authentication, passwords are definitely the most commonly used method of accessing modern computer networks and information systems [1]. Password authentication, usually integrated with key exchange simultaneously, has been proven to enjoy many advantages. For example, it can be easily operated and deployed as it only requires users to remember low-entropy passwords [2]; it is naturally compatible with various authentication means such as smart cards and biometric templates because passwords are convenient to obtain [3]. As a consequence, much attention has been paid on the fundamental security of passwords [4], the theoretical analyses and designing techniques of password authenticated key exchange (PAKE) protocols [5], as well as the standardization of password authentication schemes [6].

© Springer Nature Switzerland AG 2020
Z. Liu and M. Yung (Eds.): Inscrypt 2019, LNCS 12020, pp. 499–516, 2020.
https://doi.org/10.1007/978-3-030-42921-8_30

Anonymous Password Authenticated Key Exchange. Along with the increased awareness of security and privacy, there is an urgent need for strengthening the widely deployed PAKE protocols with additional anonymity property [7]. To be specific, it is desirable to authenticate legitimate users via low-entropy passwords while keeping their actual identities private to outside adversaries and even to the server.

To address this need, Viet et al. [8] proposed the first anonymous password authenticated key exchange (APAKE) protocol, through neatly blending an oblivious transfer (OT) protocol into a traditional two-party PAKE scheme. Since then, many research results have been put forward on the construction of more secure and more efficient APAKE protocols, either in password-only setting [9–12] or in the storage-extra setting [13–16]. Furthermore, the international organization for standardization (ISO) has developed and published the international standard ISO/IEC 20009-4 [17], which standardizes both the above two types of APAKE protocols.

For password-only APAKE protocols [8,11,12], a password is the only long-term secret that a user needs. They are very convenient from a user's point of view, but enjoy poor scalability since the computational complexity is in linear proportion to the number N of possible users [13]. As an innovative solution, APAKE protocols in the storage-extra setting were proposed by Yang et al. [13,14], and further improved by Zhang et al. [15] and Shin et al. [16], in which each user obtains a credential from the server, protects it by her password and stores the password-wrapped credential on some public storage (e.g., a public directory or an ipad). Then, in the authentication phase, the user recovers the credential via using her password and shows the possession of a valid credential to the server in an anonymous way, usually relying on some appropriate zero-knowledge proof systems. Notably, the computational cost needed by the server in the storage-extra setting is independent of the number of registered users, thus breaking the lower bound $\mathcal{O}(N)$ in the password-only setting.

However, we note that most of the existing APAKE protocols are developed in the random oracle model. Specifically, only few of APAKE protocols in the password-only setting [12,18] and, to the best of our knowledge, no APAKE protocols in the storage-extra setting have been proven secure in the standard model. Note that the random oracle model is only an ideal abstraction for the cryptographic hash functions, and there exists no random oracle definition that a public PPT algorithm can hope to satisfy [19]. Therefore, it is urgent to design storage-extra APAKE protocols in the standard model.

Storage-Extra APAKE in the Standard Model. Although we might be able to adapt classic storage-extra APAKE protocols to provide security in the standard model by adopting zero-knowledge proof schemes in the standard model (and without pairing-based assumptions). This approach will inherently increase the round number of the resulting APAKE protocols, since it is well known that one zero-knowledge proof typically requires at least three moves in this scenario.

The situation seems to be changed with the introduction of designated-verifier non-interactive zero-knowledge (DVNIZK) proof systems by Chaidos

et al. [20], which could provide proof of knowledge for a wide variety of algebraic statements related with some public words. One may wish to construct storage-extra APAKE protocols in standard model by starting from the efficient APAKE protocol [15] in the random oracle model based on algebraic message authentication codes (MACs) [21], and simply replacing the underlying zero-knowledge proof systems by some kind of DVNIZK proof schemes. Nevertheless, it is shown by Couteau et al.'s work [22] that constructing DVNIZK proof schemes compatible with algebraic MACs is not as simple as we first thought, because that the secret MAC keys have been re-used in the verification of the DVNIZK proofs. The solution developed by Couteau et al. consists of not only introducing additional random masks (i.e., $t_i \cdot G$ in Sect. 5.2 of [22]), but also requiring the underlying algebraic MAC scheme to satisfy a stronger (and cumbersome) notion of unforgeability called extended unforgeability.

Our Contributions. In this paper, we present a new storage-extra APAKE protocol in the standard model by further exploiting the construction of DVNIZK proof system compatible with algebraic MACs. Our main contributions can be summarized as follows:

- We give out a new construction of the oblivious DVNIZK proof scheme compatible with a new class of algebraic MAC schemes. Recall that Couteau et al. constructed oblivious DVNIZK proofs only for the algebraic MAC scheme abstracted from $MAC_{\texttt{GGM}}$, which is one of the two MAC schemes presented in [23]. We present an oblivious DVNIZK proof system for a class of algebraic MAC schemes generalized from $MAC_{\texttt{wBB}}$, which is based on the weak Boneh-Boyen signature [24].
- We avoid the requirement of the cumbersome security notion of extended unforgeability for the algebraic MAC scheme, which is quite hard to be verified and brings additional difficulties to the security proof. Note that the main reason for such a complicated definition is that the secret MAC key has been reused in the verification process of the DVNIZK proof. We overcome this obstacle by simulating verification oracles of the DVNIZK proof through the outputs of the MAC scheme instead of the MAC key.
- We present the first storage-extra APAKE protocol in the standard model without pairing-based assumptions, based on algebraic MACs and oblivious DVNIZK proofs. Beyond proving possession of a credential on a single value of identity, our construction can support credentials certifying many attributes at once and thus could handle more complex access policies such as expiration dates and access rights. Our APAKE protocol needs only 2 flows of messages during the authentication phase, which is very efficient in terms of round efficiency.

Organization. In Sect. 2, we briefly recall the necessary preliminaries. In Sect. 3, our construction of an oblivious DVNIZK proof system for a new class of algebraic MAC scheme is presented. Then, a new storage-extra APAKE protocol in the standard model is proposed in Sect. 4.

2 Preliminaries

In this section, we review the main cryptographic primitives needed in our construction. Throughout this paper, λ denotes the security parameter.

2.1 Algebraic Message Authentication Codes

A message authentication code (MAC) is defined by the following four PPT algorithms $M = (M.Setup, M.KeyGen, M.Mac, M.Verify)$ with an associated tag space $\mathcal{T}$, such that

- $M.Setup(1^\lambda)$ sets up the public parameters $\mathtt{pp}$ of the MAC, which will be implicitly (or explicitly) passed as an argument to the algorithms below.
- $M.KeyGen(\mathtt{pp})$ is a key generation algorithm which takes as input the public parameters $\mathtt{pp}$, outputs a secret key sk and public issuer parameters $\mathtt{ipp}$;
- $M.Mac(sk, m)$ is a MAC algorithm which takes as input the key sk and a message m, generates an authentication tag σ on the message;
- $M.Verify(sk, m, \sigma)$ is the verification algorithm which takes as input the key sk, a message m and a tag σ, outputs $b = 1$ when σ is a valid tag with respect to sk and m and outputs $b = 0$ otherwise.

We will need MAC schemes that are existentially unforgeable under chosen message and verification attacks (UF-CMVA).

Definition 1 (UF-CMVA Security). *A MAC scheme M is* UF-CMVA *secure if for any PPT adversary $\mathcal{A}$ which has access to the public issuer parameters* ipp *as well as the MAC and verification oracles, it holds that*

$$\Pr\left[\begin{array}{l} Q \leftarrow \emptyset, \mathtt{pp} \leftarrow M.Setup(1^\lambda), \\ (sk, \mathtt{ipp}) \leftarrow M.KeyGen(\mathtt{pp}), : \\ (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}_{sk}(\cdot)}(\mathtt{pp}, \mathtt{ipp}) \end{array} \begin{array}{l} M.Verify(sk, m, \sigma) = 1, \\ \wedge\, m \notin Q \end{array}\right] \leq negl(\lambda),$$

where the oracle $\mathcal{O}_{sk}(\cdot)$ treats the MAC and verification queries as follows: $\mathcal{O}.Mac(m)$ outputs $M.Mac(sk, m)$ and sets $Q \leftarrow Q \cup \{m\}$; $\mathcal{O}.Verify(m, \sigma)$ outputs $M.Verify(sk, m, \sigma)$.

For our purpose, we additionally require the MAC scheme to satisfy pseudorandomness property, which means that, as long as the MAC key is kept secret, no PPT adversary could distinguish a valid MAC tag from a random one. Based on the definitions of pseudorandom functions (PRFs) and weak pseudorandomness [15], we define the pseudorandomness property as follows.

Definition 2 (Pseudorandomness). *A MAC scheme M is said to satisfy pseudorandomness property, if for any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{l} \mathtt{pp} \leftarrow M.Setup(1^\lambda), \\ (sk, \mathtt{ipp}) \leftarrow M.KeyGen(\mathtt{pp}), \\ (m^*, \mathtt{st}) \leftarrow \mathcal{A}^{\mathcal{O}.Mac(\cdot)}(\mathtt{ipp}), \\ \sigma_0 = M.Mac(sk, m^*), \sigma_1 \leftarrow \mathcal{T}, \\ b \leftarrow \{0,1\}, b' = \mathcal{A}^{\mathcal{O}.Mac(\cdot)}(y_b, \mathtt{st}) \end{array} : b' = b\right] \leq \frac{1}{2} + negl(\lambda).$$

Algebraic MACs are a special kind of MACs that consist of only group operations instead of block ciphers or hash functions, thus easily suitable for efficient zero-knowledge proof of statements related to these MAC tags. In [23], Chase et al. proposed the first two algebraic MACs with efficient protocols for proof of knowledge, which are based on generic group model (GGM) and decisional Diffie-Hellman (DDH) assumption respectively. Since then, several improved algebraic MAC schemes have been put forward [15,25,26].

In this paper, we will use the algebraic MAC scheme $MAC_{\texttt{wBB}}$ proposed in [26], which is based on the weak Boneh-Boyen signature [24]. Denote by β a positive integer and $\mathbf{m} = (m_1, m_2, \cdots, m_\beta)$ a vector of message. Let $GGen(1^\lambda)$ be an efficient algorithm which generates a multiplicative group $\mathbb{G}$ of order p and a generator g of this group. The scheme consists of the following algorithms.

- $M.Setup(1^\lambda)$ outputs $\texttt{pp} = (\mathbb{G}, g, p) \leftarrow GGen(1^\lambda)$;
- $M.KeyGen(\texttt{pp})$ chooses randomly $\beta + 1$ elements $x_i \leftarrow \mathbb{Z}_p^*$, computes $X_i = g^{x_i}$ for every $i \in \{0, 1, \cdots, \beta\}$. Then, the secret key is $sk = (x_0, x_1, \cdots, x_\beta)$ and the public issuer parameter is $\texttt{ipp} = (X_0, X_1, \cdots, X_\beta)$;
- $M.Mac(sk, \mathbf{m})$ takes as input the key $sk = (x_0, x_1, \cdots, x_\beta)$ and a vector of messages $\mathbf{m} = (m_1, m_2, \cdots, m_\beta)$, computes $\sigma = g^{1/(x_0 + \sum_{j=1}^{\beta} x_j \cdot m_j)}$ and $\sigma_j = \sigma^{x_j}$ for $j = 1, 2, \cdots, \beta$[1], and sets the MAC tag as $\Sigma = (\sigma, \sigma_1, \cdots, \sigma_\beta)$;
- $M.Verify(sk, \mathbf{m}, (\sigma, \sigma_1, \cdots, \sigma_\beta))$ is the verification algorithm, which outputs $b = 1$ iff $\sigma = g^{1/(x_0 + \sum_{j=1}^{\beta} x_j \cdot m_j)}$ and $\sigma_j = \sigma^{x_j}$ for $j = 1, 2, \cdots, \beta$.

With respect to the security, it has been proven by Camenisch et al. [26] that this algebraic MAC scheme is UF-CMVA under the SCDHI assumption, which is a computational variation of the SDDHI assumption [27]. Moreover, we can easily prove that under the SDDHI assumption this algebraic MAC scheme satisfies the pseudorandomness property.

Theorem 1. *If the SDDHI and SCDHI assumptions hold in group $\mathbb{G}$, then the algebraic MAC scheme $MAC_{\texttt{wBB}}$ satisfies the pseudorandomness property.*

2.2 Additively Homomorphic Encryption Schemes

A public key encryption scheme is defined as a triple of PPT algorithms $\mathcal{E} = (\mathcal{E}.KeyGen, \mathcal{E}.Enc, \mathcal{E}.Dec)$ with an associated message space $\mathbb{Z}_N$ and a random space $\mathbb{Z}_R$, such that

- $\mathcal{E}.KeyGen(1^\lambda)$ takes as input the security parameter 1^λ and outputs a pair of encryption and decryption keys (ek, dk);
- $\mathcal{E}.Enc(ek, m; r)$ takes as input a public encryption key ek, a message $m \in \mathbb{Z}_N$ and a random value $r \in \mathbb{Z}_R$, and outputs a ciphertext c;
- $\mathcal{E}.Dec(dk, c)$ takes as input a decryption key dk and a ciphertext c, and outputs a message m or $\perp$.

[1] As pointed out by Camenisch et al. [26], the auxiliary information σ_j are not required for the MAC verification, but they are useful to improve the efficiency of credential presentation, and additionally remove the requirement of extended unforgeability.

The IND-CPA security for a public key encryption scheme is defined as follows.

Definition 3 (IND-CPA Security). *A public key encryption scheme $\mathcal{E}$ is* IND-CPA *secure, if for any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{l}(ek, dk) \leftarrow \mathcal{E}.KeyGen(1^\lambda),\\ (m_0, m_1, \mathtt{st}) \leftarrow \mathcal{A}(ek), b \leftarrow \{0,1\}, \qquad : b' = b\\ r \leftarrow \mathbb{Z}_R, c \leftarrow \mathcal{E}.Enc(ek, m_b; r), b' = \mathcal{A}(\mathtt{st}, c)\end{array}\right] \leq \frac{1}{2} + negl(\lambda).$$

In this paper, we focus on public key encryption schemes that are additively homomorphic for both the message and the random value. We say that an encryption scheme is *strongly additive*, if there exists an efficient operation $\oplus$ such that for any key pair $(ek, dk) \leftarrow \mathcal{E}.KeyGen(1^\lambda)$, any two ciphertexts $c_i = \mathcal{E}.Enc(ek, m_i; r_i)$ of messages $m_i \in \mathbb{Z}_N$ under the randomness $r_i \in \mathbb{Z}_R$ for $i \in \{1, 2\}$, it holds that $c_1 \oplus c_2 = \mathcal{E}.Enc(ek, m_1 + m_2 \mod N; r_1 + r_2 \mod R)$. For an integer $\rho \in \mathbb{Z}$, we denote by $\rho \odot c$ the ciphetext $\mathcal{E}.Enc(ek, \rho m \mod N; \rho r \mod R)$, which can be efficiently computed using the formula of the form $\mathcal{E}.Enc(ek, m; r) \oplus \cdots \oplus \mathcal{E}.Enc(ek, m; r)$. Moreover, given two ciphertexts c, c', we denote by $c \ominus c'$ the operation $c \oplus ((-1) \odot c')$.

A strongly additive encryption scheme $\mathcal{E}$ is further said to be *DVNIZK-friendly*, if $gcd(N, R) = 1$ and for any $(ek, dk) \leftarrow \mathcal{E}.KeyGen(1^\lambda)$, the value $\mathcal{E}.Enc(ek, m; 0)$ is efficiently decodable to get the plaintext $m \mod N$. For DVNIZK-friendly encryption scheme over groups $\mathbb{Z}_N$ of composite order of the form $N = pq$, we can resort to a slight variation [20,28] of the well-known Paillier encryption scheme [29]; For groups $\mathbb{Z}_N$ of prime order $N = p$, we could instantiate it with the Castagnos-Laguillaumie encryption scheme [30].

Note that the above encryption/decryption algorithms and scalar product could be extended to vectors in a natural way. For example, given vectors $\mathbf{m} = (m_1, m_2, \cdots, m_\beta)$ and $\mathbf{r} = (r_1, r_2, \cdots, r_\beta)$ of the length $\beta \geq 1$, we could view $\mathcal{E}.Enc(ek, \mathbf{m}; \mathbf{r})$ as the vector $(\mathcal{E}.Enc(ek, m_i; r_i))_{i=1}^{\beta}$. Given $\rho = (\rho_1, \rho_2, \cdots, \rho_\beta)$ and $c = \mathcal{E}.Enc(ek, m; r)$, we let $\rho \odot c$ denote the vector $(\mathcal{E}.Enc(ek, \rho_i m; \rho_i r))_{i=1}^{\beta}$.

2.3 Designated-Verifier Non-interactive Zero-Knowledge Proof

A designated-verifier non-interactive zero-knowledge (DVNIZK) proof system [20] for a family of languages $\{\mathfrak{L}_{\mathtt{crs}}\}$ consists of four algorithms $\Pi = (\Pi.Setup, \Pi.KeyGen, \Pi.Prove, \Pi.Verify)$. The setup algorithm $\Pi.Setup(1^\lambda)$ outputs a common reference string crs; The algorithm $\Pi.KeyGen(\mathtt{crs})$ outputs a public proving key pk and a secret verification key vk; The proving algorithm $\Pi.Prove(pk, x, w)$ takes as input the proving key pk, a word x and a witness w for the statement $x \in \mathfrak{L}_{\mathtt{crs}}$, generates a proof π; The verification algorithm $\Pi.Verify(pk, vk, x, \pi)$ outputs $b \in \{0, 1\}$ indicating either accept or reject.

However, when taking an algebraic MAC tag with respect to a user's identity as her credential, the original definition of DVNIZK proof system cannot be directly used to prove knowledge of the identity. The main difficulty is that the MAC verification cannot be carried out by the user while she does not know the

secret MAC key. To tackle this problem, Couteau et al. [22] introduced a new primitive called oblivious DVNIZK proof system, which can be used to prove knowledge of a witness w corresponding to a secret relation $R(sk, w, x) = 1$ with sk unknown to the prover.

In this paper, we will focus on secret relations $\{R_{\mathtt{crs}}(sk, \cdot, \cdot)\}$ and languages $\{\mathfrak{L}_{\mathtt{crs}}\}$ that are defined by algebraic MAC schemes. Given a MAC scheme M with secret MAC key sk, we will simply set sk as the secret relation key, and take a MAC message and tag pair as a witness and word pair, in the sense that $R_{\mathtt{crs}}(sk, w, x) = 1$ and $x \in \mathfrak{L}_{\mathtt{crs}}$ if and only if $M.Verify(sk, w, x) = 1$.

Definition 4 (Oblivious DVNIZK Proof [22]). *An oblivious DVNIZK proof system for a family of languages related with secret relations $\{R_{\mathtt{crs}}\}$ is defined by the following algorithms $\Pi = (\Pi.Setup, \Pi.RelSetup, \Pi.KeyGen, \Pi.Prove, \Pi.Verify)$, such that*

- *$\Pi.Setup(1^\lambda)$ takes as input the security parameter 1^λ, outputs a common reference string $\mathtt{crs}$ and a trapdoor $\mathtt{td}$;*
- *$\Pi.RelSetup(\mathtt{crs})$ generates a secret key sk for the secret relation, together with some public issuer parameters $\mathtt{ipp}$;*
- *$\Pi.KeyGen(\mathtt{crs})$ outputs a key pair (pk, vk), consisting of a public proving key pk and a secret verification key vk;*
- *$\Pi.Prove(\mathtt{crs}, \mathtt{ipp}, pk, (x_p, x_s), w)$ takes as input the parameters $\mathtt{crs}, \mathtt{ipp}$, the proving key pk, a word $x = (x_p, x_s) \in \mathfrak{L}_{\mathtt{crs}}$ consisting of a public subword x_p and a secret subword x_s, and a witness for the relation $R_{\mathtt{crs}}(sk, w, x) = 1$, then outputs a proof π;*
- *$\Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, vk, sk, x_p, \pi)$ is the verification algorithm which verifies whether a proof π is valid with respect to the relation $R_{\mathtt{crs}}(sk, w, x) = 1$. It outputs $b = 1$ if the proof is valid and $b = 0$ otherwise.*

We say that an oblivious DVNIZK proof system is secure, if it satisfies completeness, knowledge extractability (which is a strengthening of soundness) and oblivious zero-knowledge properties defined as follows.

Definition 5 (Completeness). *An oblivious DVNIZK proof system Π satisfies completeness property, if for all parameters $(\mathtt{crs}, \mathtt{td}) \leftarrow \Pi.Setup(1^\lambda)$, $(sk, \mathtt{ipp}) \leftarrow \Pi.RelSetup(\mathtt{crs})$, $(pk, vk) \leftarrow \Pi.KeyGen(\mathtt{crs})$, and every proof $\pi \leftarrow \Pi.Prove(\mathtt{crs}, \mathtt{ipp}, pk, (x_p, x_s), w)$, it holds that $\Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, vk, sk, x_p, \pi) = 1$.*

Definition 6 (Knowledge Extractability). *An oblivious DVNIZK proof system Π for secret relations $\{R_{\mathtt{crs}}\}$ defined by an algebraic MAC M is said to satisfy knowledge extractability property, if for every PPT adversary $\mathcal{A}$, there exists an efficient extracting algorithm Ext such that*

$$\Pr\left[\begin{array}{ll} (\mathtt{crs}, \mathtt{td}) \leftarrow \Pi.Setup(1^\lambda), & \\ (sk, \mathtt{ipp}) \leftarrow \Pi.RelSetup(\mathtt{crs}), & R_{\mathtt{crs}}(sk, w, (x_p, x_s)) = 0, \\ (pk, vk) \leftarrow \Pi.KeyGen(\mathtt{crs}), & : \Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, \\ (\pi, x_p) \leftarrow \mathcal{A}^{O_{M.Mac}, O_{\Pi.Ver}}(\mathtt{crs}, \mathtt{ipp}, pk) & \quad vk, sk, x_p, \pi) = 1 \\ (x_s, w) \leftarrow Ext(\mathtt{crs}, \mathtt{ipp}, pk, x_p, \pi, \mathtt{td}) & \end{array}\right] \leq negl(\lambda),$$

where the oracle $O_{M.Mac}(\cdot)$ *denotes* $M.Mac(sk,\cdot)$*, and the oracle* $O_{\Pi.Ver}(\cdot,\cdot)$ *denotes* $\Pi.Verify(\mathtt{crs},\mathtt{ipp},pk,vk,sk,\cdot,\cdot)$*. In addition, it is required that the oracle* $O_{\Pi.Ver}(\cdot,\cdot)$ *should be efficiently simulated, when the secret key* sk *is replaced by oracle access to* $M.Verify(sk,\cdot,\cdot)$*.*

Definition 7 (Oblivious Zero Knowledge). *An oblivious DVNIZK proof system* Π *for secret relations* $\{R_{\mathtt{crs}}\}$ *defined by an algebraic MAC* M *is said to satisfy oblivious zero knowledge property, if for every PPT adversary* $\mathcal{A}$*, there exists an efficient algorithm* Sim *such that*

$$\left|\Pr\left[\begin{array}{l}(\mathtt{crs},\mathtt{td})\leftarrow \Pi.Setup(1^\lambda),\\ (pk,vk)\leftarrow \Pi.KeyGen(\mathtt{crs}),\\ (x,w,sk,\mathtt{ipp},\mathtt{st})\leftarrow \mathcal{A}(\mathtt{crs},pk,vk)\\ \pi\leftarrow \Pi.Prove(\mathtt{crs},\mathtt{ipp},pk,x,w)\end{array}:\begin{array}{l}R_{\mathtt{crs}}(sk,w,x)=1,\\ \wedge\,\mathcal{A}(\mathtt{st},\pi)=1\end{array}\right]-\right.$$

$$\left.\Pr\left[\begin{array}{l}(\mathtt{crs},\mathtt{td})\leftarrow \Pi.Setup(1^\lambda),\\ (pk,vk)\leftarrow \Pi.KeyGen(\mathtt{crs}),\\ (x,w,sk,\mathtt{ipp},\mathtt{st})\leftarrow \mathcal{A}(\mathtt{crs},pk,vk)\\ \pi\leftarrow Sim(\mathtt{crs},\mathtt{ipp},pk,x_p,vk,sk)\end{array}:\begin{array}{l}R_{\mathtt{crs}}(sk,w,x)=1,\\ \wedge\,\mathcal{A}(\mathtt{st},\pi)=1\end{array}\right]\right|\leq negl(\lambda),$$

where $x=(x_p,x_s)$ *consists of a public subword* x_p *and a secret subword* x_s*.*

3 A New Construction of Oblivious DVNIZK Proof

In this section, we introduce a new construction of oblivious DVNIZK proof system for languages related to the algebraic MAC scheme $MAC_{\mathtt{wBB}}$ presented in Sect. 2.1, and prove the security properties of our construction.

3.1 The Construction of Oblivious DVNIZK Proof

We will take an algebraic MAC tag on a user's attributes as a credential for this user, which is treated in a similar way as in [22,23]. Nevertheless, the specific property of the algebraic MAC scheme $MAC_{\mathtt{wBB}}$ based on weak Bonel-Boyen signature (see definition in Sect. 2.1) will allow us to build a more efficient oblivious DVNIZK proof system than before. Recall that, when the algebraic MAC scheme $MAC_{\mathtt{wBB}}$ is considered, a MAC tag on a vector of attributes $\mathbf{m}=(m_1,m_2,\cdots,m_\beta)$ under the secret key $sk=(x_0,x_1,\cdots,x_\beta)$ is of the form $\Sigma=(\sigma,\sigma_1,\cdots,\sigma_\beta)$, where $\sigma=g^{1/\left(x_0+\sum_{j=1}^{\beta}x_j\cdot m_j\right)}$. It can be rewritten as $\sigma^{-\sum_{j=1}^{\beta}x_j\cdot m_j}\cdot g=\sigma^{x_0}$, where the first part $\sigma^{-\sum_{j=1}^{\beta}x_j\cdot m_j}\cdot g$ has exponents linear in both attributes $\mathbf{m}=(m_1,m_2,\cdots,m_\beta)$ and secret keys $(x_1,x_2,\cdots,x_\beta)$. This property would be preserved even after some re-randomize technique has been applied on the credential σ. For example, when it is re-randomized as $T=\sigma^a$ for a random a, it still holds that $T^{-\sum_{j=1}^{\beta}x_j\cdot m_j}\cdot g^a=T^{x_0}$.

Based on the above observation, we are now ready to present our construction of oblivious DVNIZK proof system for secret relations defined by the algebraic MAC scheme $MAC_{\mathtt{wBB}}$. Given the algebraic MAC scheme $MAC_{\mathtt{wBB}}$ denoted by $MAC_{\mathtt{wBB}} = (M.Setup, M.KeyGen, M.Mac, M.Verify)$ and a DVNIZK-friendly encryption scheme $\mathcal{E} = (\mathcal{E}.KeyGen, \mathcal{E}.Enc, \mathcal{E}.Dec)$ with message space $\mathbb{Z}_N$ of prime order $N = p$, the concrete steps of the oblivious DVNIZK proof system Π are as follows.

- $\Pi.Setup(1^\lambda)$ takes as input the security parameter 1^λ, computes $(ek, dk) \leftarrow \mathcal{E}.KeyGen(1^\lambda)$ and $\mathtt{pp} = (\mathbb{G}, g, p) \leftarrow M.Setup(1^\lambda)$, sets the common reference string as $\mathtt{crs} = (ek, \mathtt{pp})$ and the trapdoor as $\mathtt{td} = dk$. Without loss of generality, we assume that the public key ek also determines the message space $\mathbb{Z}_N$, the random source $\mathbb{Z}_R$ and a public bound B on R;
- $\Pi.RelSetup(\mathtt{crs})$ is essentially the key generation algorithm of the underlying MAC scheme $MAC_{\mathtt{wBB}}$. It chooses randomly $\beta+1$ elements $x_i \leftarrow \mathbb{Z}_p^*$ and computes $X_i = g^{x_i}$ for every $i \in \{0, 1, \cdots, \beta\}$. Then, the secret key is $sk = (x_0, x_1, \cdots, x_\beta)$ and the public issuer parameter is $\mathtt{ipp} = (X_0, X_1, \cdots, X_\beta)$;
- $\Pi.KeyGen(\mathtt{crs})$ chooses at random a value $e \leftarrow \mathbb{Z}_l$ where $l = 2^\lambda \cdot N \cdot B$, then sets the secret verification key as $vk = e$ and the public proving key as $pk = \mathcal{E}.Enc(ek, 0; e)$;
- $\Pi.Prove(\mathtt{crs}, \mathtt{ipp}, pk, (x_p, x_s), w)$ takes as input the parameters $\mathtt{crs}, \mathtt{ipp}$, the proving key pk, a credential $x = (x_p = \perp, x_s = \sigma)$ and a vector of attributes $w = (m_1, m_2, \cdots, m_\beta)$, selects a random value $a \leftarrow \mathbb{Z}_N$ and computes $T = \sigma^a$. It then chooses at random $a' \leftarrow \mathbb{Z}_N$, $m'_j \leftarrow \mathbb{Z}_N$ for $j = 1, 2, \cdots, \beta$, and computes $T' = \Pi_{j=1}^{\beta} {\sigma_j}^{-a \cdot m'_j} \cdot g^{a'}$. Next, it chooses a random vector $\mathbf{r} = (r_0, r_1, \cdots, r_\beta) \leftarrow \mathbb{Z}_R^{\beta+1}$, sets $\overline{\mathbf{m}} = (a, m_1, m_2, \cdots, m_\beta)$ and $\overline{\mathbf{m}}' = (a', m'_1, m'_2, \cdots, m'_\beta)$, and computes $\mathbf{X} = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{r})$ and $\mathbf{X}' = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}'; \mathbf{0}) \ominus (\mathbf{r} \odot pk)$. At last, the proving algorithm outputs a proof $\pi = (T, T', \mathbf{X}, \mathbf{X}')$.
- $\Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, vk, sk, x_p = \perp, \pi)$ verifies the proof π as follows. It first parses π as $\pi = (T, T', \mathbf{X}, \mathbf{X}')$, computes $\mathbf{X}' \oplus (e \odot \mathbf{X})$ and checks that the values in this vector are decodable, then decodes them to a vector $\mathbf{d} = (d_0, d_1, \cdots, d_\beta)$. Next, it checks that $T \neq 1$ and

$$(T^{x_0})^e \cdot T' = T^{-\sum_{j=1}^{\beta} x_j \cdot d_j} \cdot g^{d_0}. \tag{1}$$

  Finally, it outputs $b = 1$ if π can be parsed correctly, $\mathbf{X}' \oplus (e \odot \mathbf{X})$ is decodable and the above equation holds; otherwise, it outputs $b = 0$.

3.2 Security Analysis

In this section, we show that our oblivious DVNIZK proof system satisfies completeness, knowledge extractability and oblivious zero-knowledge properties.

Theorem 2. *If the underlying encryption scheme $\mathcal{E}$ is DVNIZK-friendly, then the oblivious DVNIZK proof system Π satisfies the completeness property.*

Proof. Firstly, if a proof $\pi = (T, T', \mathbf{X}, \mathbf{X}')$ is generated honestly, one can easily deduce that

$$\begin{aligned}\mathbf{X}' \oplus (e \odot \mathbf{X}) &= (\mathcal{E}.Enc(ek, \overline{\mathbf{m}}'; \mathbf{0}) \ominus (\mathbf{r} \odot pk)) \oplus (e \odot \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{r})) \\ &= \mathcal{E}.Enc(ek, \overline{\mathbf{m}}'; -e \cdot \mathbf{r}) \oplus \mathcal{E}.Enc(ek, e \cdot \overline{\mathbf{m}}; e \cdot \mathbf{r}) \\ &= \mathcal{E}.Enc(ek, \overline{\mathbf{m}}' + e \cdot \overline{\mathbf{m}}; \mathbf{0}),\end{aligned}$$

which is decodable according to the definition of DVNIZK-friendly encryption schemes. Moreover, we can obtain that $\mathbf{d} = \overline{\mathbf{m}}' + e \cdot \overline{\mathbf{m}} \mod N$, yielding that $d_0 = a' + e \cdot a \mod N$ and $d_j = m'_j + e \cdot m_j \mod N$ for all $j = 1, 2, \cdots, \beta$. Therefore, by combining with the equation $T^{-\sum_{j=1}^{\beta} m_j \cdot x_j} \cdot g^a = T^{x_0}$, we have

$$\begin{aligned}(T^{x_0})^e \cdot T' &= \left(T^{-\sum_{j=1}^{\beta} x_j \cdot m_j} \cdot g^a\right)^e \cdot \left(\Pi_{j=1}^{\beta} \sigma_j{}^{-a \cdot m'_j} \cdot g^{a'}\right) \\ &= \left(T^{-\sum_{j=1}^{\beta} x_j \cdot m_j} \cdot g^a\right)^e \cdot \left(T^{-\sum_{j=1}^{\beta} x_j \cdot m'_j} \cdot g^{a'}\right) \\ &= T^{-\sum_{j=1}^{\beta} x_j \cdot (m'_j + e \cdot m_j)} \cdot g^{a' + e \cdot a} \\ &= T^{-\sum_{j=1}^{\beta} x_j \cdot d_j} \cdot g^{d_0}.\end{aligned}$$

Theorem 3. *If the underlying encryption scheme $\mathcal{E}$ is DVNIZK-friendly, and the algebraic MAC scheme $MAC_{\mathtt{wBB}}$ is* `UF-CMVA` *secure, then the oblivious DVNIZK proof system Π satisfies the knowledge extractability property.*

Proof. Our proof starts with the construction of an extracting algorithm Ext. Given a proof $\pi = (T, T', \{T'_j\}_{j=1}^{\beta}, \mathbf{X}, \mathbf{X}')$ and the trapdoor $\mathtt{td} = dk$, the algorithm Ext computes $\overline{\mathbf{m}} = (a, m_1, m_2, \cdots, m_\beta) = \mathcal{E}.Dec(dk, \mathbf{X})$, sets $\sigma = T^{1/a}$ and then outputs $(x_s = \sigma, w = (m_1, m_2, \cdots, m_\beta))$.

We next turn to estimate the probability of the event $R_{\mathtt{crs}}(sk, w, (x_p, x_s)) = 0 \ \wedge \ \Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, vk, sk, \ x_p, \pi) = 1$. Recall that we only focus on secret relations that are defined by algebraic MAC schemes in the sense that $R(sk, w, x) = 1 \Leftrightarrow M.Verify(sk, w, x) = 1$. It is then sufficient to show that the oracle $O_{\Pi.Ver}(\cdot, \cdot) = \Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, vk, sk, \cdot, \cdot)$ can be efficiently simulated, with sk replaced by oracle access to $M.Verify(sk, \cdot, \cdot)$.

We denote by $O_{SimVer}(\cdot, \cdot) = SimVerify(\mathtt{crs}, \mathtt{ipp}, pk, vk, dk, \cdot, \cdot)$ the simulated verification algorithm with access to the oracle $M.Verify(sk, \cdot, \cdot)$, and proceed as follows. Assuming that $(\mathtt{crs}, \mathtt{td} = dk), (sk, \mathtt{ipp}), (pk, vk)$ are generated as before and $(\mathtt{crs}, \mathtt{ipp}, pk, vk, dk)$ are provided to $SimVerify$ as input. Then, for each query $(x_p = \perp, \pi)$ asked by the adversary $\mathcal{A}$, we can decrypt the ciphertexts $\mathbf{X}, \mathbf{X}'$ to vectors $\overline{\mathbf{m}} = (a, m_1, m_2, \cdots, m_\beta) \leftarrow \mathcal{E}.Dec(dk, \mathbf{X})$, $\overline{\mathbf{m}}' = (a', m'_1, m'_2, \cdots, m'_\beta) \leftarrow \mathcal{E}.Dec(dk, \mathbf{X}')$. Finally, we compute $\sigma = T^{1/a}$, and check that all the following equations are true:

$$-e \odot (\mathbf{X} \ominus \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{0})) = \mathbf{X}' \ominus \mathcal{E}.Enc(ek, \overline{\mathbf{m}}'; \mathbf{0}), \tag{2}$$

$$M.Verify(sk, (m_1, m_2, \cdots, m_\beta), \sigma) = 1, \tag{3}$$

$$T' = T^{-\sum_{j=1}^{\beta} x_j \cdot m'_j} \cdot g^{a'}. \tag{4}$$

If all the checks succeeded, $SimVerify$ outputs 1; otherwise, it outputs 0.

Here we remark that, under the conditions $\sigma = T^{1/a}$ and $M.Verify(sk, (m_1, m_2, \cdots, m_\beta), \sigma) = 1$, the check Eq. (4) could in fact be calculated without the knowledge of the secret key $sk = (x_0, x_1, \cdots, x_\beta)$. Alternatively, we can ask to the MAC oracle to get $(\sigma, \sigma_1, \cdots, \sigma_\beta) = M.Mac(sk, (m_1, m_2, \cdots, m_\beta))$ and then check that

$$T' = \Pi_{j=1}^{\beta} {\sigma_j}^{-a \cdot m'_j} \cdot g^{a'}. \tag{5}$$

This property is very attractive for the context of DVNIZK proof, since it enables us to avoid resorting to a stronger notion of unforgeability (called extended unforgeability [22]) for the underlying algebraic MAC schemes.

In the following, we will prove that the simulated oracle $O_{SimVer}(\cdot, \cdot)$ is indistinguishable from the real oracle $O_{\Pi.Ver}(\cdot, \cdot)$. First, we show that, given a query π, if $O_{SimVer}(\perp, \pi) = 1$, then $O_{\Pi.Ver}(\perp, \pi) = 1$. Recall that $\overline{\mathbf{m}} \leftarrow \mathcal{E}.Dec(dk, \mathbf{X})$ and $\overline{\mathbf{m}}' \leftarrow \mathcal{E}.Dec(dk, \mathbf{X}')$, it follows immediately that the Eq. (2) implies $\mathbf{X}, \mathbf{X}'$ are of the form $\mathbf{X} = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{r})$ and $\mathbf{X}' = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}'; -e \cdot \mathbf{r})$ for some random vector $\mathbf{r}$. Henceforth, the vector $\mathbf{X}' \oplus (e \odot \mathbf{X})$ is decodable, and the decoded vector is $\mathbf{d} = \overline{\mathbf{m}}' + e \cdot \overline{\mathbf{m}}$. On the other hand, if the Eq. (3) is satisfied, it yields that $\sigma = g^{1/\left(x_0 + \sum_{j=1}^{\beta} x_j \cdot m_j\right)}$, which in turn implies that $T^{-\sum_{j=1}^{\beta} x_j \cdot m_j} \cdot g^a = T^{x_0}$. Combining these facts with Eq. (4), we can easily get the equation $(T^{x_0})^e \cdot (\Pi_{j=1}^{\beta} (T'_j)^{-x_j} \cdot T') = T^{-\sum_{j=1}^{\beta} x_j \cdot d_j} \cdot g^{d_0}$. This indicates that $O_{\Pi.Ver}(\perp, \pi) = 1$.

Next, we prove that, if $O_{\Pi.Ver}(\perp, \pi) = 1$, then $O_{SimVer}(\perp, \pi) = 1$. Assume that $\mathbf{X} = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{r})$ and $\mathbf{X}' = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}'; \mathbf{r}')$ for some random vectors $\mathbf{r}$ and $\mathbf{r}'$. We would easily deduce from the fact $\mathbf{X}' \oplus (e \odot \mathbf{X})$ is decodable that $\mathbf{r}' = -e \cdot \mathbf{r}$, which thus yields that the Eq. (2) is satisfied. Furthermore, to obtain a contradiction, we now suppose that Eq. (1) holds, while the Eqs. (3) or (4) is rejected. Note that the Eq. (1) can be rewritten as

$$\left(T^{x_0} / T^{-\sum_{j=1}^{\beta} x_j \cdot m_j} \cdot g^a\right)^e = T^{-\sum_{j=1}^{\beta} x_j \cdot m'_j} \cdot g^{a'} / T'. \tag{6}$$

If the Eq. (3) does not hold, we get $T^{x_0} / T^{-\sum_{j=1}^{\beta} x_j \cdot m_j} \cdot g^a \neq 1$; if (4) does not hold, we have $T^{-\sum_{j=1}^{\beta} x_j \cdot m'_j} \cdot g^{a'} / T' \neq 1$. Since e is randomly chosen from some sufficiently large space, it holds with overwhelming probability that $e \neq 0 \mod N$. Hence, we can conclude that $T^{x_0} / T^{-\sum_{j=1}^{\beta} x_j \cdot m_j} \cdot g^a \neq 1$ and $T^{-\sum_{j=1}^{\beta} x_j \cdot m'_j} \cdot g^{a'} / T' \neq 1$ will happen simultaneously. Using a similar technique as in [20], we can then get from Eq. (6) the value $e \mod N$, which is supposed to be statistically hidden.

Now, since the simulated oracle $O_{SimVer}(\cdot, \cdot)$ is indistinguishable from the real oracle $O_{\Pi.Ver}(\cdot, \cdot)$, and the valid of secret relation has essentially been checked through Eq. (3), we conclude that the event $R(sk, w, (x_p, x_s)) = 0 \ \wedge \ \Pi.Verify(\mathtt{crs}, \mathtt{ipp}, pk, vk, sk, x_p, \pi) = 1$ happens with only negligible probability. This completes the proof.

Theorem 4. *If the underlying encryption scheme $\mathcal{E}$ is* IND-CPA *secure, then the oblivious DVNIZK proof system Π is obliviously zero-knowledge.*

Proof. The proof will be divided into two steps: constructing a simulator and proving the indistinguishability. We first construct a simulator $Sim(\mathtt{crs}, \mathtt{ipp}, pk, x_p, vk, sk)$ which producing simulated zero-knowledge proof π as follows. It first selects randomly $T \leftarrow \mathbb{G}$ and $\mathbf{d} = (d_0, d_1, \cdots, d_\beta) \leftarrow \mathbb{Z}_N^{\beta+1}$, and computes

$$T' = T^{-\sum_{j=1}^{\beta} x_j \cdot d_j} \cdot g^{d_0} / (T^{x_0})^e. \tag{7}$$

The simulator then chooses at random $\overline{\mathbf{m}} = (a, m_1, m_2, \cdots, m_\beta) \leftarrow \mathbb{Z}_N^{\beta+1}$, $\overline{\mathbf{m}}' = (a', m_1', m_2', \cdots, m_\beta') \leftarrow \mathbb{Z}_N^{\beta+1}$ and $\mathbf{r} = (r_0, r_1, \cdots, r_\beta) \leftarrow \mathbb{Z}_R^{\beta+1}$, and computes

$$\mathbf{X} = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{r}), \tag{8}$$

$$\mathbf{X}' = \mathcal{E}.Enc(ek, \mathbf{d} - e \cdot \overline{\mathbf{m}}; -e \cdot \mathbf{r}). \tag{9}$$

We now show that, given an adversary $\mathcal{A}$ against the indistinguishability of $\Pi.Prove$ and Sim, we can construct an adversary $\mathcal{A}_S$ against the IND-CPA security of S. The IND-CPA adversary $\mathcal{A}_S$ first obtains $(x = (x_p = \perp, x_s = \sigma), w = (m_1, m_2, \cdots, m_\beta))$ from the adversary $\mathcal{A}$, then it chooses randomly $a \leftarrow \mathbb{Z}_N$, sets $\overline{\mathbf{m}} = (a, m_1, m_2, \cdots, m_\beta)$ and $T = \sigma^a$, picks at random $\widetilde{\mathbf{m}} = (\tilde{a}, \tilde{m}_1, \tilde{m}_2, \cdots, \tilde{m}_\beta) \leftarrow \mathbb{Z}_N^{\beta+1}$, and sends the $(\overline{\mathbf{m}}, \widetilde{\mathbf{m}})$ to the IND-CPA challenger to get back a challenging ciphertext $\mathbf{X}$. Next, it selects randomly $\mathbf{d} = (d_0, d_1, \cdots, d_\beta) \leftarrow \mathbb{Z}_N^{\beta+1}$, computes $\mathbf{X}' = \mathcal{E}.Enc(ek, \mathbf{d}; \mathbf{0}) \ominus (e \cdot \mathbf{X})$, and sets T' as in Eq. (7). Finally, the adversary $\mathcal{A}_S$ sends $\pi^* = (T, T', \mathbf{X}, \mathbf{X}')$ to the adversary $\mathcal{A}$, and takes the bit $b \in \{0, 1\}$ outputted by $\mathcal{A}$ as its own output.

While the relation $R_{\mathtt{crs}}(sk, w, x) = 1$ holds, it is easy to check that, if the challenging ciphertext $\mathbf{X} = \mathcal{E}.Enc(ek, \overline{\mathbf{m}}; \mathbf{r})$, then the proof π^* is distributed identical to a proof in the real game; if $\mathbf{X} = \mathcal{E}.Enc(ek, \widetilde{\mathbf{m}}; \mathbf{r})$, then the proof π^* is distributed exactly as that is produced by the simulator. Therefore, if the adversary $\mathcal{A}$ has non-negligible probability in distinguishing $\Pi.Prove$ and Sim, then the adversary $\mathcal{A}_S$ will win the IND-CPA game with non-negligible probability.

4 A New Storage-Extra APAKE Protocol

In this section, we first describe the construction of our new storage-extra APAKE protocol. Then, the design rationale and detailed comparisons of our protocols, in terms of both efficiency and security, are presented.

4.1 The Construction of the APAKE Protocol

Assume that $MAC_{\mathtt{wBB}} = (M.Setup, M.KeyGen, M.Mac, M.Verify)$ is the algebraic MAC scheme presented in Sect. 2.1, $\mathcal{E} = (\mathcal{E}.KeyGen, \mathcal{E}.Enc, \mathcal{E}.Dec)$ is a DVNIZK-friendly homomorphic encryption scheme as defined in Sect. 2.2, and $\Pi = (\Pi.Setup, \Pi.RelSetup, \Pi.KeyGen, \Pi.Prove, \Pi.Verify)$ is the oblivious DVNIZK proof scheme introduced in Sect. 3.1. We also use a traditional MAC scheme $\mathtt{M} = (\mathtt{KeyGen}, \mathtt{Mac}, \mathtt{Verify})$ and a traditional signature scheme $S = (S.KeyGen, S.Sign, S.Verify)$.

For each user $U \in \mathbf{U}$, denote by $\mathbf{m} = (m_1, m_2, \cdots, m_\beta)$ the vector of attributes and by pw the password held by this user. The construction of the APAKE protocol consists of the following steps.

Setup. In the setup phase, we first run $\Pi.Setup(1^\lambda)$ to obtain $\mathtt{crs} = (ek, \mathtt{pp}) = (ek, (\mathbb{G}, g, p))$ and the trapdoor $\mathtt{td} = dk$, run $\Pi.KeyGen(\mathtt{crs})$ to generate a secret relation key $sk = (x_0, x_1, \cdots, x_\beta)$ and $\mathtt{ipp} = (X_0, X_1, \cdots, X_\beta)$, run $S.KeyGen(1^\lambda)$ to get a signing key SK and the related signature verification key VK. Then, we select a random element $h \leftarrow \mathbb{G}$, set the common reference string of the APAKE protocol as $(\mathtt{crs}, \mathtt{ipp}, VK, h)$, and provide to the server with the secret keys (sk, SK).

Registration. In this phase, each user registers to the server to prepare for subsequent anonymous authentication. The registration phase is assumed to be executed over secure channels. To begin with, each user sends her attributes[2] $\mathbf{m} = (m_1, m_2, \cdots, m_\beta)$ to the server. Upon receiving this registration request, the server generates a MAC tag $\Sigma = (\sigma, \sigma_1, \cdots, \sigma_\beta) \leftarrow M.Mac(sk, \mathbf{m})$ and sends it to the user as its credential[3]. When the credential Σ is received, the client encrypts it with her password pw to obtain a password-wrapped credential $[\Sigma]_{pw}$, and puts it on some (publicly) extra-storage.

Authentication. To login the server, a user interacts with the server as follows.

1. At the beginning, the server runs $\Pi.KeyGen(\mathtt{crs})$ to generate a proof verification key $vk = e$ and the corresponding proving key $pk = \mathcal{E}.Enc(ek, 0; e)$, picks at random $\gamma \leftarrow \mathbb{Z}_p$ and computes $Y = h^\gamma$, $\sigma_S = S.Sign(SK, (pk, Y))$. Then, the server sends to the client the message (pk, Y, σ_S).
2. Upon receiving the message (pk, Y, σ_S) from the server, the client first checks the validity of the signature σ_S. Next, she fetches back the password-wrapped credential $[\Sigma]_{pw}$ and decrypts it with her password pw to recover the credential $\Sigma = (\sigma, \sigma_1, \cdots, \sigma_\beta)$. Then, the user generates a DVNIZK proof $\pi \leftarrow \Pi.Prove(\mathtt{crs}, \mathtt{ipp}, pk, (x_p, x_s), w)$, where $x = (x_p = \perp, x_s = \sigma)$ and $w = (m_1, m_2, \cdots, m_\beta)$. The user also chooses randomly $\xi \leftarrow \mathbb{Z}_p$, computes $X = h^\xi$, $tk_U^{(1)} = Y^\xi$,$tk_U^{(2)} = \Pi_{j=1}^{\beta} \sigma_j^{-a \cdot m_j} \cdot g^a$ and $\sigma_U = \mathtt{Mac}(tk_U^{(2)}, \sigma_S || X || \pi)$. Finally, the user sends (X, π, σ_U) to the server, and computes the session key as $K_U = tk_U^{(1)} \cdot tk_U^{(2)}$.
3. When the server receives (X, π, σ_U), it ensures that $T \neq 1$, computes $tk_S^{(1)} = X^\gamma, tk_S^{(2)} = T^{x_0}$, checks that $\sigma_U = \mathtt{Mac}(tk_S^{(2)}, \sigma_S || X || \pi)$, verifies the DVNIZK

[2] Beyond a single value of identity, here we consider a vector of attributes, which could handle more complex access policies such as expiration dates and access rights.

[3] Together with the credential Σ, the server perhaps, if needed, sends a zero-knowledge proof proving that this MAC tag is honestly generated. The ZK proof could be either a NIZK proof secure in the random oracle model, or a DVNIZK proof secure in the standard model where the proving key is sent to the server along with the attributes.

proof π, and aborts if any of these checks is failed. If all checks are valid, the server computes the session key as $K_S = tk_S^{(1)} \cdot tk_S^{(2)}$.

4.2 Design Rationale

The core of our construction is a DVNIZK proof π to prove that the algebraic MAC tag σ held by the user is valid, without compromising the privacy of this credential and the user's attributes. The privacy protection property is guaranteed by the zero-knowledge property of the underlying DVNIZK proof scheme; and the soundness property of the DVNIZK proof system ensures that, the user in communication is a legitimate member with a valid algebraic MAC tag as her credential.

Based on these observations, we could even obtain a one-pass variant of the above protocol for anonymous entity authentication, through sending only one flow of message consisting of the DVNIZK proof π to the server. The resulting protocol guarantees privacy-preserving non-interactive authentication, as expected by [20]. However, as a one-pass protocol, it is inherently open to replay attacks [31]. Although it is well-known that replay attacks can be prevented by maintaining synchronized state (via counters or timestamps) between the sender and receiver, we emphasize that synchronized timestamps are actually quite tedious in practical applications.

In order to prevent replay attacks and to establish a secure session key for subsequent use, we alternatively choose to have the server send an additional message (pk, Y, σ_S) to the client. In this message, the server generates and sends a fresh proving key pk for every session, which guarantees that the DVNIZK proof π is newly generated as well. Moreover, with the extra flow of message, we can embed in this protocol of a Diffie-Hellman tuple (X, Y), which offers forward security for both participants.

4.3 Comparisons with Existing Storage-Extra APAKE Protocols

In the following, we compare our storage-extra APAKE protocol with similar anonymous authentication protocols, in terms of both security and efficiency.

Security Comparisons. First, recall that our main purpose is to design a storage-extra APAKE protocol secure in the standard model, instead of in the random oracle model. As indicated in Table 1, our storage-extra APAKE protocol is the only one with proven security in the standard model, while all the existing protocols [14–16] are analyzed in the random oracle model. However, the random oracle model is arguably "unnatural" and differs from real-world constructions significantly [19]. We thus have reasons to believe that an APAKE protocol with provable security in the standard model would provide a stronger guarantee of security than those only proven secure in the random oracle model. In addition, our protocol not only satisfies the same mutual authentication property as the

existed protocols, but also permits more flexibility in terms of identity type, as our protocol allows users to prove a vector of personal attributes rather than a single value of identity.

Efficiency Comparisons. With respect to efficiency, we compare our protocol with the existing storage-extra APAKE protocols in terms of rounds, communication and computational cost. The details are illustrated in Table 2. We stress that our protocol requires only two flows of messages during the authentication phase, which is very efficient for a protocol with explicitly mutual authentication. Although our protocol is less efficient, which is similar to those protocols with provable security in the standard model, than its counterpart proven secure in the random oracle, it is still considerably efficient. In particular, when the length of attributes is set to $\beta = 1$, we get a protocol that is even more efficient than Yang et al.'s scheme [14], which is right the storage-extra APAKE protocol contained in the standard ISO/IEC 20009-4 [17].

Table 1. Security comparisons among storage-extra APAKE protocols

Protocols	Model	Mutu-Auth	Identity-Type
Yang et al.'s [14]	Random oracle	Yes	Single value
Zhang et al.'s [15]	Random oracle	Yes	Single value
Shin et al.'s [16]	Random oracle	Yes	Single value
Our protocol	Standard model	Yes	Attribute vector

Legend: Mutu-Auth represents explicitly mutual authentication, Identity-Type denotes the type of user's identity which the protocol supports.

Table 2. Efficiency comparisons among storage-extra APAKE protocols

Protocols	Rounds	Comm.	Computational cost	
			User side	Server side
Yang et al.'s [14]	3	$7\|\mathbb{G}_1\| + \|\mathbb{G}_T\| + 6\|p\|$	$9E_{\mathbb{G}_1} + 1E^2_{\mathbb{G}_1} + 1E^5_{\mathbb{G}_T} + 2P$	$E_{\mathbb{G}_1} + 3E^2_{\mathbb{G}_1} + 1E^6_{\mathbb{G}_T} + 4P$
Zhang et al.'s [15]	2	$4\|\mathbb{G}\| + 4\|p\|$	$3E_{\mathbb{G}} + 2E^2_{\mathbb{G}}$	$3E_{\mathbb{G}} + 1E^2_{\mathbb{G}}$
Shin et al.'s [16]	3	$3\|\mathbb{G}\| + 2\|\mathcal{H}\|$	$5E_{\mathbb{G}}$	$3E_{\mathbb{G}}$
Our protocol	2	$10\|\mathbb{G}\| + 1\|\mathcal{T}\|$	$3E_{\mathbb{G}} + 2E^{\beta+1}_{\mathbb{G}} + 2\beta Enc$	$(\beta+3)E_{\mathbb{G}} + 2E^2_{\mathbb{G}}$

Legend: $|\mathbb{G}_1|$ denotes the bit size of an element from the group $\mathbb{G}_1$, $|p|$ represents the size of an element from $\mathbb{Z}_p$, $|\mathcal{H}|$ and $|\mathcal{T}|$ denote the size of an output of a hash function and a MAC scheme, respectively; $E_{\mathbb{G}}$ represents 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 denotes a homomorphic encryption operation.

4.4 Security Analysis of the APAKE Protocol

By utilizing a security model for storage-extra APAKE protocol formalized by Zhang et al. in [15], we could prove that the APAKE protocol presented above guarantees AKE security of session keys and achieves user anonymity with respect to the honest-but-curious server. However, the detailed security model and rigorous proofs are omitted here due to the page limit. We refer the reader to our full paper for more details.

5 Conclusions

In this paper, we first give out a new construction of the oblivious DVNIZK proof system compatible with a new class of algebraic MAC schemes, which avoids the requirement of the cumbersome security notion called extended unforgeability. Then, we present a new APAKE protocol in the standard model by combing the technique of algebraic MAC with oblivious designated-verifier non-interactive zero-knowledge (DVNIZK) proof. Comparisons show that our protocol enjoys stronger security guarantees as well as achieves considerably communication and computation performance.

Acknowledgments. Qihui Zhang and Wenfen Liu are supported by the National Nature Science Foundation of China (Grant Nos. 61862011, 61872449), and Guangxi Natural Science Foundation (Grant No. 2018GXNSFAA138116) and the Guangxi Key Laboratory of Cryptography and Information Security (Grant No. GCIS201704). Kang Yang is supported by the National Key Research and Development Program of China (Grant No. 2018YFB0804105), and the National Natural Science Foundation of China (Grant Nos. 61932019, 61802021).

References

1. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: an underestimated threat. In: ACM CCS 2016, pp. 1242–1254. ACM (2016)
2. Bellovin, S., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72–84 (1992)
3. Jiang, Q., Qian, Y., Ma, J., Ma, X., Cheng, Q., Wei, F.: User centric three-factor authentication protocol for cloud-assisted wearable devices. Int. J. Commun. Syst. **32**(6), e3900 (2019)
4. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Trans. Inf. Forensics Secur. **12**(11), 2776–2791 (2017)
5. Abdalla, M.: Password-based authenticated key exchange: an overview. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) ProvSec 2014. LNCS, vol. 8782, pp. 1–9. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12475-9_1
6. Schmidt, J.: Requirements for password-authenticated key agreement (PAKE) schemes. RFC 8125 (2017)
7. Lindell, Y.: Anonymous authentication. J. Priv. Confid. **2**(2), 35–63 (2007)

8. Viet, D.Q., Yamamura, A., Tanaka, H.: Anonymous password-based authenticated key exchange. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 244–257. Springer, Heidelberg (2005). https://doi.org/10.1007/11596219_20
9. Shin, S.H., Kobara, K., Imai, H.: A secure threshold anonymous password-authenticated key exchange protocol. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 444–458. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75651-4_30
10. Shin, S.H., Kobara, K., Imai, H.: Very-efficient anonymous password-authenticated key exchange and its extensions. In: Bras-Amorós, M., Høholdt, T. (eds.) AAECC 2009. LNCS, vol. 5527, pp. 149–158. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02181-7_16
11. Yang, J., Zhang, Z.: A new anonymous password-based authenticated key exchange protocol. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 200–212. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89754-5_16
12. Zhang, Q., Chaudhary, P., Kumari, S., Kong, Z., Liu, W.: Verifier-based anonymous password-authenticated key exchange protocol in the standard model. Math. Biosci. Eng. **16**(5), 3623–3640 (2019)
13. Yang, Y., Zhou, J., Weng, J., Bao, F.: A new approach for anonymous password authentication. In: the 25th Annual Computer Security Applications Conference, pp. 199–208, December 2009
14. Yang, Y., Zhou, J., Wong, J.W., Bao, F.: Towards practical anonymous password authentication. In: the 26th Annual Computer Security Applications Conference, pp. 59–68. ACM (2010)
15. Zhang, Z., Yang, K., Hu, X., Wang, Y.: Practical anonymous password authentication and TLS with anonymous client authentication. In: ACM CCS 2016, pp. 1179–1191. ACM (2016)
16. Shin, S., Kobara, K.: Simple anonymous password-based authenticated key exchange (SAPAKE), reconsidered. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **100**(2), 639–652 (2017)
17. ISO/IEC 20009-4: Information technology - security techniques - anonymous entity authentication - part 4: Mechanisms based on weak secrets. Standard (2019). https://www.iso.org/standard/64288.html
18. Hu, X., Zhang, J., Zhang, Z., Liu, F.: Anonymous password authenticated key exchange protocol in the standard model. Wirel. Pers. Commun. **96**(1), 1451–1474 (2017)
19. Leurent, G., Nguyen, P.Q.: How risky is the random-oracle model? In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 445–464. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_26
20. Chaidos, P., Couteau, G.: Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 193–221. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_7
21. Dodis, Y., Kiltz, E., Pietrzak, K., Wichs, D.: Message authentication, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 355–374. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_22
22. Couteau, G., Reichle, M.: Non-interactive keyed-verification anonymous credentials. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 66–96. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_3

23. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: ACM CCS 2014, pp. 1205–1216. ACM (2014)
24. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008)
25. Barki, A., Brunet, S., Desmoulins, N., Traoré, J.: Improved algebraic MACs and practical keyed-verification anonymous credentials. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 360–380. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_20
26. Camenisch, J., Drijvers, M., Dzurenda, P., Hajny, J.: Fast keyed-verification anonymous credentials on standard smart cards. In: Dhillon, G., Karlsson, F., Hedström, K., Zúquete, A. (eds.) SEC 2019. IAICT, vol. 562, pp. 286–298. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22312-0_20
27. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: ACM CCS 2006, pp. 201–210. ACM (2006)
28. Damgård, I., Jurik, M., Nielsen, J.B.: A generalization of Paillier's public-key system with applications to electronic voting. Int. J. Inf. Secur. **9**(6), 371–385 (2010)
29. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
30. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_26
31. Halevi, S., Krawczyk, H.: One-pass HMQV and asymmetric key-wrapping. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 317–334. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_20