Password-Based Authenticated Key Exchange without Centralized Trusted Setup

Kazuki Yoneyama

NTT Secure Platform Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan
`yoneyama.kazuki@lab.ntt.co.jp`

Abstract. Almost all existing password-based authenticated key exchange (PAKE) schemes achieve concurrent security in the standard model by relying on the common reference string (CRS) model. A drawback of the CRS model is to require a centralized trusted authority in the setup phase; thus, passwords of parties may be revealed if the authority ill-uses trapdoor information of the CRS. There are a few secure PAKE schemes in the plain model, but, these are not achievable in a constant round (i.e., containing a linear number of rounds). In this paper, we discuss how to relax the setup assumption for (constant round) PAKE schemes. We focus on the multi-string (MS) model that allows a number of authorities (including malicious one) to provide some reference strings independently. The MS model is a more relaxed setup assumption than the CRS model because we do not trust any single authority (i.e., just assuming that a majority of authorities honestly generate their reference strings). Though the MS model is slightly restrictive than the plain model, it is very reasonable assumption because it is very easy to implement. We construct a (concurrently secure) three-move PAKE scheme in the MS model (justly without random oracles) based on the Groce-Katz PAKE scheme. The main ingredient of our scheme is the multi-string simulation-extractable non-interactive zero-knowledge proof that provides both the simulation-extractability and the extraction zero-knowledge property even if minority authorities are malicious. This work can be seen as a milestone toward constant round PAKE schemes in the plain model.

Keywords: authenticated key exchange, password, multi-string model, concurrent security.

1 Introduction

Password-based authenticated key exchange (PAKE) is one of most attractive cryptographic primitives because authentication with short PINs or human memorable passwords is getting popular in web-based or cloud services. PAKE provides both the authentication property by passwords and the secrecy of the session key generation (often used to establish a secure channel) simultaneously. We consider the standard two-party setting: two parties share a common short password in advance and can generate a common long session key over an insecure channel like the Internet.

The first PAKE scheme was proposed by Bellovin and Merritt [1]. However, the security of their scheme is not proved formally. To construct a provably secure PAKE scheme is not so easy because a password has only low entropy. Since a password dictionary is small, an adversary can attempt *off-line dictionary attacks* that the adversary guesses the correct password and *locally* tests guessed passwords with transcripts repeatedly. Also, the adversary can attempt *on-line dictionary attacks* that the adversary guesses the correct password and tries to impersonate some honest party. Though on-line dictionary attacks cannot be prevented essentially, resistance to off-line dictionary attacks must be guaranteed. Thus, the required security of PAKE is that the advantage of the adversary is bounded by $Q/|\mathcal{D}|$, where Q is the number of impersonations that the adversary attempts and $|\mathcal{D}|$ is the size of a password dictionary $\mathcal{D}$.

First provably secure PAKE schemes [2,3,4] rely on the random oracle (RO) model or the ideal cipher (IC) model. Then, a secure PAKE scheme without ideal primitives (i.e., RO and IC) is proposed by Katz et al. [5] by adopting the *common reference string (CRS) model*. The CRS model assumes that a reference string (e.g., a public-key of a trapdoor function) is honestly created before the beginning of all interactions, and later available to all parties; thus, it is a setup assumption by a trusted third party. Though the CRS model may be practical in some situations where a trusted setup is guaranteed exactly, but it must be very restrictive in general; if an untrustworthy or a corrupted party chooses the reference string, he may be able to learn all parties' passwords by just eavesdropping on the communications. Furthermore, parties cannot detect if CRS is maliciously generated. Almost all existing secure and practical PAKE schemes in the standard model are constructed in the CRS model [5,6,7,8,9,10,11,12,13,14,15,16].

There is another methodology to construct PAKE schemes via secure function evaluation [17,18,19]. These schemes avoid both ideal primitives and the CRS model; that is, these schemes are proved to be secure in the plain model. However, the security is only guaranteed in the case of the non-concurrent (i.e., multiple sessions must be executed sequentially) or bounded concurrent setting. Goyal et al. [20] proposed a PAKE scheme (the GJO scheme) that is secure in the plain model under *concurrent* self-composition. Unfortunately, it is a theoretical scheme because many rounds of communication is necessary (i.e., containing a polynomial number of rounds). Thus, our interest is to construct a concurrently secure and round efficient PAKE scheme without relying on any of ideal primitives and the CRS model.

Our Results. In this paper, we give a three-move PAKE scheme that is secure without assuming both ideal primitives and a (centralized) trusted setup.

Naturally, the most desirable goal is to construct a practical PAKE scheme in the plain model. However, we have a lot of hurdles to get an efficient construction in the plain model, even for getting constant round. Thus, this paper aims to avoid the drawback of the CRS model as a milestone toward a practical PAKE scheme in the plain model. Our key idea is to adopt the *multi-string (MS) model* [21].

The MS model is a decentralized model of the ordinary CRS model. In the CRS model, a single-authority creates the CRS and parties must trust the authority. For example, in typical PAKE schemes, a public-key is set as the CRS, and then, a password is encrypted with the public-key and the ciphertext is sent in the protocol. If the authority keeps the secret-key corresponding to the public-key, the password can be decrypted. Conversely, in the MS model, there are multiple authorities and each authority creates a random string, respectively. Honest authorities independently generates their reference strings (i.e., each reference string does not depend on other reference strings) while malicious authorities can set their reference strings arbitrarily. It is clearly decentralized because we only assume a majority of authorities generate random strings honestly. The MS model is very realistic because it is possible to be easily implemented in the real world so that authorities just publish their random strings on the Internet, and parties just need to agree on which authorities' strings they want to use. Obviously, the MS model is more relaxed than the CRS model in necessity of a trusted setup.

We introduce a new round-efficient PAKE scheme in the MS model. Our scheme has two attractive points beside existing schemes as follows:

1. *Round-Efficiency.* The drawback of the GJO scheme is in communication complexity; it needs linear number of rounds in the security parameter. Though Groth and Ostrovsky [21] show that any universally composable (UC) multi-party computation protocol can be securely realized in the MS model, such a general construction also needs a large number of rounds. Conversely, our scheme only needs three-moves. The construction is based on the three-move PAKE scheme by Groce and Katz [12] (GK scheme). The GK scheme is (concurrently) secure in the CRS model and most efficient in known PAKE schemes secure in the standard model. To avoid the single CRS (we will mention later), our scheme is not achieved in one-round; but round efficiency of our scheme is comparable with existing schemes. As far as we know, our scheme is first secure round-efficient PAKE scheme without a centralized trusted setup.
2. *Decentralized Setup.* The GK scheme needs a CRS to prove its security. The CRS contains public-keys of semantically secure public-key encryption (CPA-PKE) and chosen ciphertext secure labelled public-key encryption (CCA-lPKE) [22]. Since the simulator needs to know secret keys corresponding to the CRS in order to respond to adversarial messages in the security proof, public-keys have to be the CRS in order that the simulator can generate secret keys. To achieve security in the MS model, we must change the way to use strings in the protocol. Our technique is twofold. One is that public-keys are chosen by parties temporarily in each session, not included in reference strings. Thus, authorities cannot decrypt the ciphertexts; but, the simulator cannot know secret keys in the security proof. To solve this problem, we use the multi-string simulation-extractable non-interactive zero-knowledge (SENIZK) proof [21]. We can simulate responses to adversarial messages without knowing secret keys by relying on simulation-extractability of the SENIZK proof even if minority authorities are malicious. Hence, our scheme does not need a centralized CRS and enjoys the concurrent security as the GK scheme. We will discuss how to avoid the single CRS in detail in Section 3.2.

To show usefulness of our technique, we also show two extended PAKE schemes in the MS model. One is the lattice-based scheme, and the other is the UC scheme. These schemes are obtained by applying our technique to existing PAKE in the CRS model. Thus, while our technique is not generic transformation, it is applicable to a wide class of PAKE schemes such as [7] and its variants.

2 Preliminaries

Throughout this paper we use the following notations. If Set is a set, then by $m \in_R \mathsf{Set}$ we denote that m is sampled uniformly from Set. If $\mathcal{ALG}$ is an algorithm, then by $y \leftarrow \mathcal{ALG}(x; r)$ we denote that y is output by $\mathcal{ALG}$ on input x and randomness r (if $\mathcal{ALG}$ is deterministic, r is empty).

2.1 Password-Based Authenticated Key Exchange in Multi-string Model

A PAKE scheme contains two parties (an initiator and a responder, or a client and a server) who will engage in the protocol. We suppose that the total number of parties in the system is at most N. Let passwords for all pairs of parties be uniformly and independently chosen from fixed dictionary $\mathcal{D}$. This uniformity requirement is made for simplicity and can be easily removed by adjusting security of an individual password to be the min-entropy of the distribution, instead of $1/|\mathcal{D}|$. Parties P and P' share a password $pw_{PP'}$. Also, we suppose that the total number of authorities in the system is at most n. Each honest authority publishes reference string ρ without recognizing each other or any other parties.

We denote with Π_P^i the i-th instance of key exchange sessions that party P runs. Parties use reference strings to execute instances. Each party can concurrently execute the protocol multiple times with different instances. We suppose that the total number of instances of a party is at most ℓ. The adversary is given oracle access to these instances and may also control some of the instances itself. We remark that unlike the standard notion of an "oracle", in this model instances maintain state which is updated as the protocol progresses. Also, the adversary can corrupt $n - t_p$ authorities and publish $n - t_p$ reference strings (i.e., there are at least t_p honest reference strings), possibly in a malicious and adaptive manner (i.e., generating corrupted reference strings after seeing honest strings). In particular the state of an instance Π_P^i includes the following variables (initialized as null):

- sid_P^i : the session identifier which is the ordered concatenation of all messages sent and received by Π_P^i;
- pid_P^i : the partner identifier whom Π_P^i believes it is interacting ($\mathsf{pid}_P^i \neq P$);
- acc_P^i : a Boolean variable corresponding to whether Π_P^i accepts or rejects at the end of the execution.

We say that two instances Π_P^i and $\Pi_{P'}^j$ are partnered if the following properties hold: $\mathsf{pid}_P^i = P'$ and $\mathsf{pid}_{P'}^j = P$, and $\mathsf{sid}_P^i = \mathsf{sid}_{P'}^j \neq null$ except possibly for the final message.[1] Partnered parties must accept and conclude with the common session key.

[1] The exception of the final message for matching of sid is needed to rule out a trivial attack that an adversary forwards all messages except the final one.

Security Definition. Following [2,12], we show the security definition of PAKE. An adversary is given total control of the external network connecting parties. This adversarial capability is modeled by giving some oracle accesses[2] as follows:

- $\mathsf{Execute}(P, i, P', j)$: This query models passive attacks. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.
- $\mathsf{Send}(P, i, m)$: This query models active attacks. The instance Π_P^i runs according to the protocol specification and updates state. The output of this query consists of the message that the party P would generate on receipt of message m. If the inputted message is empty (say $\perp$), the query means activating the initiator and the output of the query consists of the first move message.
- $\mathsf{Reveal}(P, i)$: This query models leakage of session keys by improper erasure of session keys after use or compromise of a host machine. The output of this query consists of the session key SK of Π_P^i if $\mathsf{acc}_P^i = 1$.
- $\mathsf{Test}(P, i)$: At the beginning a hidden bit b is chosen. If no session key for instance Π_P^i is defined, then return the undefined symbol $\perp$. Otherwise, return the session key for instance Π_P^i if $b = 1$ or a random key from the same domain if $b = 0$. This query is posed just once.

The adversary is considered successful if it guesses b correctly or if it breaks correctness of a session. We say that an instance Π_P^i is fresh unless one of the following is true at the conclusion of the experiment:

- the adversary poses $\mathsf{Reveal}(P, i)$,
- the adversary poses $\mathsf{Reveal}(P', j)$ if Π_P^i and $\Pi_{P'}^j$ are partnered.

We say that an adversary $\mathcal{A}$ succeeds if either:[3]

- $\mathcal{A}$ poses $\mathsf{Test}(P, i)$ for a fresh instance Π_P^i and outputs a bit $b' = b$,
- Π_P^i and $\Pi_{P'}^j$ are partnered, and $\mathsf{acc}_P^i = \mathsf{acc}_{P'}^i = 1$, but session keys are not identical.

The adversary's advantage is formally defined by $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathcal{A} \text{ succeeds}] - 1|$, where κ is a security parameter.

Definition 1 (PAKE). *We say a PAKE protocol is t_p-secure if for a dictionary $\mathcal{D}$ and any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that makes at most Q_{Send} queries of* Send *to different instances the advantage $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is only negligibly larger than $Q_{\mathsf{Send}}/|\mathcal{D}|$ for κ, where t_p is the number of honest reference strings.*

[2] The model does not contain any explicit corruption oracle access (i.e., to reveal passwords). In the password-only setting, such an oracle is unnecessary because an adversary can internally simulate these oracles by itself. Please see [23, pp.190, footnote 8] for details.

[3] If a PAKE scheme requires mutual authenticity (i.e., an adversary cannot cause an instance to accept without any partnered peer), we add the following condition: $\mathsf{acc}_P^i = 1$ but Π_P^i is not partnered with any other instance.

2.2 Smooth Projective Hash Functions

Smooth projective hash functions play a central role in our scheme. This notion is introduced by Cramer and Shoup [24], and our scheme uses its variant.

Let R be an efficiently computable binary relation. For pairs $(x, w) \in R$ we call x the statement and w the witness. Let X be a set and $L \subset X$ be a NP-language consisting of statements in R. Loosely speaking, hash function H_{hk} that maps X to some set is *projective* if there exists a projection key that defines the action of H_{hk} over the subset L of the domain X. That is, there exists deterministic projection function $F(\cdot)$ that maps key hk into its projection key $hp = F(hk)$. The projection key hp is such that for every $x \in L$ it holds that the value of $H_{hk}(x)$ is uniquely determined by hp and x. In contrast, nothing is guaranteed for $x \notin L$, and it may not be possible to compute $H_{hk}(x)$ from hp and x. A *smooth* projective hash function (SPHF) has the additional property (smoothness) that for $x \notin L$, the projection key hp actually says nothing about the value of $H_{hk}(x)$. More specifically, given x and $hp = F(hk)$, the value $H_{hk}(x)$ is uniformly distributed (or statistically close) to a random element in the range of H_{hk}.

An interesting feature of SPHF is that if L is an NP-language, then for every $x \in L$ it is possible to efficiently compute $H_{hk}(x) = h_{hp}(x, w)$ using projection key $hp = F(hk)$ and witness w of the fact that $x \in L$. Alternatively, given hk itself, it is possible to efficiently compute $H_{hk}(x)$ even without knowing the witness. When L is a hard-on-the-average NP-language, for a random $x \in_R L$, given x and $hp = F(hk)$ the value $H_{hk}(x)$ is computationally indistinguishable from a random value in the range of $H_{hk}(x)$. Thus, even if $x \in L$, the value $H_{hk}(x)$ is pseudorandom, unless the witness is known.

2.3 Multi-string Simulation-Extractable Non-interactive Zero-Knowledge Proof

We borrow the definition of multi-string SENIZK proof from [21].

A multi-string proof system for a relation R consists of PPT algorithms $\mathsf{K}, \mathsf{P}, \mathsf{V}$, that K means the key generator, P means the prover and V means the verifier respectively. The key generation algorithm can be used to produce common reference string ρ. The prover takes as input $(\boldsymbol{\rho}, x, w)$ where $\boldsymbol{\rho}$ is a set of n different common reference strings and $(x, w) \in R$, and outputs a proof π. The verifier takes as input $(\boldsymbol{\rho}, x, \pi)$ and outputs 1 if the proof is acceptable and 0 otherwise. We call $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ a (t_c, t_s, t_z, n)-simulation-extractable NIZK proof system for R if it has the completeness, simulation-extractability and extraction zero-knowledge properties described below. Parameters t_c, t_s and t_z affect these properties in a threshold manner as follows: If t_c out of n reference strings are honestly generated, then the prover holding an NP-witness for the truth of the statement should be able to create a convincing proof (i.e., corresponding to completeness). If t_s out of n reference strings are honestly generated, then it should be infeasible to convince the verifier about a false statement (i.e., corresponding to soundness). If t_z out of n reference strings are honestly generated, then it should be possible to simulate the proof without knowing the witness (i.e., corresponding to zero-knowledge).

Definition 2 (Completeness). *We say* $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ *is computationally* (t_c, t_s, t_z, n)*-complete if for any non-uniform polynomial time adversary* $\mathcal{S}$*, we have* $\Pr[(\boldsymbol{\rho}, x, w) \leftarrow \mathcal{S}^{\mathsf{K}}(1^{\kappa});$

$\pi \leftarrow \mathsf{P}(\boldsymbol{\rho}, x, w) : \mathsf{V}(\boldsymbol{\rho}, x, \pi) = 1$ and $(x, w) \in R] \geq 1 - negl$, where K is an oracle on query i outputting $\rho_i \leftarrow \mathsf{K}(1^\kappa)$ and $\mathcal{S}$ outputs $\boldsymbol{\rho}$ such that at least t_c of the ρ_i's generated by K are included.

Simulation-extractability is a combining notion of simulation-soundness and proof of knowledge. As simulation-soundness, it guarantees that an adversary cannot prove any false statement even after seeing simulated proofs of arbitrary statements. Also, as proof of knowledge, it guarantees that there are PPT algorithms that can extract a witness from a valid proof. An algorithm SE generates reference string ρ with trapdoor information δ and ξ. An algorithm S uses δ to produce a valid proof from a statement without knowing the corresponding witness. An algorithm E uses ξ to extract the witness from a valid proof.

Definition 3 (Simulation-Extractability). *We say $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ is (t_c, t_s, t_z, n)-simulation-extractable if SE is a PPT algorithm that outputs (ρ, δ, ξ) and for any non-uniform polynomial time adversary $\mathcal{S}$, we have $|\Pr[\rho \leftarrow \mathsf{K}(1^\kappa) : \mathcal{S}(\rho) = 1] - \Pr[(\rho, \delta, \xi) \leftarrow SE(1^\kappa) : \mathcal{S}(\rho) = 1]| \leq negl$, and if SE is a PPT algorithm that outputs (ρ, δ, ξ), E is a PPT algorithm that outputs w on input (ρ, ξ, x, π) and for any non-uniform polynomial time adversary $\mathcal{S}$, we have $|\Pr[(\boldsymbol{\rho}, x, \pi) \leftarrow \mathcal{S}^{SE_s, S}(1^\kappa); w \leftarrow E(\boldsymbol{\rho}, \boldsymbol{\xi}, x, \pi) : (\boldsymbol{\rho}, x, \pi) \notin \mathcal{L}$ and $(x, w) \notin R$ and $\mathsf{V}(\boldsymbol{\rho}, x, \pi) = 1] \leq negl$, where SE_s is an oracle on query i outputting (ρ_i, ξ_i) from $(\rho_i, \delta_i, \xi_i) \leftarrow SE(1^\kappa)$, S is an oracle on input $(\boldsymbol{\rho}_j, \boldsymbol{\delta}_j, x_j)$ outputting π_j such that $\boldsymbol{\delta}_j$ contains t_z δ_i's corresponding to ρ_i's generated by SE, $\mathcal{L}$ is a list of statements and corresponding proofs $(\boldsymbol{\rho}_j, x_j, \pi_j)$ made by S, and $\boldsymbol{\xi}$ contains t_s ξ_i's corresponding to ρ_i's generated by SE.*

Extraction zero-knowledge is an extended notion of zero-knowledge according to simulation-extractability. It guarantees that even after seeing many extractions it should still be hard to distinguish real proofs and simulated proofs from one another.

Definition 4 (Extraction Zero-Knowledge). *We say $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ is (t_c, t_s, t_z, n)-extraction zero-knowledge if SE is a PPT algorithm that outputs (ρ, δ, ξ), E is a PPT algorithm that outputs w on input (ρ, ξ, x, π), S is a PPT algorithm that outputs π on input $(\boldsymbol{\rho}, \boldsymbol{\delta}, x)$ and for any non-uniform polynomial time adversary $\mathcal{S}$, we have $|\Pr[(\boldsymbol{\rho}, x, w) \leftarrow \mathcal{S}^{SE_z, E}(1^\kappa); \pi \leftarrow \mathsf{P}(\boldsymbol{\rho}, x, w) : \mathcal{S}^E(\pi) = 1$ and $(x, w) \in R] - \Pr[(\boldsymbol{\rho}, x, w) \leftarrow \mathcal{S}^{SE_z, E}(1^\kappa); \pi \leftarrow S(\boldsymbol{\rho}, \boldsymbol{\delta}, x) : \mathcal{S}^E(\pi) = 1$ and $(x, w) \in R]| \leq negl$, where SE_z is an oracle on query i outputting (ρ_i, δ_i) from $(\rho_i, \delta_i, \xi_i) \leftarrow SE(1^\kappa)$, and E is an oracle on input $(\boldsymbol{\rho}_j, \boldsymbol{\xi}_j, x_j, \pi_j)$ outputting w such that the query contains t_s ρ_i's generated by SE and π_j is not the challenge proof for $\mathcal{S}$.*

3 Three-Move PAKE in Multi-string Model

In this section, we show a PAKE scheme in the MS model based on the Groce-Katz PAKE scheme [12] (GK scheme) in the CRS model. Though our scheme is less efficient both in communication and computational complexity than the GK scheme, the setup assumption is relaxed to a decentralized setup.

3.1 Recalling the GK Scheme

First, we recall the GK scheme that is secure in the CRS model. Fig. 1 shows an overview of the GK scheme.

$$\begin{array}{rcl}
\multicolumn{3}{c}{\textbf{Common private input : } \text{password } pw = pw_{AB}} \\
\multicolumn{3}{c}{\textbf{Common reference string : } pk, pk'} \\
\underline{\text{Party } A \text{ (Initiator)}} & & \underline{\text{Party } B \text{ (Responder)}} \\
r' \leftarrow \{0,1\}^* & & \\
CT' = \mathsf{Enc}'_{pk'}(pw; r') & \xrightarrow{trans_1 := A \| CT'} & \\
 & & hk \leftarrow KS \\
 & & hp = F(hk, pk', CT') \\
 & & r_B \| \tau_B \| SK_B = H_{hk}(pk', CT', pw) \\
 & & label := trans_1 \| B \| hp \\
 & \xleftarrow{trans_2 := B \| hp \| CT} & CT = \mathsf{Enc}^{label}_{pk}(pw; r_B) \\
r_A \| \tau_A \| SK_A = h_{hp}(pk', CT', pw, r') & & \\
label := trans_1 \| B \| hp & & \\
\hat{CT} = \mathsf{Enc}^{label}_{pk}(pw; r_A) & & \\
\text{if } \hat{CT} \neq CT, \text{ abort} & \xrightarrow{trans_3 := A \| \tau_A} & \\
 & & \text{if } \tau_A \neq \tau_B, \text{ abort} \\
\text{output } SK_A & & \text{output } SK_B
\end{array}$$

Fig. 1. A high-level overview of the GK scheme

The GK scheme uses a CCA-lPKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where the message space $MS = \mathcal{D}$, PKS is the public-key space and CTS is the ciphertext space, and a CPA-PKE $\Sigma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ with an associated SPHF where the message space $MS' = \mathcal{D}$, PKS' is the public-key space and CTS' is the ciphertext space. We define sets X, $\{L_m\}_{m \in \mathcal{D}}$ and language L for Σ' with respect to pk'. Let X be a set $\{(pk', CT', m) | pk' \in PKS'; CT' \in CTS'; m \in \mathcal{D}\}$, L_m be a set $\{(pk', CT', m) | (pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa); \mathsf{Dec}'_{sk'}(CT') = m\}$ and L be a set $\cup_{m \in \mathcal{D}} L_m$.

As described in Section 2.2, the GK scheme uses a family of SPHFs $\mathcal{H} = \{H_{hk}\}$ such that for every hk in the key space KS, $H_{hk} : X \rightarrow \{0,1\}^{3\kappa}$ and $F : KS \times PKS' \times CTS' \rightarrow PS$ where PS is the projection key space. Formally, the SPHF that the GK scheme uses is defined by a sampling algorithm that outputs $(KS, \mathcal{H}, PS, F)$ such that:

- There are efficient algorithms that sample a uniform $hk \in KS$, compute H_{hk} for $hk \in KS$ and $x \in X$, and compute $F(hk, pk', CT')$ for all $hk \in KS$, $pk' \in PKS'$ and $CT' \in CTS'$.
- For any $x = (pk', CT', m) \in L$, there is an efficient algorithm that given inputs $hp = F(hk, pk', CT')$ and (pk', CT', m, r') such that $CT' = \mathsf{Enc}'_{pk'}(m; r')$ computes $h_{hp}(x, r') = H_{hk}(x)$.
- For any $x = (pk', CT', m) \in X \setminus L$, distributions $\{hk \leftarrow KS; hp = F(hk, pk', CT') : (hp, H_{hk}(x))\}$ and $\{hk \leftarrow KS; hp = F(hk, pk', CT'); v \leftarrow \{0,1\}^{3\kappa} : (hp, v)\}$ are statistical indistinguishable in κ.

The GK scheme ensures both correctness and authenticity by relying on the SPHF. First, correctness is guaranteed because of the projection property. Initiator A only knows the projection key hp, but he also knows the randomness r' in generating CT'. Thus, A can derive the correct session key SK with h_{hp} and r'. On the other hand, responder B does not know r', but knows the hash key hk. Thus, B can derive the correct session key SK only with hk. Also, authenticity is guaranteed because of the smoothness property. If a party does not know correct password pw, he cannot determine $r_A = r_B$ and $\tau_A = \tau_B$; hence, the verification of the peer is failed.

3.2 Design Principle

Here, we show our strategy to avoid centralized trusted setup with the MS model.

First, we observe how the CRS model contributes to prove security of the GK scheme. In some step of the security proof, a simulator needs to check if a ciphertext that is generated by the adversary is valid for the correct password. If so, the simulator regards the adversary successful. The simulator can know secret keys sk, sk' for pk, pk' because the CRS is generated by him. Thus, the simulator successfully works in this case with the power of trapdoors of the CRS. In the MS model, the simulator cannot generate all public-keys $(pk_1, \ldots, pk_n)$ because an adversary may publish malicious public-keys as corrupt authorities. Then, the simulator can know trapdoors of only honest authorities. Next, we discuss what is non-trivial to construct PAKE under this setting.

Simple Solutions Do Not Work. A naive idea is that parties encrypt the password with all public-keys and send n ciphertexts. In this case, the simulator can check if a ciphertext that is generated by the adversary is valid for the correct password with a secret key of an honest authority. Obviously, to directly encrypt the password is flawed because the adversary knows secret keys of corrupted authorities and can obtain the password.

Threshold cryptography is widely used in cryptographic protocols involving a multi-authority setting. If we apply it to the PAKE setting, parties transform the password into n shares with a secret sharing scheme like [25], then, encrypt each share with each public-key, and send n ciphertexts. If malicious public-keys are less than the threshold, the password is perfectly protected, and the simulator can reconstruct the password and check the validity of ciphertexts. However, there is a problem that each party cannot know shares of the peer because he does not have any secret key and the share generation algorithm must be probabilistic with local randomness. To compute a common value with the SPHF, parties can input the same encrypted messages to H_{hk} or h_{hp}. Therefore, the approach using threshold cryptography does not work.

Another way is to accumulate all public-keys to a combined public-key. For example, with the ElGamal encryption, the public-keys $(pk_1, \ldots, pk_n)$ are combined as $pk = \prod_1^n pk_i$, and the password is encrypted by pk. In this case, parties can compute a common value with the SPHF. Unfortunately, this approach does not work because a malicious authority can reveal the password. In the MS model, it is not guaranteed that malicious authorities determine their reference strings without seeing other authorities' strings. If a malicious authority observes all other authorities' public-keys before publishing her public-key, she can easily set $pk = \prod_1^n pk_i$ to an arbitrary value that she knows the underlying secret key.

Therefore, these orthodox techniques are not useful in the PAKE setting.

Our Technique. Our main idea is twofold: to use *ad-hoc* public-key and to use a *multi-string SENIZK proof*. Problems we discussed above come from the fact that corrupted authorities can decrypt ciphertexts. Thus, we modify the protocol as each party generates an ad-hoc public-key[4] and encrypts the password with the ad-hoc public-key. Then, the adversary cannot decrypt it. Conversely, the simulator cannot also decrypt the ciphertext that is generated by the adversary because the ad-hoc public-key is also generated by the adversary. We find that what the simulator must do is *not to decrypt the ciphertext but to check if it is valid for the correct password.* Hence, we can solve this problem by using the simulation-extractability of SENIZK. Each party proves that he knows a plaintext and randomness that are used to generate the ciphertext with the public-key. The simulation-extractability ensures that there is a PPT algorithm to extract a witness from a valid proof. The simulator can extract the encrypted plaintext, and check if the plaintext is the correct password.

More formally, the SENIZK proves that ciphertext CT is well-formed in that there exists (pw, r) such that $CT = \mathsf{Enc}_{pk}(pw; r)$. Nobody knows the secret key corresponding to pk. Though the adversary can pose Send query, the simulator can handle such a query with oracle S in the definition of the simulation-extractability. Also, even if arbitrary simulated proofs are provided, the adversary cannot distinguish the simulation environment and the real experiment from the extraction zero-knowledge, and cannot prove any false statement from the simulation-extractability. We use $(0, \frac{n+1}{2}, \frac{n+1}{2}, n)$-SENIZK (i.e., a majority of reference strings are honest). Such a SENIZK is given in [21] for any NP-language. The SENIZK not only allowed us to prove security, but allows us to do so without CCA-lPKE for the responder. A semantically secure labelled public-key encryption (CPA-lPKE) is sufficient.

A remaining problem is how to deal with the SPHF. In the GK scheme, since the SPHF is parameterized by a public-key in the CRS, the public-key is put in the CRS. In our scheme, the SPHF is parameterized by an ad-hoc public-key; thus, we cannot put it in some reference string. This problem is easily resolved. We can simply put the specification of the SPHF in the system-wide parameter without parameterizing by a public-key because the hash key sampling algorithm does not depend on the public-key and other functions are deterministic. The responder can compute projection key hp with an ad-hoc public-key and a ciphertext from the peer. Here, we show an example of the SPHF for the ElGamal encryption: $pk = (g, h)$ is the public-key, r is the randomness to encrypt message m, and $CT = (u = g^r, e = mh^r)$ is the ciphertext. The hash key generation algorithm $HKGen(\mathbb{Z}_p)$ chooses hash key $hk = (a_1, a_2)$ from $\mathbb{Z}_p^2$, and the projection key generation algorithm $HPGen(pk, hk)$ generates projection key hp as $g^{a_1}h^{a_2}$. The hash key-based hash value derivation algorithm $HashHK(CT, m, hk)$ computes the hash value as $u^{a_1}(e/m)^{a_2}$, and the projection key-based hash value derivation algorithm $HashHP(r, hp)$ computes the hash value as hp^r. In this case, the specification of the SPHF without parameterizing by a public-key is set as $(HKGen(\mathbb{Z}_p), HPGen(\cdot,\cdot), HashHK(\cdot,\cdot,\cdot), HashHP(\cdot,\cdot))$. Thus, it can be put in the system-wide parameter without depending on ad-hoc public-keys.

3.3 Our Protocol

A high-level overview of the protocol appears in Fig. 2. Our scheme is formally described as follows:

[4] It is unnecessary to keep the secret-key.

Common private input : password $pw = pw_{AB}$
Multi-string : $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_n)$

Party A (Initiator)		Party B (Responder)
$r' \leftarrow \{0,1\}^*$		
$(pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa)$		
$CT' = \mathsf{Enc}'_{pk'}(pw; r')$		
$\pi' \leftarrow \mathsf{P}(\boldsymbol{\rho}, (pk', CT'), (pw, r'))$	$\xrightarrow{trans_1 := A \Vert pk' \Vert CT' \Vert \pi'}$	
		if $\mathsf{V}(\boldsymbol{\rho}, (pk', CT'), \pi') = 0$, abort
		$hk \leftarrow KS$
		$hp = F(hk, pk', CT')$
		$r_B \Vert \tau_B \Vert SK_B = H_{hk}(pk', CT', pw)$
		$(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
		$label := trans_1 \Vert B \Vert hp \Vert pk$
		$CT = \mathsf{Enc}^{label}_{pk}(pw; r_B)$
	$\xleftarrow{trans_2 := B \Vert hp \Vert pk \Vert CT \Vert \pi}$	$\pi \leftarrow \mathsf{P}(\boldsymbol{\rho}, (pk, CT, label), (pw, r_B))$
$label := trans_1 \Vert B \Vert hp \Vert pk$		
if $\mathsf{V}(\boldsymbol{\rho}, (pk, CT, label), \pi) = 0$, abort		
$r_A \Vert \tau_A \Vert SK_A = h_{hp}(pk', CT', pw, r')$		
$\hat{CT} = \mathsf{Enc}^{label}_{pk}(pw; r_A)$		
if $\hat{CT} \neq CT$, abort	$\xrightarrow{trans_3 := A \Vert \tau_A}$	
		if $\tau_A \neq \tau_B$, abort
output SK_A		output SK_B

Fig. 2. A high-level overview of our basic PAKE scheme

Public Parameters. κ is a security parameter. Let $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-lPKE where the message space $MS = \mathcal{D}$, PKS is the public-key space and CTS is the ciphertext space, and $\Sigma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a CPA-PKE with an associated SPHF where the message space $MS' = \mathcal{D}$, PKS' is the public-key space and CTS' is the ciphertext space. We define sets X, $\{L_m\}_{m \in MS}$ and L for Σ'. Let X be a set $\{(pk', CT', m) | pk' \in PKS'; CT' \in CTS'; m \in MS'\}$, L_m be a set $\{(pk', CT', m) | (pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa);$ $\mathsf{Dec}'_{sk'}(CT') = m\}$ and L be a set $\cup_{m \in MS} L_m$. We use a family of SPHFs $\mathcal{H} = \{H_{hk}\}$ such that for every hk in the key space KS, $H_{hk} : X \to \{0,1\}^{3\kappa}$ and $F : KS \times PKS' \times CTS' \to PS$ where PS is the projection key space. Each authority generates reference string $\rho_i \leftarrow \mathsf{K}(1^\kappa)$ for SENIZK. The multi-strings is $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_n)$ where ρ_i is generated by i-th authority.

Protocol Execution. The initiator A generates a randomness $r' \in \{0,1\}^*$ and a public-key $(pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa)$, and computes the ciphertext $CT' = \mathsf{Enc}'_{pk'}(pw; r')$ with the password pw. Define a language L'_{ZK} as $L'_{ZK} := \{(pk', CT') : \exists pw \text{ and } r' \text{ s.t. } CT' = \mathsf{Enc}'_{pk'}(pw; r')\}$. A produces a SENIZK proof π' that $(pk', CT') \in L'_{ZK}$, with the multi-string $\boldsymbol{\rho}$. A sends $trans_1 := A \Vert pk' \Vert CT' \Vert \pi'$ to the responder B. Upon receiving $A \Vert pk' \Vert CT'$ $\Vert \pi'$, B verifies π' with pk', CT' and $\boldsymbol{\rho}$. If π' is invalid, B aborts. Otherwise, B generates a hash key $hk \leftarrow KS$ and a public-key $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$, derives the projection key $hp = F(hk, pk', CT')$ and $r_B \Vert \tau_B \Vert SK_B = H_{hk}(pk', CT', pw)$, sets $label :=$ $trans_1 \Vert B \Vert hp \Vert pk$, and computes the ciphertext $CT = \mathsf{Enc}^{label}_{pk}(pw; r_B)$. Define a language L_{ZK} as $L_{ZK} := \{(pk, CT, label) : \exists pw \text{ and } r_B \text{ s.t. } CT = \mathsf{Enc}^{label}_{pk}(pw; r_B)\}$. B produces a SENIZK proof π that $(pk, CT, label) \in L_{ZK}$, with $\boldsymbol{\rho}$. B sends $trans_2 :=$

$B||hp||pk||CT||\pi$ to A. Upon receiving $B||hp||pk||CT||\pi, A$ sets $label := trans_1||B||hp||pk$, and verifies π with $pk, CT, label$ and ρ. If π is invalid, A aborts. Otherwise, A derives $r_A||\tau_A||SK_A = h_{hp}(pk', CT', pw, r')$, computes the ciphertext $\hat{CT} = \mathsf{Enc}_{pk}^{label}(pw; r_A)$, and checks whether $\hat{CT} \neq CT$. If so, A aborts. Otherwise, A sends τ_A to B and outputs the session key SK_A. Upon receiving τ_A, B checks whether $\tau_A \neq \tau_B$. If so, B aborts. Otherwise, B outputs the session key SK_B.

Correctness. When both parties A and B have the common password, the session keys that they compute are the same. This is because the same hash value is obtained when using the hash key hk and when using the projection key hp. This implies that the correctness property holds for the protocol.

Concrete Instantiation. Σ and Σ' can be instantiated by the ElGamal encryption because it can admit a smooth projective hash function. A concrete SENIZK proof in the MS model is introduced in [21]. Specifically, the SENIZK proof is constructed from a CCA-PKE, a zap, and a strong one-time signature. We have efficient CCA-PKE schemes such as the Cramer-Shoup encryption [26]. Zaps are public-coin witness-indistinguishable proofs. While the original SENIZK proof uses a two-move zap [27], we can replace it with an efficient non-interactive zap [28,29] for circuit SAT based on the decisional linear assumption. We can also use an efficient strong one-time signature scheme [30] based on the discrete-log assumption.

3.4 Security

Theorem 1. *Assume that Σ' is a CPA-PKE with an associated SPHF, Σ is a CPA-lPKE, and $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ is $(0, \frac{n+1}{2}, \frac{n+1}{2}, n)$-simulation-extractable and $(0, \frac{n+1}{2}, \frac{n+1}{2}, n)$-extraction zero-knowledge. Then, our scheme in Fig. 2 is $(\frac{n+1}{2})$-secure in the MS model.*

The security proof of our scheme follows the manner of the GK scheme. We proceed to prove the security through a series of hybrid experiment.

Due to space limitation, the proof of Theorem 1 is shown in the full paper. We show a sketch of the proof.

First, we modify the generation of the reference string ρ_i by each honest authority so that ρ_i is generated with trapdoor information by the algorithm SE in the definition of multi-string SENIZK proof. Since honest authorities are majority, this change is indistinguishable from simulation-extractability of SENIZK. Trapdoor information helps to extract the passwords from adversary-generated messages, and the simulator can check the validity of adversary-generated messages in later.

Next, we modify the output of $\mathsf{Execute}$ oracle so that all proofs π' and pi are changed to simulated proofs by the algorithm SE in the definition of multi-string SENIZK proof, ciphertexts CT' and CT are changed to encryptions of a fake password, and $H_{hk}(pk', CT', pw)$ is changed to random. Since honest authorities are majority, these changes are indistinguishable from extraction zero-knowledge of SENIZK, CPA security and smoothness of SPHF, respectively. Then, the session key becomes truly random and all transcripts are independent from the real passwords for the output of Execute oracle.

Finally, we modify the output of Send oracle similarly as the case of Execute oracle. We note that the simulator must check the validity of adversary-generated messages by verifying that the correct password is used. However, the simulator does not know the secret key of Σ and Σ' because it is not contained in reference strings. It can be solved by using trapdoor information of reference strings of honest authorities. The simulator can extract a password from the adversary-generated proof. and can check the validity of it. This part is the distinguished point from existing proof scenarios in the CRS model. From extraction zero-knowledge of SENIZK, CPA security and smoothness of SPHF, we can change the output of Send oracle so that the session key becomes truly random and all transcripts are independent from the real passwords.

4 Other Constructions

In this section, we briefly discuss if our technique is applicable to construct other PAKE schemes that have different benefit than our basic scheme in Section 3.3.

4.1 Lattice-Based Three-Move PAKE in Multi-string Model

Katz and Vaikuntanathan [11] propose a lattice-based three-move PAKE scheme (the KV1 scheme) in the CRS model. Because perfect correctness is hard to achieve in lattice-based cryptosystems, the KV1 scheme uses the notion of *approximate* SPHF (i.e., correctness is guaranteed approximately).

We can apply our technique (i.e., generating ad-hoc public-keys and adding the SENIZK proof) to the KV1 scheme. Fig. 3 shows a high-level overview of the lattice-based protocol. We use a family of ϵ-approximate SPHFs $\mathcal{H} = \{H_{hk}\}$ such that for every hk in the key space KS, $H_{hk} : X \to \{0,1\}^{\ell}$ and $F : KS \times PKS \times CTS \times LS \to PS$, where KS is the hash key space, PTS is the public-key space, CTS is the ciphertext space, LS is the label space and PS is the projection key space. Let $\mathsf{ECC} : \{0,1\}^{\kappa} \to \{0,1\}^{\ell}$ be an error-correcting code correcting a 2ϵ-fraction of errors. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-lPKE with an associated ϵ-approximate SPHF. Let $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Ver})$ be a one-time signature scheme. The SENIZK proof $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ is for the same language as Section 3.3. Please see [11] for definitions of ϵ-approximate SPHFs and error-correcting codes.

The security proof is similar to the proof in [11]. The main difference is the way to extract the password from the ciphertext that is generated by an adversary. As the proof of Theorem 1, we use the power of simulation-extractability of the SENIZK proof.

For an instantiation, we use the same ingredients as the KV1 scheme except the SENIZK proof. As a building block of the SENIZK proof, while we can use an efficient non-interactive zap [28,29] in the instantiation of our scheme in Section 3.3, it is not suitable for the lattice-based protocol because of the assumption. Thus, we use a two-move zap [27] that is constructed from a general NIZK proof. Therefore, our lattice-based scheme is less efficient than our basic scheme; but, it has an advantage in immunity against quantum attacks.

Common private input : password $pw = pw_{AB}$
Multi-string : $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_n)$

Party A (Initiator) | Party B (Responder)

$(vk, ssk) \leftarrow \mathsf{SigGen}(1^\kappa)$
$r' \leftarrow \{0,1\}^*$
$(pk', sk') \leftarrow \mathsf{Gen}(1^\kappa)$
$label' := A\|B\|vk$
$CT' = \mathsf{Enc}_{pk'}^{label'}(pw; r')$
$\pi' \leftarrow \mathsf{P}(\boldsymbol{\rho}, (pk', CT', label'), (pw, r'))$ $\xrightarrow{A\|pk'\|CT'\|vk\|\pi'}$

$label' := A\|B\|vk$
if $\mathsf{V}(\boldsymbol{\rho}, (pk', CT', label'), \pi') = 0$, abort
$hk \leftarrow KS$
$hp = F(hk, pk', CT', label')$
$(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$label := \epsilon$
$r \leftarrow \{0,1\}^*$
$CT = \mathsf{Enc}_{pk}^{label}(pw; r)$
$\xleftarrow{B\|hp\|pk\|CT\|\pi}$ $\pi \leftarrow \mathsf{P}(\boldsymbol{\rho}, (pk, CT, label), (pw, r))$

$label := \epsilon$
if $\mathsf{V}(\boldsymbol{\rho}, (pk, CT, label), \pi) = 0$, abort
$hk' \leftarrow KS$
$hp' = F(hk', pk, CT, label)$
$tk' = H_{hk'}(pk, label, CT, pw)\cdot$
$h_{hp}(pk', label', CT', pw, r')$
$SK_A \leftarrow \{0,1\}^\kappa$
$c := \mathsf{ECC}(SK_A)$
$\varDelta := tk' \oplus c$
$\sigma = \mathsf{Sign}_{ssk}(pk', CT', pk, CT, hp', hp)$ $\xrightarrow{A\|hp'\|\varDelta\|\sigma}$

if $\mathsf{Ver}_{vk}((pk', CT', pk, CT, hp', hp), \sigma) = 0$, abort
$tk = h_{hp'}(pk, label, CT, pw, r)\cdot$
$H_{hk}(pk', label', CT', pw)$
$SK_B = \mathsf{ECC}^{-1}(tk \oplus \varDelta)$

output SK_A | output SK_B

Fig. 3. A high-level overview of the lattice-based PAKE scheme

4.2 Universally Composable Three-Move PAKE in Multi-string Model

Katz and Vaikuntanathan [13] propose a UC one-round PAKE scheme (the KV2 scheme) in the CRS model. The KV2 scheme achieves the UC security by adding a simulation-sound NIZK (SSNIZK) proof that proves that 1) there exists a hash key which is the plaintext of a ciphertext, and 2) a projection key is generated from the hash key.

We also can apply our technique to the KV2 scheme. Fig. 4 shows a high-level overview of the UC protocol. We use the family of SPHFs $\mathcal{H} = \{H_{hk}\}$ that is constructed in [13]. The difference between SPHFs in Section 3.3 and in [13] is that the projection key is generated without inputting the ciphertext in the latter SPHFs (i.e., $hp = F(hk, pk)$).[5] Let $(\mathsf{Gen1}, \mathsf{Enc1}, \mathsf{Dec1})$ be a CCA-PKE. Let $(\mathsf{Gen2}, \mathsf{Enc2}, \mathsf{Dec2})$ be a CPA-lPKE with an associated SPHF. The SENIZK proof $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ is for the same language as Section 3.3. Also, we use a SSNIZK proof $(\mathsf{K}', \mathsf{P}', \mathsf{V}')$ for the following

[5] The SPHF based on the ElGamal encryption in [12] indeed satisfies the definition in [13]. Thus, we use the same SPHFs as Section 3.3.

Common private input : password $pw = pw_{AB}$
Multi-string : $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_n)$ and $\boldsymbol{\rho}' := (\rho'_1, \ldots, \rho'_n)$

Party A (Initiator)		Party B (Responder)
$(pk'_1, sk'_1) \leftarrow \mathsf{Gen1}(1^\kappa)$ $(pk'_2, sk'_2) \leftarrow \mathsf{Gen2}(1^\kappa)$	$\xrightarrow{A \Vert pk'_1 \Vert pk'_2}$	
		$(pk_1, sk_1) \leftarrow \mathsf{Gen1}(1^\kappa)$ $(pk_2, sk_2) \leftarrow \mathsf{Gen2}(1^\kappa)$ $r_1, r_2 \leftarrow \{0,1\}^*$ $hk \leftarrow KS$ $hp = F(hk, pk'_2)$ $CT_1 = \mathsf{Enc1}_{pk_1}(hk; r_1)$ $\pi_1 \leftarrow \mathsf{P}'(\boldsymbol{\rho}', (pk_1, hp, CT_1), (hk, r_1))$ $label := ssid \Vert B \Vert A \Vert pk_1 \Vert pk_2 \Vert hp \Vert CT_1 \Vert \pi_1$ $CT_2 = \mathsf{Enc2}^{label}_{pk_2}(pw; r_2)$
	$\xleftarrow{B \Vert pk_1 \Vert pk_2 \Vert CT_1 \Vert CT_2 \Vert hp \Vert \pi_1 \Vert \pi_2}$	$\pi_2 \leftarrow \mathsf{P}(\boldsymbol{\rho}, (pk_2, CT_2, label), (pw, r_2))$
if $\mathsf{V}'(\boldsymbol{\rho}', (pk_1, hp, CT_1), \pi_1) = 0$, abort $label := ssid \Vert B \Vert A \Vert pk_1 \Vert pk_2 \Vert hp \Vert CT_1 \Vert \pi_1$ if $\mathsf{V}(\boldsymbol{\rho}, (pk_2, CT_2, label), \pi_2) = 0$, abort $r'_1, r'_2 \leftarrow \{0,1\}^*$ $hk' \leftarrow KS$ $hp' = F(hk', pk_2)$ $CT'_1 = \mathsf{Enc1}_{pk'_1}(hk'; r'_1)$ $\pi'_1 \leftarrow \mathsf{P}'(\boldsymbol{\rho}', (pk'_1, hp', CT'_1), (hk', r'_1))$ $label' := ssid \Vert A \Vert B \Vert pk'_1 \Vert pk'_2 \Vert hp' \Vert CT'_1 \Vert \pi'_1$ $CT'_2 = \mathsf{Enc2}^{label'}_{pk'_2}(pw; r'_2)$		
$\pi'_2 \leftarrow \mathsf{P}(\boldsymbol{\rho}, (pk'_2, CT'_2, label'), (pw, r'_2))$	$\xrightarrow{A \Vert CT'_1 \Vert CT'_2 \Vert hp' \Vert \pi'_1 \Vert \pi'_2}$	

		if $\mathsf{V}'(\boldsymbol{\rho}', (pk'_1, hp', CT'_1), \pi'_1) = 0$, abort
		 $label' := ssid \Vert A \Vert B \Vert pk'_1 \Vert pk'_2 \Vert hp' \Vert CT'_1 \Vert \pi'_1$
		 if $\mathsf{V}(\boldsymbol{\rho}, (pk'_2, CT'_2, label'), \pi'_2) = 0$, abort

$SK_A = H_{hk'}(pk_2, label, CT_2, pw)$ $\cdot h_{hp}(pk'_2, label', CT'_2, pw, r'_2)$ output SK_A		$SK_B = h_{hp'}(pk_2, label, CT_2, pw, r_2)$ $\cdot H_{hk}(pk'_2, label', CT'_2, pw)$ output SK_B

Fig. 4. A high-level overview of the UC PAKE scheme

language: $\{(pk, hp, CT) : \exists hk \in KS$ and r s.t. $CT = \mathsf{Enc1}_{pk}(hk; r)$ and $hp = F(hk, pk)\}$. where the corresponding multi string is $\boldsymbol{\rho}' := (\rho'_1, \ldots, \rho'_n)$. Because SENIZK implies SSNIZK, we can use the SENIZK proof as the SSNIZK proof in the MS model. Please see [31,32,13] for the UC framework and the definition of the simulation-sound NIZK.

The security proof is similar to the proof in [13]. The simulator extracts the password from the ciphertext that is generated by an adversary by simulation-extractability of the SENIZK proof.

The KV2 scheme is one-round protocol, but our UC PAKE scheme needs three-move. We add the first move in order to send initiator's public-keys. The reason we must do that is that the projection key is generated from the hash key and the public-key. In the CRS model, the public-key is put as the CRS, and each party can generate the projection key without interacting with the peer. However, in the MS model, the public-key cannot be put as reference strings because of the discussion in Section 3.2. Hence, we add an additional move to send the public-key.

4.3 Is One-Round PAKE in Multi-string Model Possible?

All our constructions are three-move protocols. Some existing PAKE schemes are one-round such as a scheme in [13] which is secure in the game-based model (i.e., the BPR model [2]). One may wonder why we do not construct a one-round PAKE scheme. Here, we show the reason as follows:

In SPHF-based PAKE, each party must send a projection key without receiving any message from the peer in one-round protocols. However, our methodology requires that public-keys are not put as a CRS but are sent in messages, and the generation of a projection key depends on the public-key of the peer. Therefore, we need an additional message to send the public-key of the initiator. It is the essential reason why our PAKE scheme is not achieved in one-round. All known SPHF-based constructions fall into this problem by adapting our methodology.

The last resort is to use another methodology than the SPHF-based. For example, Canetti et al. [15] proposed a secure PAKE scheme in the CRS model based on an oblivious transfer (OT). Unfortunately, their scheme is not achieved in one-round because it follows the design principle of the GK scheme.

Therefore, a secure one-round PAKE in the MS model is an open problem.

4.4 Semi-Generic Transformation from CRS Model

Our technique is able to be interpreted as a semi-generic transformation from secure PAKE scheme to remove the centralized CRS. Our scheme indeed attaches public-keys to messages, and the SENIZK to prove possession of a password to the GK scheme. However, the technique cannot be applied to any secure PAKE in the black-box manner because it heavily depends on the structure of the underlying scheme. In the GK scheme, a password is encrypted (or committed) by PKE schemes (or a commitment scheme) and the initiator and the responder exchange ciphertexts (or commitments). This structure allows us to prove possession of a password under self-generated public-keys with the SENIZK.

On the other hand, it is applicable to schemes having the same structure like the Gennaro-Lindell framework [7,9], the lattice-based PAKE [11], and the UC PAKE[13] as in 4.1 and 4.2. The common feature of these protocols is that the CRS just contains public-keys to encrypt passwords (and some public information) and the SENIZK property is sufficient to replace NIZKs used in protocols (if any). In this case, we can give the semi-generic transformation from a secure PAKE scheme π is as follows:

1. If π includes NIZKs in the CRS model, the CRS is replaced with the MS for the SENIZK.
2. The initiator I and responder R in π generate public-keys (or commitment CRSs) which are used to encrypt (or commit) the shared password. They exchange their public-keys.
3. When I or R computes a ciphertext or a commitment of the password, the party also generates a proof that public-keys are correctly generated under the password with the SENIZK. Also, if generations of NIZK proofs are contained, these proofs are replaced with the SENIZK.
4. In other parts, I and R execute π.

Though this transformation needs extra two moves for exchanging public-keys, we can optimize the number of moves depending on the structure of the underlying protocol. For example, our constructions in Section 3.3 and 4.1 do not need any extra move, but the construction in Section 4.2 needs an additional move.

5 Concluding Remark

We introduced a first three-move PAKE scheme which is secure without any centralized trusted setup (i.e., in the MS model).

Our idea to construct the PAKE scheme in the MS model is not trivially applicable to a construction in the plain model. The proof in the MS model depends on the fact that there are honest trusted authorities. Thus, we can use the SENIZK proof. Conversely, it is impossible to construct NIZK proofs for non-trivial languages in the plain model [33]. A possible way is to use some (concurrently secure) interactive zero-knowledge proof protocol in the plain model instead of using the SENIZK proof. However, such a protocol needs a large number of rounds.

Therefore, a secure constant round PAKE scheme in the plain model is still hard and an open problem.

References

1. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In: IEEE S&P, pp. 72–84 (1992)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
3. Boyko, V., MacKenzie, P.D., Patel, S.: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
4. MacKenzie, P.D., Patel, S., Swaminathan, R.: Password-Authenticated Key Exchange Based on RSA. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 599–613. Springer, Heidelberg (2000)
5. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
6. Katz, J., Ostrovsky, R., Yung, M.: Forward Secrecy in Password-Only Key Exchange Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 29–44. Springer, Heidelberg (2003)
7. Gennaro, R., Lindell, Y.: A Framework for Password-Based Authenticated Key Exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 408–432. Springer, Heidelberg (2003)
8. Jiang, S., Gong, G.: Password Based Key Exchange with Mutual Authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004)
9. Gennaro, R.: Faster and Shorter Password-Authenticated Key Exchange. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 589–606. Springer, Heidelberg (2008)
10. Katz, J., Ostrovsky, R., Yung, M.: Efficient and secure authenticated key exchange using weak passwords. J. ACM 57(1), 1–39 (2009)
11. Katz, J., Vaikuntanathan, V.: Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 636–652. Springer, Heidelberg (2009)
12. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: ACM Conference on Computer and Communications Security 2010, pp. 516–525 (2010)

13. Katz, J., Vaikuntanathan, V.: Round-Optimal Password-Based Authenticated Key Exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011)
14. Jutla, C.S., Roy, A.: Relatively-Sound NIZKs and Password-Based Key-Exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 485–503. Springer, Heidelberg (2012)
15. Canetti, R., Dachman-Soled, D., Vaikuntanathan, V., Wee, H.: Efficient Password Authenticated Key Exchange via Oblivious Transfer. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 449–466. Springer, Heidelberg (2012)
16. Ben Hamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 272–291. Springer, Heidelberg (2013)
17. Goldreich, O., Lindell, Y.: Session-Key Generation Using Human Passwords Only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
18. Nguyen, M.H., Vadhan, S.P.: Simpler Session-Key Generation from Short Random Passwords. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 428–445. Springer, Heidelberg (2004)
19. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure Computation Without Authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
20. Goyal, V., Jain, A., Ostrovsky, R.: Password-Authenticated Session-Key Generation on the Internet in the Plain Model. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 277–294. Springer, Heidelberg (2010)
21. Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
22. Shoup, V. (ed.): Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers. International Organization for Standardization, ISO/IEC 18033–2 (2006)
23. Gennaro, R., Lindell, Y.: A Framework for Password-Based Authenticated Key Exchange. ACM Trans. Inf. Syst. Secur. 9(2), 181–234 (2006)
24. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal on Computing 33, 167–226 (2004)
25. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
26. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
27. Dwork, C., Naor, M.: Zaps and Their Applications. In: FOCS 2000, pp. 283–293 (2000)
28. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive Zaps and New Techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
29. Groth, J., Ostrovsky, R., Sahai, A.: New Techniques for Noninteractive Zero-Knowledge. J. ACM 59(3), 11 (2012)
30. Mohassel, P.: One-Time Signatures and Chameleon Hash Functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 302–319. Springer, Heidelberg (2011)
31. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Cryptology ePrint Archive: 2000/067 (2005), `http://eprint.iacr.org/2000/067/`
32. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
33. Goldreich, O., Oren, Y.: Definitions and Properties of Zero-Knowledge Proof Systems. J. Cryptology 7(1), 1–32 (1994)