Password-based Authenticated Key Exchange in the Bare Public-Key Model

Abstract. Almost all of previous 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 trusted third party in the setup phase; thus, passwords of parties may be revealed if a third party ill-uses trapdoor information of CRS. A PAKE scheme by Goyal et al. is only known to be concurrently secure in the plain model. However, their scheme is 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 bare public-key (BPK) model that allows parties (including malicious one) to freely register their public-keys in a public file without certification authority before starting protocol executions. The BPK model is more relaxed setup assumption than the CRS model because any trusted third party is unnecessary. Though the BPK model is slightly restrictive than the plain model, it is very realistic assumption. Our aim is to construct concurrently secure PAKE schemes in the BPK model (justly without random oracles). First, we propose a pretty simple conversion of PAKE schemes, that can relax the setup assumption from the CRS model to the BPK model. The proposed conversion needs at least an additional move than underlying PAKE schemes; hence, next, we also give a concrete PAKE scheme secure in the BPK model without round overheads. This work will be a milestone toward constant round PAKE schemes in the plain model.

keywords: authenticated key exchange, password, bare public-key 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 achieves both authentication by passwords and session key generation (often used to establish a secure channel) simultaneously. With PAKE schemes, two parties share a common short password in advance and can generate a common long session key over an insecure channel.

The first PAKE scheme was proposed by Bellovin and Merritt [BM92]. However, the security of their scheme is not proved formally. To construct a provably secure PAKE scheme is not so easy due to low entropy password. 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 impersonates 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 $\mathcal{Q}/|\mathcal{D}|$ where $\mathcal{Q}$ is the number of impersonations that the adversary attempts and $|\mathcal{D}|$ is size of a dictionary $\mathcal{D}$.

First provably secure PAKE schemes [BPR00,BMP00,MPS00] rely on the random oracle (RO) model or the ideal cipher (IC) model. Several years later, many PAKE schemes [KOY01,KOY02,GL03,JG04,CHK$^+$05,Gen08,KOY09,KV09,GK10,KV11] that are proved to be secure in the standard model (i.e., without relying on ROs or ICs) are proposed 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 at 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 trust-worthily generated.

There are some studies [GL01,NV04,BCL$^+$05] to avoid the RO model, the IC model and the CRS model. Though these schemes are proved to be secure in the plain model (i.e., without any idealized building block or setup assumption), security is only guaranteed in the case of the non-concurrent (i.e., the treatment of multiple protocol execution is limited to the case of sequential composition) or bounded concurrent setting. The only known scheme which is secure in the plain model under *concurrent* self-composition is recently proposed by Goyal et al. [GJO10]. Unfortunately, it is only theoretical scheme because tremendous rounds of communication is necessary (i.e., containing a polynomial number of rounds). Thus, our interest is to construct a concurrently secure and practical PAKE scheme without relying on ROs, ICs or a trusted setup.

Our Results. In this paper, we show a methodology to construct secure PAKE scheme without relying on the RO model, the IC model or the CRS model.

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, in this paper, we concentrate to avoid the fault of the CRS model as a milestone toward a practical PAKE scheme in the plain model. Our key idea is to adopt the *bare public-key (BPK) model* [CGGM00].

– *BPK model vs. PKI model.* The BPK model is a relaxation of the ordinary PKI model. In the PKI model, each party must prove that his public-key is validly generated according to a protocol description; thus, parties' public-keys are certified by a certification authority who must be a trusted party. In the

BPK model, each party just register their public-keys[1] in a public file before any execution of the protocol begins. There is no interactive preprocessing stage, trusted string or third party, or assumption on the asynchronicity of the network. The main difference from the PKI model is that public-keys are not certified. Thus, anyone cannot attribute an uploaded public-key to the party who registers it. In other words, any trusted authority is unnecessary in the BPK model. The BPK model is very realistic because it is possible to achieve in the real world by setting a date where the protocol begins running and closing key registration before this time. Obviously, the BPK model is more relaxed than the CRS model in necessity of a trusted setup.

This paper considers concurrently secure and practical PAKE schemes in the BPK model. First, we give a general conversion from secure schemes in the CRS model to secure schemes in the BPK model. The conversion is pretty simple and, as far as we know, applicable to all of previously known secure schemes [KOY01,KOY02] [GL03,JG04,Gen08,KOY09,KV09,GK10,KV11] in the CRS model. It needs to add two moves to the underlying PAKE scheme in order to exchange indices of parties' public-keys in the public file. Any additional computation is not required. Thus, if an underlying PAKE scheme (in the CRS model) is practical, then the converted PAKE scheme (in the BPK model) is also practical.

The proposed conversion has round overhead compared with the underlying scheme; that is, at least an additional move is necessary. We can minimize round overhead by setting in the exchange of indices into a concrete PAKE scheme. Specifically, we give a PAKE scheme based on the Groce-Katz (GK) scheme [GK10]. The GK scheme is very efficient both in communication and computational complexity. Our scheme contains same number of moves (three moves) and computational costs as the GK scheme while security can be proved in the BPK model. Therefore, our scheme achieves both practicality and security without a trusted setup.

2 Preliminaries

2.1 Password-based Authenticated Key Exchange

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 $\mathcal{D}$.[2] Parties A and B share a password pw_{AB}. We denote with Π_P^i the i^{th} instance that party P runs. Each party can concurrently execute the protocol multiple times with different instances. We suppose

[1] Note that 'public-key' means not only a key used in an asymmetric key cryptographic primitive. 'Public-key' contains all kinds of public information such that corresponding secret trapdoor information exists. For example, 'public-key' contains a CRS for a (non-interactive) commitment scheme.

[2] 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 $1/|\mathcal{D}|$.

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. 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 $\mathsf{sid}_P^i = \mathsf{sid}_{P'}^j \neq \mathit{null}$ except possibly for the final message.[3] Partnered parties must accept and conclude with the common session key.

Security Definition. Following [BPR00,GK10], 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[4] 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 $\mathsf{Reveal}(P, i)$,

[3] 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.

[4] 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 [GL06, pp.190, footnote 8] for details.

– 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:[5]

– $\mathcal{A}$ 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 $\mathsf{acc}_P^i = \mathsf{acc}'^i_P = 1$, 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 secure if for a dictionary $\mathcal{D}$ and any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that makes at most $\mathcal{Q}_{\mathsf{Send}}$ queries of* Send *to different instances the advantage $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is only negligibly larger than $\mathcal{Q}_{\mathsf{Send}}/|\mathcal{D}|$ for κ.*

2.2 Smooth Projective Hash Functions

Smooth projective hash plays a central role in our concrete scheme. This notion is introduced by Cramer and Shoup [CS04] and the GK scheme uses its variant.

Let X be a set and $L \subset X$ be a language. Loosely speaking, a 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 a projection function $F(\cdot)$ that maps a 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 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 smooth projective hashing 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 the projection key $hp = F(hk)$ and a 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 a 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 a witness is known.

[5] 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.

2.3 Bare Public-Key Model

The BPK model assumes that:

- there exists a public file $\mathcal{L}_{pk}$ that is a collection of records, each containing a public key;
- on input a security parameter, each party generates a public-key pk and stores pk in one entry of the file $\mathcal{L}_{pk}$;
- for simplicity, stored public-keys are indexed by the order of storing in the public file;
- the first interaction between two parties starts after all parties complete storing public-keys, and parties can freely show the fixed public file $\mathcal{L}_{pk}$.

3 A Simple Conversion for General Relaxation of Setup Assumption

In this section, we give a simple conversion of PAKE schemes, which relaxes the setup assumption from the CRS model to the BPK model.

3.1 Design Principle

Here, we show our strategy to avoid use of CRSs with the BPK model.

First, we observe how the CRS model contributes to prove security of previous PAKE schemes. In some step of security proofs, we often need to construct a reduction that simulates the attack environment. In the real environment, no one knows trapdoors of CRSs. Conversely, in the simulation, the simulator can know trapdoors because he can generate CRSs. Thus, the simulator successfully simulates the attack environment with the power of trapdoors.

In the BPK model, the simulator cannot generate all public-keys in the public file because an adversary may register malicious public-keys. Then, the simulator can know trapdoors of only honest parties. Fortunately, the owner of the test session must be honest as in the security definition of PAKE. Hence, the simulator can use the trapdoor of the public-key of the owner of the test session to simulate oracle queries from the adversary. Passwords of sessions other than the test session are freely set by the simulator, and so the simulation of these sessions can be done identically with the real environment.

The remaining hurdle is how two parties share information about public-keys that is used in a session. In the CRS model, public-keys can be exactly shared because the CRS is unique. But, in the BPK model, any party other than a party P who registers a public-key cannot know which is the public-key of P in the public file. Thus, we must consider how to share public-keys information. A naive idea is that we use the public-key of the initiator instead of the CRS and the initiator notifies the peer of indices of public-keys in the public file. It seems likely to work correctly because parties do not need to know trapdoors of public-keys in protocol executions. However, this naive idea is not sure to work, indeed. An adversary can interfere in the communication and substitute the index of a public-key with one of a malicious public-key that the adversary knows the trapdoor. If the responder uses such a public-key to encrypt a data including the password, then partial information of the password may be revealed by the adversary (i.e., the adversary succeeds in the off-line dictionary attack).

To avoid off-line dictionary attacks, parties must split the CRS into two parts. One part is used for the initiator and the other part is used for the responder. Specifically, a public-key will be used to derive some message (e.g., a ciphertext and a commitment) that is sent to the peer by either of the initiator and the responder. Our solution is that a party registers *only* public-keys that he uses to derive such a message. Note that, even if a public-key is commonly used both by the initiator or the responder in the protocol, parties must register the public-key respectively. Thus, the final form of the conversion contains an exchange of indices of the initiator's public-keys and indices of the responder's public-keys. The adversary cannot perform off-line dictionary attacks, because each party exactly derives messages with the correct public-key even if the adversary substitutes indices.

To summarize, our conversion adds an exchange of indices of public-keys and runs the underlying PAKE scheme with the concatenation of specified public-keys as the CRS. Though it contains round overhead, there is no computational gain for parties.

3.2 Protocol of Conversion

A high-level overview of the conversion appears in Fig. 1. Our conversion is formally described as follows:

Public Parameters. κ is a security parameter. Let $\mathcal{L}_{pk}$ be a public file that can contain $q(\kappa)$ public-keys at a maximum where q is a polynomial function. All public information (except CRSs) in the underlying PAKE scheme PAKE are contained public information in the conversion.

Protocol Execution. CRSs in PAKE are divided into two sets of public keys (pk_A and pk_B) used by the initiator A and the responder B respectively. In the setup phase, A generates pk_A and registers these to the public file $\mathcal{L}_{pk}$. B also generates pk_B and registers these to $\mathcal{L}_{pk}$. That is, $\mathsf{pk}_A, \mathsf{pk}_B \subseteq \mathcal{L}_{pk}$ holds. Note that parties do not have to keep generated secret keys.

For a key exchange session, the initiator A sends indices of pk_A in $\mathcal{L}_{pk}$ to the responder B. Upon receiving indices of pk_A, B sends indices of pk_B in $\mathcal{L}_{pk}$ to A. A and B set $CRS = \mathsf{pk}_A || \mathsf{pk}_B$ and start the protocol execution of PAKE. Finally, they output the session key.

Remark 1. If B initiates PAKE, indices of pk_B and the first move of PAKE can be simultaneously sent. That is, B plays the initiator role in PAKE. By this parallelization, the round overhead of a converted scheme becomes only one move than the underlying scheme.

3.3 Security

Theorem 1. *Assume that* PAKE *is a secure PAKE scheme in the CRS model. Then, the converted scheme in Fig. 1 is a secure PAKE scheme in the BPK model.*

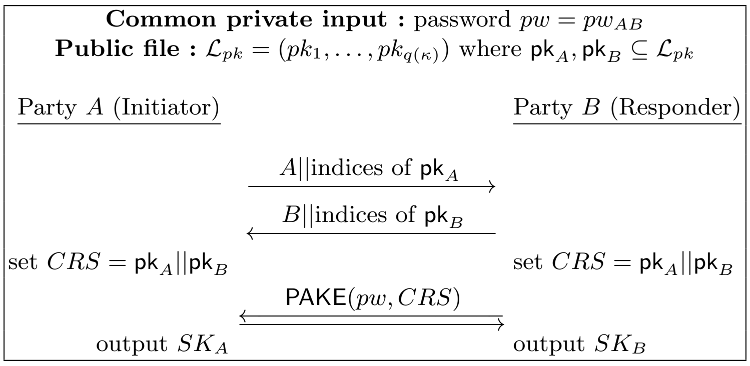
Common private input : password $pw = pw_{AB}$
Public file : $\mathcal{L}_{pk} = (pk_1, \dots, pk_{q(\kappa)})$ where $\mathsf{pk}_A, \mathsf{pk}_B \subseteq \mathcal{L}_{pk}$
Party A (Initiator)
Party B (Responder)
$A||$indices of pk_A
$B||$indices of pk_B
set $CRS = \mathsf{pk}_A||\mathsf{pk}_B$
set $CRS = \mathsf{pk}_A||\mathsf{pk}_B$
$\mathsf{PAKE}(pw, CRS)$
output SK_A
output SK_B

Fig. 1. A high-level overview of the generic conversion

Proof. The proof outline is to show that there exists a successful adversary $\mathcal{A}_{CRS}$ for PAKE if there exists a successful adversary $\mathcal{A}_{BPK}$ for the converted scheme. We construct $\mathcal{A}_{CRS}$ from $\mathcal{A}_{BPK}$ and show $\mathsf{Adv}_{\mathcal{A}_{CRS}}(\kappa)$ is non-negligible if $\mathsf{Adv}_{\mathcal{A}_{BPK}}(\kappa)$ is non-negligible. The simulation is quite simple; $\mathcal{A}_{CRS}$ registers CRSs to the public file in the simulated environment as the public-keys of parties for the test session. $\mathcal{A}_{CRS}$ performs the following steps.

Init. $\mathcal{A}_{CRS}$ receives public information for PAKE. Also, $\mathcal{A}_{CRS}$ can observe CRSs (CRS).

Setup. First, $\mathcal{A}_{CRS}$ randomly chooses a party A from N parties and an instance i from L instances as the test session. In the simulation, if the guess of the test session is failed, $\mathcal{A}_{CRS}$ aborts the simulation. We suppose that Π_A^i is the initiator in the converted scheme.

Secondly, $\mathcal{A}_{CRS}$ sets all public-keys of parties other than A with trapdoors. $\mathcal{A}_{CRS}$ maintains a public file $\mathcal{L}_{pk}$ for the simulation. $\mathcal{A}_{CRS}$ generates a set of public-keys pk_P for a party P with the corresponding set of trapdoors t_P and registers pk_P to $\mathcal{L}_{pk}$ instead of P. Also, $\mathcal{A}_{CRS}$ extracts pk_A from CRS where pk_A is the set of public-keys that A uses to derive messages in the test session and registers pk_A to $\mathcal{L}_{pk}$.

Thirdly, $\mathcal{A}_{CRS}$ sets all passwords other than the test session's. That is, $\mathcal{A}_{CRS}$ impersonates parties other than A in the password sharing phase and generates passwords other than the password pw_{AB} in the test session where B is the intended peer of A in the test session. If $\mathcal{A}_{BPK}$ impersonates a party and specifies a password, $\mathcal{A}_{CRS}$ adopts it. Note that $\mathcal{A}_{CRS}$ must not know pw_{AB}.

Finally, $\mathcal{A}_{CRS}$ inputs public information to $\mathcal{A}_{BPK}$ as it is. If $\mathcal{A}_{BPK}$ registers some public-key, $\mathcal{A}_{CRS}$ adds it into $\mathcal{L}_{pk}$.

Simulation. $\mathcal{A}_{CRS}$ simulates oracle queries by $\mathcal{A}_{BPK}$ as follows.

1. $\mathsf{Send}(P, j, \perp)$: $\mathcal{A}_{CRS}$ returns indices of pk_P in $\mathcal{L}_{pk}$.
2. $\mathsf{Send}(P, j, \bar{P}||$indices of $\mathsf{pk}_{\bar{P}})$: If P is the responder, $\mathcal{A}_{CRS}$ returns indices of pk_P in $\mathcal{L}_{pk}$. Else if $P = A$, $\mathcal{A}_{CRS}$ poses $\mathsf{Send}(A, j, \perp)$ query for the converted scheme,

receives the message of the first move of PAKE and returns it. Otherwise, $\mathcal{A}_{CRS}$ locally simulates the message of the first move of PAKE with pk_P and $pw_{P\bar{P}}$, and return it.

3. $\mathsf{Send}(P, j,$ a message of $\mathsf{PAKE})$: If $P = A$, $\mathcal{A}_{CRS}$ poses $\mathsf{Send}(A, j,$ a message of $\mathsf{PAKE})$ query for the converted scheme, receives the message of the next move of PAKE or acc^j_A, and returns it. Otherwise, $\mathcal{A}_{CRS}$ locally simulates the message of the next move of PAKE or the completion of the session (with updating internal states including acc^j_P and SK) with pk_P and $pw_{P\bar{P}}$, and return it.
4. $\mathsf{Execute}(P, j, \bar{P}, k)$: If $P = A$ or $\bar{P} = A$, $\mathcal{A}_{CRS}$ poses Send queries to A for the converted scheme, locally simulates Π^j_P or $\Pi^k_{\bar{P}}$ with pk_P, $\mathsf{pk}_{\bar{P}}$ and $pw_{P\bar{P}}$, combines received messages and simulated messages, and returns it as the transaction. Otherwise, $\mathcal{A}_{CRS}$ locally simulates messages of Π^j_P and $\Pi^k_{\bar{P}}$ with pk_P, $\mathsf{pk}_{\bar{P}}$ and $pw_{P\bar{P}}$, and returns it as the transaction.
5. $\mathsf{Reveal}(P, j)$:
   (a) If $\mathsf{acc}^j_P = 0$, $\mathcal{A}_{CRS}$ returns an error message.
   (b) Otherwise, $\mathcal{A}_{CRS}$ returns SK in the internal states of Π^j_P.
6. $\mathsf{Test}(P, j)$: If $P \neq A$ or $j \neq i$, $\mathcal{A}_{CRS}$ aborts the simulation. Otherwise, $\mathcal{A}_{CRS}$ poses $\mathsf{Test}(A, i)$ query for the converted scheme and obtains a key SK^*. $\mathcal{A}_{CRS}$ returns SK^*.
7. If $\mathcal{A}_{BPK}$ outputs a guess b', $\mathcal{A}_{CRS}$ outputs b'.

Analysis. First, we have to estimate the probability that $\mathcal{A}_{CRS}$ aborts. $\mathcal{A}_{CRS}$ aborts when $P \neq A$ or $j \neq i$ occurs for the Test query. $\mathcal{A}_{CRS}$ successfully guess the test session with probability $1/N^2L$.

Next, we show the simulation of Send queries is perfect. In the case that $P \neq A$, the simulation is identical with the real experiment because $\mathcal{A}_{CRS}$ knows the password of the session. In the other case that $P = A$, the simulation also perfect, because A's part of CRS is identical to pk_A in $\mathcal{L}_{pk}$ and $\mathcal{A}_{CRS}$ forwards Send queries to A for the converted scheme. Thus, even though $\mathcal{A}_{CRS}$ do not know pw for A, $\mathcal{A}_{CRS}$ can return the correct transaction to $\mathcal{A}_{BPK}$.

To summarize, $\mathsf{Adv}_{\mathcal{A}_{CRS}}(\kappa) \geq \frac{1}{N^2L}\mathsf{Adv}_{\mathcal{A}_{BPK}}(\kappa)$ holds. Therefore, if $\mathsf{Adv}_{\mathcal{A}_{BPK}}(\kappa)$ is non-negligible, $\mathsf{Adv}_{\mathcal{A}_{CRS}}(\kappa)$ is also non-negligible.

□

4 Round Efficient PAKE in BPK Model

In this section, we show a concrete PAKE scheme that achieves shorter communication rounds than schemes obtained from the general conversion.

4.1 Protocol of GK Scheme

First, we recall the GK scheme that is secure in the CRS model. Fig. 2 shows an overview of the GK scheme.

The GK scheme uses a chosen ciphertext secure labeled public key encryption (CCA-lPKE) $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where the message space $MS = \mathcal{D}$, PKS is the

Common private input : password $pw = pw_{AB}$
Common reference string : pk, pk'

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

Fig. 2. A high-level overview of the GK scheme

public key space and CTS is the valid ciphertext space with respect to a public key pk, and a semantically secure public key encryption (CPA-PKE) $\Sigma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ with an associated smooth projective hash function where the message space $MS' = \mathcal{D}$, PKS' is the public key space and CTS' is the valid ciphertext space with respect to a public key pk'. We define sets X, $\{L_m\}_{m\in\mathcal{D}}$ and L for Σ'. Let X be a set $\{(CT', pk', m) | pk' \in PKS'; CT' \in CTS'; m \in \mathcal{D}\}$, L_m be a set $\{(CT', pk', m) | (pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa); \mathsf{Dec}'_{sk'}(CT') = m\}$ and L be a set $\cup_{m\in\mathcal{D}} L_m$.

As described in Section 2.2, the GK scheme also uses a family of smooth projective hash functions $\mathcal{H} = \{H_{hk}\}$ such that for every hk in the key space KS, $H_{hk} : X \to \{0,1\}^k$ and $F : KS \times CTS' \times PKS' \to S$ where S is the projection key space. Formally, the smooth projective hash function the GK scheme uses is defined by a sampling algorithm that outputs $(KS, \mathcal{H}, S, F)$ such that:

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

If our general conversion is simply applied to the GK scheme, we need five moves communication to obtain a secure scheme in the BPK model.

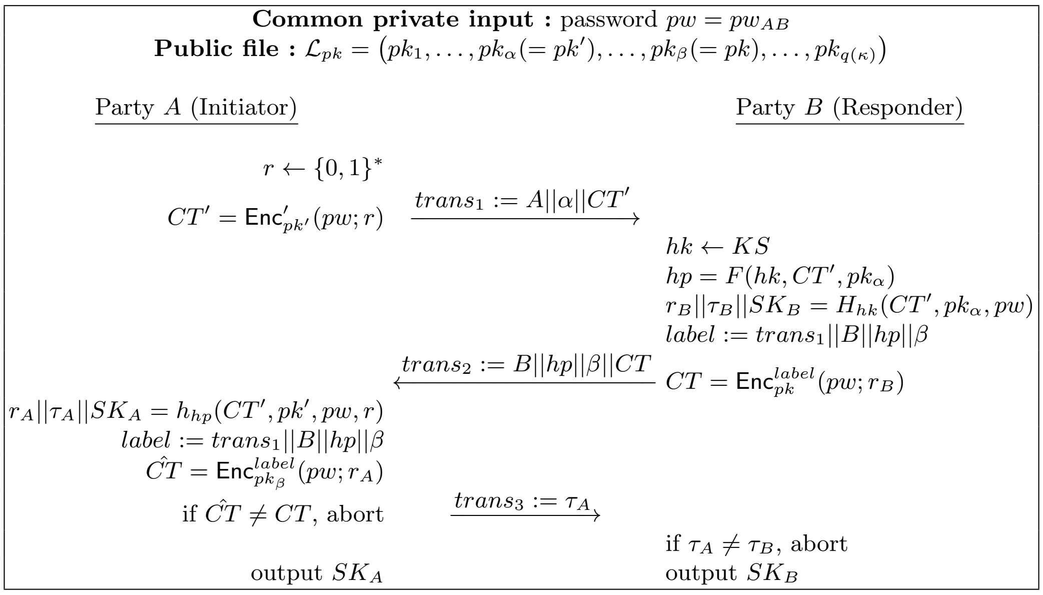
Common private input : password $pw = pw_{AB}$
Public file : $\mathcal{L}_{pk} = \big(pk_1, \ldots, pk_\alpha(= pk'), \ldots, pk_\beta(= pk), \ldots, pk_{q(\kappa)}\big)$
Party A (Initiator)
Party B (Responder)
$r \leftarrow \{0,1\}^*$
$CT' = \mathsf{Enc}'_{pk'}(pw; r)$
$\xrightarrow{trans_1 := A||\alpha||CT'}$
$hk \leftarrow KS$
$hp = F(hk, CT', pk_\alpha)$
$r_B||\tau_B||SK_B = H_{hk}(CT', pk_\alpha, pw)$
$label := trans_1||B||hp||\beta$
$\xleftarrow{trans_2 := B||hp||\beta||CT}$
$CT = \mathsf{Enc}^{label}_{pk}(pw; r_B)$
$r_A||\tau_A||SK_A = h_{hp}(CT', pk', pw, r)$
$label := trans_1||B||hp||\beta$
$\hat{CT} = \mathsf{Enc}^{label}_{pk_\beta}(pw; r_A)$
if $\hat{CT} \neq CT$, abort
$\xrightarrow{trans_3 := \tau_A}$
if $\tau_A \neq \tau_B$, abort
output SK_A
output SK_B

Fig. 3. A high-level overview of our scheme

4.2 Our Protocol

To reduce round overhead, we must include the exchange of indices of public-keys in some messages of the GK scheme. CRSs in the GK scheme can be divided into pk' for the initiator A and pk for the responder B. Thus, in our construction, A registers pk' in a public file $\mathcal{L}_{pk}$ with the index α and B registers pk in a public file $\mathcal{L}_{pk}$ with the index β. We add α to $trans_1$ and β to $trans_2$.

A high-level overview of the protocol appears in Fig. 3. Our scheme is formally described as follows:

Public Parameters. κ is a security parameter. Let $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-lPKE where the message space is $\mathcal{D}$, the public key space is PKS and the valid ciphertext space with respect to a public key pk is CTS. Let $\Sigma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a CPA-PKE with an associated smooth projective hash function where the message space is $\mathcal{D}$, the public key space is PKS' and the valid ciphertext space with respect to a public key pk' is CTS'. We define sets X, $\{L_m\}_{m\in\mathcal{D}}$ and L for Σ'. Let X be a set $\{(CT', pk', m) | pk' \in PKS'; CT' \in CTS'; m \in \mathcal{D}\}$, L_m be a set $\{(CT', pk', m) | (pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa); \mathsf{Dec}'_{sk'}(CT') = m\}$ and L be a set $\cup_{m\in\mathcal{D}} L_m$. As same as the GK scheme, we use a family of smooth projective hash functions $\mathcal{H} = \{H_{hk}\}$ such that for every hk in the key space KS, $H_{hk} : X \rightarrow \{0,1\}^\kappa$ and $F : KS \times CTS' \times PKS' \rightarrow S$ where S is the projection key space.

Protocol Execution. In the setup phase, A generates $(pk', sk') \leftarrow \mathsf{Gen}'(1^\kappa)$ and registers pk' to the public file $\mathcal{L}_{pk}$ as α-th public key. B also generates $(pk, sk) \leftarrow$

$\mathsf{Gen}(1^\kappa)$ and registers pk to the public file $\mathcal{L}_{pk}$ as β-th public key. Note that parties do not have to keep generated secret keys.

For a key exchange session, the initiator A generates a randomness $r \in \{0,1\}^*$ and computes the ciphertext $CT' = \mathsf{Enc}'_{pk'}(pw; r)$ with the password pw. A sends $A||\alpha||CT'$ to the responder B. Upon receiving $A||\alpha||CT'$, B generates a hash key $hk \leftarrow KS$, finds pk_α from $\mathcal{L}_{pk}$, and computes the projection key $hp = F(hk, CT', pk_\alpha)$ and $r_B||\tau_B||SK_B = H_{hk}(CT', pk_\alpha, pw)$. Then, B sets $label := trans_1||B||hp||\beta$ and computes the ciphertext $CT = \mathsf{Enc}^{label}_{pk}(pw; r_B)$. B sends $B||hp||\beta||CT$ to A. Upon receiving $B||hp||\beta||CT$, A finds pk_β from $\mathcal{L}_{pk}$ and computes $r_A||\tau_A||SK_A = h_{hp}(CT', pk', pw, r)$. Then, A sets $label := trans_1||B||hp||\beta$, computes the ciphertext $\hat{CT} = \mathsf{Enc}^{label}_{pk_\beta}(pw; r_A)$ 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 see the same messages, the session keys that they compute are the same. This is because the same hash value is obtained when using the hash keys (hk and hk') and when using the projection keys (hp and hp'). This implies that the correctness property holds for the protocol.

Concrete Instantiation. Σ' can be instantiated by the ElGamal encryption because it can admit a smooth projective hash function. Furthermore, a more efficient instantiation is possible by using a CPA-PKE by Boneh et al. [BBS04] based on bilinear maps. Katz and Vaikuntanathan [KV11] show that this CPA-PKE can also admit a smooth projective hash function. Σ can be instantiated by arbitrary CCA-PKE like the Cramer-Shoup encryption [CS98,CS04] and the Kurosawa-Desmedt encryption [KD04] because Σ does not need to admit a smooth projective hash function and CCA-lPKE is easily constructed from CCA-PKE.

4.3 Security

Theorem 2. *Assume that Σ' is a CPA-PKE with associated smooth projective hash functions and Σ is a CCA-lPKE. Then, our scheme in Fig. 3 is a secure PAKE scheme in the BPK model.*

The security proof of our scheme is almost same as that of the GK scheme. We proceed to prove the security through a series of hybrid experiment.

The main difference between the proof of the GK scheme and our scheme is that we must deal with multiple public-keys in the BPK model though the public-key is fixed and common for all parties in the CRS model. Thus, we must fix the test session in the earlier step of hybrid experiments. By fixing the test session, we can concentrate only the public-key corresponding to this session when we construct reductions to security of PKEs from some game transition.

The proof of Theorem 2 is shown in Appendix A.

5 Concluding Remark

We introduce a first constant round PAKE scheme which is concurrently secure without any trusted setup (i.e., in the BPK model). Our general conversion is applicable to any secure PAKE scheme in the CRS model. Also, our concrete scheme has same computational and communication complexity as the GK scheme; that is, surprisingly, our scheme relaxes the setup assumption without any overhead.

Unfortunately, our security proof technique in the BPK model is not trivially applicable to proofs in the plain model. The proof in the BPK model depends on the fact that parties' public-keys are exactly contained in the public file though the public-keys are not certified. Thus, the simulator can embed challenge public-keys in the public file. Conversely, since there is no place embed public-keys in the plain model, we must do a simulation without them. A bypass is to use some zero-knowledge protocol having witness-extended property; that is, parties send their public-keys with proofs of validity of public-keys. However, such a protocol needs large number of rounds. Therefore, concurrently secure constant round PAKE in the plain model is still hard and an open problem.

One may wonder that we do not give an extension to universally composable (UC) security as precious schemes [CHK+05,GK10,KV11]. However, Kidron and Lindell [KL11] prove that any secure two-party protocol is impossible to realize in the UC framework in the BPK model. Thus, a secure UC PAKE scheme in the BPK model is not achievable as well as impossibility in the plain model [CHK+05].

References

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO 2004*, pages 41–55, 2004.

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure Computation Without Authentication. In *CRYPTO 2005*, pages 361–377, 2005.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *IEEE S&P 1992*, pages 72–84, 1992.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In *EUROCRYPT 2000*, pages 156–171, 2000.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *EUROCRYPT 2000*, pages 139–155, 2000.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *STOC 2000*, pages 235–244, 2000.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally Composable Password-Based Key Exchange. In *EUROCRYPT 2005*, pages 404–421, 2005.

[CS98] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO 1998*, pages 13–25, 1998.

[CS04] Ronald Cramer and Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. In *SIAM Journal on Computing 33*, pages 167–226, 2004.

[Gen08] Rosario Gennaro. Faster and Shorter Password-Authenticated Key Exchange. In *TCC 2008*, pages 589–606, 2008.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-Authenticated Session-Key Generation on the Internet in the Plain Model. In *CRYPTO 2010*, pages 277–294, 2010.

[GK10] Adam Groce and Jonathan Katz. A new framework for efficient password-based authenticated key exchange. In *ACM Conference on Computer and Communications Security 2010*, pages 516–525, 2010.

[GL01] Oded Goldreich and Yehuda Lindell. Session-Key Generation Using Human Passwords Only. In *CRYPTO 2001*, pages 408–432, 2001.

[GL03] Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange. In *EUROCRYPT 2003*, pages 408–432, 2003.

[GL06] Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange. In *ACM Trans. Inf. Syst. Secur. 9(2)*, pages 181–234, 2006.

[JG04] Shaoquan Jiang and Guang Gong. Password Based Key Exchange with Mutual Authentication. In *Selected Areas in Cryptography 2004*, pages 267–279, 2004.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *CRYPTO 2004*, pages 426–442, 2004.

[KL11] Dafna Kidron and Yehuda Lindell. Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs. In *J. Cryptology 24(3)*, pages 517–544, 2011.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In *EUROCRYPT 2001*, pages 475–494, 2001.

[KOY02] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Forward Secrecy in Password-Only Key Exchange Protocols. In *SCN 2002*, pages 29–44, 2002.

[KOY09] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient and secure authenticated key exchange using weak passwords. In *J. ACM 57(1)*, pages 1–39, 2009.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices. In *ASIACRYPT 2009*, pages 636–652, 2009.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-Optimal Password-Based Authenticated Key Exchange. In *TCC 2011*, pages 293–310, 2011.

[MPS00] Philip D. MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-Authenticated Key Exchange Based on RSA. In *ASIACRYPT 2000*, pages 599–613, 2000.

[NV04] Minh-Huyen Nguyen and Salil P. Vadhan. Simpler Session-Key Generation from Short Random Passwords. In *TCC 2004*, pages 428–445, 2004.

A Proof of Theorem 2

We begin by making session keys and transcripts be independent from the password for protocol instances for which the Execute oracle is called. We then proceed to make session keys and transcripts be independent from the password for instances that receive Send oracle calls. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, all session keys are independent from the password and uniformly chosen. Thus, the adversary clearly cannot distinguish them from random. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_9$ and by $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_i)$ the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$.

The real adversarial attack is denoted by $\mathbf{H}_0$ and in this experiment all the oracles are as defined in the protocol.

Hybrid experiment $\mathbf{H}_1$: In this experiment, the experiment selects a party A and integer $i \in [1, L]$ randomly in advance. If $\mathcal{A}$ poses Test query to a session except i-th session of A, the experiment halts.

Since guess of the test session matches with $\mathcal{A}$'s choice with probability $1/N^2L$, $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_1) \geq 1/N^2L \cdot \mathsf{Adv}(\mathcal{A}, \mathbf{H}_0)$.

Hybrid experiment $\mathbf{H}_2$: In this experiment, the $\mathsf{Execute}$ oracle is modified so that we now compute $CT' = \mathsf{Enc}'_{pk'}(pw'; r_1)$ for $\mathsf{Execute}(P_1, j_1, P_2, j_2)$ query where pw' is some fake password not in the dictionary. All other transcripts are computed as the same way in $\mathbf{H}_0$ except that SK_{P_1} is set to be equal to SK_{P_2}.

Lemma 1. $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_2}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_1})| < negl(k)$.

Proof. We show that if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_1}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_0})|$ is non-negligible, we can construct an attacker $\mathcal{S}$ against Σ'. Given the public key pk^*, $\mathcal{S}$ sets $pk' = pk^*$ and chooses random passwords for each pair of parties. For the response to $\mathsf{Execute}(P_1, j_1, P_2, j_2)$ query, $\mathcal{S}$ poses the real password pw and the fake password pw' to the challenge oracle, and responds the received challenge ciphertext CT^* as CT' in $trans_1$. Then, $\mathcal{S}$ can compute the session key SK_{P_2} as the real protocol. Finally, if $\mathcal{A}$ succeeds, $\mathcal{S}$ outputs 1. Therefore, if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_2}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_1})|$ is non-negligible, $\mathcal{S}$ also breaks CPA security of Σ'. □

Hybrid experiment $\mathbf{H}_3$: In this experiment, the $\mathsf{Execute}$ oracle is modified so that we now randomly choose $r_B||\tau_B||CT'$ from $\{0,1\}^k$ instead of computing $H_{hk}(CT', pk_\alpha, pw)$ for $\mathsf{Execute}(P_1, j_1, P_2, j_2)$ query. All other transcripts are computed as the same way in $\mathbf{H}_2$.

Lemma 2. $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_3}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_2})| < negl(k)$.

Proof. Since CT' is always an encryption of the fake password pw', $(CT', pk_\alpha, pw) \notin L$. Thus, from smoothness property of the smooth projective hash function H for Σ', the output of $H_{hk}(CT', pk_\alpha, pw)$ is statistically close to uniform even conditioned hp. □

Hybrid experiment $\mathbf{H}_4$: In this experiment, the $\mathsf{Execute}$ oracle is modified so that we now compute $CT = \mathsf{Enc}_{pk}^{label}(pw')$ for $\mathsf{Execute}(P_1, j_1, P_2, j_2)$ query where pw' is some fake password not in the dictionary. All other transcripts are computed as the same way in $\mathbf{H}_3$ except that checking of CT performed by the initiator is removed.

Lemma 3. $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_4}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_3})| < negl(k)$.

Proof. We show that if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_4}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_3})|$ is non-negligible, we can construct an attacker $\mathcal{S}$ against Σ. Given the public key pk^*, $\mathcal{S}$ sets $pk = pk^*$ and chooses random passwords for each pair of parties. For the response to $\mathsf{Execute}(P_1, j_1, P_2, j_2)$ query, $\mathcal{S}$ poses pw and pw' to the challenge oracle, and responds the received challenge ciphertext CT^* as CT in $trans_2$. Session keys SK_{P_1} and SK_{P_2} are chosen randomly. Finally, if $\mathcal{A}$ succeeds, $\mathcal{S}$ outputs 1. Therefore, if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_4}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_3})|$ is non-negligible, $\mathcal{S}$ also breaks CPA security of Σ. □

Note that the session key becomes truly random and all transcripts are independent from the real passwords in $\mathbf{H}_5$ for $\mathsf{Execute}$ query.

Now, in order to proceed to show that Send oracles are also not of "too much" help to the adversary, we divide Send oracles into four different types:

– $\mathsf{Send}_0(A, i, B)$: this oracle returns the first message $trans_1$ in the protocol as the initiator upon receiving the activation of the session.
– $\mathsf{Send}_1(B, j, trans_1)$: this oracle returns the second message $trans_2$ in the protocol as the responder upon receiving $trans_1$.
– $\mathsf{Send}_2(A, i, trans_2)$: this oracle returns the third message $trans_3$ in the protocol and outputs the session key as the initiator upon receiving $trans_2$.
– $\mathsf{Send}_3(B, j, trans_3)$: this oracle updates state and outputs the session key as the responder upon receiving $trans_3$.

Hybrid experiment $\mathbf{H}_5$: We now record the secret keys sk' and sk corresponding to the public keys pk' and pk of A and B, respectively. In this experiment, the $\mathsf{Send}_2(A, i, B||hp||\beta||CT)$ oracle is modified as follows:

– If $\mathsf{pid}_A^i \neq B$, such a Send_2 query is rejected. We can simply assume that this case does not occur.
– If $B||hp||\beta||CT$ was output by a previous $\mathsf{Send}_1(B, *, A||\alpha||CT')$ query (in this case, we say that the message $B||hp||\beta||CT$ is *oracle-generated*), the experiment continues as $\mathbf{H}_4$ except that we set $\tau_A = \tau_B$ and $SK_A = SK_B$.
– If $B||hp||\beta||CT$ is not oracle-generated (in this case, we say that the message $B||hp||\beta||CT$ is *adversary-generated*), we check whether $\mathsf{Dec}_{sk}(CT) = pw$ with sk. If so, we regard the adversary successful and the experiment aborts. Otherwise, such a Send_2 query is rejected.

Lemma 4. $\mathsf{Adv}(\mathcal{A}, \mathbf{H_4}) \leq \mathsf{Adv}(\mathcal{A}, \mathbf{H_5})$.

Proof. In the case of oracle-generated inputs, to set $\tau_A = \tau_B$ and $SK_A = SK_B$ is only a syntactic rewriting of $\mathbf{H}_4$. Thus, the advantage of $\mathcal{A}$ remains unchanged. The difference between $\mathbf{H}_4$ and $\mathbf{H}_5$ occurs in the case that $B||hp||\beta||CT$ is adversary-generated and CT is an encryption of the correct password, and this case only increase the advantage of $\mathcal{A}$. □

Hybrid experiment $\mathbf{H}_6$: In this experiment, the $\mathsf{Send}_0(A, i, B)$ oracle is modified so that we now compute $CT' = \mathsf{Enc}'_{pk'}(pw')$ where pw' is some fake password not in the dictionary. All other transcripts are computed as the same way in $\mathbf{H}_5$.

Lemma 5. $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_6}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_5})| < negl(k)$.

Proof. We show that if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_6}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_5})|$ is non-negligible, we can construct an attacker $\mathcal{S}$ against Σ'. Given the public key pk^*, $\mathcal{S}$ sets $pk' = pk^*$. For the response to the Send_0 query, $\mathcal{S}$ poses the real password pw and the fake password pw' to the challenge oracle, and responds the received challenge ciphertext CT^* as CT' in $trans_1$. Note that $\mathcal{S}$ can correctly respond to Send_2 query because CT can be verified with sk without knowing randomness used to generate CT'. Finally, if $\mathcal{A}$ succeeds, $\mathcal{S}$ outputs 1. Therefore, if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_6}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_5})|$ is non-negligible, $\mathcal{S}$ also breaks CPA security of Σ'. □

Hybrid experiment $\mathbf{H}_7$: In this experiment, the $\mathsf{Send}_1(B, j, A||\alpha||CT')$ oracle is modified as follows:

- If $\mathsf{pid}_B^j \neq A$, such a Send_1 query is rejected. We can simply assume that this case does not occur.
- If $A||\alpha||CT'$ is oracle-generated, the experiment continues as $\mathbf{H}_6$.
- If $A||\alpha||CT'$ is adversary-generated, we check whether $\mathsf{Dec}'_{sk'}(CT') = pw$ with sk'. If so, we regard the adversary successful and the experiment aborts. Otherwise, such a Send_1 query is rejected.

Lemma 6. $\mathsf{Adv}(\mathcal{A}, \mathbf{H_6}) \leq \mathsf{Adv}(\mathcal{A}, \mathbf{H_7})$.

Proof. The difference between $\mathbf{H}_6$ and $\mathbf{H}_7$ occurs in the case that $A||\alpha||CT'$ is adversary-generated, and CT' is an encryption of the correct password. This case only increase the advantage of $\mathcal{A}$. □

Hybrid experiment $\mathbf{H}_8$: In this experiment, the $\mathsf{Send}_1(B, j, A||\alpha||CT')$ oracle is modified so that we now randomly choose $r_B||\tau_B||SK_B$ from $\{0,1\}^k$ instead of computing $H_{hk}(CT', pk_\alpha, pw)$. All other transcripts are computed as the same way in $\mathbf{H}_7$.

Lemma 7. $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_8}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_7})| < negl(k)$.

Proof. If $\mathsf{Dec}'_{sk'}(CT') \neq pw$, CT' is always an encryption of the fake password pw' and $(CT', pk', pw) \notin L$. Thus, from smoothness property of the smooth projective hash function H for Σ', the output of $H_{hk}(CT', pk_\alpha, pw)$ is statistically close to uniform even conditioned hp. □

Note that CT is generated with truly random r_B in $\mathbf{H}_8$.

Hybrid experiment $\mathbf{H}_9$: In this experiment, $\mathsf{Send}_1(B, j, A||\alpha||CT')$ oracle is modified so that we now compute $CT = \mathsf{Enc}_{pk}^{label}(pw')$ where pw' is some fake password not in the dictionary. All other transcripts are computed as the same way in $\mathbf{H}_8$.

Lemma 8. $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_9}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_8})| < negl(k)$.

Proof. We show that if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_9}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_8})|$ is non-negligible, we can construct an attacker $\mathcal{S}$ against Σ. Given the public key pk^*, $\mathcal{S}$ sets $pk = pk^*$. For the response to the Send_1 query, $\mathcal{S}$ poses pw and pw' to the challenge oracle, and responds the received challenge ciphertext CT^* as CT in $trans_2$. Note that $\mathcal{S}$ can correctly respond to Send_1 query because CT' can be verified with sk'. Also, $\mathcal{S}$ can correctly respond to the Send_2 query because $\mathcal{S}$ can pose CT to the decryption oracle and can verify CT. Finally, if $\mathcal{A}$ succeeds, $\mathcal{S}$ outputs 1. Therefore, if $|\mathsf{Adv}(\mathcal{A}, \mathbf{H_9}) - \mathsf{Adv}(\mathcal{A}, \mathbf{H_8})|$ is non-negligible, $\mathcal{S}$ also breaks CCA security of Σ. $\square$

Bounding the advantage in $\mathbf{H}_9$: In the experiment $\mathbf{H}_9$, $\mathcal{A}$ may succeed in the following possible ways:

1. $(A||\alpha||CT')$ is adversary-generated and $\mathsf{Dec}'_{sk'}(CT') = pw$ holds.
2. $(B||hp||\beta||CT)$ is adversary-generated and $\mathsf{Dec}_{sk}(CT) = pw$ holds.
3. $\mathcal{A}$ successfully guesses the bit b for the Test oracle.
4. $(B||\tau_B)$ is adversary-generated where $\tau_B = \tau_A$ and τ_B is not output by any instance partnered with Π^i_A.

First, the case 4 occurs with negligible probability because τ_A is chosen randomly as in $\mathbf{H}_5$ and $\mathbf{H}_8$, and is independent from $\mathcal{A}$'s view if τ_B is not output by any instance partnered with Π^i_A. Thus, Pr[Case 4 occurs] $\leq negl(k)$. Next, the case 1 and 2 occur with the success probability of on-line dictionary attacks because all passwords are independent from $\mathcal{A}$'s view in $\mathbf{H}_9$. Thus, Pr[Case 1 or 2 occur] $\leq \mathcal{Q}_{\mathsf{Send}}/|\mathcal{D}|$ where $\mathcal{Q}_{\mathsf{Send}}$ is the total number of Send queries. Finally, the case 3 occurs with probability 1/2 if the case 1 and 2 do not occur because all session keys are chosen randomly as in $\mathbf{H}_8$. Thus, Pr[Case 3 occurs$|\neg$Case 1 or 2 occur] = 1/2. Then, we have

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ succeeds in } \mathbf{H}_9] \\
&= \Pr[\text{Case 3 occurs} \wedge \text{Case 1 or 2 occur}] \\
&\qquad + \Pr[\text{Case 3 occurs} \wedge \neg\text{Case 1 or 2 occur}] + \Pr[\text{Case 4 occurs}] \\
&\leq \Pr[\text{Case 3 occurs} \wedge \neg\text{Case 1 or 2 occur}] \cdot (1 - \Pr[\text{Case 1 or 2 occur}]) \\
&\qquad + \Pr[\text{Case 1 or 2 occur}] + negl(k) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr[\text{Case 1 or 2 occur}] + negl(k) \\
&\leq \frac{1}{2} + \frac{1}{2} \cdot \frac{\mathcal{Q}_{\mathsf{Send}}}{|\mathcal{D}|} + negl(k).
\end{aligned}$$

Thus, we have

$$\mathsf{Adv}(A, \mathbf{H}_9) \leq \mathcal{Q}_{\mathsf{Send}}/|\mathcal{D}| + negl(k).$$

By combining Lemmas, this completes the proof of Theorem 2. $\square$