Round-Optimal Password-Based Group Key Exchange Protocols in the Standard Model

Jing Xu[1](✉), Xue-Xian Hu[1,2], and Zhen-Feng Zhang[1]

[1] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{xujing,huxuexian,zfzhang}@tca.iscas.ac.cn

[2] State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, Zhengzhou 450002, China

Abstract. Password-based group key exchange protocols allow group users who share only a short, low entropy password to agree on a cryptographically strong session key. One fundamental complexity measure of such protocols is its *round complexity.* In this paper, we present the first *one-round* password-based group key exchange protocol in the common random string model. Furthermore, we propose a completely new approach to remove the need for the common random string and then construct a *two-round* password-based group key exchange protocol that does not require any setup assumption. This is - to the best of our knowledge - the first password-based group key exchange protocol without trusted setup. Using indistinguishability obfuscation as main tool, both protocols are provably secure in the standard model.

Keywords: Group key exchange protocol · Password based authentication · Round complexity · Indistinguishability obfuscation

1 Introduction

Password-based authenticated key exchange (PAKE) protocols [1] allow users who share only a short, low-entropy password to agree on a cryptographically strong session key. PAKE protocols are fascinating from a theoretical perspective, as they can be viewed as a means of "bootstrapping" a common cryptographic key from the (essentially) minimal setup assumption of a short, shared secret. PAKE protocols are also important in practice, since passwords are perhaps the most common and widely-used means of authentication. In this paper, we consider PAKE protocols in the group setting where the number of users involved in the computation of a common session key can be large.

This work was supported by the National Grand Fundamental Research (973) Program of China under Grant 2013CB338003, China Postdoctoral Science Foundation under Grant 2014M552524, and the National Natural Science Foundation of China (NSFC) under Grants 61170279, U1536205 and 61170278.

T. Malkin et al. (Eds.): ACNS 2015, LNCS 9092, pp. 42–61, 2015.
DOI: 10.1007/978-3-319-28166-7_3

The difficulty in designing password-based protocols is to prevent *off-line* dictionary attacks whereby an eavesdropping adversary exhaustively enumerates passwords, attempting to match the correct password to the eavesdropped session. However, the adversary can always correctly determine the correct password via an *on-line* dictionary attack in which the adversary tries to impersonate one of the parties using each possible password. Although an *on-line* dictionary attack is not avoidable, the damage it may cause can be mitigated by other means such as limiting the number of failed login attempts. Roughly, a secure password-based protocol guarantees that an exhaustive *on-line* dictionary attack is the "best" possible strategy for an adversary.

1.1 Related Work

Group Key Exchange Protocols. Bresson et al. [2] introduced a formal security model for group key exchange protocols and proposed the first provably secure protocol for this setting. Their protocol use a ring structure for the communication, in which each user has to wait for the message from his predecessor before producing his own. Unfortunately, the nature of their communication structure makes their protocols quite impractical for large groups due to the number of rounds of communication linear in the number of group users. Later, Burmester and Desmedt [3,4] proposed a more efficient and practical group key exchange protocol, in which the number of rounds of communication is constant. Their protocol has been formally analyzed by Katz and Yung [5], who also proposed the first constant round and fully scalable authenticated group key exchange protocol which is provably secure in the standard model. Recently, Boneh and Zhandry [6] constructed the first multiparty non-interactive key exchange protocol requiring no trusted setup, and gave the formal security proof in the static and semi-static models.

Password-Based Group Key Exchange Protocols. Adding password authentication services to a group key exchange protocol is not trivial since redundancy in the flows of the protocol can open the door to password dictionary attacks. Bresson et al. [7] proposed the first solution to the group Diffie-Hellman key exchange problem in the password-based scenario. However, their protocol has a total number of rounds which is linear in the number of group users and their security analysis requires ideal models, which is impractical for large groups. Later, two different password-based versions [8,9] of Burmester-Desmedt protocol were proposed, and unfortunately, both of them are not secure [10]. Also, Abdalla et al. [10] demonstrated the first password-based group key exchange protocol in a constant number of rounds. Their protocol is provably secure in the random oracle and ideal cipher models.

To date, there are only a few general approaches for constructing password-based group key exchange protocols in the standard model (i.e., without random oracles). Abdalla and Pointcheval [11] constructed the first such protocol with a proof of security in the standard model. Their protocol combines smooth projective hash function with the construction of Burmester and Desmedt [3,4] and includes only 5 rounds communication, but requires a common reference string model. Later, Abdalla et al. [12] presented a compiler, that transforms any provably secure (password-based) two-party key exchange protocol into a provably secure (password-based) group key exchange protocol with two more rounds of communication. Their compiler uses non-interactive and non-malleable commitment schemes as main technical tools, also requires a common reference string model.

1.2 Technical Contributions

Round complexity is a central measure of efficiency for any interactive protocol. In this paper, our main goal is to further improve bounds on the round complexity of password-based group key exchange protocol.

Towards this goal, we propose the first *one-round* password-based group key exchange protocol which is provably secure in the standard model. Our main tool is indistinguishability obfuscation, for which a candidate construction was recently proposed by Garg et al. [14]. The essential idea is the following: the public parameter consists an obfuscated program for a pseudorandom function PRF which requires knowledge of the password pw to operate, so that each user in the group can independently evaluate the obfuscated program to obtain the output session key. To prevent the *off-line* dictionary attack, we require the random value r_i used for generating the ciphertext c_i also as input of the obfuscated program.

Our second contribution is *two-round* password-based group key exchange protocol without any setup. The existing constructions require a trusted setup to publish public parameters, which means whoever generates the parameters can obtain all group users' passwords and compute the agreed session key. However, this may be less appealing than the "plain" model where there is no additional setup. Motivated by this observation, we propose a completely new approach to password-based group key exchange protocol with no trusted setup. The resulting scheme is the first secure password-based group key exchange protocol which does not rely on a random oracle or a setup, only requires two rounds of communication. Our central challenge is how to create a way to let each group user run setup for himself securely. In fact, at a first glance, it seems that letting each user publish an obfuscated program might fully resolve this problem. However, such an approach fails because a potentially malicious program can be replaced by an adversary. Specifically, an adversary may publish a malicious program that simply outputs the input password. To prevent such attacks, we extend the Burmester-Desmedt protocol framework [3,4] to the password setting, where the Diffie-Hellman key exchanges are replaced by indistinguishability obfuscation, and let each user generate two obfuscated programs. The first obfuscated program is used to obtain other users' random value, and the second program is used to generate the shared key with the user's neighbors, where the output of the first program is only as the input of the second program. Moreover, each user's partial message broadcasted for computing the session key is generated by his own program and cannot be replaced. Thus, even if the adversary replace some programs, any password information will not be disclosed.

1.3 Outline of the Paper

The rest of this paper is organized as follows. Section 2 recalls the security model usually used for password-based group key exchange protocol, and Sect. 3 recalls the definition of different cryptographic primitives essential for our study. We then propose two round-optimal constructions for password-based group key exchange protocol in Sects. 4 and 5, respectively. Sections 6 concludes.

2 Password-Based Group Key Exchange

In this section, we briefly recall the formal security model for password-based group key exchange protocols as presented in [10] (which is based on the model by Bresson [13]).

In a password-based group key exchange protocol, we assume for simplicity a fixed, polynomial-size set $\mathcal{U} = \{U_1, \ldots, U_l\}$ of potential users. Each user $U \in \mathcal{U}$ may belong to several subgroup $\mathcal{G} \subseteq \mathcal{U}$, each of which has a unique password $pw_{\mathcal{G}}$ associated to it. The password $pw_{\mathcal{G}}$ is known to all the users $U_i \in \mathcal{G}$ wishing to establish a common session key.

Let $U^{\langle i \rangle}$ denote the *i-th* instance of a participant U and b be a bit chosen uniformly at random. During the execution of the protocol, an adversary $\mathcal{A}$ could interact with protocol participants via several oracle queries, which model adversary's possible attacks in the real execution. All possible oracle queries are listed in the following:

- $Execute(U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the user instances $U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle}$. It returns the messages that were exchanged during an honest execution of the protocol.
- $Send(U^{\langle i \rangle}, m)$: This oracle query is used to simulate active attacks, in which the adversary may tamper with the message being sent over the public channel. It returns the message that the user instance $U^{\langle i \rangle}$ would generate upon receipt of message m.
- $Reveal(U^{\langle i \rangle})$: This query models the possibility that an adversary gets session keys. It returns to the adversary the session key of the user instance $U^{\langle i \rangle}$.
- $Test(U^{\langle i \rangle})$: This query tries to capture the adversary's ability to tell apart a real session key from a random one. It returns the session key for instance $U^{\langle i \rangle}$ if $b = 1$ or a random number of the same size if $b = 0$. This query is called only once.

Besides the above oracle queries, some terminologies are defined as follows.

- **Partnering:** Let the session identifier sid^i of a user instance $U^{\langle i \rangle}$ be a function of all the messages sent and received by $U^{\langle i \rangle}$ as specified by the protocol. Let the partner identifier pid^i of a user instance $U^{\langle i \rangle}$ be the set of all participants with whom $U^{\langle i \rangle}$ wishes to establish a common session key. Two instances $U_1^{\langle i_1 \rangle}$ and $U_2^{\langle i_2 \rangle}$ are said to be partnered if and only if $\mathrm{pid}_1^{i_1} = \mathrm{pid}_2^{i_2}$ and $\mathrm{sid}_1^{i_1} = \mathrm{sid}_2^{i_2}$.

- **Freshness:** We say an instance $U^{\langle i \rangle}$ is *fresh* if the following conditions hold: (1) $U^{\langle i \rangle}$ has accepted the protocol and generated a valid session key; (2) No *Reveal* queries have been made to $U^{\langle i \rangle}$ or to any of its partners.

Correctness. The correctness of password-based group key exchange protocol requires that, whenever two instances $U_1^{\langle i_1 \rangle}$ and $U_2^{\langle i_2 \rangle}$ are partnered and have accepted, both instances should hold the same non-null session key.

Security. For any adversary $\mathcal{A}$, let $Succ(\mathcal{A})$ be the event that $\mathcal{A}$ makes a single $Test$ query directed to some fresh instance $U^{\langle i \rangle}$ at the end of a protocol P and correctly guesses the bit b used in the $Test$ query. Let $\mathcal{D}$ be the user's password dictionary (i.e., the set of all possible candidate passwords). The advantage of $\mathcal{A}$ in violating the semantic security of the protocol P is defined as:

$$\mathrm{Adv}_{P,\mathcal{D}}(\mathcal{A}) = |2Pr[Succ(\mathcal{A})] - 1|.$$

Definition 1 (Security). A password-based group key exchange protocol P is said to be secure if for every dictionary $\mathcal{D}$ and every (non-uniform) polynomial-time adversary $\mathcal{A}$,

$$\mathrm{Adv}_{P,\mathcal{D}}(\mathcal{A}) < \mathcal{O}(q_s)/|\mathcal{D}| + negl(\lambda),$$

where q_s is the number of *Send* oracle queries made by the adversary to different protocol instances and λ is a security parameter.

3 Preliminaries

In this section we start by briefly recalling the definition of different cryptographic primitives essential for our study. Let $x \leftarrow S$ denote a uniformly random element drawn from the set S.

3.1 Indistinguishability Obfuscation

We will start by recalling the notion of indistinguishability obfuscation (iO) recently realized in [14] using candidate multilinear maps [15].

Definition 2 (Indistinguishability Obfuscation). An *indistinguishability obfuscator* iO for a circuit class $\mathcal{C}_\lambda$ is a PPT uniform algorithm satisfying the following conditions:

- iO(λ, C) preserves the functionality of C. That is, for any $C \in \mathcal{C}_\lambda$, if we compute C' =iO(λ, C), then $C'(x) = C(x)$ for all inputs x.
- For any λ and any two circuits $C_0, C_1 \in \mathcal{C}_\lambda$ with the same functionality, the circuits iO(λ, C_0) and iO(λ, C_1) are indistinguishable. More precisely, for all pairs of PPT adversaries (Samp, D) there exists a negligible function α such that, if

$$\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \tau) \leftarrow \mathrm{Samp}(\lambda)] > 1 - \alpha(\lambda)$$

then

$$|\Pr[\mathrm{D}(\tau, \mathrm{iO}(\lambda, C_0)) = 1] - \Pr[\mathrm{D}(\tau, \mathrm{iO}(\lambda, C_1)) = 1]| < \alpha(\lambda)$$

In this paper, we will make use of such indistinguishability obfuscators for all polynomial-size circuits:

Definition 3 (Indistinguishability Obfuscation for P/*poly*). A uniform PPT machine iO is called an indistinguishability obfuscator for P/*poly* if the following holds: Let $\mathcal{C}_\lambda$ be the class of circuits of size at most λ. Then iO is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}$.

3.2 Constrained Pseudorandom Functions

A pseudorandom function (PRF) [16] is a function PRF: $\mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ where PRF$(k, \cdot)$is indistinguishable from a random function for a randomly chosen key k. Following Boneh and Waters [17], we recall the definition of constrained pseudorandom function.

Definition 4 (Constrained Pseudorandom Function). A PRF F: $\mathcal{K}\times\mathcal{X} \rightarrow \mathcal{Y}$ is said to be *constrained* with respect to a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if there is an additional key space $\mathcal{K}_\mathcal{C}$ and two additional algorithms:

- F.constrain(k, S): On input a PRF key $k \in \mathcal{K}$ and the description of a set S$\in \mathcal{S}$ (so that S $\subseteq \mathcal{X}$), the algorithm outputs a constrained key $k_S \in \mathcal{K}_\mathcal{C}$.
- F.eval(k_S, x): On input $k_S \in \mathcal{K}_\mathcal{C}$ and $x \in \mathcal{X}$, the algorithm outputs

$$F.eval(k_S, x) = \begin{cases} F(k, x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

For ease of notation, we write $F(k_S, x)$ to represent $F.eval(k_S, x)$.

Security. Intuitively, we require that even after obtaining several constrained keys, no polynomial time adversary can distinguish a truly random string from the PRF evaluation at a point not queried. This intuition can be formalized by the following security game between a challenger and an adversary $\mathcal{A}$.

Let F: $\mathcal{K}\times\mathcal{X} \rightarrow \mathcal{Y}$ be a constrained PRF with respect to a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$. The security game consists of three phases:

Setup Phase. The challenger chooses a random key $K \leftarrow \mathcal{K}$ and a random bit $b \leftarrow \{0, 1\}$.

Query Phase. In this phase, $\mathcal{A}$ is allowed to ask for the following queries:
- Evaluation Query: On input $x \in \mathcal{X}$, it returns $F(K, x)$.
- Key Query: On input $S \in \mathcal{S}$, it returns F.constrain(K, S).
- Challenge Query: $\mathcal{A}$ sends $x \in \mathcal{X}$ as a challenge query. If $b = 0$, the challenger outputs $F(K, x)$. Else, the challenger outputs a random element $y \leftarrow \mathcal{Y}$.

Guess Phase. $\mathcal{A}$ outputs a guess b' of b.

Let $E \subseteq \mathcal{X}$ be the set of evaluation queries, $C \subseteq \mathcal{S}$ be the set of constrained key queries and $Z \subseteq \mathcal{X}$ the set of challenge queries. $\mathcal{A}$ wins if $b = b'$ and $E \bigcap Z = \phi$ and $C \bigcap Z = \phi$. The advantage of $\mathcal{A}$ is defined to be $\text{Adv}_\mathcal{A}^F(\lambda) = |\Pr[\mathcal{A}\text{wins}] - 1/2|$.

Definition 5. The PRF F is a secure constrained PRF with respect to $\mathcal{S}$ if for all probabilistic polynomial time adversaries $\mathcal{A}$, $\text{Adv}_\mathcal{A}^F(\lambda)$ is negligible in λ.

3.3 CCA Secure Encryption

Definition 6 (Public-Key Encryption). A public-key encryption scheme Σ consist of three algorithms:

- Gen: (randomized) key generation algorithm. It outputs a pair (pk, sk) consisting of a public key and a secret key, respectively.
- Enc: (randomized) encryption algorithm. It outputs a ciphertext $c = Enc_{pk}(m)$ for any message m and a valid public key pk.
- Dec: deterministic decryption algorithm. It outputs $m = Dec_{sk}(c)$ or $\perp = Dec_{sk}(c)$ for a ciphertext c and a secret key sk.

In order to make the randomness used by Enc explicit, we write $Enc_{pk}(m; r)$ to highlight the fact that random coins r are used to encrypt the message m.

Perfect Correctness. We say that the encryption scheme has perfect correctness if for overwhelming fraction of the randomness used by the key generation algorithm, for all messages we have $\Pr[Dec_{sk}(Enc_{pk}(m)) = m] = 1$.

CCA Security [18]**.** The CCA security of the $\Sigma = (\text{Gen}; \text{Enc}; \text{Dec})$ is defined via the following security game between a challenger and an adversary $\mathcal{A}$:

1. The challenger generates $(pk; sk) \leftarrow Gen(1^\lambda)$ and $b \leftarrow \{0, 1\}$, and gives pk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ asks decryption queries c, which are answered with the message $Dec_{sk}(c)$.
3. The adversary $\mathcal{A}$ inputs (m_0, m_1) with $|m_0| = |m_1|$ to the challenger, and receives a challenge ciphertext $c^* = Enc_{pk}(m_b)$.
4. The adversary $\mathcal{A}$ asks further decryption queries $c \neq c^*$, which are answered with the message $Dec_{sk}(c)$.
5. The adversary $\mathcal{A}$ outputs a bit b', and wins the game if $b' = b$.

We say that a PKE scheme Σ is CCA secure if for all (non-uniform) probabilistic polynomial time adversaries $\mathcal{A}$, $|\Pr[b' = b] - 1/2|$ is negligible.

4 One-Round Password-Based Group Key Exchange Protocol

In this section we present our construction of a one-round password-based group key exchange protocol. The idea is the following: each user broadcasts a ciphertext c_i of the password pw using random r_i. In the setup phase, a key K is chosen for a constrained pseudorandom function PRF. The shared session key will be the function PRF evaluated at the concatenation of the ciphertexts c_i and pw. To allow each user to compute the session key, the setup will publish an obfuscated program for PRF which requires knowledge of the password pw to operate. However, the adversary may obtain the obfuscated program for PRF and then mount an *off-line* dictionary attack, that is, the adversary guesses password pw^*

and inputs it to the obfuscated program. By observing whether the program outputs $\perp$, the adversary can find the correct password. Therefore, besides the password pw, the random r_i is also required as input of the obfuscated program. In this way, all users can compute the session key, but anyone else without the password, will therefore be unable to compute the session key.

A formal description appears in Fig. 1. The correctness is trivial by inspection. For security, we have the following theorem.

Protocol I

Consider an execution of the protocol among users $\mathrm{U}_1, \cdots, \mathrm{U}_n$ wishing to establish a common session key and let pw be their joint password chosen uniformly at random from a dictionary **Dict** of size N. Let $\Sigma = (\mathrm{Gen}; \mathrm{Enc}; \mathrm{Dec})$ be a public-key encryption scheme and iO be a program indistinguishability obfuscator.

Setup: Run the key generation algorithm Gen on input 1^k, where $k \in \mathbb{N}$ is a security parameter, to obtain a pair (pk, sk) of public and secret keys (i.e., $(pk, sk) \leftarrow \mathbf{Gen}(1^k)$). Choose a random key K to obtain an instance of a pseudorandom function PRF. Build the program P_{PGKE} in Figure 2, and then output pk and $P_{iO} = \mathrm{iO}\ (P_{PGKE})$ as the public parameters.

Round 1: Each user U_i proceeds as:

1. Choose r_i randomly, encrypt the password pw using r_i with respect to the public key pk, and generate the ciphertext $c_i = \mathbf{Enc}_{pk}(pw; r_i)$.
2. Broadcast c_i.

Key Generation: Each user U_i runs P_{iO} on $(c_1, c_2, \cdots, c_n, pw, i, r_i)$ to obtain the session key SK or $\perp$.

Fig. 1. An honest execution of the password-based group key exchange protocol

Theorem 1. If Σ is a CCA-secure public-key encryption scheme, PRF a secure constrained PRF, and iO a secure indistinguishability obfuscator, then the protocol in Fig. 1 is a secure password-based group key exchange protocol.

Proof. Fix a PPT adversary $\mathcal{A}$ attacking the password-based group key exchange protocol. We use a hybrid argument to bound the advantage of $\mathcal{A}$. Let $\mathbf{Hyb}_0$ represent the initial experiment, in which $\mathcal{A}$ interacts with the real protocol as defined in Sect. 2. We define a sequence of experiments $\mathbf{Hyb}_1, \ldots$, $\mathbf{Hyb}_5$, and denote the advantage of adversary $\mathcal{A}$ in experiment $\mathbf{Hyb}_i$ as:

$$\mathrm{Adv}_i(\mathcal{A}) \overset{\mathrm{def}}{=} 2 \cdot \Pr[\mathcal{A} \text{ succeeds in } \mathbf{Hyb}_i] - 1.$$

We bound the difference between the adversary's advantage in successive experiments, and then bound the adversary's advantage in the final experiment.

```
Inputs: $c_1, c_2, \cdots, c_n$, password $pw$, $i, r_i$
Constants: PRF key K, the public key $pk$
    1. If $c_i \neq Enc_{pk}(pw; r_i)$, output $\perp$
    2. Otherwise, output PRF$(K, c_1, c_2, \cdots, c_n, pw)$
```

Fig. 2. The program P_{PGKE}

Finally, combining all the above results gives the desired bound on $\text{Adv}_0(\mathcal{A})$, the adversary's advantage when attacking the real protocol.

Experiment $\mathbf{Hyb}_1$. In this experiment, whenever a session key is needed to be computed by an honest simulated user instance $U^{\langle i \rangle}$, we directly compute it as $sk_U^{\langle i \rangle} = PRF(K, c_1, \cdots, c_n, pw_i)$ instead of by calling the obfuscated program $P_{iO}(c_1, \cdots, c_n, pw_i, i, r_i)$.

Lemma 1. *$Adv_0(\mathcal{A}) = Adv_1(\mathcal{A})$.*

Proof. Notice that, for an honest simulated instance, the verification procedure $c_i = \mathbf{Enc}_{pk}(pw; r_i)$ in program P_{PGKE} will always holds. Therefore, this verification step could be omitted without changing the adversary's view and advantage.

Experiment $\mathbf{Hyb}_2$. For each honest simulated user instance $U_i^{\langle s \rangle}$, which is involved in either an **Execute** or a **Send** query, we compute $c_i = \mathbf{Enc}_{pk}(pw_0; r_i)$ instead of $c_i = \mathbf{Enc}_{pk}(pw_i; r_i)$, where pw_0 represents some *dummy password* not in the dictionary **Dict** but in the plaintext space of the encryption scheme Σ.

Lemma 2. *$|Adv_1(\mathcal{A}) - Adv_2(\mathcal{A})| < negl(\lambda)$.*

Proof. First note that, with respect to the honest simulated users, the verification procedure in program P_{PGKE} has been removed in the last experiment. Denote by $q_{es} = q_{exe} + q_{send}$. We define $\mathbf{Hyb}_1^{(\eta)}$ $(0 \leq \eta \leq n \cdot q_{es})$ to be a sequence of hybrid variants of experiment $\mathbf{Hyb}_1$ such that, for every $\eta = n \cdot \xi + \gamma, 0 \leq \xi < q_{es}, 0 \leq \gamma \leq n$, the first ξ **Execute** or **Send** queries are answered according to experiment $\mathbf{Hyb}_2$, the last $q_{es} - \xi - 1$ queries are replied the same as in experiment $\mathbf{Hyb}_1$; when the $(\xi + 1)$-th **Execute** or **Send** oracle is asked, the first γ ciphertexts of $(c_1, c_2, \cdots, c_n)$ are computed according to experiment $\mathbf{Hyb}_2$ and the rest $n - \gamma$ ciphertexts are treated the same as in experiment $\mathbf{Hyb}_1$. As one can easily verify, the hybrids $\mathbf{Hyb}_1^{(0)}$ and $\mathbf{Hyb}_1^{(n \cdot q_{es})}$ are equivalent to the experiments $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$, respectively.

In such case, if there is an adversary $\mathcal{A}$ whose advantage gap between $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are non-negligible in security parameter, there would exist an η such that the adversary's advantage gap between $\mathbf{Hyb}_1^{(\eta-1)}$ and $\mathbf{Hyb}_1^{(\eta)}$ are non-negligible. Then, we would be able to build an adversary $\mathcal{B}$ violating the CPA security of the encryption scheme Σ with non-negligible advantage from the adversary $\mathcal{A}$ as follows.

Upon receiving the public key pk of the encryptions scheme Σ from his challenger, the adversary $\mathcal{B}$ initializes the public parameters for the group key exchange protocol. It selects a random $K \in \mathcal{K}$, chooses password pw_i for every users $U_i \in \mathbf{U}$, and picks a bit $b \in \{0,1\}$ for answering the **Test** oracle. Then, for $\eta = n \cdot \xi + \gamma$, it simulates the **Execute**, **Send**, **Reveal** and **Test** oracles exactly as in hybrid $\mathbf{Hyb}_1^{(\eta)}$ except for the γ-th ciphertext of $(c_1, c_2, \cdots, c_n)$ in the $(\xi+1)$-th **Execute** or **Send** oracle. In this case, the adversary $\mathcal{B}$ gives pw and pw_0 to its challenger to obtain a challenging ciphertext c_i^* that is either $\mathbf{Enc}_{pk}(pw)$ or $\mathbf{Enc}_{pk}(pw_0)$, and it uses this ciphertext in place of c_i to answer the $(\xi+1)$-th *Execute* query. At last, $\mathcal{B}$ checks whether $\mathcal{A}$ succeeds or not. If $\mathcal{A}$ succeeds in this hybrid game, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

The distinguishing advantage of $\mathcal{B}$ is exactly equal to the adversary $\mathcal{A}$'s advantage gap between $\mathbf{Hyb}_1^{(\eta-1)}$ and $\mathbf{Hyb}_1^{(\eta)}$. Then, the lemma follows by notice that the encryption scheme Σ is a CPA secure one.

Experiment $\mathbf{Hyb}_3$. In this experiment, we first let the simulator record the corresponding decryption key sk when generating the public key pk. Then, we define the following event:

`PwdGuess` : During the experiment, an honest user instance $U^{\langle i \rangle}$ with password pw_i is activated by some input message $(c_1, \cdots, c_{i-1}, \perp, c_{i+1}, \cdots, c_n)$, such that there exists some index $j \in [n]$ and $j \neq i$ satisfying $Dec_{sk}(c_j) = pw_i$.
Whenever the event `PwdGuess` happens, the adversary is declared successful and the experiment ends; Otherwise, the experiment is simulated in the same way as in the last experiment.

Lemma 3. *$Adv_2(\mathcal{A}) \leq Adv_3(\mathcal{A})$.*

Proof. Even when the event `PwdGuess` happens in experiment $\mathbf{Hyb}_2$, the adversary would not necessarily succeed in this case. As a result, the modification made in experiment $\mathbf{Hyb}_3$ introduces a new way for the adversary to succeed.

Experiment $\mathbf{Hyb}_4$. Replace the $PRF(\cdot)$ in P_{PGKE} by an constrained pseudorandom function $PRF^C(\cdot)$, arriving at the program P'_{PGKE} given in Fig. 3. The constrained set C is defined as $C = \mathcal{M}^n \times \mathbf{Dict} \setminus \{(c_1, c_2, \cdots, c_n, pw) : pw \in \mathbf{Dict}, \forall i \in [n], c_i \notin Enc_{pk}(pw), \text{ and } \exists j \in [n], c_j \in Enc_{pk}(pw_0)\}$.

Lemma 4. *$|Adv_3(\mathcal{A}) - Adv_4(\mathcal{A})| < negl(\lambda)$.*

Proof. Because the dummy password pw_0 in experiment $\mathbf{Hyb}_2$ is derived from the plaintext space randomly, then with overwhelming probability (in fact, bigger than $(1 - n/|\mathbf{Dict}| \cdot 2^\lambda)$), the input to the pseudorandom function PRF in program P_{PGKE} will belong to the set C defined as above. Therefore, with overwhelming probability, the modified program P'_{PGKE} has the same functionality with the original program P_{PGKE}. The security of the indistinguishable obfuscator iO implies that the adversary's advantage gap between the experiment $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_3$ is no more than the probability that P'_{PGKE} differs from P_{PGKE}, thus is negligible. The lemma's result follows.

```
Inputs: $c_1, c_2, \cdots, c_n$, password $pw$, $i, r_i$
Constants: Constrained PRF key $K_{\mathcal{C}}$, the public key $pk$
    1. If $c_i \neq Enc_{pk}(pw; r_i)$, output $\perp$
    2. Otherwise, output $\mathrm{PRF}^{\mathcal{C}}(K_{\mathcal{C}}, c_1, c_2, \cdots, c_n, pw)$
```

Fig. 3. The program P'_{PGKE}

Experiment $\mathbf{Hyb}_5$. Recall that only the situation when event PwdGuess does not happen (i.e., $(c_1, c_2, \cdots, c_n, pw_i) \notin C$) is considered since $\mathbf{Hyb}_3$. Then, when a session key is needed to be computed by an honest user instance, we evaluate it as $sk_U^{\langle i \rangle} \leftarrow_R \{0,1\}^\lambda$ instead of $sk_U^{\langle i \rangle} = PRF(K, c_1, c_2, \cdots, c_n, pw_i)$.

Lemma 5. $|Adv_4(\mathcal{A}) - Adv_5(\mathcal{A})| < negl(\lambda)$.

Proof. We reduce the problem of distinguishing the experiments $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_5$ to the security of constrained PRF presented above. Assume that $\mathcal{A}$ is a protocol adversary that is defined as in $\mathbf{Hyb}_4$. We construct a PRF adversary $\mathcal{B}$ against the security of the constrained pseudorandom function PRF as follows. When the adversary $\mathcal{B}$ receives the constrained key k_C of PRF with respected to the constrained set C, it simulates the protocol execution for $\mathcal{A}$ as in $\mathbf{Hyb}_4$. Note that the program P'_{PGKE} is used in this experiment and all the queries asked by $\mathcal{A}$ could be answered with overwhelming probability by utilizing it. However, when a honest simulated user instance needs to generate a session key, $\mathcal{B}$ asks its own challenge query, getting back either a value computed from the function PRF or a value selected uniformly at random, and used it as the session key. Finally, $\mathcal{B}$ checks whether $\mathcal{A}$ succeeds or not. If $\mathcal{A}$ succeeds, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

It follows that the advantage of $\mathcal{B}$ is exactly equal to the adversary $\mathcal{A}$'s advantage gap between $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_5$.

Bounding the Advantage in $\mathbf{Hyb}_5$. Consider the different ways for the adversary to succeed in Hyb_5:

1. The Event PwdGuess happens;
2. The adversary successfully guesses the bit used by the **Test** oracle.

Since all oracle instances are simulated using dummy passwords, the adversary's view is independent of the passwords that are chosen for each group of users. Then we have Pr[PwdGuess]$\leq Q(\lambda)/D_\lambda$, where $Q(\lambda)$ denotes the number of *Send* oracle queries and D_λ denotes the dictionary size. Conditioned on PwdGuess not occurring, the adversary can succeed only in case 2. But since all session keys defined throughout the experiment are chosen uniformly and independently at random, the probability of success in this case is exactly 1/2. Then, we have

$$\begin{aligned} Pr[Success] &\leq Pr[\texttt{PwdGuess}] + Pr[Success|\overline{\texttt{PwdGuess}}] \cdot (1 - Pr[\texttt{PwdGuess}]) \\ &= \frac{1}{2} + \frac{1}{2} \cdot Pr[\texttt{PwdGuess}] \\ &\leq \frac{1}{2} + \frac{Q(\lambda)}{2 \cdot D_\lambda} \end{aligned}$$

and so $\mathrm{Adv}_5(\mathcal{A}) \leq \frac{Q(\lambda)}{D_\lambda}$. Taken together, Lemmas 1–5 imply that $\mathrm{Adv}_0(\mathcal{A}) \leq \frac{Q(\lambda)}{D_\lambda} + negl(\lambda)$ as desired. □

5 Two-Round Password-Based Group Key Exchange Protocol with No Setup

In this section, we show how to remove the trusted setup and common reference string (CRS) from the password-based group key exchange protocol in the previous section. Intuitively, letting each user publish an obfuscated program and run setup for himself might fully resolve this problem. However, unlike the protocol I in the previous section, the obfuscated programs generated by group users are susceptible to a "replace" attack - i.e., the adversary may replace the program with a malicious program that simply outputs the input password. Then, the message broadcasted by an honest user may disclose the information about password. With this message, the adversary can mount an *off-line* dictionary attack and obtain the password, thus breaking the security of protocol. We believe that such attacks are the principle reason that the existing constructions require a trusted setup to publish public parameters.

To overcome the above difficulties, we present a new methodology for constructing password-based group key exchange protocol with no setup. The basic idea of our construction follows the Burmester-Desmedt [3,4] construction where the Diffie-Hellman key exchanges are replaced by indistinguishability obfuscation. As in the Burmester-Desmedt protocol, our protocol assumes a ring structure for the users so that we can refer to the predecessor and successor of a user. Each user in the group will run setup for himself and his neighbors (predecessor and successor), and generate two obfuscated programs.

- The first obfuscated program P^{iO-dec} is used to obtain other users' random value s. This program takes as input "ciphertext" c and user password pw, and outputs the corresponding "plaintext" s.
- The second obfuscated program P^{iO} is used to generate the shared key with the user's neighbors. This program takes as input two random value s_i and s_{i+1} generated by the user U_i and its neighbor U_{i+1} respectively, and outputs the shared K_i. However, to make the key K_i shared only between U_i and its neighbor U_{i+1} (i.e., other users cannot obtain the key K_i), an obfuscated program P^{iO} for PRF will be required the knowledge of a seed r to operate. More precisely, each user generates a seed r_i and computes $s_i = PRG(r_i)$, where PRG is a pseudorandom generator.

Protocol II

Consider an execution of the protocol among users $\mathrm{U}_1, \cdots, \mathrm{U}_n$ wishing to establish a common session key and let pw be their joint password chosen uniformly at random from a dictionary **Dict** of size N. Let F_1 and F_2 be two pseudorandom functions (PRF), PRG_1 and PRG_2 be two pseudorandom generators, and iO be a program indistinguishability obfuscator.

Round 1: Each user U_i proceeds as:

1. Choose r_i^L and r_i^R randomly, compute $s_i^L = PRG_1(r_i^L)$ and $s_i^R = PRG_1(r_i^R)$.
2. Choose a PRF key K_{i-enc} for PRF F_1 and a PRF key K_i for PRF F_2.
3. Compute $c_i^L = s_i^L + F_1(K_{i-enc}, pw, U_i, U_{i-1})$ and $c_i^R = s_i^R + F_1(K_{i-enc}, pw, U_{i+1}, U_i)$.
4. Build the program P_i^{dec} in Figure 5, and the program P_i in Figure 6.
5. Broadcast c_i^L, c_i^R, $P_i^{iO-dec} = \mathrm{iO}\ (P_i^{dec})$ and $P_i^{iO} = \mathrm{iO}\ (P_i)$.

Round 2: Each user U_i proceeds as:

1. Run P_{i+1}^{iO-dec} on $(c_{i+1}^L, pw, U_{i+1}, U_i)$ to obtain s_{i+1}^L.
2. Run P_{i-1}^{iO-dec} on $(c_{i-1}^R, pw, U_i, U_{i-1})$ to obtain s_{i-1}^R.
3. Run P_i^{iO} on $(s_{i+1}^L, s_i^R, r_i^R, U_{i+1}, U_i)$ to obtain K_i.
4. Run P_{i-1}^{iO} on $(s_i^L, s_{i-1}^R, r_i^L, U_i, U_{i-1})$ to obtain K_{i-1}.
5. Compute $X_i = K_i/K_{i-1}$ and broadcast X_i.

Key Generation: Each user U_i computes the MSK $= K_i^n \cdot \prod_{j=1}^{n-1} X_{i+j}^{n-j}$ and the session key SK $= PRG_2(MSK)$.

Fig. 4. An honest execution of No-Setup password-based group key exchange protocol

In our protocol, each group user U_i executes two correlated instances to obtain K_{i-1} and K_i, one with his predecessor and one with his successor so each user can authenticate his neighbors, and then computes and broadcasts $X_i = K_i/K_{i-1}$. After this round, each user is capable of computing the group session key SK. For the message $X_i = K_i/K_{i-1}$ broadcasted by U_i in the second round, K_i is generated by U_i's own program P_i, which cannot be replaced. Moreover, the output of the first program P^{iO-dec} is only as the input of the second program P^{iO}. Thus, even if the adversary replace P^{iO-dec} and P^{iO} into malicious programs, from the messages broadcasted, he cannot obtain any information about the password or the session key.

A formal description appears in Fig. 4. In an honest execution of the protocol, it is easy to verify that all honest users in the protocol will terminate by accepting and computing the same $MSK = \prod_{j=1}^{n} K_j$ and the same session key SK. Therefore, the correctness of the protocol follows directly. For the security, we have the following theorem.

Theorem 2. If PRG is a secure pseudorandom generator, PRF a secure constrained PRF, and iO a secure indistinguishability obfuscator, then the protocol in Fig. 4 is a secure password-based group key exchange protocol with no setup.

Inputs: c, password pw, U_1, U_2
Constants: PRF F_1 key K_{i-enc}
Outputs: $c - F_1(K_{i-enc}, pw, U_1, U_2)$

Fig. 5. The program P_i^{dec}

Inputs: s_1, s_2, r, U_1, U_2
Constants: PRF F_2 key K_i

1. If $PRG(r) = s_1$ or $PRG(r) = s_2$, output $F_2(K_i, s_1, s_2, U_1, U_2)$
2. Otherwise, output $\perp$

Fig. 6. The program P_i

Proof. Fix a PPT adversary $\mathcal{A}$ attacking the password-based group key exchange protocol. We construct a sequence of experiments $\mathbf{Hyb}_0, \ldots, \mathbf{Hyb}_{13}$, with the original experiment corresponding to $\mathbf{Hyb}_0$. Let $\text{Adv}_i(\mathcal{A})$ denote the advantage of $\mathcal{A}$ in experiment $\mathbf{Hyb}_i$. To prove the desired bound on $\text{Adv}(\mathcal{A}) = \text{Adv}_0(\mathcal{A})$, we bound the effect of each change in the experiment one the advantage of $\mathcal{A}$, and then show that $\text{Adv}_{13}(\mathcal{A}) \leq \frac{Q(\lambda)}{D(\lambda)}$ (where, recall, $Q(\lambda)$ denotes the number of on-line attacks made by $\mathcal{A}$, and $D(\lambda)$ denotes the dictionary size).

Experiment $\mathbf{Hyb}_1$. Here we change the way **Execute** queries are answered. Specifically, for $i = 1, \ldots, n$, we will choose random $s_i^L, s_i^R \in \{0,1\}^{2\lambda}$ instead of generating them from PRG. Let the set $\widehat{S} = \{s_i^L, s_i^R | i = 1, \ldots, n\}$. The security of PRG yields the Lemma 6.

Lemma 6. *$| Adv_0(\mathcal{A}) - Adv_1(\mathcal{A}) | \leq negl(\lambda)$.*

Experiment $\mathbf{Hyb}_2$. In this experiment, We constrain the PRF F_2 so that it can only be evaluated at points (s_1, s_2, U_1, U_2) where $s_1 \notin \widehat{S}$ or $s_2 \notin \widehat{S}$. Then we replace F_2 with F_2^C in the program P_i, arriving at the program P_i' given in Fig. 7. In respond to a query **Execute**, output $P_i^{iO} = \text{iO}\ (P_i')$.

Lemma 7. *$| Adv_1(\mathcal{A}) - Adv_2(\mathcal{A}) | \leq negl(\lambda)$.*

Proof. Note that with overwhelming probability, none of $s \in S$ in **Experiment** Hyb_1 has a pre-image under PRG. Therefore, with overwhelming probability, there is no input to P_i^{iO} that will cause PRF F_2 to be evaluated on points of the form (s_1, s_2, U_1, U_2) where $s_1 \in \widehat{S}$ and $s_2 \in \widehat{S}$. We can conclude that the programs P_i and P_i' are functionally equivalent. Then based on the indistinguishability obfuscation property, it is easy to see that the hybrids Hyb_1 and Hyb_2 are computationally indistinguishable. Thus, security of iO yields the lemma.

```
Inputs: $s_1, s_2, r, U_1, U_2$
Constants: Constrained PRF $F_2$ key $K_i^C$
    1. If $PRG(r) = s_1$ or $PRG(r) = s_2$, output $F_2^C(K_i^C, s_1, s_2, U_1, U_2)$
    2. Otherwise, output $\perp$
```

Fig. 7. The program P_i'

Experiment Hyb$_3$. In this experiment, we change once again the simulation of the **Execute** queries so that the value K_i for $i = 1, \ldots, n$ are chosen as a random string of the appropriate length.

Lemma 8. $\mid Adv_2(\mathcal{A}) - Adv_3(\mathcal{A}) \mid \leq negl(\lambda)$.

Proof. This follows from the security of PRF as a constrained PRF (as in Definition 4). We construct a PRF adversary $\mathcal{B}$ that breaks the security of PRF as a constrained PRF as follows: adversary $\mathcal{B}$ simulates the entire experiment for $\mathcal{A}$. In response to **Execute**$(U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle})$ query, $\mathcal{B}$ computes c_i^L, c_i^R with correct password pw exactly as in experiment Hyb$_2$. $\mathcal{B}$ also asks its PRF F_2 oracle and thus always reveals the correct key. At the end of the experiment, $\mathcal{B}$ makes a real-or-random challenge query for the constrained function PRF^C as defined above. One can easily see that, $\mathcal{B}$ is given a real PRF or a random value, then its simulation is performed exactly as in experiment Hyb$_2$ or experiment Hyb$_3$, respectively. Thus, the distinguishing advantage of $\mathcal{B}$ is exactly $\mid$ Adv$_2(\mathcal{A})$ – Adv$_3(\mathcal{A}) \mid$.

Experiment Hyb$_4$. In this experiment, we change once again the simulation of the **Execute** queries so that the value MSK is chosen as a random string of the appropriate length.

Lemma 9. $Adv_3(\mathcal{A}) = Adv_4(\mathcal{A})$.

Proof. Note that in the simulation of **Execute** oracle in experiment Hyb$_3$, the values K_i for $i = 1, \ldots, n$ are chosen at random. Then, from the transcript $T = \{X_1, \ldots, X_n\}$ that the adversary receives as output for an **Execute** query, the values K_i are constrained by the following n equations.

$$X_1 = K_1/K_n$$
$$\vdots$$
$$X_n = K_n/K_{n-1}$$

Of these equations, only $n-1$ are linearly independent. Furthermore, we have

$$MSK = \prod_{i=1}^{n} K_i.$$

Since the last equation is linearly independent of the previous ones, MSK that each user computes in an **Execute** query is independent of the transcript T that the adversary sees. Thus, no computationally unbounded adversary can tell the experiment Hyb_3 apart from Hyb_4, i.e. $\text{Adv}_3(\mathcal{A}) = \text{Adv}_4(\mathcal{A})$.

Experiment Hyb$_5$. In this experiment, we change once more the simulation of the **Execute** queries so that the session key SK is chosen uniformly at random. The security of PRG guarantees that its output is statistically close to be uniform when given a random value as input, which yields the Lemma 10.

Lemma 10. $\mid Adv_4(\mathcal{A}) - Adv_5(\mathcal{A}) \mid \leq negl(\lambda)$.

Experiment Hyb$_6$. In this experiment, we change last time the simulation of the **Execute** queries. Specifically, in response to a query **Execute**$(U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle})$ we now compute $c_i^L = s_i^L + F_1(pw_0, U_i, U_{i-1})$ and $c_i^R = s_i^R + F_1(pw_0, U_{i+1}, U_i)$ for $i = 1, \ldots, n$, where pw_0 represents some dummy password not in the dictionary **Dict**. We note that in the simulation of **Execute** oracle in experiment Hyb_1, the values s_i^L, s_i^R for $i = 1, \ldots, n$ are chosen at random, and the function F_1 is pseudorandom. So the Lemma 11 holds.

Lemma 11. $\mid Adv_5(\mathcal{A}) - Adv_6(\mathcal{A}) \mid \leq negl(\lambda)$.

Experiment Hyb$_7$. In this experiment we begin to modify the **Send** oracle. Let **Send**$_0(\Pi_{U_i}^j,\ U_1, \cdots, U_n)$ denote a "prompt" message that causes the user instance $\Pi_{U_i}^j$ to initiate the protocol in a group $\mathcal{G} = \{U_1, \cdots, U_n\}$ that contains user U_i; let **Send**$_1(\Pi_{U_i}^j, \{(c_1^L, c_1^R, P_1^{iO-dec},\ P_1^{iO}), \ldots, (c_n^L, c_n^R, P_n^{iO-dec}, P_n^{iO})\})$ denote sending the message $\{(c_1^L, c_1^R, P_1^{iO-dec}, P_1^{iO}), \ldots,\ (c_n^L, c_n^R, P_n^{iO-dec}, P_n^{iO})\}$ to user instance $\Pi_{U_i}^j$; let **Send**$_2(\Pi_{U_i}^j, \{X_1, \ldots, X_n\})$ denote sending the message $\{X_1, \ldots, X_n\}$ to user instance $\Pi_{U_i}^j$.

In experiment Hyb_7 we modify the way **Send**$_0$ query is handled. In response to a query **Send**$_0(\Pi_{U_i}^j, U_1, \cdots, U_n)$, $\Pi_{U_i}^j$ chooses random $s_i^L, s_i^R \in \{0,1\}^{2\lambda}$ instead of generating them from PRG and computes the $c_i^L = s_i^L + F_1(pw_0, U_i, U_{i-1})$ and $c_i^R = s_i^R + F_1(pw_0, U_{i+1}, U_i)$, where pw_0 represents some dummy password not in the dictionary **Dict**.

Lemma 12. $\mid Adv_6(\mathcal{A}) - Adv_7(\mathcal{A}) \mid \leq negl(\lambda)$.

Proof. The proof is similar to those of Lemmas 6 and 11, and follows easily from the security of PRG and PRF.

Experiment Hyb$_8$. In this experiment, we change again the simulation of the **Send**$_0$ query. We constrain the PRF F_1 so that it can only be evaluated at points (pw, U_1, U_2) where pw is contained in the dictionary **Dict**. Then we replace F_1 with F_1^C in the program P_i^{dec}, arriving at the program $\hat{P}_i^{dec}$ given in Fig. 8. In respond to a query **Send**$_0$, output $P_i^{iO-dec} = \text{iO}\ (\hat{P}_i^{dec})$.

Lemma 13. $\mid Adv_7(\mathcal{A}) - Adv_8(\mathcal{A}) \mid \leq negl(\lambda)$.

Inputs: c, password pw, U_1, U_2
Constants: Constrained PRF F_1 key K^C_{i-enc}
Outputs: $c - F^C_1(K^C_{i-enc}, pw, U_1, U_2)$

Fig. 8. The program $\hat{P}^{dec}_i$

Proof. Since the group users share a password chosen uniformly at random from the dictionary **Dict**, we can conclude that the programs P^{dec}_i and $\hat{P}^{dec}_i$ are functionally equivalent. Then based on the indistinguishability obfuscation property, it is easy to see that the hybrids Hyb_7 and Hyb_8 are computationally indistinguishable. Thus, security of iO yields the lemma.

Experiment $\mathbf{Hyb_9}$. In this experiment, we change the simulation of the $\mathbf{Send}_1$ query. In response to a query $\mathbf{Send}_1$, if $\{(c^L_1, c^R_1, P^{iO-dec}_1, P^{iO}_1), \ldots, (c^L_n, c^R_n, P^{iO-dec}_n, P^{iO}_n)\}$ was output by a previous query of the form $\mathbf{Send}_0$, the values K_i and K_{i-1} are chosen uniformly at random. As the lemma below shows, the difference in the advantage between Hyb_8 and Hyb_9 is negligible. The proof of Lemma 14 is omitted here since it follows easily from the security of PRF F_1 as a constrained PRF, where the outputs of $\mathbf{Send}_0$ are always using dummy password.

Lemma 14. $\mid Adv_8(\mathcal{A}) - Adv_9(\mathcal{A}) \mid \leq negl(\lambda)$.

Experiment $\mathbf{Hyb_{10}}$. In this experiment, we change again the simulation of the $\mathbf{Send}_1$ query. In response to a query $\mathbf{Send}_1$, if $\{(c^L_1, c^R_1, P^{iO-dec}_1, P^{iO}_1), \ldots, (c^L_n, c^R_n, P^{iO-dec}_n, P^{iO}_n)\}$ was generated by the adversary using correct password pw, the experiment ends.

Lemma 15. $Adv_9(\mathcal{A}) \leq Adv_{10}(\mathcal{A})$.

Proof. The only situation in which Hyb_{10} proceeds differently from Hyb_9 occurs when the adversary correctly guess the password. All this does is introduce a new way for the adversary to succeed, so $\text{Adv}_9(\mathcal{A}) \leq \text{Adv}_{10}(\mathcal{A})$.

Experiment $\mathbf{Hyb_{11}}$. In this experiment, we change once more the simulation of the $\mathbf{Send}_1$ query. In response to a query $\mathbf{Send}_1$, if $\{(c^L_1, c^R_1, P^{iO-dec}_1, P^{iO}_1), \ldots, (c^L_n, c^R_n, P^{iO-dec}_n, P^{iO}_n)\}$ was generated by the adversary using incorrect password, the values K_i is chosen uniformly at random.

Lemma 16. $\mid Adv_{10}(\mathcal{A}) - Adv_{11}(\mathcal{A}) \mid \leq negl(\lambda)$.

Proof. Note that, for user instance $\Pi^j_{U_i}$, the program P^{iO}_i can not be altered by the adversary. Moreover, since s^R_i as one of inputs of P^{iO}_i is generated, the adversary can not get it or change it. Then the proof follows easily from the security of PRF F_2 as a constrained PRF. We construct a PRF adversary $\mathcal{B}$

that breaks the security of PRF as a constrained PRF as follows: adversary $\mathcal{B}$ simulates the entire experiment for $\mathcal{A}$. In response to **Send** query, $\mathcal{B}$ responds with correct password pw exactly as in experiment Hyb_{10}. $\mathcal{B}$ also asks its PRF F_2 oracle and thus always reveals the correct key. At the end of the experiment, $\mathcal{B}$ makes a real-or-random challenge query for the constrained function PRF^C as defined above. One can easily see that, $\mathcal{B}$ is given a real PRF or a random value, then its simulation is performed exactly as in experiment Hyb_{10} or experiment Hyb_{11}, respectively. Thus, the distinguishing advantage of $\mathcal{B}$ is exactly $\mid \mathrm{Adv}_{10}(\mathcal{A}) - \mathrm{Adv}_{11}(\mathcal{A}) \mid$.

Experiment $\mathbf{Hyb}_{12}$. In this experiment, we change once more the simulation of the $\mathbf{Send}_2$ query. In response to a query $\mathbf{Send}_2(\Pi_{U_i}^j, \{X_1, \ldots, X_n\})$, the value MSK is chosen uniformly at random.

Lemma 17. *$Adv_{11}(\mathcal{A}) = Adv_{12}(\mathcal{A})$.*

Proof. The proof of Lemma 17 uses arguments similar to those in the proof of Lemma 9, omitted.

Experiment $\mathbf{Hyb}_{13}$. In this experiment, we change again the simulation of the $\mathbf{Send}_2$ query so that the session key SK is chosen uniformly at random. The security of PRG guarantees that its output is statistically close to be uniform when given a random value as input, which yields the Lemma 18.

Lemma 18. *$\mid Adv_{12}(\mathcal{A}) - Adv_{13}(\mathcal{A}) \mid \leq negl(\lambda)$.*

Bounding the Advantage in $\mathbf{Hyb}_{13}$. We now conclude the experiment Hyb_{13}. First, the session keys of all accepting instances are chosen at random. Second, all oracle instances are simulated using dummy passwords, so the adversary's view of the protocol is independent of the passwords that are chosen for each group of users. Finally, the probability that an adversary guesses the correct password is at most $\frac{Q(\lambda)}{D_\lambda}$. Similar to the proof of Theorem 1, we have $\mathrm{Adv}_{13}(\mathcal{A}) \leq \frac{Q(\lambda)}{D_\lambda}$. Taken together, Lemmas 6–18 imply that $\mathrm{Adv}_0(\mathcal{A}) \leq \frac{Q(\lambda)}{D_\lambda} + negl(\lambda)$ as desired. $\square$

6 Conclusion

In this paper, we proposed two round-optimal constructions for password-based group key exchange protocol. In particular we obtain a *one-round* protocol in the common reference string model and a *two-round* protocol in the "plain" model where there is no additional setup. Both protocols are provably secure in the standard model. It remains an interesting open problem to further reduce the computational costs of group users, whilst maintaining its optimal communication rounds.

References

1. Pointcheval, D.: Password-based authenticated key exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 390–397. Springer, Heidelberg (2012)
2. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group Diffie-Hellman key exchange. In: Proceedings of the 8th Annual ACM Conference on Computer and Communications Security, pp. 255–264. ACM (2001)
3. Burmester, M., Desmedt, Y.G.: A secure and efficient conference key distribution system. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
4. Burmester, M., Desmedt, Y.: A secure and scalable group key exchange system. Inf. Process. Lett. **94**(3), 137–143 (2005)
5. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
6. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014)
7. Bresson, E., Chevassut, O., Pointcheval, D.: Group Diffie-Hellman key exchange secure against dictionary attacks. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 497–514. Springer, Heidelberg (2002)
8. Dutta, R., Barua, R.: Password-based encrypted group key agreement. Int. J. Netw. Secur. **3**(1), 30–41 (2006)
9. Lee, S.M., Hwang, J.Y., Lee, D.-H.: Efficient password-based group key exchange. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 191–199. Springer, Heidelberg (2004)
10. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-based group key exchange in a constant number of rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
11. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer, Heidelberg (2006)
12. Abdalla, M., Chevalier, C., Granboulan, L., Pointcheval, D.: Contributory password-authenticated group key exchange with join capability. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 142–160. Springer, Heidelberg (2011)
13. Bresson, E., Chevassut, O., Pointcheval, D.: A security solution for IEEE 802.11s Ad-Hoc mode: password authentication and group Diffie-Hellman key exchange. Int. J. Wirel. Mob. Comput. **2**(1), 4–13 (2007)
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS 2013, pp. 40–49. IEEE (2013)
15. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
16. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **34**(4), 792–807 (1986)

17. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
18. Naor, M., Yung, M.: Pulbic-key cryptosystems provably secure against chosen ciphertext attacks. In 22nd Annual ACM Symposium on Theory of Computing, pp. 427–437 (1990)