UC-secure and Contributory Password-Authenticated Group Key Exchange

Lin Zhang and Zhenfeng Zhang[✉]

Trusted Computing and Information Assurance Laboratory,
Institute of Software, Chinese Academy of Sciences, Beijing, China
{zhanglin,zfzhang}@tca.iscas.ac.cn

Abstract. The contributory property allows participants of group key exchange fairly to engage in the generation of the random session key rather than an entity or some part of members solely to determinate it or force it to lie in an undesired distribution. In this paper, we put forth a password-authenticated group key exchange (GPAKE) in which principals cooperate to agree a strong session key just in possession of a short password. The scheme realizes the optimality of contributory property—full-contributiveness—as long as there is one honest party, the uniform distribution of final session keys can be guaranteed. Moreover, it reaches the security definitions in the well-known universal composability (UC) framework under the random oracle model based on the one-more gap Diffie-Hellman assumption. In particular, our scheme that achieves these results with only two-round messages, has better performances on round complexity in comparison with the existing UC-secure schemes.

Keywords: Group key exchange · Password-based protocols · Contributiveness · Universal composability

1 Introduction

In recent decades, as electronic communications and information systems become more and more complicated, applications, such as video- or tele-conferencing involving multiple participants, are widespread throughout the Internet. In order to satisfy the requirement of secure communication channels within the insecure public network, it is necessary to design authenticated key exchange protocols for groups of principals.

To date, a collection of schemes has been designed elegantly. Bresson, et al. [10] is the first to usher in a formal model of security for group key exchange protocols, and the first to give a concrete scheme with a rigorous proof. However, in their protocol, the number of communication rounds depends upon the number of group members, so that this construction is impractical in the situation where the number of players is very large. Fewer rounds generally mean easier implementation and more effective reducing to synchronization problems. Subsequent to their work, several solutions [8,24] to constant-round protocol for group key exchange are provided and proven secure in formal models. In addition, the desirable security goals of this kind of protocol not just focus on resistance against the outsider adversary who lies outside of the target group and seeks to get any information about the session key with observing and modifying protocol messages. Additionally, a certain degree of security properties against malicious insiders are expected in the designs of protocols. In Katz and Shin's work [23], they define the insider security for group key exchange protocols: one prevents the adversary from determining the session key entirely, unless at least there exists one corrupted party in the group.

Many schemes, including ones mentioned above, relies on possession of shared keys with other peers or authentic public/private pairs. In some scenarios, passwords, the ubiquitous keys to on-line communications, are the proper alternative in the group key exchanges, which benefits from password's convenience and low-cost. Namely, in the password-authenticated group key exchange (GPAKE), members only share a low-entropy password that can be reliably remembered by humans to bootstrap a high-entropy session key. Compared with other group key exchange protocols, GPAKEs, as password-based protocols, bear additional vulnerabilities to off-line dictionary attacks and the inevitable on-line dictionary attacks because of relatively small dictionary space. Thus, how to resist off-line attacks and restrict to adversaries eliminating at most one password per party instance, is also the basic security requirement in designing password-authenticated group key exchange protocols.

The first solution to the GPAKE is proposed by Bresson et al. [9]. Still, their protocol's round complexity is related to the number of group users. Abdalla et al. [1,4] demonstrate the first password-based group key exchange protocols in a constant number of rounds, in the random oracle /ideal cipher models [7,25], and in the standard model, respectively. Later, they give provably secure schemes [2,3] universal composability (UC) framework [15]. Recently, Xu et al. [27] present an one-round scheme in the standard model, using indistinguishability obfuscation as the main tool. Specifically, the works of [2,3] achieve a strong notion of (t, n)-contributiveness which captures that no adversary can bias the key if no more t players in the group of n players have been corrupted.

These schemes have shown important outcomes in GPAKE, yet much work remains to be done to enhance the efficiency and practicality of existing schemes. We focus on how to realize as few rounds as possible for the design of GPAKE scheme, without the expense of desirable security involved in the preceding description.

1.1 Our Contributions

In this literature, we put forward a UC-secure solution for password-authenticated group key exchange protocol against the static adversary in the password-only model, where the players do not have public keys authenticated by a certificate authority, pre-shared symmetric keys or other auxiliary equipments.

In the aspect of security models, this scheme is provably secure in the UC framework by the help of random oracles under the one-more gap Diffie-Hellman assumption. Compared with the game-based security models, such as [6,9], not only does the UC framework inherently provide the secure composition property, but it also has conspicuous advantages in distribution of passwords. Specifically, UC framework brings about strong composability properties: (1) UC-secure protocols remain secure even if many protocol instances (may be various kinds protocols) execute concurrently, and (2) The powerful universal composition theorem guarantees that they can be securely used as sub-routine protocols of other UC protocols. Besides, rather ideal assumptions on passwords independently chosen from pre-determined dictionary space in the game-based models, UC framework designates the environment (i.e. the distinguisher) to provide passwords to parties, which models arbitrary distributions and dependencies between passwords. Thereout, it captures the cases in real-life settings where the honest parties with incorrect but related passwords interacts with others—when a user obliviously makes typos.

Furthermore, we incorporate the full-contributiveness property (or called as (n, n)-contributiveness) into our rewritten ideal functionality for GPAKE, which means that the adversary cannot bias the key if there exists at least one honest player in the group, while our scheme has proven to be capable of realizing it. In fact, the notion of contributiveness brings several advantages in group key exchange protocols. First, it pledges each party equally contributes to the session key, which makes one intuitively feel that key agreement is "fairer" than key distribution. Second, it still results in high quality random keys even though some malicious parties improperly choose their contributions. Third, it deters the case where the insider adversary determinates session keys to specific values known by an outsider adversary in advance, so that the latter can eavesdrop on the later communications without the former's direct divulging to them. To a certain degree, the destructibility of the insider adversary is decreased.

An important measure of a protocol's efficiency is the communication complexity (number of protocol rounds) of the given protocol. Our protocol achieves the properties above with only two rounds. It distinctly has better performance on round complexity than the other UC-secure ones [2,3]. On the minus side, our scheme also has to perform $O(\mathrm{n})$ exponent calculations per member.

Table 1. Comparison of GPAKE schemes

Scheme	Security	Contributeness	Model	Rounds	Computation
BCP [9]	Game	$(1, n)$	RO&IC	n	$O(n)$
ABCP [1]	Game	$(1, n)$	RO&IC	4	$O(1)$
AP [4]	Game	$(1, n)$	Std	5	$O(n)$
ACCP [2]	UC	$(n/2, n)$	RO&IC	5	$O(n)$
ACGP [3]	UC	(n, n)	Std	6	$O(n)$
XHZ [27]	Game	$(1, n)$	Std	1	iO[1]
Ours	UC	(n, n)	RO	2	$O(n)$

[1] The "iO" means a program indistinguishability obfuscator.

However, according to the Moore's laws which declare the computing power grows faster than communication power, it is therefore an acceptable and reasonable compromise trade communication power for computing power in a group key exchange protocol. Comparison of some existing schemes for GPAKE is shown in Table 1.

2 Security Definitions

In this section, we will begin by reviewing the UC framework and the general split functionality. Then a detailed description of the ideal functionality for the password-authenticated group key exchange and related discussion are followed.

2.1 Universal Composability Framework

Universal composability [15] is the definition of secure computation that considers an execution of a protocol in the setting involving an *environment* $\mathcal{Z}$, an adversary and parties. This framework involves two worlds—the real world and the ideal world. $\mathcal{Z}$'s aim is to distinguish two worlds. For it, the environment provides the inputs to the parties and observes their outputs. On one hand, as usual, the real world consists of participants of the protocol and an *adversary* $\mathcal{A}$ that controls the communication channel and potentially attacks protocols. On the other hand, in the ideal world, there exists an entirely trusted entity $\mathcal{F}$ called *ideal functionality*, and dummy participates of the target protocol simply hand their inputs to $\mathcal{F}$. The *ideal adversary* $\mathcal{S}$ directly interacts with $\mathcal{F}$, and their communication essentially models the information it can obtain and its abilities to attack the protocols. Namely, the functionality describes the security goals we expect. Intuitively, the adversary, with a variety of means of attacks, should not learn more information than the functionality's outputs to it. Thus, security requires that, for any adversary $\mathcal{A}$ attacking a protocol ρ, there exists an ideal adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish the case that it is interacting with $\mathcal{A}$ and parties in the real world, and the case which it is interacting with $\mathcal{S}$ and a functionality $\mathcal{F}$ in the ideal world. If so, we say that ρ *UC-realizes* $\mathcal{F}$. From the point of view of the environment, the real-world protocol is at least as secure as the ideal-world one. In particular, the universal composability theorem guarantees that the protocol ρ continues to behave like the ideal functionality $\mathcal{F}$ even if it is executed in an arbitrary network environment. The complete details refer to [15].

2.2 Split Functionalities

In a network, without any authentication mechanism, an adversary of the network can simply "disconnect" the parties completely, and engage in separate executions with the other parties on behalf of the honest ones. Such an attack is inevitable. Players cannot distinguish the case in which they interact with the expected ones from the case where they interact with the adversary. Hence, in

For a given functionality $\mathcal{F}$, the split version $s\mathcal{F}$ proceeds as follows:

Initialization:

- Upon receiving (Init, sid) from P_i, send $(\mathsf{Init}, sid, P_i)$ to the adversary $\mathcal{S}$.
- Upon receiving $(\mathsf{Init}, sid, P_i, \mathcal{H}, sid_{\mathcal{H}})$ from $\mathcal{S}$, where $\mathcal{H}$ is the set of party identities, check that P_i has sent (Init, sid) and that for all previous sets $\mathcal{H}'$, either (1) $\mathcal{H} = \mathcal{H}', sid_{\mathcal{H}} = sid_{\mathcal{H}'}$, or (2) $\mathcal{H} \cap \mathcal{H}' = \varnothing, sid_{\mathcal{H}} \neq sid_{\mathcal{H}'}$. If so, record $(\mathcal{H}, sid_{\mathcal{H}})$, send $(\mathsf{Init}, sid, sid_{\mathcal{H}})$ to P_i, and initialize a new instance of $\mathcal{F}$ with $sid_{\mathcal{H}}$, denoted as $\mathcal{F}^{\mathcal{H}}$. Otherwise, ignore this message.

Computation:

- Upon receiving (Input, sid, m) from P_i, find the set $\mathcal{H}$ such that $P_i \in \mathcal{H}$, and forward m to $\mathcal{F}^{\mathcal{H}}$.
- Upon receiving $(\mathsf{Input}, sid, \mathcal{H}, P_j, m)$ from $\mathcal{S}$, if $\mathcal{F}^{\mathcal{H}}$ exists and $P_j \notin \mathcal{H}$, then forward m to $\mathcal{F}^{\mathcal{H}}$ as if coming from P_j. Otherwise, do nothing.
- When a copy $\mathcal{F}^{\mathcal{H}}$ generates an output m for party $P_i \in \mathcal{H}$, send m to P_i. if m is for a party $P_j \notin \mathcal{H}$ or for $\mathcal{S}$, $s\mathcal{F}$ sends the output to $\mathcal{S}$.

Fig. 1. The split version of ideal functionality $\mathcal{F}$

the work of [5], Barak et al. propose a new model based on split functionalities which guarantees that this attack is the only one available to the adversary.

The split functionality is a generic construction based upon a normal ideal functionality $\mathcal{F}$. Its formal description can be found on Fig. 1. It models security by allowing the adversary to carry out such an "attack" in the ideal world. In the initialization stage, the adversary adaptively chooses subsets of the honest parties' $\mathcal{H}$ under two constraints: (1) these subsets are disjoint; (2) the adversary must choose a unique session identifier $sid_{\mathcal{H}}$ for each authentication set $\mathcal{H}$. That is, the subsets create a partition of the players. During the computation stage of $s\mathcal{F}$, each subset $\mathcal{H}$ activates a different and independent instance of the ideal functionality $\mathcal{F}$, denoted as $\mathcal{F}^{\mathcal{H}}$. In each such execution, the parties $P_i \in \mathcal{H}$ provide their own inputs, and the adversary $\mathcal{S}$ provides the inputs for all $P_i \notin \mathcal{H}$. Similarly, the parties $P_i \in \mathcal{H}$ all receive their specified outputs as computed by their copy of $\mathcal{F}$. However, the adversary receives all of its own outputs, as well as the outputs of the parties $P_i \notin \mathcal{H}$ who are controlled by $\mathcal{S}$. It's important to note that there is no interaction between different instances of $\mathcal{F}$ run by $s\mathcal{F}$.

2.3 The Ideal Functionality for GPAKE

The formalized description of GPAKE's ideal functionality $\mathcal{F}_{\mathsf{GPAKE}}$ is presented in Fig. 2. In order to reduce repeated representations, assume that the ideal functionality only takes notice of the first query or input for each sid and party, The functionality $\mathcal{F}_{\mathsf{GPAKE}}$ parameterized by the security parameter κ, interacts with an adversary $\mathcal{S}$ and a ordered set of parties $\mathcal{H} = \{P_1, \ldots, P_n\}$ (where $n \geq 3$) via the following queries:

- **Initialization.** Upon receiving $(\mathsf{NewSession}, sid, P_i, pw_i)$ from P_i, record (sid, P_i, pw_i), if P_i is honest, mark it fresh, and send $(\mathsf{NewSession}, sid, P_i)$ to $\mathcal{S}$. Otherwise, this record is marked as $\mathsf{corrupted}$ instead. If there exists $n-1$ recorded tuples (sid, P_j, pw_j) for $P_j \in \mathcal{H}\backslash\{P_i\}$, then record $(sid, \mathcal{H}, \mathsf{ready})$ and send it to $\mathcal{S}$.
- **Key Generation.** Upon receiving $(\mathsf{NewKey}, sid, P_i, \mathcal{H}^*, sk^*)$ from $\mathcal{S}$, abort if there is no record of the form $(sid, \mathcal{H}^*, \mathsf{ready})$ or $\mathcal{H} \neq \mathcal{H}^*$. Otherwise, proceed for record (sid, P_i, pw_i) as follows:
  - If all the records whose identities belong to $\mathcal{H}^*$ are corrupted, then output (sid, sk^*) to player P_i.
  - If this record is fresh, and there is a record (sid, P_j, pw') with $pw' = pw_i$, and a key sk' was sent to P_j, then output (sid, sk') to P_i.
  - In any other case, pick a new random key sk' of length κ, and send (sid, sk') to P_i.

  Either way, mark the record (sid, P_i, pw_i) as $\mathsf{completed}$.

Fig. 2. The ideal functionality $\mathcal{F}_{\mathsf{GPAKE}}$

and subsequent ones for the same sid and party are straightly ignored. What's more, the session identifier sid are considered to be globally unique so that several sessions running in parallel can be distinguished. Note that we take into account the static corruption—the adversary could selectively designate and corrupt some participants, but only prior to the beginning of a protocol instance. From the corruption on, it not only obtains their inputs resulted from $\mathcal{Z}$, and also fully controls their behaviors in the following executions.

In the ideal world, the functionality $\mathcal{F}_{\mathsf{GPAKE}}$ interacts with an adversary $\mathcal{S}$ (i.e. the simulator), n parties $P_1, \ldots, P_n$ and the environment $\mathcal{Z}$ (through the parties). Before beginning, $\mathcal{Z}$ chooses the passwords pw_i (may be unequal) on its own for participants, which captures the arbitrary password distribution, including the case of making typos. As a bonus, this approach provides forward secrecy for free, which preserve the security of session keys even if the password is used for other purposes.

Though such a query $(\mathsf{NewSession}, sid, P_i, pw_i)$, every party initiates a new session with the expected ones in the group $\mathcal{H}$. Then $\mathcal{F}_{\mathsf{GPAKE}}$ is triggered to create the corresponding records, such as (sid, P_i, pw_i), for them to store their inputs locally, and labeled it as fresh. Actually, among these group members, some may be impersonated or corrupted by the adversary S to take part in the protocol instance with $\mathcal{S}$'s own password attempt. In both cases, the records are marked as $\mathsf{corrupted}$. Once all the parties in the group $\mathcal{H}$ have sent $\mathsf{NewSession}$ queries, the ideal functionality $\mathcal{F}_{\mathsf{GPAKE}}$ stores a record $(sid, \mathcal{H}, \mathsf{ready})$, and informs the adversary $\mathcal{S}$ with it as a notification. When the adversary $\mathcal{S}$ commences with impersonating a party P_i with the NewSession query, it is allowed temporarily to submit a character $\perp$ instead of the password, and replenish it before sending the corresponding NewKey query. This stipulation contributes to a more smooth simulation in the security proof. In principle, after this phase, the parties basically wait for receiving the session keys.

When receiving $(\mathsf{NewKey}, sid, P_i, \mathcal{H}^*, sk^*)$ query from the adversary $\mathcal{S}$, the ideal functionality $\mathcal{F}_{\mathsf{GPAKE}}$ is instructed to release the session key to P_i. Note that $\mathcal{H}^*$ is the set of participants that is specified by the adversary and may not be equal to $\mathcal{H}$ in the NewSession queries. When $\mathcal{H} = \mathcal{H}^*$, the computation happen within pre-assigned members, while $\mathcal{H} \neq \mathcal{H}^*$ means that the adversary introduces outsiders (may be fictional entities) into the group to replace some honest ones. The latter case is forbidden in our definition. Besides, if there no exists a ready record for $\mathcal{H}^*$, i.e. members of $\mathcal{H}^*$ do not entirely join this session, $\mathcal{F}_{\mathsf{GPAKE}}$ also abort this execution. Unlike previous key exchange functionalities [16,23], in that if one of NewSession records is corrupted the adversary is given the ability to fully determine the resulting session key into sk^*, ours deprives of this ability of $\mathcal{S}$ unless all parties are corrupted simultaneously. By this definition, we integrate the full-contributiveness property in the functionality. Participants shall share the same, uniformly distributed session keys with whom have the matching password. Namely, $\mathcal{F}_{\mathsf{GPAKE}}$ has to traverse the records and the session keys sent to some parties, and return the corresponding ones. If no one is found out, $\mathcal{F}_{\mathsf{GPAKE}}$ chooses a random value from the range of session keys. In consideration of implicit authentication, the protocol will not end up with the case where no key is established for parties unless the inevitable abortions occur.

When $\mathcal{F}_{\mathsf{GPAKE}}$ outputs the session key to the specified party, the corresponding NewSession record is marked as completed to avoid undesired on-line guessing attacks from $\mathcal{S}$ even after the authentication has ended.

2.4 Discussions

For the completeness of the ideal functionality, the adversary $\mathcal{S}$ should be acquiescently allowed to abort the instance at any time to capture some trivial cases. For instance, in the real network, the attackers can always delay, hijack messages or revise them into irregular formats in the communication channel, resulting in a failed session among parties.

In our context of the ideal functionality $\mathcal{F}_{\mathsf{GPAKE}}$, the TestPwd query is completely abandoned, since split functionality has modeled the adversary's active attacks which enable it to take apart in the group. In the view of security analysis, the simulator does have to learn the results whether the passwords are matching, which is totally left to the functionality along with generation of session keys. Moreover, an outsider should not get the final results of protocol executions without the later communications in realistic scenarios.

In the UC framework, as per the formalism of [15], assume that multiple protocol instances are running concurrently. As the case in the real world, numerous execution instances often invoke the same common random strings or random oracles. Roughly, we have to consider the multi-session extension $\widehat{\mathcal{F}}$ through the JUC theorem [18]. We refer to [16,18] for more discussions.

3 Our GPAKE Scheme

The basic idea of this protocol is inspired by Jarecki et al's (verifiable) oblivious pseudorandom functions (V-OPRF) in [21,22] and Camenisch et al's constructions for distributed password verification protocol of [12], and then utilized to build our GPAKE scheme. Briefly, each participant $P_i \in \{P_1, \ldots, P_n\}$ has the shared password pw_i, along with its own ephemeral secret key x_i. The session key computed by parties for session and sub-session identifiers sid, $ssid$ is $H_2(sid, ssid, H_1(sid, pw_i)^X)$, where $X = \sum_{i=1}^{n} x_i \bmod q$, and H_1, H_2 are hash functions. In order to get this value, each party chooses $r_i \leftarrow_{\mathrm{R}} \mathbb{Z}_q^*$ to blind the password hash $a_i := (H_1(sid, pw_i))^{r_i}$ and sends the result to the others. When $(b_{i,j} := a_i^{x_j})_{j\in[n],\, j\neq i}$ are returned, it can compute $v_i := (a_i^{x_i} \cdot \prod_{j=1,\, j\neq i}^{n} b_{j,i})^{1/r_i} = H_1(sid, pw_i)^X$. Simultaneously, P_i proceed to the similar power operation to $\{a_j\}_{j\neq i}$ from the others with its own secret key x_i. Nevertheless, such simple proposition of GPAKE cannot reach the UC-security in the unauthenticated channel. Hence, we provide other primitives, such as the zero-knowledge proof of knowledge and the extra hash function to ensure the simulator that the participants always use the coincident public/secret keys, and also to help it extract the secret key for simulation. More details are presented as follows.

3.1 Concrete Construction

Let $\mathbb{G}$ be a multiplicative group of prime order q with the generator g generated through an algorithm of parameters generation by the security parameter κ. The hash functions $H_1 : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \{0,1\}^{2\kappa}$ and $H_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \rightarrow \{0,1\}^{\kappa}$ are modeled as random oracles. The public parameters also consist of the common random strings crs for the zero-knowledge proofs of knowledge. PK denotes the non-interactive proof of knowledge (which is formally defined by Camenisch et al. [11,14]), showing $y_i = g^{x_i} \wedge (b_{i,j} = a_j^{x_i})_{j\in[n],\, j\neq i}$.

Assume that the actual members are known in advance, and we simply denote them as $P_1, \ldots, P_n$ according to a certain order. In a protocol instance, the parties communicate over an unauthenticated broadcast channel, where messages can be arbitrarily observed, modified, and delayed by the adversary $\mathcal{A}$. Particularly, the adversary can corrupt or impersonate the valid ones to join the group as an insider with its own password attempt.

When a protocol execution begins, each party randomly chooses a blinding factor r_i and ephemeral secret key x_i from $\mathbb{Z}_q^*$, and then generates the blinded password hash $a_i := (H_1(sid, pw_i))^{r_i}$ and the ephemeral public key $y_i := g^{x_i}$, respectively. By the end of this round, it computes a hash to the values a_i and y_i, i.e. $h_i := H_2(sid, a_i, y_i)$. Then each party broadcasts $\langle P_i, a_i, h_i\rangle$ to others.

Shared information: Generator g of group $\mathbb{G}$. Hash functions H_1, H_2, H_3.
Common reference strings for proofs of knowledge crs.
Information held by P_i: Password pw_i.

Round 1:

1). Choose $x_i \leftarrow_R \mathbb{Z}_q^*$ and generate $y_i := g^{x_i}$;
2). Pick $r_i \leftarrow_R \mathbb{Z}_q^*$ and compute $a_i := (H_1(sid, pw_i))^{r_i}$;
3). Make a hash $h_i := H_2(sid, a_i, y_i)$;
4). Broadcast $\langle P_i, a_i, h_i \rangle$.

Round 2:

1). On receiving $\langle P_j, a_j, h_j \rangle$ from all $P_j \in \mathcal{SP} \backslash \{P_i\}$, compute $b_{i,j} := a_j^{x_i}$, and set $ssid := (a_1, h_1)||\dots||(a_n, h_n)$;
2). Produce the non-interactive proof of knowledge π_i;
3). Broadcast $\left\langle P_i, y_i, (b_{i,j})_{j\in[n], j\neq i}, \pi_i \right\rangle$.

Key Generation:

1). Upon receiving $\left\langle P_j, (b_{j,k})_{k\in[n], k\neq j}, \pi_j \right\rangle$ from all $P_j \in \mathcal{SP} \backslash \{P_i\}$, check $h_j = H_2(sid, a_j, y_j)$ and continue. If not, abort this instance;
2). Verify $(\pi_j)_{j\in[n], j\neq i}$, and abort if one of them is invalid;
3). Compute $v_i := (a_i^{x_i} \cdot \prod_{j=1, j\neq i}^{n} b_{j,i})^{1/r_i}$, and then output the session key $sk_i := H_3(sid, ssid, pw_i, v_i)$;

Fig. 3. The description of our GPAKE protocol for each party P_i

Note that the sub-session identifier is defined as messages received in this round $ssid := a_1, h_1||\dots||a_n, h_n$, which be included in the subsequent hash evaluations. Specifically, it means that parties are partitioned by the shared messages among them. The purpose of usage of H_2 is, during the formal security proof, to embed the one-more gap Diffie-Hellman problem in the next round rather than this round when the group has not partitioned by the adversary yet (Fig. 3).

In the second round, each party computes blinded responses $b_{i,j} := a_j^{x_i}$ for the others in this group using its ephemeral secret key x_i. Moreover, it is required to generate a non-interactive zero-knowledge proof of knowledge PK that $b_{i,j}$ is generated correctly using y_i's discrete logarithm. That is,

$$\pi_i := \mathsf{PK}\{x_i : y_i = g^{x_i} \wedge (b_{i,j} = a_j^{x_i})_{j\in[n],\, j\neq i}\}$$

Note that these zero-knowledge proofs should be on-line extractable, since the simulator $\mathcal{S}$ needs to extract the adversary's ephemeral secret keys to obtain the solutions to the one-more gap Diffie-Hellman problem in the simulation of random oracle H_3. It ends this round communication with broadcasting the message $\langle P_i, y_i, (b_{i,j})_{j\in[n],\, j\neq i}, \pi_i\rangle$ to the other participants.

In the end, the parties check the hash values and the proofs of knowledge. As soon as a value received by P_i doesn't be verified correctly, it aborts this instance and outputs nothing. Otherwise, it computes the key material $v_i := (a_i^{x_i} \cdot \prod_{j=1,\, j\neq i}^{n} b_{j,i})^{1/r_i}$ and the session key $sk_i := H_3(sid, ssid, pw_i, v_i)$, outputs $(sid, ssid, sk)$, and terminates this session.

Throughout this scheme, the hash values in the first round and the proofs of knowledge play important roles in ensuring the full-contributory property. For the existence of hash values and proofs of knowledge, it is impossible for a malicious party P_i adaptively to choose its ephemeral secret key x_i after it gets $\prod_{j=1,\, j\neq i}^{n} H_1(sid, pw_i)^{x_j}$. Namely, even if there is only one honest party, the remaining $n-1$ ones still cannot have the sum of secret keys depend on its.

Remarkably, this scheme achieves the implicit authentication by only two rounds communications among the participants. Using general techniques, such as the hash values of session key materials along with new tags, it is easy to get explicit authentication at the cost of one more round messages.

In the scheme, PK is a non-interactive transformation of a proof of knowledge with the Fiat-Shamir heuristic [19] in the random oracle model. It can be extended to be online-extractable, by verifiably encrypting the witness with a public key defined in the common reference string. The witness can be extracted from the CRS by the simulator without rewinding by decrypting the ciphertext. A practical instantiation is given by the CPA-secure version of Camenisch and Shoup's encryption scheme [13], which is secure under the DCR assumption [26].

4 Security Analysis

In this section, we prove the security of our scheme utilizing the (N, Q) one-more gap Diffie-Hellman assumption, which states that for the group $\mathbb{G}$ there no exists polynomial-time adversary $\mathcal{A}$ so that the following probability is non-negligible:

$$\mathsf{Prob}[\mathcal{A}^{(\cdot)^k, \mathsf{DDH}(\cdot)}(g, g^k, g_1, \ldots, g_N) = \{(g_{j_s}, g_{j_s}^k) | s = 1, \ldots, Q+1\}]$$

where $k \in \mathbb{Z}_q^*$ and Q is the number of the queries $\mathcal{A}$ makes to the $(\cdot)^k$ oracle. Moreover, $\mathcal{A}$'s other inputs $g_1, \ldots, g_N$ are assumed to be sampled from $\mathbb{G}$.

We can draw a conclusion for the GPAKE scheme in this theorem below:

Theorem 1. *Under the one-more gap Diffie-Hellman assumption in $\mathbb{G}$, if the zero-knowledge proofs involved are online extractable, then the password-authenticated group key exchange presented in Sect. 3 securely realizes $\widehat{s\mathcal{F}}_{\mathsf{GPAKE}}$ under static corruptions in the $(\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{RO}})$-hybrid model.*

In order to prove this theorem, it is an ideal-world adversary (i.e. simulator) $\mathcal{S}$ that needs to be constructed such that an arbitrary environment $\mathcal{Z}$ cannot distinguish between protocol executions in the ideal world and ones in the real world, which is described in Sect. 4.1. Then, in Sect. 4.2, we demonstrate the indistinguishability between two worlds through a sequence of games.

4.1 Description of Simulator

The simulator $\mathcal{S}$ not only interacts with the functionality $\mathcal{F}_{\mathsf{GPAKE}}$ in the ideal world, but also acts as honest parties and environment $\mathcal{Z}$ against a copy of the real-world adversary $\mathcal{A}$ invoked by $\mathcal{S}$ internally, and provide it a simulated real world. Moreover, $\mathcal{S}$ faithfully forwards all messages between $\mathcal{A}$ and $\mathcal{Z}$.

Simulating Random Oracles and Common Random Strings. When the simulator receives the queries to random oracles H_1, H_2 and H_3, it chooses random values from appropriate ranges to provide answers, and then records inputs and outputs. Here, $\mathcal{S}$ is allowed to maintain a list Λ to store them, which is also helpful to ensure the consistency of simulated random oracles. The simulator answers $\mathcal{A}$'s queries and updates the lists according to the rules which are analogous to the description in Fig. 4. Particularly, the random oracle H_3 is answered by the help of NewKey queries in some points.

Furthermore, the simulated common reference string is chosen by $\mathcal{S}$ for the adversary $\mathcal{A}$ as $\mathcal{F}_{\mathsf{CRS}}$ presented in the Appendix A.2. $\mathcal{S}$ runs the initial simulator for proofs of knowledge and gets (crs, τ). $\mathcal{S}$ sets the common reference string to crs and locally stores τ as the trapdoor for generating simulated PKs and extracting the adversary's witnesses.

Simulating the Party $\boldsymbol{P_i}$. Once receiving $(\mathsf{Init}, sid, P_i)$ and $(\mathsf{NewSession}, sid, P_i)$ from the functionality, the simulator $\mathcal{S}$ randomly samples an element g_i from the group $\mathbb{G}$, and then sets $a_i := g_i$, due to the fact that it has no access to the correct password for the honest party P_i. And it randomly chooses the value h_i from $\{0,1\}^*_{H_2}$. Such messages from honest participants are delivered to the adversary $\mathcal{A}$ in the simulated real world.

The adversary $\mathcal{A}$ can make its decision about the subgroups participants belong to, on account of lack of strong authentication assumptions. It sends the first flow to target parties on behalf of ones it wants to impersonate (or they have been corrupted since the beginning of the session). These subgroups $\mathcal{H}$s are defined according to the messages $(a_i, h_i)_{i\in[n]}$ in the first round. $\mathcal{S}$ forwards these $\mathcal{H}$s, which make a partition of all parties, to the split functionality. Namely, the players in the same session receive and share the same $(a_i, h_i)_{i\in[n]}$. The simulator $\mathcal{S}$ also sends $\mathsf{NewSession}$ queries for the corrupted parties or ones in disguise correspondingly. Note that the simulator might have no knowledge about the latter's passwords in this moment, and thus it has to pass $(\mathsf{NewSession}, sid, P_j, \perp)$ for the dishonest party P_j to the ideal functionality instead. Moreover, we assume that the simulator is permitted to fill in the blanks later.

During the second round, on the behalf of honest parties, $\mathcal{S}$ just follows the protocol description to send $\langle P_i, y_i, (b_{i,j})_{j\in[n], j\neq i}, \pi_i\rangle$ back to $\mathcal{A}$ in the broadcast channel, where π_i is a simulated one, and H_2 is programmed as $h_i := H_2(sid, a_i, y_i)$. Finally, $\mathcal{S}$ makes a call $(\mathsf{NewKey}, sid, P_i, \perp)$ to $\widehat{\mathcal{F}}_{\mathsf{GPAKE}}$.

To make the session keys indistinguishable in the view of $\mathcal{Z}$, the simulator deals with $\mathcal{A}$'s queries $H_3(sid, ssid, pw_j, v_j)$ for some party P_j as follows. If $v_j \neq H_1(sid, pw_j)^{\Sigma x_l^* + \Sigma x_k}$, where x_k results from an honest party P_k, while x_l^*

is extracted from the proofs of knowledge provided by the dishonest one P_l, $\mathcal{S}$ just return a random value. Otherwise, $\mathcal{S}$ fills the blank $\perp$ in P_j's NewSession record with pw_j, and sends $(\mathsf{NewKey}, sid, P_j, \perp)$ to $\widehat{\mathcal{F}}_{\mathsf{GPAKE}}$, and then obtain sk. Finally, it sets $H_3(sid, ssid, pw_j, v_j) := sk$ and output it to $\mathcal{A}$.

4.2 Sequence of Games

Here, via a sequence of games $\mathbf{G}_{\mathsf{i}}$, we will prove that the real world with the arbitrary $\mathcal{A}$ and the ideal world with $\widehat{\mathcal{F}}_{\mathsf{GPAKE}}$ and $\mathcal{S}$ as defined above are indistinguishable in the view of environment $\mathcal{Z}$. This needs to be stressed that, the simulator $\mathcal{S}$ "magically" obtains the inputs of honest parties provided by $\mathcal{Z}$ in the intermediate games, but they are no longer needed at the end of simulation. Following is the sequence of concrete games:

Game $\mathbf{G}_0$: Let $\mathbf{G}_0$ be the real-world game. As we noted above, the simulator $\mathcal{S}$ "magically" receives inputs from $\mathcal{Z}$, and just simply runs the real-world protocol executions for all the honest parties.

Game $\mathbf{G}_1$: It is identical to $\mathbf{G}_0$, except that we change the generation of crs and proofs of knowledge in the protocol. More specifically, on one hand, the common random strings are replaced with the simulated ones, and $\mathcal{S}$ knows the secret keys. On the other hand, whenever the honest parties perform the proofs of knowledge, $\mathcal{S}$ provides the simulated ones instead. The indistinguishability between them follows from the zero-knowledge properties of the proof system.

Game $\mathbf{G}_2$: Since $\mathbf{G}_2$, $\mathcal{S}$ begins to simulate the hash functions H_1, H_2 and H_3 instead of the real ones. It is distinguishable with the previous game in the view of the environment $\mathcal{Z}$, if there happen collisions that multiple inputs of oracles correspond to an output. Obviously, this case occurs with negligible probability, due to the birthday paradox.

Game $\mathbf{G}_3$: Let $\mathbf{G}_3$ be the modification of $\mathbf{G}_2$, where the honest party P_i replaces a normal $a_i := H_1(sid, pw_i)^{r_i}$ with a random element g_i from $\mathbb{G}$ in the first round. Actually, in the previous game, r_i is randomly chosen from $\mathbb{Z}_q^*$ by P_i locally without leakage to the adversary $\mathcal{A}$. Therefore, $H_1(sid, pw_i)^{r_i}$ cannot be distinguished from the random g_i, from $\mathcal{Z}$'s view.

Game $\mathbf{G}_4$: Compared with $\mathbf{G}_3$, the simulator $\mathcal{S}$ makes the party P_i output sk which $\widehat{\mathcal{F}}_{\mathsf{GPAKE}}$ forwards to it after the NewKey query, rather than the normal value $H_3(sid, ssid, pw_i, v_i)$. Only by querying the $H_3(\cdot)$ oracle can the environment distinguish between these two outputs. Concretely, When the adversary $\mathcal{A}$ makes a query $(sid, ssid, pw_j, H_1(sid, pw_j)^{\Sigma x_l^* + \Sigma x_k})$, $\mathcal{S}$ interacts with the functionality $\widehat{\mathcal{F}}_{\mathsf{GPAKE}}$, and obtains the proper output value sk as the response. Besides, We define an event Γ that the adversary $\mathcal{A}$ queries the oracle $H_3(\cdot)$ on the input $(sid, ssid, pw_j, H_1(sid, pw_j)^{\Sigma x_l^* + \Sigma x_k})$ without communicating with some honest parties of $\mathcal{H}$, which gives rise to an abort in $\mathbf{G}_4$, since $\mathcal{S}$ has to send $(\mathsf{NewKey}, sid, P_j, \mathcal{H}^*, \perp)$ to the functionality, where $\mathcal{H}^* \neq \mathcal{H}$. It is observed that, provided that the event Γ does not occur, the environment is not able to distinguish between two cases.

Here, by the help of reduction from the one-more gap Diffie-Hellman problem, we conclude that the event Γ just happens with a negligible probability. Given an instance of the one-more gap Diffie-Hellman problem $(Q, g, y = g^k, g_1, \ldots, g_N)$, we revise the simulator's behaviors as follows. Before the beginning, the simulator initializes a counter $c(k) := 0$ modeling the number of times that the $(\cdot)^k$ oracle is invoked and a set of pairs of the form (z, z^k), where z is in $\{g_1, \ldots, g_N\}$, denoted as $\mathcal{T}(k) := \varnothing$. It uses the challenges $g_1, \ldots, g_N$ as the responses to $H_1(\cdot)$ queries instead of random values from $\mathbb{G}$, and as the value a_j for the honest party P_j to the other members in the first round.

In the second round, without loss of generality, assume that there exists s $(s \geq 1)$ honest parities in the target group $\mathcal{H}$, the simulator randomly samples $s-1$ values from $\mathbb{Z}_q^*$ and implicitly sets $k_t = k - \sum_{i=1, i \neq t}^{s} k_i$ for a honest party P_t. Note that $\mathcal{S}$ has no access to k_t, but it still can provide the P_t's ephemeral public key $y_t := y/(\prod_{i=1, i\neq t}^{s} g^{k_i})$. If a_j is the first round message sent to P_t, the simulator calls the $(\cdot)^k$ oracle to compute $b_{t,j} := a_j^k/(\prod_{i=1, i\neq t}^{s} a_j^{k_i})$ and simulates the proof of knowledge π_t using the trapdoor τ corresponding to the specified CRS. Moreover, once the oracle $(\cdot)^k$ is invoked, it increases the counter $c(k)$. For the other honest ones, the simulations proceeds as before. When $\mathcal{A}$ later queries $H_2(sid, ssid, pw_i, v_i)$, the simulator invokes the DDH oracle to check whether it satisfies $v_i = H_1(sid, pw_i)^{k+\Sigma x_i^*}$. If so, it adds $(H_1(sid, pw_i), v_i/H_1(sid, pw_i)^{\Sigma x_i^*})$ to the set $\mathcal{T}(k)$. Once the event $c(k) < |\mathcal{T}(k)|$, which the adversary never communicates with P_t since the end of the first round but submits an appropriate H_3 query corresponding the solution, occurs, the simulator $\mathcal{S}$ addresses the one-more gap DH problem by returning the set $\mathcal{T}(k)$.

Now, the ideal-world is identical to $\mathbf{G}_4$, except that $\mathcal{S}$ no longer owns the specified inputs from $\mathcal{Z}$. Thus, the proof of theorem is completed.

Acknowledgement. We would like to thank the anonymous reviewers for their beneficial comments. This work is supported by the National Natural Science Foundation of China (No. U1536205, 61170278) and the National Basic Research Program of China (No.2013CB338003).

A Auxiliary Ideal Functionalities

In this section, we list the formal ideal functionalities of random oracles and common random strings used as setup assumptions in our work.

A.1 Random Oracles

The random oracle model (e.g. [7]) captures an idealization of a hash function. In particular, it allows only black-box access and cannot be "predicted" without explicitly evaluating it. The outputs are uniformly selected random strings of specified size. We present the random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ that has been defined by Hofheinz and Müller-Quade [20] in Fig. 4.

$\mathcal{F}_{\mathsf{RO}}$ proceeds as follows, running on security parameter κ, with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$:

- $\mathcal{F}_{\mathsf{RO}}$ keeps a list L (which is initially empty) of pairs of bitstrings.
- Upon receiving a value (RO, sid, m), where $m \in \{0,1\}^*$ from some party P_i or from $\mathcal{S}$, do:
  - If there is a pair $(m, \tilde{h})$ for some $\tilde{h} \in \{0,1\}^\kappa$ in the list L, set $h := \tilde{h}$;
  - If there is no such pair, choose uniformly $h \in \{0,1\}^\kappa$ and store the pair (m, h) in L.

  Once h is set, reply to the activating machine (i. e., either P_i or $\mathcal{S}$) with (RO, sid, h).

Fig. 4. The ideal functionality $\mathcal{F}_{\mathsf{RO}}$

A.2 Common Reference Strings

The common reference string functionality $\mathcal{F}_{\mathsf{CRS}}$ [15,17] captures that a common string drawn from a pre-specified distribution D can be accessible by all parties in the system, including the adversary. Furthermore, it guarantees that no party can be aware of the information related to the process of generating this string. The functionality illustrated in Fig. 5 results from the 2005 version of [15].

The functionality $\mathcal{F}_{\mathsf{CRS}}$ running on distribution D proceeds as follows:

- When receiving input (CRS, sid) from party P, first verify that $sid = (\mathcal{P}, sid')$ where $\mathcal{P}$ is a set of identities, and that $P \in \mathcal{P}$; else ignore the input. Next, if there is no value r recorded then choose and record $r \xleftarrow{R} D$. Finally, send a public delayed output (CRS, sid, r) to P.

Fig. 5. The ideal functionality $\mathcal{F}_{\mathsf{CRS}}$

References

1. 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). doi:10.1007/11745853_28
2. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Password-authenticated group key agreement with adaptive security and contributiveness. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 254–271. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02384-2_16

3. 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). doi:10.1007/978-3-642-19074-2_11
4. 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). doi:10.1007/11935230_22
5. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005). doi:10.1007/11535218_22
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_11
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM (1993)
8. Boyd, C., Nieto, J.M.G.: Round-optimal contributory conference key agreement. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 161–174. Springer, Heidelberg (2003). doi:10.1007/3-540-36288-6_12
9. 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). doi:10.1007/3-540-36178-2_31
10. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably authenticated group Diffie-Hellman key exchange. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 255–264. ACM (2001)
11. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized Schnorr proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01001-9_25
12. Camenisch, J., Lehmann, A., Neven, G.: Optimal Distributed Password Verification. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security, pp. 182–194. ACM (2015)
13. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45146-4_8
14. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997). doi:10.1007/BFb0052252
15. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd IEEE Symposium on Foundations of Computer Science, pp. 136–145. IEEE (2001)
16. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005). doi:10.1007/11426639_24
17. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-party and Multi-party Secure Computation. In: Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing, pp. 494–503. ACM (2002)
18. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45146-4_16

19. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12
20. Hofheinz, D., Müller-Quade, J.: Universally composable commitments using random oracles. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 58–76. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24638-1_4
21. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: Highly-efficient and composable password-protected secret sharing (or: how to protect your bitcoin wallet online). In: 2016 IEEE European Symposium on Security and Privacy, pp. 276–291 (2016)
22. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 233–253. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45608-8_13
23. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 180–189. ACM (2005)
24. 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). doi:10.1007/978-3-540-45146-4_7
25. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_3
26. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_16
27. Xu, J., Hu, X.-X., Zhang, Z.-F.: Round-optimal password-based group key exchange protocols in the standard model. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 42–61. Springer, Heidelberg (2015). doi:10.1007/978-3-319-28166-7_3