The Fairy-Ring Dance: Password Authenticated Key Exchange in a Group

Feng Hao[*], Xun Yi[†], Liqun Chen[‡], Siamak F. Shahandashti[§]

Abstract

In this paper, we study Password Authenticated Key Exchange (PAKE) in a group. First, we present a generic "fairy-ring dance" construction that transforms any secure two-party PAKE scheme to a group PAKE protocol while preserving the round efficiency in the optimal way. Based on this generic construction, we present two concrete instantiations based on using SPEKE and J-PAKE as the underlying PAKE primitives respectively. The first protocol, called SPEKE+, accomplishes authenticated key exchange in a group with explicit key confirmation in just two rounds. This is more round-efficient than any existing group PAKE protocols in the literature. The second protocol, called J-PAKE+, requires one more round than SPEKE+, but is computationally faster. Finally, we present full implementations of SPEKE+ and J-PAKE+ with detailed performance measurements. Our experiments suggest that both protocols are feasible for practical applications in which the group size may vary from three to several dozen. This makes them useful, as we believe, for a wide range of applications – e.g., to bootstrap secure communication among a group of smart devices in the Internet of Things (IoT).

1 Introduction

The need for secure group communication naturally rises in many practical applications. For example, in the Internet of Things (IoT), a plethora of smart devices with computing and network capabilities interconnect with each other based on an existing Internet infrastructure. The inter-device communication often carries security-sensitive information, which should be protected. On the other hand, the open nature of the Internet invites all sorts of attacks – for example, an attacker may intercept a data packet, maliciously change its content and pass it on.

[*]Newcastle University, UK
[†]RMIT University, Australia
[‡]HP Labs, UK
[§]Newcastle University, UK

Just as secure communication between two remote parties can be established by an Authenticated Key Exchange (AKE) protocol, secure communication among a group of parties can be realized by using a Group AKE protocol.

In this paper, we focus on studying Group AKE protocols based on password authentication without any PKI. In the past literature, a two-party password-based authenticated key exchange protocol is commonly known as PAKE [4]. In the multi-party setting, we will call a password-based authenticated key exchange protocol within a group of participants as Group PAKE (or GPAKE).

As compared with the extensive research on PAKE, the subject of GPAKE has received much less attention. Only a few papers have devoted to this subject (e.g., [1, 2, 19]. However, with the rise of the Internet of Things, we believe the subject of GPAKE deserves more attention. A GPAKE protocol can be naturally fit for the IoT because it does not require any Public Key Infrastructure or trusted Certificate Authority. Hence, to deploy a set of out-of-the-box smart devices that do not have any preinstalled secrets, one just needs to enter a common password onto each device to bootstrap the initial secure communication among the group of devices over an insecure Internet.

One major challenge in designing a practical GPAKE protocol is to make it as round-efficient as possible. Many GPAKE protocols require $O(n)$ rounds, which linearly increase with the number of participants [6, 9]. This can be problematic in practice. For example, for a group of 20 participants, if it requires over 20 rounds of interactions between participants over a distributed network, the latency will be determined by the slowest responder in each round. The overall latency can prove high.

Some researchers strive to improve the $O(n)$ round complexity in GPAKE [2]. One of the most notable improvements comes from Abdalla et al. at PKC'06 [2]. Their approach is to trade more computation for fewer rounds. The authors propose a GPAKE protocol that requires a constant number of 4 round. This is a significant improvement over the $O(n)$ round complexity in other works.

In this paper, we aim to improve the round-efficiency even further. Our contributions are summarized as follows:

1. We propose a generic construction to extend any secure two-party PAKE protocol with explicit key confirmation to a multi-party GPAKE protocol with explicit key confirmation without adding any more rounds. This serves to preserve round efficiency in the optimal way.

2. Based on the generic construction, we propose two concrete GPAKE protocols called SPEKE+ and J-PAKE+, which require 2 and 3 rounds respectively. These are signifiant improvements over Abdalla et al's 4-round [2].

3. We present concrete implementations of the SPEKE+ and J-PAKE+ protocols with detailed latency measurements. (The open source code of our implementations can be found at the end of the paper.)

2 A generic construction

We first describe a generic construction that extends any secure two-party PAKE to a multi-party GPAKE while preserving optimal round-efficiency. Our construction works similarly to the fairy-ring dance[1] in two aspects. First, all n participants form a logical ring (see Figure 1), each indexed in the circular order from 1 to n. In practice, participants can be indexed by the sequence of being added to the group or by using the canonical order of their unique identifiers. Second, all participants fully interact with everyone else in the ring. In electronic communication, this can be easily realized by broadcasting.

To explain the basic intuition, we take "dancing" as a physical analogy to "communicating". When two partners "dance" together, they do a handshake to establish a common secret key (based on a two-party PAKE protocol). As they constantly rotate in the circular order to "dance" with the next partner, all participants establish shared keys with everyone else in the group. Since all pairs "dance" independently, the pairwise shared keys can be established simultaneously through broadcasting messages in a network.

The generic construction consists of two processes: 1) establishing pairwise keys based on two-party PAKE (which is represented by the inner dashed lines in Figure 1; 2) creating a common group key based on a variant of the Burmester-Desmedt protocol [3] (which is represented by the outer solid line in Figure 1). The two processes can start in parallel. Once the pairwise keys are established in the inner dashed lines, they then are used (via MAC signatures) to authenticate each participant's contribution in the group key agreement protocol at the outer ring. If every participant finds that everyone else in the group is authenticated, then the whole group is authenticated.

We term those who know the correct password as "authentic participants". By contrast, attackers do not know the password. In this paper, we do not consider the case that authentic participants maliciously disrupt each other's communication.

In the following two sections, we will present two concrete instantiations of the generic construction based on using SPEKE and J-PAKE as the underlying PAKE protocols. We choose these two, because they are among the few PAKE schemes that have been implemented and used in practical applications. SPEKE has been used in Blackberry phones and been included into the IEEE P1363.2 and ISO/IEC 11770-4 standards. J-PAKE has been used in Mozilla Firefox sync and it has recently been adopted by ISO/IEC 11770-4.

3 First protocol: SPEKE+

In this section, we will present an instantiation of our generic construction based on using SPEKE [14] as the underlying PAKE protocol. The result is a new

[1] The fairy-ring dance is a traditional Scottish country dance in which ladies and gentlemen form a circle, dance in front and round the partner, rotate in the circular order to repeat dancing with the next partner. Thus, one dances with everyone else in the ring.

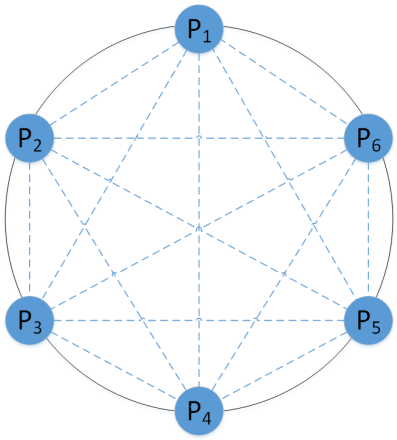


Figure 1: An example of six people in a fairy-ring dance. The outer line indicates a circular ring that all dancers form and the dashed lines indicate interactions between the dancers.

protocol, which we call SPEKE+.

3.1 Protocol setting

Let p be a large safe prime, $p = 2q + 1$ where q is also a prime. Denote G_q as the subgroup of $\mathbb{Z}_p^*$ with prime order q. Let g be a generator of the subgroup (any non-identity element in the subgroup can serve as a generator). We assume there are a group of n participants ($n \geq 3$) who share a common password s and wish to establish a common group key. All participants agree the domain parameters (p, q, g) before starting the protocol. Due to the human being's inability to memorise long passwords, we assume s has low entropy.

The two-party SPEKE protocol as defined in IEEE P1363.2 uses the function f to map a password s to a group element: $f(s) = (H(s))^2 \bmod p$ where H is a hash function. Let $g_s = f(s)$. Essentially, g_s has an order of q and could serve as a generator of G_q.

3.2 Protocol specification

The SPEKE+ protocol performs password-based authenticated group key exchange with explicit authentication in the following two rounds. (Unless stated otherwise, all the modular operations are performed with respect to the modulus p. Hence, the explicit 'mod$p$' notation is omitted for simplicity.)

Round 1 *Every participant P_i selects $x_i \in_R [1, q-1]$, $y_i \in_R [0, q-1]$ and broadcasts $g_s^{x_i}$, g^{y_i} together with a zero-knowledge proof, denoted as* ZKP$\{y_i\}$, *for proving the knowledge of the exponent y_i.*

We define $z_i = y_{i+1} - y_{i-1}$. Every one is able to compute $g^{z_i} = g^{y_{i+1}}/g^{y_{i-1}}$. When the above round finishes, every participant P_i checks that:

- $g^{z_i} \neq 1$ for $i = 1, \cdots, n$ (this ensures g^{z_i} can be used as a valid generator in the next round);
- $g_s^{x_j} \neq 1$ and $g_s^{x_j} \neq p-1$ for $j \in \{1, \cdots, n\} \setminus \{i\}$ (this ensures that the received values do not fall within the small subgroup of order two [14]);
- the received ZKP$\{y_j\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid.

We adopt the Schnorr non-interactive ZKP to prove the knowledge of the exponent without leaking any information about its value [18]. The non-interactive Schnorr technique is obtained from the three-pass Schnorr identification scheme by applying the standard Fiat-Shamir heuristics [10].

Round 2 *Every participant P_i broadcasts $(g^{z_i})^{y_i}$ and a zero-knowledge proof,* ZKP$\{\widetilde{y_i}\}$*, for proving the equality of the discrete logarithm of $(g^{z_i})^{y_i}$ to the base g^{z_i} and the discrete logarithm of g^{y_i} to the base g. Furthermore, P_i computes two pairwise keys: 1) $\kappa_{ij}^{\mathrm{MAC}} = H(g_s^{x_i x_j} \| \text{“MAC”})$ for authentication; 2) $\kappa_{ij}^{\mathrm{KC}} = H(g_s^{x_i x_j} \| \text{“KC”})$ for key confirmation. Let $A_{ij} = g^{y_i} \| \mathrm{ZKP}\{y_i\} \| (g^{z_i})^{y_i} \| \mathrm{ZKP}\{\widetilde{y_i}\}$. P_i broadcasts $t_{ij}^{\mathrm{MAC}} = \mathrm{HMAC}(\kappa_{ij}^{\mathrm{MAC}}, A_{ij})$ (message authentication tag), and $t_{ij}^{\mathrm{KC}} = \mathrm{HMAC}(\kappa_{ij}^{\mathrm{KC}}, \text{“KC”} \| i \| j \| g_s^{x_i} \| g_s^{x_j})$ (key confirmation string) for $j \in \{1, \cdots, n\} \setminus \{i\}$.*

When the above round finishes, each participant P_i checks:

- the received ZKP$\{\widetilde{y_j}\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid;
- the received key confirmation strings t_{ji}^{KC} for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid[2];
- the received message authentication tags t_{ji}^{MAC} for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid.

We adopt the Chaum-Pedersen non-interactive ZKP [8] to prove the equality of the discrete logarithm of $(g^{z_i})^{y_i}$ to the base g^{z_i} and the discrete logarithm of g^{y_i} to the base g without revealing any information about the secret y_i.

Key computation. Upon successful verifications as detailed above, every participant P_i proceeds to compute a group key K_i based on Burmester-Desmedt's cyclic key computation technique [3] (the definition of the index i is cyclic, so the index before 1 is n, and conversely, the index after n is 1).

$$\begin{aligned} K_i = & \ (g^{y_{i-1}})^{n \cdot y_i} \cdot (g^{z_i y_i})^{n-1} \cdot (g^{z_{i+1} y_{i+1}})^{n-2} \cdots (g^{z_{i-2} y_{i-2}}) \\ = & \ g^{y_1 \cdot y_2 + y_2 \cdot y_3 + \cdots + y_n \cdot y_1} \end{aligned} \tag{1}$$

Under the assumption that authentic participants honestly follow the protocol, it is easy to verify that all K_i values $(i = 1, \ldots, n)$ are identical [3].

[2]When there are two indices, say i and j, in the subscript, one can interpret the first index indicating the sender and the second index representing the intended recipient.

Hence, after two rounds, the group key exchange is completed with explicit authentication and key confirmation. Here, the explicit authentication and key confirmation are equivalent concepts. However, in key confirmation, there are two levels of contexts that we need to clarify. In the context of pairwise PAKE sessions, the tag t_{ij}^{KC} serves to provide explicit key confirmation, and hence explicit authentication if the other party has used the correct password. In the context of the group key establishment, if every participant finds that everyone else in the ring is explicitly authenticated, then the explicit authentication for the whole group is achieved. Under the assumption that an authentic participant honestly follows the key computation formula in Equation 1, the values of their keys K_i (for $i = 1, \ldots, n$) are guaranteed to be the same. Therefore, the assurance on explicit authentication and key confirmation at the two-party level naturally extends to the group level.

4 Second protocol: J-PAKE+

In this section, we will present a second instantiation of the generic "fairy-ring" construction based on using J-PAKE as the underlying PAKE protocol. We call the resultant protocol J-PAKE+.

4.1 Protocol setting

The cyclic group setting (p, q, g) is similar as before, except that p does not need to be a safe prime. This allows more flexible choices of groups, especially those with short exponents, e.g., DSA-like groups. The J-PAKE+ protocol operates in the subgroup of Z_p^* with prime order q. As an example, for 2048-bit p, we choose 256-bit q instead of 2047-bit q as in SPEKE+. This has a significant impact on the performance as we will discuss in Section 5.

We use the same symbol s to denote the low-entropy password. Here, we assume the value of s falls within the range of $[1, q-1]$. For 256-bit q, the range should be sufficiently large to accommodate values of all practical passwords. This explicit range is needed for the precise definition of the on-line dictionary attack resistance [11], since $s + m \cdot q$ for $m \in \mathbb{Z}$ are all equivalent values on the exponent.

4.2 Protocol specification

In the original J-PAKE protocol [11], two parties complete authenticated key exchange in two rounds with *implicit* authentication, or in three rounds with *explicit* authentication. Here we extend J-PAKE from two-party to multi-party with *explicit* authentication within three rounds. Again we omit the 'mod$p$' notation in the protocol specification for simplicity.

Round 1 *Every participant* P_i *selects* $a_{ij} \in_R [0, q-1]$ *and* $b_{ij} \in_R [1, q-1]$ *for* $j \in \{1, \cdots, n\} \setminus \{i\}$. *Accordingly,* P_i *broadcasts* $g^{a_{ij}}$ *and* $g^{b_{ij}}$, *together with* ZKP$\{a_{ij}\}$ *and* ZKP$\{b_{ij}\}$ *to prove the knowledge of the exponents* a_{ij} *and* b_{ij}.

In addition, P_i selects $y_i \in_R [0, q-1]$ and broadcasts g^{y_i} together with $\mathrm{ZKP}\{y_i\}$ *to prove the knowledge of the exponent y_i.*

When this round finishes, every participant is able to compute $g^{z_i} = g^{y_{i+1}}/g^{y_{i-1}}$ for $i = 1, \ldots, n$. Every participant P_i performs the following checks:

- $g^{z_i} \neq 1$ for $i = 1, \ldots, n$ (this ensure g^{z_i} can be used as a valid generator in the next round);
- the received $\mathrm{ZKP}\{a_{ji}\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid;
- the received $b_{ji} \neq 1$ for $j \in \{1, \cdots, n\} \setminus \{i\}$;
- the received $\mathrm{ZKP}\{b_{ji}\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid;
- the received $\mathrm{ZKP}\{y_j\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$.

Note that P_i does not need to verify all ZKPs – only those in which P_i is the intended recipient. Same as before, when there are two indices in the subscript, e.g., $\mathrm{ZKP}\{a_{ji}\}$, the first index can be regarded as indicating the sender and the second representing the intended recipient.

Round 2 *Every participant P_i computes and broadcasts $\beta_{ij} = (g^{a_{ij}+a_{ji}+b_{ji}})^{b_{ij} \cdot s}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$, together with* $\mathrm{ZKP}\{b_{ij} \cdot s\}$ *to prove the knowledge of the exponent $b_{ij} \cdot s$.*

When this round finishes, each participant P_i checks that:

- validation of $\mathrm{ZKP}\{b_{ji} \cdot s\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$.

Round 3 *Every participant P_i broadcasts $(g^{z_i})^{y_i}$, and* $\mathrm{ZKP}\{\widetilde{y_i}\}$ *for proving the equality of the discrete logarithm of $(g^{z_i})^{y_i}$ to the base g^{z_i} and the discrete logarithm of g^{y_i} to the base g. Let $K_{ij} = (\beta_{ji}/g^{b_{ij} \cdot b_{ji} \cdot s})^{b_{ij}}$. Each participant P_i derives two pairwise session keys from K_{ij}: 1) $\kappa_{ij}^{\mathrm{MAC}} = H(K_{ij} \| \text{“MAC”})$ for authentication; 2) $\kappa_{ij}^{\mathrm{KC}} = H(K_{ij} \| \text{“KC”})$ for key confirmation. Let $A_{ij} = g^{y_i} \| \mathrm{ZKP} y_i \| (g^{z_i})^{y_i} \| \mathrm{ZKP}\{\widetilde{y_i}\}$. P_i broadcasts $t_{ij}^{\mathrm{MAC}} = \mathrm{HMAC}(\kappa_{ij}^{\mathrm{MAC}}, A_{ij})$ (message authentication), and $t_{ij}^{\mathrm{KC}} = \mathrm{HMAC}(\kappa_{ij}^{\mathrm{KC}}, \text{“KC”} \| i \| j \| g^{a_{ij}} \| g^{b_{ij}} \| g^{a_{ji}} \| g^{b_{ji}})$ (key confirmation) for $j \in \{1, \cdots, n\} \setminus \{i\}$.*

When this round finishes every participant P_i checks that:

- the received $\mathrm{ZKP}\{\widetilde{y_j}\}$ for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid;
- the received key confirmation strings t_{ji}^{KC} for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid (optional);
- the received message authentication tags t_{ji}^{MAC} for $j \in \{1, \cdots, n\} \setminus \{i\}$ are valid.

Key computation. If all verifications in the above round are successful, every participant P_i computes the group key K_i, using the same Burmester-Desmedt formula [3] as used in SPEKE+ (see Equation 1).

The computed K_i values ($i = 1, \cdots, n$) are guaranteed to be the same if the authentic participants honestly follow the key computation formula in Equation 1. Hence, after three rounds, the password-based authenticated group key exchange protocol is completed with explicit authentication (and key confirmation).

5 Implementation

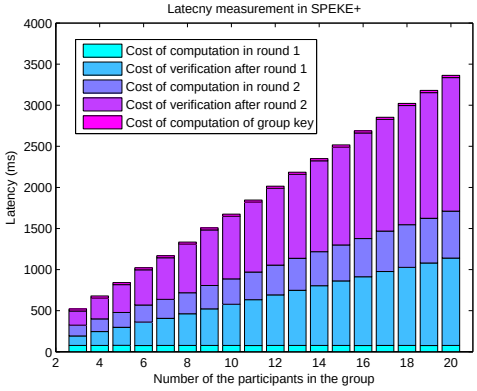
To demonstrate the practical feasibility, we implemented both protocols in Java on a PC (2.93 GHz CPU, 4 GB RAM) running 64-bit Windows 7. We use the standard Java *BigInteger* class to implement all the modular exponentiations.

Group setting. In the implementation of both protocols, we choose the 2048-bit group setting. In SPEKE+, we specify the safe prime p to be the same value as defined for SRP-6 in IETF RFC 5054. By definition, $q = (p-1)/2$. Hence, q is 2047-bit. The generator for the subgroup G_q is 3. As for J-PAKE+, we use the same group setting as specified for the 2048-bit DSA. In this setting, p is 2048-bit, and q is 256-bit.

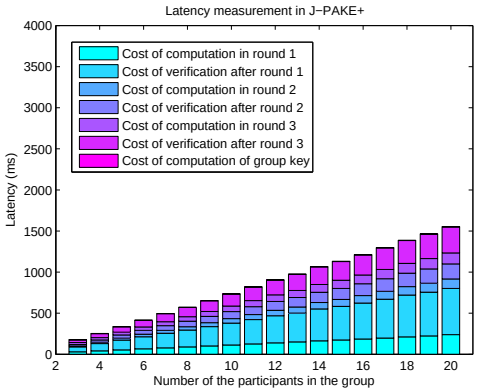
From the group size of 3 up to 20 participants, we measure the latency of computation at each round of the protocol execution. The measurement is done by repeating the same experiment thirty times and taking the average values. The results of the latency measurements are summarized in Figure 2a and Figure 2b for SPEKE+ and J-PAKE+ respectively.

As shown in Figure 2, the total computational latency in both protocols increases linearly with the size of the group. For a size of three, the total computational latency is 0.5 seconds in SPEKE+ and 0.18 seconds in J-PAKE+. For a group size of twenty, the total latency increases to about 3 seconds in SPEKE+ and 1.5 seconds in J-PAKE+. Based on linear extrapolation, the total latency for a group of fifty would be 8.4 seconds in SPEKE+ and 3.3 seconds in J-PAKE. In terms of computation, J-PAKE+ is faster than SPEKE+. However, in terms of round efficiency, SPEKE+ requires one fewer round than J-PAKE+, hence incurs lower communication latency. Overall, for small to medium group sizes, both protocols seem sufficiently efficient for practical applications.

Cost breakdown in SPEKE+. A breakdown of the cost components in SPEKE+ is summarized in Table 1. In Round 1, it takes two exponentiations to compute $g_s^{x_i}$ and g^{y_i}, and one additional exponentiation to compute a Schnorr non-interactive ZKP. After Round 1, every participant needs to verify $n-1$ Schnorr ZKPs. For each Schnorr ZKP, it takes one exponentiation to verify the order of X and one to compute $g^r \cdot X^h$ using the simultaneous computation technique (see [17]). However, since the standard *BigInteger* class in Java does not support simultaneous computation, we compute g^r and X^h in two separate operations. The size of h is 256-bit (because we use SHA-256 to implement the hash function). Since the cost of the modular exponentiation is roughly



(a) Latency measurement per participant in SPEKE+



(b) Latency measurement per participant in J-PAKE+

Figure 2: Performance evaluation

linear to the bit length of the exponent, the cost of computing X^h is roughly $256/2047 \approx 1/8$ of a full exponentiation. Hence, the total cost of verifying the $n-1$ Schnorr ZKPs is $(n-1) \times 2.125$ full exponentiations. In Round 2, it needs one exponentiation to compute $(g^{z_i})^{y_i}$, two to compute a Chaum-Pedersen ZKP, and $n-1$ exponentiations to compute $g^{x_i x_j}$ with all the others in the group ($j \neq i$). After Round 2, it requires roughly $1 + 2 \times 1.125 = 3.25$ full exponentiations to verify each of the $n-1$ Chaum-Pederen ZKPs. Finally, the group key computation requires one full exponentiation (see Table 1).

In Figure 3a, we compare the practical measurements of the computational costs in SPEKE+ with the theoretical estimates. The theoretical estimates are based on counting the number of modular exponentiations (see Table 1) and the cost of performing one modular exponentiation. Through experiments, we

	Cost breakdown	Complexity	No of exponentiations
1	Computation in R1	$O(1)$	3
2	Verification after R1	$O(n)$	$(n-1) \times 2.215$
3	Computation in R2	$O(n)$	$3 + (n-1) \times 1$
4	Verification after R2	$O(n)$	$(n-1) \times 3.25$
5	Compute group key	$O(1)$	1

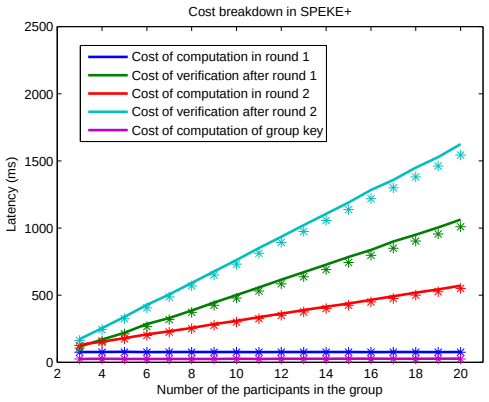
Table 1: Analysis of computational cost in SPEKE+

	Cost breakdown	Complexity	No of exponentiations
1	Computation in R1	$O(n)$	$2 + (n-1) \times 4$
2	Verification after R1	$O(n)$	$(n-1) \times 9$
3	Computation in R2	$O(n)$	$(n-1) \times 2$
4	Verification after R2	$O(n)$	$(n-1) \times 4$
5	Computation in R3	$O(n)$	$5 + (n-1) \times 2$
6	Verification after R3	$O(n)$	$(n-1) \times 5$
7	Compute group key	$O(1)$	1

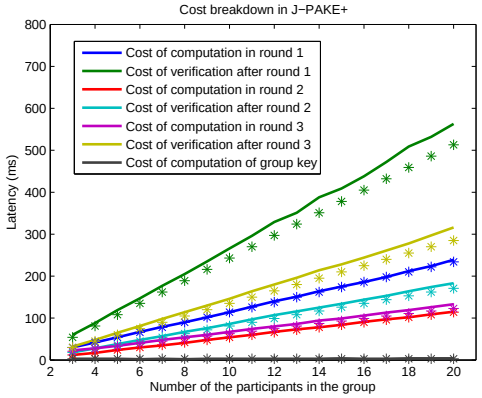
Table 2: Analysis of computational cost in J-PAKE+

determine that the cost of performing one full exponentiation in SPEKE+ is 25 ms. All experimental measurements are plotted as solid lines in Figure 3a, while the theoretical estimates are plotted as '*' markers. The practical and theoretical results are found to match. In some cases, the practical measurements are slightly higher than the theoretical estimates. This is because in the theoretical estimates, we only consider the cost of modular exponentiations. However, there are other cost components, such as the computation and verification of HMAC tags, which are negligible when the group size is small, but gradually become noticeable when there are many participants involved.

Cost breakdown in J-PAKE+. We perform a similar analysis on the cost breakdown in J-PAKE+. The results are summarized in Table 2. It is worth noting that in J-PAKE+, the exponent is relatively short, only 256-bit, while in SPEKE+, the exponent is 2047-bit. This means for the same 2048-bit modulus p, the cost of one exponentiation in SPEKE+ is about 8 times the cost in J-PAKE+. Through experiments, we find it takes 3 ms to do one exponentiation in J-PAKE+ (recall that it takes 25 ms in SPEKE+). We plot all experimental measurements in J-PAKE+ in Figure 3b as solid lines, and the theoretical estimates as '*' markers. Similar as in SPEKE+, the practical measurements in J-PAKE are found to be consistent with the theoretical estimates. In some cost components, the practical latencies are slightly higher than the theoretical estimates. This is because the theoretical analysis only considers modular exponentiations as pre-dominant costs and do not take into account the symmetric key operations such as the computation of the HMAC tags.



(a) Cost analysis in SPEKE+



(b) Cost analysis in J-PAKE+

Figure 3: Analysis of computational cost. The solid lines represent experimental data and the '*' the theoretical estimates.

6 Security analysis

In this section, we analyse the security of SPEKE+ and J-PAKE+ by naturally extending the security requirements (namely, offline dictionary attack resistance, forward secrecy, known-session security and online dictionary attack resistance [11]) from a two-party setting to a multi-party setting.

6.1 Security of J-PAKE+

J-PAKE+ is built on J-PAKE, the Burmester-Desmedt group key agreement protocol, and Schnorr non-interactive ZKP. J-PAKE has been proved to fulfil the above requirements under the assumptions that the Decision Diffie-Hellman problem is intractable and that the underlying Schnorr non-interactive ZKP is secure [11]. The Schnorr NIZKP is obtained from the provably secure Schnorr interactive identification scheme through Fiat-Shamir heuristics (which requires a publicly secure hash function).

Off-line Dictionary Attack: In an off- line dictionary attack, an adversary exhaustively tries all possible passwords in a dictionary in order to determine the password of the client on the basis of the exchanged messages.

J-PAKE+ is directly resting on J-PAKE. In J-PAKE+, all the pairwise sessions are *independently* established using J-PAKE. If J-PAKE+ is insecure against off-line dictionary attack, at least one J-PAKE session is insecure against off-line dictionary attack. In other words, if the underlying J-PAKE is secure against off-line dictionary attack, we can conclude that J-PAKE+ is also secure against off-line dictionary attack.

Forward Secrecy: Forward secrecy ensures that a session key derived from the long-term key cannot be compromised if the long-term key is compromised in the future. In J-PAKE+, the long-term key is the password.

In J-PAKE+, two kinds of session keys are established at the end of the protocol. One is the pairwise session key established between each pair of participants through J-PAKE. Another is the group key established by the Burmester-Desmedt group key agreement.

If J-PAKE has forward secrecy, the first kind of session key cannot be compromised even if the password is compromised in the future.

If the Burmester-Desmedt group key agreement protocol is secure against the passive attack (i.e., the attacker cannot interact with any of the parties involved, attempting to determine the group key based upon observed communication data), the second kind of session key cannot be compromised even if the password is compromised in the future.

In summary, If J-PAKE has forward secrecy and the Burmester-Desmedt group key agreement protocol is secure against the passive attack, J-PAKE+ has forward secrecy.

Known-Session Security: The known-session security for J-PAKE+ requires 1) the disclosure of one J-PAKE session does not affect other J-PAKE sessions

and 2) the disclosure of one group session key does not affect the secrecy of the group keys in other sessions.

If the underlying J-PAKE has known-session security, J-PAKE+ satisfies the first requirement. In addition, all secrets that contribute to the Burmester-Desmedt group key computation are randomly generated in each run of the protocol. This results that the group session keys are independent of each other and thus J-PAKE+ satisfies the second requirement.

Online-Dictionary Attack: In on-line dictionary attack, an adversary simply attempts to pass the authentication repeatedly, trying each possible password.

Given that J-PAKE limits an on-line attacker to guess only "one" password in one protocol execution, the number of passwords an on-line attacker can guess in J-PAKE+ in a group setting clearly depends on the number of on-line attackers who are present in the group. Out of the n participants, we assume α of them are legitimate users and $(n - \alpha)$ are illegitimate (i.e., attackers). We use η to denote the number of password guesses by the attacker in one protocol execution. By interacting with each of the α users, the attackers can test in total $\eta = \alpha \times (n - \alpha)$ passwords in one run. For a fixed group size n, η is maximum when $\alpha = n - \alpha$, i.e., half of the participants are attackers.

We will use some concrete figures to illustrate the effect of the on-line dictionary attack. We use the telephone conference call as an example. Assume a group of 20 callers who try to set up a secure group session key. We assume the worst case that half of the participants are attackers. Thus, the maximum number of passwords that can be guessed in one run is $\eta = 10 \times 10 = 100$. We assume the password is an eight-digit PIN, which only has an entropy of 26 bits. (We only consider PIN here as the user can conveniently enter that on the phone keypad.) For the on-line attack to successfully discover the password, the protocol needs to be executed on average $10^8/100/2 = 500,000$ times. In practice, this on-line attack is unlikely to succeed as the users normally refuse further engagement if the authentication has failed consecutively for a limited number of times (say 3).

Key Confirmation: Key confirmation in key agreement is the property whereby one party is assured that another party actually have derived the same secret key.

Let us examine the "fairy-ring" construction that transforms J-PAKE to J-PAKE+. In this construction, the inner pairwise interactions start concurrently with the outer group key establishment. The resultant pairwise keys obtained from the inner PAKE sessions are used to authenticate the group key created at the outer ring. The group key establishment at the outer ring is based on a variant of the Burmester-Desmedt scheme. We make two modifications to the original 2-round Burmester-Desmedt protocol. First, we add a Schnorr ZKP in Round 1 to make the sender prove the knowledge of the ephemeral private key. Second, we add a Chaum-Pedersen ZKP in Round 2 to make the sender prove the equality of the exponent to the one in the previous round. These ZKPs serve to enforce that 1) the ephemeral public keys sent in the first round are "independently" generated (e.g., one cannot just replay another participant's

public key as the ZKP verification will fail); 2) all data sent in the protocol are well-formed according to the specification in the original Burmester-Desmedt scheme [3]. The authenticity of all data is further assured (via MAC signatures) by the pairwise sessions keys created between participants. Because of these ZKPs, the security proofs in the original Burmester-Desmedt protocol are directly applicable in our construction: the security of the group key is reducible to the same Decision Diffie-Hellman problem as in the original scheme [15].

In summary, J-PAKE+ has key confirmation for the pairwise session key established between each pair of participants because MAC signatures are exchanged for authentication. J-PAKE+ also has key confirmation for the group session key established in the group because of ZKPs and MAC signatures.

6.2 Security of SPEKE+

The original SPEKE paper [14] does not provide formal reductionist proofs for the protocol. Instead, the protocol is heuristically argued to be secure. Although some subtle issues are reported [12], no major flaws seem to have been found. In practice, SPEKE has been included into the IEEE 1363.2 and ISO/IEC 11770-4 standards and used in commercial products.

The design of SPEKE+ is actually based on an enhanced version of SPEKE instead of the original scheme. The enforcement is to address the two attacks recently reported by Hao and Shahandashti [12]. The first attack exploits the lack of identities in the original specification of SPEKE. The attacker launches two parallel sessions with a victim and relays messages from one session to the other in order to pass the authentication in both sessions. In the second attack, the attacker is able to modify the key exchange message, and thus manipulate the session key but without being detected by the end users. Both attacks are addressed in the SPEKE+ protocol by including the identities (which are indices in our context) and the key exchange messages in the explicit key confirmation string. We specify the key confirmation string as "mandatory". The cost of mandating this explicit key confirmation is nearly negligible, because 1) this does not increase the number of rounds and 2) the symmetric operations of generating/verifying HMAC tags is insignificant as compared to the asymmetric counterparts in the protocol.

The structural design of SPEKE+ is a bit different from that of J-PAKE+. Each participant P_i only generates one ephemeral public key $g_s^{x_i}$ rather than one for each of the other $n-1$ participants. This is because the single public key $g_s^{x_i}$ broadcast by P_i can be used by all the rest participants in a non-interactive manner to derive the pairwise shared session keys. Hence, the extension from two-party to multi-party occurs naturally without needing to send more key exchange messages. (By contrast, J-PAKE requires one more round of user interaction than SPEKE, which tightly binds the ephemeral keys used in one J-PAKE session to be between two specific parties.)

Although SPEKE does not have formal security proofs, it is believed, to a large extent, that it satisfies the four requirements as listed above. Hence, we will analyze the security properties of SPEKE+ based on the assumption that

the underlying SPEKE is a secure PAKE scheme.

Off-line Dictionary Attack: Under the assumption that the two-party SPEKE resists off-line dictionary attack, the same property holds in SPEKE+, because P_i does not need to send any additional data to the group key exchange. (In SPEKE+, P_i needs to send $n-1$ key confirmation string instead of just one as in two-party SPEKE. This does not have effect on the off-line dictionary attack since each key confirmation string reveals just one bit to an active attacker: if the two passwords are equal. However, that affects on-line dictionary attack, as we will explain later.)

Forward secrecy: This requirement concerns the security of a past session key when the password is revealed. This essentially assumes a passive attacker (who knows the password but only passively observes) during the establishment of the past session key. We consider this requirement at two levels. At the level of the pairwise two-party interaction, each SPEKE session assures the forward secrecy in the created session key. At the level of the group key establishment, the forward secrecy in each group session key trivially holds (since the password and the group session key are completely unrelated in terms of entropy).

Known-session security: This requirement concerns the effect of a disclosed session to other sessions. In SPEKE+, the key confirmation is "mandatory". Hence a session key is only established after the key confirmation has been successfully verified; in other words, the other party has been successfully authenticated. Here, we assume authentic participants honestly follow the protocol. Hence, all ephemeral secrets are randomly and independently chosen by honest users. At the level of the pairwise two-party interaction, SPEKE assures that the disclosure of a session key does not affect the security of other sessions. At the level of the group key establishment, the disclosure of one group session key clearly does not affect other sessions, as the ephemeral secrets are freshly generated in each session by authentic participants.

On-line dictionary attack: This is similar to the analysis in J-PAKE+. Assume that SPEKE limits an on-line attacker to guess only "one" password in one protocol execution and that there are α authentic participants and $n-\alpha$ on-line attackers. The number of passwords the on-line attackers can test in one execution of SPEKE+ is $\eta = \alpha \times (n-\alpha)$.

6.3 Summary

Based on the security analysis above, under the assumption that the underlying SPEKE and J-PAKE protocol are secure PAKE primitives, we conclude that:

- Both SPEKE+ and J-PAKE+ provide off-line dictionary attack resistance.
- They both provide forward secrecy at two levels: the level of pairwise session keys established from the two-party PAKE interaction and the level of the group session key created from the Burmester-Desmedt group key computation method.

- They both provide known-session security at the same two levels as above.
- They both limit on-line attackers to test at most $\alpha \times (n - \alpha)$ passwords, where α is the number of authentic participants and n is the total number of participants.

7 Related work

Existing GPAKE protocols are generally designed in two approaches. The first approach adopts a similar design as the Encrypted Key Exchange (EKE) protocol, which was proposed in 1992 by Belloven and Merrit [4]. In particular, it assume an "ideal cipher" that is able to encrypt data using a low-entropy password as the encryption key without leaking any information about the data or the key. This assumption was introduced in the theoretical analysis of EKE in 2000 [5]. Examples of such GPAKE protocols include Bresson et al.'s at ASIACRYPT'02 [6], Dutta and Barua's at IJNS'06 [9], Abdalla et al.'s at PKC'06 [2], Wan et al.'s at ICICS'07 [20], and He et al.'s at IJCS'11 [13]. The second approach involves using a password-derived element as a secret generator in a cyclic group, just like in the original design of the SPEKE protocol in 1996 by Jablon [14]. The GPAKE protocol designed by Tang and Choo at ACNS'06 [19] is an example of this approach.

However, practical deployment of the above protocols has been rather limited so far. There are several issues in limiting the implementation. First, although the "ideal cipher" is a common assumption made in the GPAKE literature, its concrete instantiation has yet to be specified [2, 6, 9, 13, 20]. In 2003, Boyd and Mathuria [7] examined the "ideal cipher" assumption and found it could not be realized by any existing symmetric ciphers. In 2006, Zhao et al. reviewed several proposals in instantiating an "ideal cipher" and found none of them was secure [21]. Second, while some protocols try to avoid the "ideal cipher" assumption, they instead assume a trusted third party to define two *independent* generators [16]. The involvement of a trusted third party defeats the fundamental goal in GPAKE. Finally, GPAKE protocols [6, 20] commonly require $O(n)$ rounds of interactions between participants, which is rather inefficient. Dependence on interactions over a distributed network (waiting for each other's output for the next operation) can easily cause a bottleneck in the overall performance.

Among the existing GPAKE protocols, the Tang-Choo protocol [19] seems the closest to being ready-to-implement. The protocol does not rely on any "ideal cipher" or trusted third party. It follows a SPEKE-styled approach: the modulus is defined to be a safe prime and the password is used to derive a generator for the cyclic subgroup group. The protocol has 4 rounds with explicit key confirmation. The computational complexity per participant is $O(1)$.

As compared with the 4-round Tang-Choo protocol, our SPEKE+ and J-PAKE+ protocols are more round-efficient, since they require only 2 and 3 rounds respectively. However, the Tang-Choo protocol has the advantage of keeping a constant amount of computation per participant while SPEKE+ and

J-PAKE+ require $O(n)$ computation. This is because of our strategy of achieving the optimal round efficiency with the trade-off on computation. Despite the linear $O(n)$ complexity, experiments in Section 5 show that SPEKE+ and J-PAKE+ are sufficiently fast for small to medium sized groups.

8 Conclusion

In this paper, we explore the practical limit of the best achievable round efficiency in Group PAKE protocols. We first present a generic construction that transforms a secure two-party PAKE (with explicit key confirmation) to a multi-party GPAKE (with explicit key confirmation) without adding any more rounds. We then show two concrete instantiations of the generic construction, called SPEKE+ and J-PAKE+. The SPEKE+ protocol needs only two rounds, which is probably the best round-efficiency one can hope for. J-PAKE+ requires one more round, but is computationally faster than SPEKE+. Experiment shows that for a medium size of 20 participants in the group, the total computational latency in SPEKE+ is about 3 seconds, while in J-PAKE+ is about 1.5 seconds. This suggests that both protocols are feasible for secure communication in practice for a small-to-medium sized group.

Availability

The source code of our prototype implementations of SPEKE+ and J-PAKE+ is freely available at: `https://github.com/FairyRing/SourceCode`

References

[1] M. Abdalla, C. Chevalier, L. Granboulan, D. Pointcheval, "Contributory Password-Authenticated Group Key Exchange with Join Capability," CT-RSA'11, LNCS 6558, pp. 142-160, 2011.

[2] M. Abdalla, E. Bresson, O. Chevassut, D. Pointcheval, "Password-Based Group Key Exchange in a Constant Number of Rounds," PKC'06, LNCS 3958, pp. 427-442, 2006.

[3] M. Burmester, Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," EUROCRYPT'95, LNCS 950, pp. 275-286, 1995.

[4] S. Bellovin and M. Merritt, "Encrypted Key Exchange: password-based protocols secure against dictionary attacks," Proc. IEEE Sym. Research in Security and Privacy, 1992.

[5] M. Bellare, D. Pointcheval, P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Eurocrypt'00, LNCS 1807, pp. 139–155, 2000.

[6] E. Bresson, O. Chevassut, D. Pointcheval, "Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks," ASIACRYPT'02, LNCS 2501, pp. 497-514, 2002.

[7] C. Boyd, A. Mathuria, *Protocols for authentication and key establishment*, Springer-Verlag, 2003.

[8] D. Chaum and T.P. Pedersen, "Transferred Cash Grows in Size," EUROCRYPT'92, LNCS 658, pp. 390-407, 1993.

[9] R. Dutta, R. Barua, "Password-based Encrypted Group Key Agreement," *International Journal of Network Security*, 3(1):23-34, 2006.

[10] A. Fiat, A. Shamir, "How to Prove Yourself: Practical Solution to Identification and Signature Problems," CRYPTO'86, LNCS 263, pp. 186-189, 1987.

[11] F. Hao, P. Ryan, "J-PAKE: Authenticated Key Exchange Without PKI," *Springer Trans. on Computational Science XI*, LNCS 6480, pp. 192-206, 2010.

[12] F. Hao, S. Shahandashti, "The SPEKE Protocol Revisted," SSR'14, LNCS 8893, PP. 26-38, 2014.

[13] D. He, C. Chen, M. Ma, S. Chan, J. Bu, "A Secure and Efficient Password Authenticated Group Key Exchange Protocol for Mobile Ad Hoc Networks." *International Journal of Communication Systems*, 26(4):495-504, 2011.

[14] D. Jablon, "Strong Password-only Authenticated Key exchange," *ACM Computer Communications Review*, 26(5):5-26, October 1996.

[15] J. Katz, M. Yung, "Scalable Protocols for Authenticated Group Key Exchange," CRYPTO'03, LNCS 2729, pp. 110-125, 2003.

[16] J.O. Kwon, I.R. Jeong, D.H. Lee, "Provably-Secure Two-Round Password-Authenticated Group Key Exchange in the Standard Model," IWSEC'06, LNCS 4266, pp. 322-336, 2006.

[17] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1996.

[18] C.P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, 4(3):161-174, 1991.

[19] Q. Tang, R. Choo, "Secure Password-Based Authenticated Group Key Agreement for Data-Sharing Peer-to-Peer Networks," ACNS'06, LNCS 3989, pp. 162-177, 2006.

[20] Z. Wan, R.H. Deng, F. Bao, B. Preneel, "nPAKE+?: A Hierarchical Group Password-Authenticated Key Exchange Protocol Using Different Passwords," ICICS'07, LNCS 4861, pp. 31-43, 2007.

[21] Z. Zhao, Z. Dong, Y. Wang, "Security analysis of a password-based authentication protocol proposed to IEEE 1363," *Theoretical Computer Science*, 352(1):280-287, 2006.