PASSWORD AUTHENTICATED KEY EXCHANGE: FROM TWO PARTY METHODS TO GROUP SCHEMES

STEPHEN MELCZER, TARAS MYCHASKIW AND YI ZHANG

Abstract. This project investigates group password authenticated key exchange methods (GPAKEs). In particular, we detail the so-called 'fairy ring dance' method recently described by Hao et. al. [9] which allows for the extension of two party password authenticated key exchange methods (PAKEs) with explicit key confirmation to a group setting with an arbitrary number of users without increasing round complexity (the computational complexity, of course, increases with the number of users). This paper presents two new GPAKEs constructed through these means, based on the Dragonfly and PAK/PPK two party protocols, and includes timings comparing them to previous GPAKEs of Hao et. al. [9].

1. Introduction

Since their introduction in the 1990s, password-authenticated key exchange (PAKE) methods – also known as password-authenticated key agreement methods – have become popular for their ability to allow agents sharing a (typically low entropy) password to securely establish shared cryptographic keys (see Bellovin and Merritt [4] or Jablon [11] for early examples, and Hao and Ryan [8] for a more recent paper). Although they have been around for decades, most research on PAKEs has focused on key establishment between two parties. For our project, we have studied the problem of establishing Group PAKEs (GPAKEs) – that is, using a low entropy password shared between many agents to set up cryptographic keys. This has modern applications with the rise of the so-called 'Internet of Things', where many consumer devices connected through a local Internet connection wish to securely communicate (such schemes would allow, for instance, secure communication between a smart television, DVD player, and cable box after their owner inputs a short shared password into each upon purchase).

The major issue in designing an efficient GPAKE is to minimize the number of rounds of communication between the agents involved, as the latency of such a protocol is determined by the slowest responder in each round. A recent pre-print of Hao et. al. [9] proposes a construction which allows for the extension of any secure two-party PAKE with key confirmation to a multi-party PAKE, without adding any extra rounds of communication (if the underlying two-party PAKE only allows for key authentication, then the associated GPAKE has one extra round of communication). The authors continue on to give two explicit schemes following from this template: SPEKE+ (using two rounds of communication, adapted from the SPEKE [11] protocol) and J-PAKE+ (using three rounds of communication, adapted from the J-PAKE [8] protocol).

The structure of this document is as follows: Section 2 begins by giving a survey of classical two-party PAKEs – including explicit descriptions of the PAKEs which will be extended into the group setting and background information on the zero knowledge proof protocols these use. Section 3 starts with a description of the theoretical methodology developed by Hao et. al. [9] to extend two-party PAKEs into a group setting. After this theoretical background, we give two explicit GPAKEs (SPEKE+ and J-PAKE+) constructed by Hao et. al. using this methodology, followed by two explicit GPAKEs which we have derived through the same means (a group variant of the IEEE 802.11-2012 standard Dragonfly protocol [10], and a variant of the PAK/PPK protocol [5]). Security properties of these GPAKEs follow from the security properties of the underlying two party PAKES, in a manner described by Hao et. al. [9] and in Section 3 of this document. In Section 4 we test the practical efficacy of our new methods against the Java implementations of SPEKE+ and J-PAKE+ given by Hao et. al. [9]. Section 5 concludes with an overview of these results and possible directions for future work.

The main original contributions found in this project come from the two new group PAKEs we have constructed – see sub-Sections 2.3 and 2.4 – and timings which compare these methods against previous Java implementations of SPEKE+ and J-PAKE+ (the code for this project is available at https://github.com/twentylemon/gpake). We also survey the relevant background material on Zero Knowledge Proofs and classical PAKEs missing from Hao et. al. [9] (which had constrained space as a conference abstract), and fixed some minor Java implementation errors which could cause the timings in that paper to be slightly inaccurate.

2. Two Party Password-Authenticated Key Exchange (PAKE) [1]

The genesis of password-authenticated key exchange is widely credited to the 1992 work of Bellovin and Merrit [4], whose protocol – known as Encrypted Key Exchange, or EKE, for short – came to be known as the first PAKE (previous password based protocols, like the one proposed in 1989 by Lomas et.al. [12], contained key features of PAKEs such as the offline dictionary attack resistance detailed below, although they still relied on one party having another's public key). All PAKEs aim for two main goals: to require their users to provide a zero knowledge proof of a short password known to both parties *a priori* (that is, before the protocol has begun) and to leverage knowledge of this password to facilitate an authenticated key exchange. As password are assumed to be low entropy – for instance, they are often treated as human memorable passwords (typically assumed to be approximately 20-30 bits of entropy) – if the passwords themselves were broadcast they would need to be protected, for instance using SSL. This would require Public Key Infrastructure, such as a Trusted Authority / Certificate Authority to maintain public keys, which can be expensive. The ability to work around such infrastructure is often the point of PAKE protocols, which essentially use pre-established shared knowledge (of the common password) as an alternative to Trusted Authorities.

Indeed, it is somewhat miraculous that PAKEs – which transform a low entropy shared secret into a much larger and more complicated shared key – exist at all. Although the EKE protocol of Bellovin and Merrit was later shown to have weaknesses (see Jaspan [11], for example) its great contribution was to show that such schemes can be achieved. Due to its historical significance, we outline the Diffie-Hellman variant of the method here (an RSA variant, also by Bellovin and Merrit, was later shown to be insecure). Given a symmetric encryption function $[\cdot]_\pi$ which uses a password π shared by agents Alice and Bob as a key, the algorithm does the following:

Setup: Let α be a primitive root of $\mathbb{Z}_p$

Alice		Bob
randomly choose $x_a \in \mathbb{Z}_p^*$		randomly choose $x_b \in \mathbb{Z}_p^*$
	$\xrightarrow{[\alpha^{x_a}]_\pi}$	
	$\xleftarrow{[\alpha^{x_b}]_\pi}$	
Compute $K = \alpha^{x_a \cdot x_b}$		Compute $K = \alpha^{x_a \cdot x_b}$

Figure 1. The flow diagram for EKE

At the end, both Alice and Bob share the key $K = \alpha^{x_a \cdot x_b}$. The weaknesses of the algorithm stems from the issues discussed above: as the password has low entropy in order for the scheme to be secure the input into $[\cdot]_\pi$ must essentially look like a random number. But a 1024 bit number modulo p is not random, and a passive attacker can try candidate passwords π' to attempt to decipher $[\alpha^{x_a}]_\pi$ and immediately rule out any passwords giving a result in the range $[p, 2^{1024} - 1]$.

[1] The background information in this section, and details about SPEKE and J-PAKE, are mainly based on the presentation in Hao and Ryan [8]. The information on the Dragonfly protocol was taken from Harkins [10] and Clarke and Hao [6]. The sub-section about PAK/PPK is based on the work of Boyko et. al. [5].

Although EKE has this, and other, weaknesses, it was extremely influential and its general characteristics are reflected in many of the more advanced protocols we outline below (there is also a minor variant known as EKE2, which was shown to be secure by Bellare, Pointcheval, and Rogaway [3]). Before giving these methods, we must outline what constitutes a good measure of security for a PAKE. To begin, a secure two party PAKE satisfies each of the following properties coming from the security of general key exchange protocols:

(Offline dictionary attack resistance)
The PAKE does not leak any information to a passive or active attacker which can be used by the attacker to determine the password through a brute force search (the protocol cannot reveal a hash of the password, for instance).

(Forward secrecy for established keys)
If the password is disclosed, past session keys cannot be computed by an attacker. This implies that a *passive* attacker who knows the password he cannot learn a session key by observing communication between Alice and Bob (of course, an active attacker could establish a shared key with one of the participants as he would have access to all of their secret information).

(Known session security)
If an attacker learns all secrets of a protocol in progress, it does not reveal any information about other established sessions.

(Online dictionary attack resistance)
An active attacker can only test one password per protocol execution (this is the best that we can reasonably assume, as any attacker can randomly guess a password and run the protocol – at some point the key must be confirmed, either explicitly through the PAKE or when the key is used in some other protocol, and the attacker will know whether or not his guess was correct). This is sometimes relaxed (for instance, in the proof of SPEKE given by MacKenzie [13]) to restricting the attacker to at most a small constant number of tests per protocol execution.

In the modern literature, a full proof of security essentially requires showing that an attacker can only gain information about established keys or a shared password if he is active, and that even an active attacker can gain extremely little information (for instance, can only guess one password per protocol execution). The formal model of Bellare, Pointcheval, and Rogaway [3] is commonly used as a standard, and three of the four PAKEs discussed later in this section have been proven secure – under various assumptions, see Figure 2 – in this model (the fourth, which is the Dragonfly protocol of sub-Section 2.3, we include despite a formal proof of security as it is an IEEE 802.11-2012 standard). We refer an interested reader to the work of Abdalla et. al. [2] and MacKenzie [14] for in-depth discussions of PAKE security and attack models.

		Assumptions[a]			Complexity	
	Rounds / Flows	ROM	AAM		Communication[b]	Time[c]
J-PAKE with Schnorr	2 / 4 or 3 / 3	✗	✗	DSDH or (CSDH + DDH)	$12 \times G + 6 \times \mathbb{Z}_p$	28 exp
SPEKE	1 / 2	✗		DIDH[d]	$2 \times G$	8 exp
PPK	2 / 2	✗		DDH	$2 \times G$	6 exp

[a] *CRS*: common reference string, *ROM*: random-oracle model, *ICM*: ideal-cipher model, *AAM*: algebraic-adversary model;
[b] G: group elements, $\mathbb{Z}_p$: scalars;
[c] *exp*: number of exponentiations;
[d] *DIDH*: decision inverted-additive Diffie-Hellman assumption

FIGURE 2. Table taken from Abdalla et. al. [2] comparing the security assumptions and complexity of the methods discussed below

[Re-do table ourselves]

Some PAKEs satisfy the additional requirement that an attacker not be allowed to impersonate other users to some fixed target after obtaining (through illicit means) password verification files for those users which were stored by the target. The schemes with this additional property are known as augmented PAKEs, although some (for instance, Hao et. al. [9]) have argued that such a requirement is not useful as the low entropy of the password means that it will soon be discovered through an offline dictionary attack on the verification files. Nevertheless, augmented variants exist for a number of balanced PAKEs (for example Augmented-EKE for EKE, B-SPEKE for SPEKE and PAK-X for PAK/PPK).

2.1. **SPEKE.** A more advanced PAKE which we consider is the Simple Password Exponential Key Exchange (SPEKE) protocol, designed by Jablon [11] in 1996. SPEKE tries to work around the deficiencies of EKE by using the shared password π of the two participants to change the generator of a Diffie-Hellman like scheme. The protocol runs as follows:

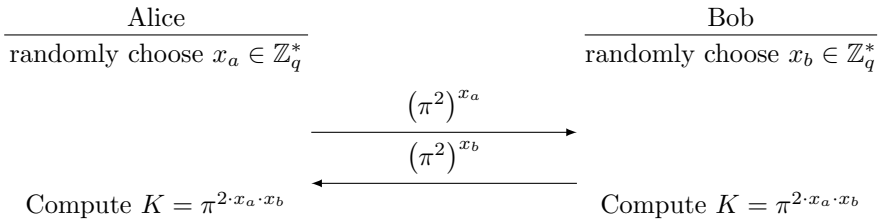


FIGURE 3. The flow diagram for SPEKE

Note that the password is squared so that the exponentiations in the protocol can occur in a subgroup of prime order q (the participants must check that that $\pi^2 \not\equiv \pm 1 \bmod p$ – if this is not the case, all work is carried out in the order 2 subgroup of $\mathbb{Z}_p^*$ which renders the protocol insecure, and a new password or prime p must be chosen).

There are drawbacks to using the password directly: mainly that an attacker can guess multiple passwords in one execution of the protocol (as multiple passwords may have the same square mod p) and that the size of the subgroup in which the protocols occur is large (if p is a 1024-bit prime, for example, then q is a 1023-bit prime). Although it is troubling to allow an active attacker multiple guesses at the password, for practical purposes as long as they are limited to a small constant number of guesses per protocol execution the security of the protocol can be safely assumed. Indeed, a variant of the basic protocol presented in Figure 3 where a hash of the password is squared (as defined by the IEEE P1363.2 standard regarding SPEKE[2]) was later proven secure by MacKenzie [13] in a common formal model (proposed by Boyko, MacKenzie and Patel [5]) under the assumptions of the random-oracle model and the hardness of the decision inverted-additive Diffie-Hellman (DIDH) problem[3]. Here, the notion of 'secure' is relaxed to allow an active attacker to rule out a (small) constant number of guesses per protocol execution.

[2]Our implementation of SPEKE uses this variant

[3]This somewhat non-standard Diffie-Hellman assumption asks one to distinguish between an element $g^{(x+y)^{-1}}$ and a random group element, given the elements $X = g^{x^{-1}}$ and $Y = g^{y^{-1}}$. It has been shown that if the typical computational Diffie-Hellman problem (CDH) is hard, so is the computational inverted-additive Diffie-Hellman problem. Furthermore, if the Decision Square Diffie-Hellman problem is hard (which is assumed in the security proof of the J-PAKE protocol, discussed below) then the DIDH problem is hard. See Figure 2 of Abdalla et. al. [2] for a comparison of all the Diffie-Hellman type assumptions used in the protocols presented here.

2.2. **J-PAKE.** In 2010, Hao and Ryan [8] proposed the Password Authenticated Key Exchange by Juggling (J-PAKE) protocol, at least in part to get around the deficiencies of the SPEKE method described in the previous section.[4] J-PAKE, which is used in Firefox and (as an optional protocol) in OpenSSL, among others, is quite straightforward and uses the shared password to make a nice simplification of randomly chosen Diffie-Hellman like exponentiations.

The protocol as specified relies on a Zero Knowledge Proof (ZKP) of an exponent: that is, a protocol such that a sender can transmit $X = g^x$ and a message which allows a receiver to determine almost certainly that the sender knows x, without revealing any knowledge of x (the element X is assumed to be a member of a group in which the computational Diffie-Hellman problem is hard). Our implementation uses the common Schnorr non-interactive ZKP: roughly, the sender transmits their ID, sID, along with the values

$$V = g^v \qquad \text{and} \qquad r = v - xh$$

where $v \in_R \mathbb{Z}_q$ and $h := H(g||V||X||sID)$ for a suitable hash function H. The receiver checks that X is in the proper group, that h is the correct hash, and that $V = g^r \cdot X^h$. This choice of ZKP is also used by Hao et. al. [9], and we refer the reader to that paper for more details.

We are now ready to describe the protocol. Let Q be a subgroup of $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ with prime order q, g be a generator of this subgroup, and $\pi \in \mathbb{Z}_q^*$ be the shared password between Alice and Bob. J-PAKE consists of a setup round followed by two rounds of communication:

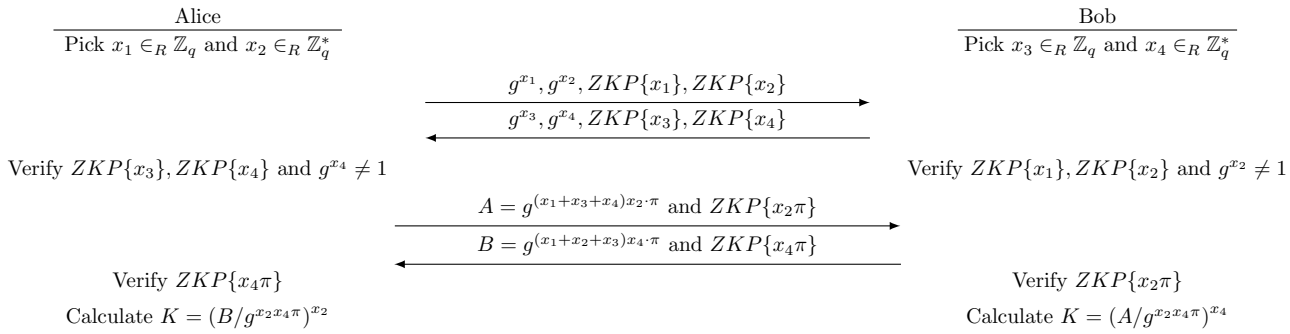


FIGURE 4. The J-PAKE protocol.

As noted in the figure, both Alice and Bob are able to determine

$$K = \underbrace{(B/g^{x_2 x_4 \pi})^{x_2}}_{\text{Computable by Alice}} = g^{(x_1+x_3)x_2 x_4 \pi} = \underbrace{(A/g^{x_2 x_4 \pi})^{x_4}}_{\text{Computable by Bob}} .$$

The shared session key is then taken to be $\kappa = H(K)$, where H is a suitable hash function. We note that J-PAKE (like SPEKE) admits only key authentication: each of Alice and Bob know that the only people who can calculate the shared key are themselves; key confirmation can additionally be performed if desired (which will increase the number of rounds of communication by one).

In their original paper, Hao and Ryan [8] gave proofs that J-PAKE satisfies the four security properties (offline dictionary attack resistance, forward secrecy for established keys, known session security, and online dictionary attack resistance) discussed at the beginning at this section. Although these proofs were straightforward, they did not take place in the framework of an established and commonly used formal model for PAKE security, and relied on unstated assumptions about an adversary's range of potentially attack techniques. As Feng Hao, one of the J-PAKE authors, wrote in a blog post[5]: "Some researchers might like to take it from here and add more 'formalism' into the paper. I'm sure that will be a valuable addition in

[4] In addition to the security flaws, such as allowing multiple guesses of the password per execution, SPEKE is also patented by Phoenix Technologies, while J-PAKE proudly presents its freedom from patents.

[5] Accessible at https://www.lightbluetouchpaper.org/2008/05/29/j-pake/#comment-9550

future work." In 2015, the security of J-PAKE was proven in the model of Bellare, Pointcheval, and Rogaway by Abdalla et. al. [2] under the assumptions of the random-oracle model, the algebraic adversary model (AAM)[6] and the hardness of the Decision Square Diffie-Hellman problem (DSDH). DSHD is the problem of determining the group element g^{x^2} from a random element, given access to g^x. Hardness of DSDH implies the hardness of the standard decision Diffie-Hellman (DDH) problem, and it is currently unknown whether or not it is harder (i.e., whether there is a separation in the complexity classes).

2.3. **Dragonfly.** Dragonfly – another PAKE created by Harkins [10] – is also based on discrete logarithm cryptography. This general setup means one can work either in a finite field using elliptic curves, and like all our implementations for this project we describe and use the finite field version. Similar to J-PAKE (and unlike SPEKE), there are no assumptions on the order of the underlying group (i.e., its order does not have to be of the form $p = 2q - 1$, for q prime).

Let p be a large prime. We will let Q denote a cyclic subgroup of $\mathbb{Z}_p^*$ with prime order q – hence $q|(p-1)$. In addition to p and q, a hash function H is also agreed upon, and it assumed that Alice and Bob share the password $\pi \in Q$. The protocol specification maps the password arbitrarily (but deterministically) to the element π, and includes some example algorithms to perform the actual mapping. These examples are omitted here. Due to a (slightly) increased complexity over the protocols discussed so far, we begin with a text description:

(Round 1) Alice chooses two random values $r_A, m_A \in_R \mathbb{Z}_q^*$ and computes $s_A = r_A + m_A \mod q$ along with the element $E_A = \pi^{-m_A} \mod p$. If $s_A < 2$ (to avoid a small subgroup attack), she repeats this step. She sends s_A and E_A to Bob.

Bob chooses two random values $r_B, m_B \in_R \mathbb{Z}_q^*$. He computes $s_B = r_B + m_B \mod q$ and the element $E_B = \pi^{-m_B} \mod p$. If $s_B < 2$, he repeats this step over. He sends s_A and E_A to Alice.

Each member verifies that one of $E_A \neq E_B$ or $s_A \neq s_B$ is true to avoid a reflection attack. [7]

(Round 2) Alice computes the shared secret $ss = (\pi^{s_B} E_B)^{r_A} = \pi^{r_A r_B} \mod p$.
Alice sends $A = H(ss||E_A||s_A||E_B||s_B)$ to Bob.

Bob computes the shared secret $ss = (\pi^{s_A} E_A)^{r_B} = \pi^{r_A r_B} \mod p$.
Bob sends $B = H(ss||E_B||s_B||E_A||s_A)$ to Alice.

Alice and Bob both confirm the received hash values are correct and compute the shared key

$$K = H(ss||E_A \times E_B||(s_A + s_B) \mod q).$$

This protocol is illustrated in Figure 5. We note that despite its status as an IEEE 802.11-2012 standard no security proof of Dragonfly has yet been derived (the document of Harkins [10] states that it possesses security features such as offline dictionary attack resistance). In fact, Rond 1 was modified in the most recent update to the Dragonfly protocol after a small subgroup attack was discovered by Clarke and Hao [6]; the protocol works around this now by checking s_A and s_B, and repeating the step until the values generated are safe. The chief benefit of Dragonfly is its speed: as the results of Section 4 show, Dragonfly+ is the fastest group protocol we tested.

[6]The AAM, originated by Dolev and Yao [7] states that an adversary can only perform operations in the underlying group of the protocol, on known messages (for instance, the attacker cannot modify the bits of messages or guess keys)

[7]An attacker can be accepted as $A = B$ in Round 2. The attacker still cannot compute the key, however this check is a precautionary measure.

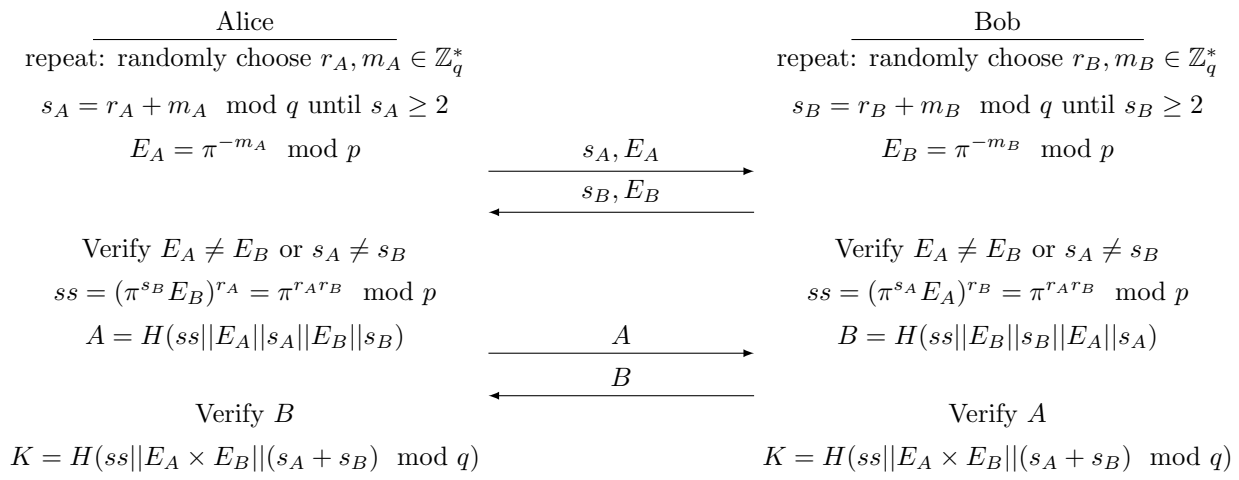


FIGURE 5. The Dragonfly protocol.

2.4. **PAK/PPK.** The PAK/PPK protocols were first introduced by Boyko, MacKenzie and Patel in [5] in 2000 as Diffie-Hellman based provably secure PAKEs, with the PAK protocol admitting key confirmation and PPK admitting only key authentication (although PPK) uses one less round of communication). An augmented variant of PAK, called PAK-X, was also introduced in the same paper.

Let π be the password shared by Alice and Bob and p and q be primes with $p = rq + 1$, where $q \nmid r$. Furthermore, let g be a generator of a subgroup of $\mathbb{Z}_p^*$ of size q where the Decision Diffie-Hellman (DDH) problem is infeasible. Finally, we take H_1, H_{2a}, H_{2b}, H_3 to be independent random hash functions. The PAK and PPK protocols are described as in Figures 6 and 7.

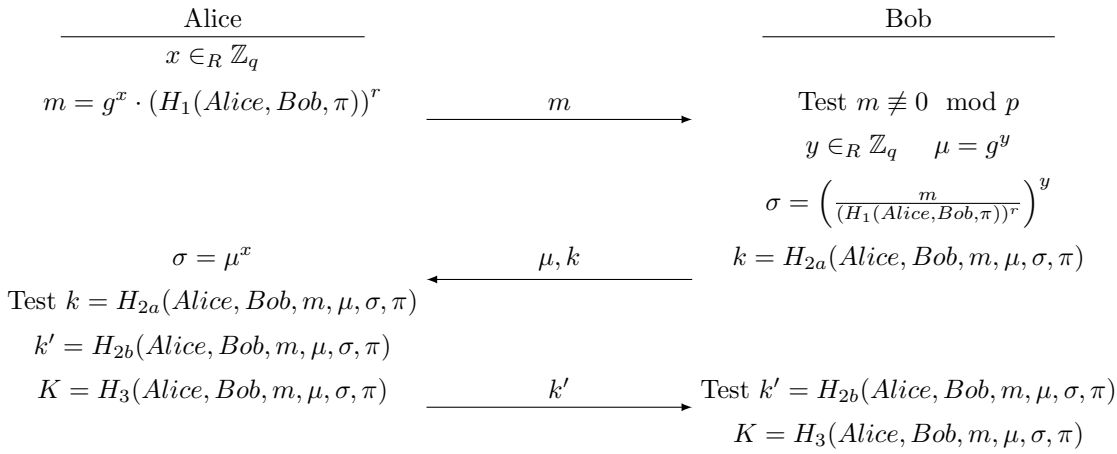


FIGURE 6. The PAK protocol.

In their original paper, Boyko, MacKenzie and Patel [5] developed a new formal model for PAKE security, in which they proved that PAK and PPK are secure under the assumptions of the random-oracle model and the hardness of the DDH is intractable. This newly proposed model was well designed, and security proofs of other PAKE protocols have been tailored to it (for instance, the security proof of SPEKE given by MacKenzie[13] and mentioned in Section 2.1).

3. Group Password-Authenticated Key Exchange (GPAKE)

Background info from paper

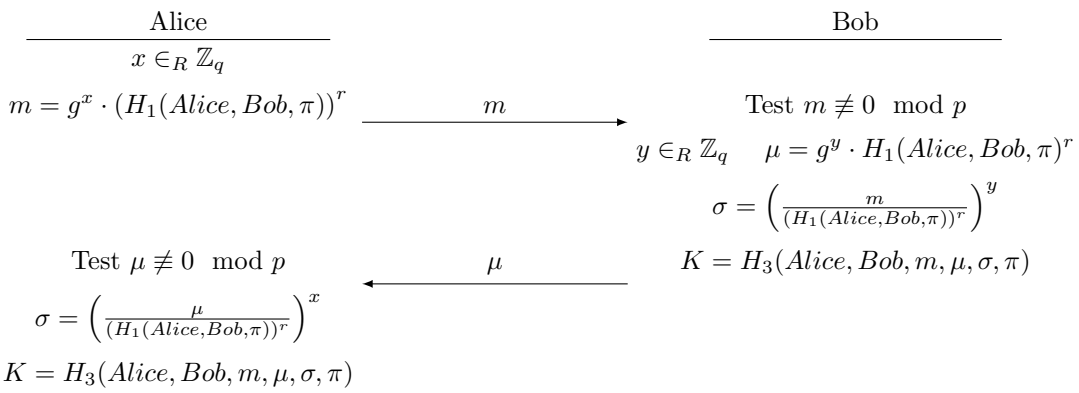


FIGURE 7. The PPK protocol.

3.1. **SPEKE+ and J-PAKE+.** Info about SPEKE+ and J-PAKE+

3.2. **Dragonfly+.** We present the group extension to the Dragonfly protocol using the general construction outlined in [9]. The construction follows the Dragonfly protocol closely, with only minor modifications to the first round of communication and of course adding the final round to establish the group key. The setup is the same as Dragonfly (see Section 2.3), except the generator of the subgroup Q is required, called g. The Dragonfly+ protocol executes as follows:

(Round 1) Every participant selects $r_{ij}, m_{ij} \in_R \mathbb{Z}_q^*$ for all $j \neq i$ and compute $s_{ij} = r_{ij} + m_{ij} \mod q$ along with the element $E_{ij} = \pi^{-m_{ij}} \mod p$. If any $s_{ij} < 2$, r_{ij} and m_{ij} must be re-established. Additionally, each member selects $y_i \in_R \mathbb{Z}_q$. Each participant broadcasts s_{ij}, E_{ij}, $g^{y_i} \mod p$ and ZKP$\{y_i\}$.

Define $z_i = y_{i+1}/y_{i-1}$ (with cyclic index i). Each member is able to compute $g^{z_i} = g^{y_{i+1}}/g^{y_{i-1}}$, and check:
- $g^{z_i} \neq 1 \mod p$;
- One of $E_{ij} \neq E_{ji}$ or $s_{ij} \neq s_{ji}$ is true for all $j \neq i$;
- the received ZKP$\{y_j\}$ for all $j \neq i$ is valid.

(Round 2) Every participant can compute the pairwise shared secrets $ss_{ij} = (\pi^{s_{ji}} E_{ji})^{r_{ij}} = \pi^{r_{ij} r_{ji}} \mod p$. They broadcast $A_{ij} = H(ss_{ij}||E_{ij}||s_{ij}||E_{ji}||s_{ji})$.

Each participant confirms the hashes are correct.

(Round 3) Every participant P_i broadcasts $(g^{z_i})^{y_i}$ and a zero knowledge proof, ZKP$\{\tilde{y_i}\}$ for providing the equality of the discrete logarithm of $(g^{z_i})^{y_i}$ to the base g^{z_i} and the discrete logarithm g^{y_i} to the base g. Each member computes the pairwise Dragonfly keys, $K_{ij} = H(ss_{ij}||E_{ij} \times E_{ji}||(s_{ij} + s_{ji}) \mod q)$ and the authentication and confirmation keys $\kappa^{\text{MAC}} = H(K_{ij}, \text{“MAC”})$ and $\kappa^{\text{KC}} = H(K_{ij}, \text{“KC’)}$. Each member additionally broadcasts $t_{ij}^{MAC} = HMAC(\kappa_{ij}^{MAC}, g^{y_i}||\text{ZKP}\{y_i\}||(g^{z_i})^{y_i}||\text{ZKP}\{\tilde{y_i}\})$ and $t_{ij}^{KC} = HMAC(\kappa_{ij}^{KC}, \text{“}KC''||i||j||E_{ij}||E_{ji})$

Finally, all members confirm:
- the received ZKP$\{\tilde{y_j}\}$ for $j \neq i$ are valid;
- the received key confirmation strings t_{ji}^{KC} for $j \neq i$ are valid;
- the received message authentication tags t_{ji}^{MAC} for $j \neq i$ are valid.

and establish the group key according to the Burmester-Desmedt group key agreement protocol. [add reference to equation]

The group scheme does not compromise what security properties Dragonfly might have, as the only additional information y_i is not related to the Dragonfly protocol.

For simplicity in our implementation of Dragonfly+, we mapped the password to g, the generator of Q. Obviously, this specific strategy is not secure since the password is not used. However, the actual latency measurements would not be affected much if at all.

3.3. **PPK+.** In this section, we present the Group PAKE extension of the PPK protocol, the simpler of the PAK/PPK suite, using the Fairy Ring Dance construction from [9]. The PPK+ protocol is described as follows:

(Round 1) Every participant P_i selects $x_i \in_R \mathbb{Z}_q$ and $y_i \in_R \mathbb{Z}_q$ and broadcasts $m_{ij} = g^{x_i} \cdot (H_1(i,j,\pi))^r$ for all $j \neq i$ as well as 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 $z_i = y_{i+1} - y_{i-1}$. Then everyone is able to compute $g^{z_i} = g^{y_{i+1}}/g^{y_{i-1}}$. Every participant P_i also computes $\sigma_{ij} = \left(\frac{m_{ji}}{H_1(j,i,\pi)^r}\right)^{x_i} = g^{x_i x_j}$ and checks:

- $g^{z_i} \neq 1$ for $i = 1, \ldots, n$.
- $m_j \neq 0$ for $j = 1, \ldots, n, j \neq i$.
- the received ZKP$\{y_j\}$ for $j = 1, \ldots, n, j \neq i$ are valid.

Similar to all GPAKE constructions, the ZKPy_i are standard Schnorr non-interactive zero knowledge proofs outlined in [9].

(Round 2) Every participant P_i broadcasts $(g^{z_i})^{y_i}$ and a zero knowledge proof, ZKP$\{\tilde{y_i}\}$ for providing the equality of the discrete logarithm of $(g^{z_i})^{y_i}$ to the base g^{z_i} and the discrete logarithm g^{y_i} to the base g. Everyone then computes the raw pairwise keys K_{ij} according to PPK, namely $K_{ij} = H_3(i, j, m_{ij}, m_{ji}, \sigma_{ij}, \pi)$, and the derived authentication and confirmation keys, $\kappa^{\text{MAC}} = H(K_{ij}, \text{“MAC”})$, $\kappa^{\text{KC}} = H(K_{ij}, \text{“KC”})$. Furthermore, let $A_{ij} = g^{y_i} \,||\, \text{ZKP}\{y_i\} \,||\, K_{ij} \,||\, \text{ZKP}\{\tilde{y_i}\}$, P_i broadcast $t_{ij}^{\text{MAC}} = \text{HMAC}(\kappa_{ij}^{\text{MAC}}, A_{ij})$ and $t_{ij}^{\text{KC}} = \text{HMAC}(\kappa_{ij}^{\text{KC}}, \text{“KC”} \,||\, i \,||\, j \,||\, m_i \,||\, m_j)$ for each $j \neq i$.

When this round finishes, everyone checks:

- the received ZKP$\{\tilde{y_j}\}$ for $j = 1, \ldots, n, j \neq i$ are valid;
- the received key confirmation strings t_{ji}^{KC} for $j = 1, \ldots, n, j \neq i$ are valid;
- the received message authentication tags t_{ji}^{MAC} for $j = 1, \ldots, n, j \neq i$ are valid.

Again, the zero knowledge proofs are standard Chaum-Pedersen ZKP used in [9]. At the end of the two rounds, the same formula is used for calculating the group key. [Yi: Steve please put the formula in section 3.1 and I'll reference it here.] *

The security of the GPAKE protocol follows directly from the security of the two party protocol PPK.

For simplicity of our demonstration, there are a few tweaks we made in our implementation of the protocol. Firstly, the supposedly independent random hash functions H_1 and H_3 are implemented as *SHA-256* shifted by two different constants. Obviously it has security implications on the protocol, but it will certainly not be used in real life implementations, nor does it have any impact on the run time performance that we are measuring.

Secondly, it should be noted that the formula for calculating the raw pairwise keys K_{ij} at the end of round 1 is not symmetric between i and j. A modification is therefore made such that when calculating K_{ij}, we set $i < j$ without loss of generality. In this case both parties would be able to calculate the same pairwise key.

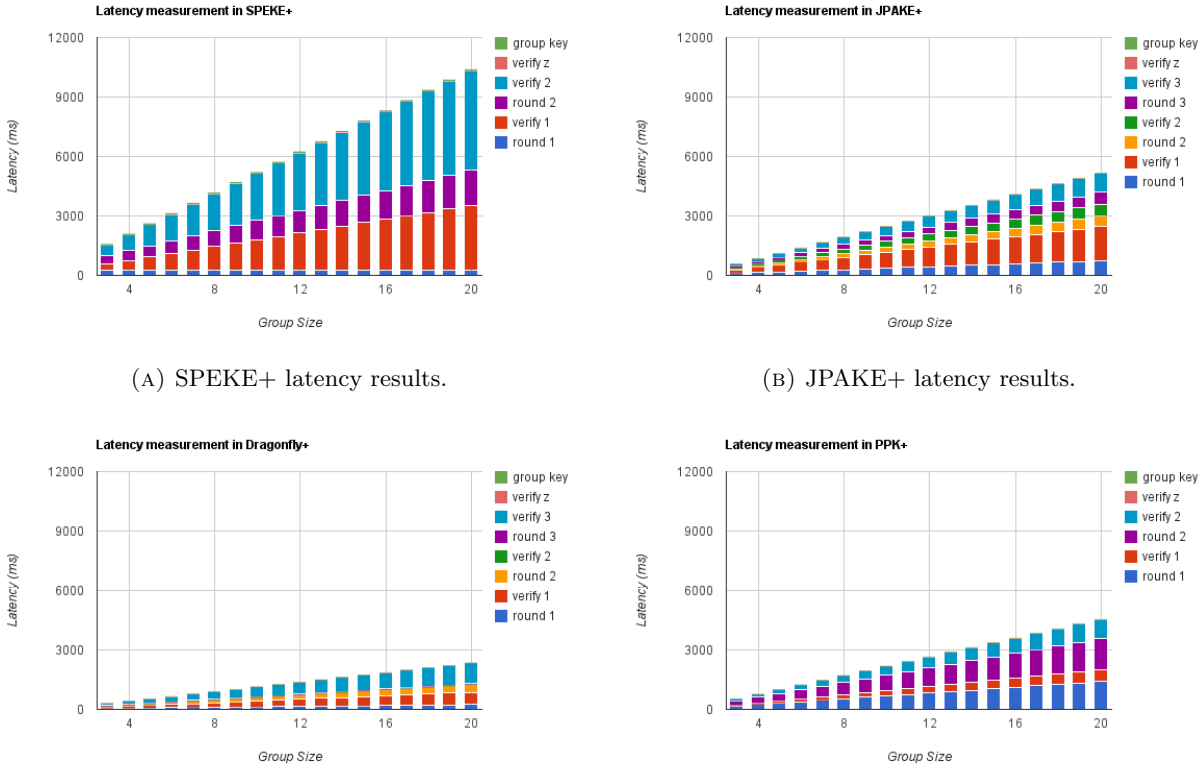
4. Implementation Results

All of the protocols were implemented in Java 6 on a server (3GHz AMD processor, 6GB of RAM) running Ubuntu 12.04. These benchmarks measured latency, the amount of work each device would have to do in the group excluding communication.

For all settings, 2048-bit primes p are used. The same values for SPEKE+ and JPAKE+ were used as in [9]. Values for PPK+ were taken from JPAKE+, and values for Dragonfly+ were taken from the NIST cryptographic toolkit ([1]).

For groups sizes from 3 to 20 participants, we measure the latency of computation at each round of the

protocol. The measurement was done by repeating the same experiment 100 times and taking the average values. The results are summarized in Figure 8.



(A) SPEKE+ latency results.

(B) JPAKE+ latency results.

(C) Dragonfly+ latency results.

(D) PPK+ latency results.

FIGURE 8. Latency measurement results for the GPAKE extensions of each of the PAKE protocols.

As shown in Figure 8, the total latency for each member increases linearly with the size of group, regardless of the protocol. [etc... etc...]

5. Conclusion

The conclusion

References

[1] NIST cryptographic toolkit, July 2014. http://csrc.nist.gov/groups/ST/toolkit/.

[2] M. Abdalla, F. Benhamouda, and P. MacKenzie. Security of the j-pake password-authenticated key exchange protocol. In *SP*, 2015.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *LNCS, B. Preneel, Ed., vol. 1807. Springer*, pages 139 – 155. EUROCRYPT 2000, May 2000.

[4] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Computer Society Symposium on Research in Security and Privacy Proceedings*, pages 72–84. IEEE, 1992.

[5] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *LNCS 1807,Springer-Verlag, Berlin*, pages 156–171. Eurocrypt 2000, 2000.

[6] Dylan Clarke and Feng Hao. Cryptanalysis of the dragonfly key exchange protocol. In *IET Information Security*, pages 283–289, 2014.

[7] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE trans. on Information Theory*, pages 198–208, 1983.

[8] F. Hao and P. Ryan. J-PAKE: Authenticated key exchange without PKI. *Transactions on Computational Science XI, Lecture Notes in Computer Science*, 6480:192–206, 2010.

[9] F. Hao, X. Yi, L. Chen, and S. F. Shahandashti. The fairy-ring dance: Password authenticated key exchange in a group. Cryptology ePrint Archive, Report 2015/080, 2015. http://eprint.iacr.org/2015/080.

[10] D. Harkins. Dragonfly key exchange – internet research task force internet draft, 2015. http://datatracker.ietf.org/doc/draft-irtf-cfrg-dragonfly/.

[11] B. Jaspan. Dual-workfactor encrypted key exchange: efficiently preventing password chaining and dictionary attacks. In *Proceedings of the Sixth Annual USENIX Security Conference*, pages 43–50, July 1996.

[12] T. M. A. Lomas, L. Gong, J. H. Saltzer, and R. M. Needham. Reducing risks from poorly chosen keys. In *ACM Operating Systems Review, 23(5)*, pages 14–18. Proceedings of the 12th ACM Symposium on Operating System Principles, Dec 1989.

[13] P. MacKenzie. On the security of the speke password-authenticated key exchange protocol. Cryptology ePrint Archive, Report 2001/057, 2001. http://eprint.iacr.org/2001/057.

[14] Philip MacKenzie. The pak suite: Protocols for password-authenticated key exchange. In *IEEE P1363.2*, 2002.