Password Authenticated Key Exchange: From Two Party Methods to Group Schemes

Stephen Melczer, Taras Mychaskiw, and Yi Zhang

Introduction

1. Classical Two Party PAKEs
   1.1 Background and Security Properties
   1.2 J-PAKE
   1.3 Dragonfly
   1.4 PAK/PPK
2. Extension to Group Setting (GPAKEs)
   2.1 Fairy-Ring Dance
   2.2 Examples of GPAKEs
3. Timings
4. Conclusion

PART 1
Classical Two Party PAKEs

Password Authenticated Key Exchange (PAKE)

PAKEs allow two parties sharing a *short/weak* password to establish a shared key

Cannot broadcast password directly – would need to be protected (expensive)

Instead, modern PAKEs use *zero-knowledge proof* and/or *hash* of password in protocol

First Protocol: EKE (Bellovin and Merrit 1992)

Pick prim. root $\alpha \in \mathbb{Z}_p$

Alice		Bob
randomly choose $x_a \in \mathbb{Z}_p^*$	$\xrightarrow{\quad [\alpha^{x_a}]_s \quad}$	
	$\xleftarrow{\quad [\alpha^{x_b}]_s \quad}$	randomly choose $x_b \in \mathbb{Z}_p^*$

Alice and Bob share $K = \alpha^{x_a \cdot x_b}$.

First Protocol: EKE (Bellovin and Merrit 1992)

Pick prim. root $\alpha \in \mathbb{Z}_p$

Alice		Bob
randomly choose $x_a \in \mathbb{Z}_p^*$	$\xrightarrow{[\alpha^{x_a}]_s}$	
	$\xleftarrow{[\alpha^{x_b}]_s}$	randomly choose $x_b \in \mathbb{Z}_p^*$

Alice and Bob share $K = \alpha^{x_a \cdot x_b}$.

Uses password directly $\implies$ many insecurities found

First Protocol: EKE (Bellovin and Merrit 1992)

Pick prim. root $\alpha \in \mathbb{Z}_p$

Alice		Bob
randomly choose $x_a \in \mathbb{Z}_p^*$	$\xrightarrow{[\alpha^{x_a}]_s}$	
	$\xleftarrow{[\alpha^{x_b}]_s}$	randomly choose $x_b \in \mathbb{Z}_p^*$

Alice and Bob share $K = \alpha^{x_a \cdot x_b}$.

Uses password directly $\implies$ many insecurities found

Ex: Decypher $[\alpha^{x_a}]_{s'}$ – rule out s' if output in $[p, 2^n - 1]$

Desired Security Properties

Offline dictionary attack resistance

Don't leak info which can be used in brute force search

Forward secrecy for established keys

Past keys secure if password disclosed

Implies passive attacker w/ password cannot compute key

Known session security

All secrets of one session reveals nothing about others

Online dictionary attack resistance

Attacker can only test one password per protocol execution

J-PAKE

Alice		Bob
Pick $x_1 \in_R \mathbb{Z}_q$ and $x_2 \in_R \mathbb{Z}_q^*$		Pick $x_3 \in_R \mathbb{Z}_q$ and $x_4 \in_R \mathbb{Z}_q^*$
	$\xrightarrow{g^{x_1}, g^{x_2}, ZKP\{x_1\}, ZKP\{x_2\}}$	
	$\xleftarrow{g^{x_3}, g^{x_4}, ZKP\{x_3\}, ZKP\{x_4\}}$	
Verify $ZKP\{x_3\}, ZKP\{x_4\}$ and $g^{x_4} \neq 1$		Verify $ZKP\{x_1\}, ZKP\{x_2\}$ and $g^{x_2} \neq 1$
	$\xrightarrow{A = g^{(x_1+x_3+x_4)x_2 \cdot s} \text{ and } ZKP\{x_2 s\}}$	
	$\xleftarrow{B = g^{(x_1+x_2+x_3)x_4 \cdot s} \text{ and } ZKP\{x_4 s\}}$	
Verify $ZKP\{x_4 s\}$		Verify $ZKP\{x_2 s\}$
$K = (B/g^{x_2 x_4 s})^{x_2}$		$K = (A/g^{x_2 x_4 s})^{x_4}$

Original uses Schnorr ZKPs

J-PAKE

Satisfies all 4 desired properties
(under DSDH assumptions)

Only two rounds of communication

No *explicit* key confirmation (only implicit)

Not patented

Dragonfly

Setup Let Q be a cyclic subgroup of $\mathbb{Z}_p^*$ with prime order q. Both members map the password to an element $\pi \in Q$.

Dragonfly

Setup Let Q be a cyclic subgroup of $\mathbb{Z}_p^*$ with prime order q. Both members map the password to an element $\pi \in Q$.

Alice		Bob
repeat: randomly choose $r_A, m_A \in \mathbb{Z}_q^*$		repeat: randomly choose $r_B, m_B \in \mathbb{Z}_q^*$
$s_A = r_A + m_A \mod q$ until $s_A \geq 2$		$s_B = r_B + m_B \mod q$ until $s_B \geq 2$
$E_A = \pi^{-m_A} \mod p$	$\xrightarrow{s_A, E_A}$	$E_B = \pi^{-m_B} \mod p$
	$\xleftarrow{s_B, E_B}$	
Verify $E_A \neq E_B$ or $s_A \neq s_B$		Verify $E_A \neq E_B$ or $s_A \neq s_B$
$ss = (\pi^{s_B} E_B)^{r_A} = \pi^{r_A r_B}$		$ss = (\pi^{s_A} E_A)^{r_B} = \pi^{r_A r_B}$
$A = H(ss\|E_A\|s_A\|E_B\|s_B)$	$\xrightarrow{A}$	$B = H(ss\|E_B\|s_B\|E_A\|s_A)$
	$\xleftarrow{B}$	
Verify B		Verify A
$K = H(ss\|E_A \times E_B\|(s_A + s_B) \mod q)$		$K = H(ss\|E_A \times E_B\|(s_A + s_B) \mod q)$

Dragonfly

No formal security proofs, but claims resistance to offline dictionary attacks

Only two rounds of communication

Very fast compared to other protocols

PAK/PPK – PAK

Setup Let $G = <g>$ be an order q subgroup of $\mathbb{Z}_p^*$

Let $p = rq + 1$ where r, q relatively prime.

Let π be the password and H_1, H_{2a}, H_{2b}, H_3 be random, independent hash functions.

PAK/PPK – PAK

Setup Let $G = <g>$ be an order q subgroup of $\mathbb{Z}_p^*$

Let $p = rq + 1$ where r, q relatively prime.

Let π be the password and H_1, H_{2a}, H_{2b}, H_3 be random, independent hash functions.

PAK

A		B
$x \in_R Z_q$		
$m = g^x \cdot (H_1(A, B, \pi))^r$	$\xrightarrow{\quad m \quad}$	Test $m \overset{?}{\not\equiv} 0 \bmod p$
		$y \in_R Z_q$
		$\mu = g^y$
		$\sigma = (\frac{m}{(H_1(A,B,\pi))^r})^y$
$\sigma = \mu^x$	$\xleftarrow{\quad \mu, k \quad}$	$k = H_{2a}(A, B, m, \mu, \sigma, \pi)$
Test $k \overset{?}{=} H_{2a}(A, B, m, \mu, \sigma, \pi)$		
$k' = H_{2b}(A, B, m, \mu, \sigma, \pi)$		
$K = H_3(A, B, m, \mu, \sigma, \pi)$	$\xrightarrow{\quad k' \quad}$	Test $k' \overset{?}{=} H_{2b}(A, B, m, \mu, \sigma, \pi)$
		$K = H_3(A, B, m, \mu, \sigma, \pi)$

PAK/PPK – PPK

Setup Let $G = <g>$ be an order q subgroup of $\mathbb{Z}_p^*$

Let $p = rq + 1$ where r, q relatively prime.

Let π be the password and H_1, H_3 be random, independent hash functions.

PAK/PPK – PPK

Setup Let $G = <g>$ be an order q subgroup of $\mathbb{Z}_p^*$

Let $p = rq + 1$ where r, q relatively prime.

Let π be the password and H_1, H_3 be random, independent hash functions.

PPK

A		B
$x \in_R Z_q$		
$m = g^x \cdot (H_1(A, B, \pi))^r$	$\xrightarrow{\quad m \quad}$	Test $m \overset{?}{\not\equiv} 0 \bmod p$
		$y \in_R Z_q$
		$\mu = g^y \cdot (H_1(A, B, \pi))^r$
		$\sigma = (\frac{m}{(H_1(A,B,\pi))^r})^y$
Test $\mu \overset{?}{\not\equiv} 0 \bmod p$	$\xleftarrow{\quad \mu \quad}$	$K = H_3(A, B, m, \mu, \sigma, \pi)$
$\sigma = (\frac{\mu}{(H_1(A,B,\pi))^r})^x$		
$K = H_3(A, B, m, \mu, \sigma, \pi)$		

PAK/PPK

Proposes a new formal security model
based on the random oracle model

Satisfies all 4 desired properties
(under DDH assumptions)

Only two/three rounds of communication

PAK has explicit key confirmation
PPK has implicit key confirmation

Comparisons

Comparison of practical Diffie-Hellman-based PAKE protocols proven secure in the BPR model [5]

		Assumptions[a]					Complexity	
	Rounds / Flows	CRS	ROM	ICM	AAM		Communication[b]	Time[c]
J-PAKE with Schnorr [24]	2 / 4 or 3 / 3		✗		✗	DSDH or (CSDH + DDH)	$12 \times G + 6 \times \mathbb{Z}_p$	28 exp (12 exp + 8 mexp)
EKE [5], [7]	1 / 2			✗		CDH	$2 \times G$	4 exp + 2 memb + 2 enc
SPEKE [29], [35]	1 / 2		✗			DIDH[d]	$2 \times G$	4 exp + 2 memb
PPK [10]	2 / 2		✗			DDH	$2 \times G$	6 exp + 2 memb
SPAKE2 [3]	1 / 2		✗			CDH	$2 \times G$	4 exp + 2 memb
GK-SPOKE [1], [21], [30]	2 / 2	✗				DDH + PRG[e]	$6 \times G$	17 exp (4 exp + 7 mexp) + 6 memb
GL-SPOKE [1], [18], [32]	2 / 2	✗				DDH	$7 \times G$	21 exp (4 exp + 7 mexp) + 7 memb
KV-SPOKE [1], [33]	1 / 2	✗				DDH	$10 \times G$	30 exp (2 exp + 12 mexp) + 10 memb

[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; *mexp*: number of multi-exponentiations; *memb*: verification of the membership of a group element to the cyclic group G. For elliptic curve with small co-factor, this only costs a small number of additions on the curve, but for subgroups of $\mathbb{Z}_q$ (q being a prime larger than p, the order of the group G), this costs an exponentiation (with exponent $p-1$); *enc*: encryption with the ideal cipher; multiplications, hash evaluations, and PRG evaluations are omitted;
[d] *DIDH*: decision inverted-additive Diffie-Hellman assumption [35] (see Fig. 2 and the Appendix);
[e] *PRG*: pseudo-random generator.

Security of the J-PAKE Password-Authenticated Key Exchange Protocol.

M. Abdalla, F. Benhamouda and P. MacKenzie, SP'2015.

PART 2
Extension to Group Setting

The Fairy-Ring Dance

Group members establish the group key and the pair-wise keys simultaneously.

The Fairy-Ring Dance

Group members establish the group key and the pair-wise keys simultaneously.

For pair-wise key:
Use plain two-party PAKE protocols. (Need at least 2 rounds)

The Fairy-Ring Dance

Group members establish the group key and the pair-wise keys simultaneously.

For pair-wise key:
Use plain two-party PAKE protocols. (Need at least 2 rounds)

For group key:

- Everyone additionally choose another random $y_i \in_R \mathbb{Z}_q$ and broadcast g^{y_i}
- Everyone can calculate $g^{z_i} := g^{y_{i+1} - y_{i-1}} = g^{y_{i+1}} / g^{y_{i-1}}$

The Fairy-Ring Dance

Group members establish the group key and the pair-wise keys simultaneously.

For pair-wise key:
Use plain two-party PAKE protocols. (Need at least 2 rounds)

For group key:

- Everyone additionally choose another random $y_i \in_R \mathbb{Z}_q$ and broadcast g^{y_i}
- Everyone can calculate $g^{z_i} := g^{y_{i+1}-y_{i-1}} = g^{y_{i+1}}/g^{y_{i-1}}$

Group key:

$$K_i = (g^{y_{i-1}})^{ny_i} \cdot (y^{z_i y_i})^{n-1} \cdot (y^{z_{i+1} y_{i+1}})^{n-2} \cdots (g^{z_{i_2} y_{i-2}}) \tag{1}$$
$$= g^{y_1 y_2 + y_2 y_3 + \cdots + y_n y_1} \tag{2}$$

Dragonfly+

Setup Let Q be a cyclic subgroup of $\mathbb{Z}_p^*$ with prime order q with generator g. Each member maps the password to an element $\pi \in Q$.

Every member P_i chooses $r_{ij}, m_{ij} \in_R \mathbb{Z}_q^*$ for all $j \in \{1, \ldots, n\} \setminus \{i\}$

Each member also chooses $y_i \in_R \mathbb{Z}_q$ and computes g^{y_i} mod p

Dragonfly+

Setup Let Q be a cyclic subgroup of $\mathbb{Z}_p^*$ with prime order q with generator g. Each member maps the password to an element $\pi \in Q$.

Every member P_i chooses $r_{ij}, m_{ij} \in_R \mathbb{Z}_q^*$ for all $j \in \{1, \ldots, n\} \setminus \{i\}$

Each member also chooses $y_i \in_R \mathbb{Z}_q$ and computes $g^{y_i} \mod p$

Rd 1 Each member broadcasts $s_{ij} = r_{ij} + m_{ij} \mod q$ and $E_{ij} = \pi^{-m_{ij}} \mod p$

They also broadcast $g_{y_i} \mod p$ along with ZKP$\{y_i\}$

[All members verify the ZKP, verify $g_{z_i} \neq 1$ and check for reflection attacks]

Dragonfly+

Rd 2 Each member computes their pairwise shared secrets:
$ss_{ij} = (\pi^{s_{ji}} E_{ji})^{r_{ij}}$
Each member broadcasts $H(ss_{ij}||E_{ij}||s_{ij}||E_{ji}||s_{ji})$
[All members verify the pairwise hash values]

Dragonfly+

Rd 2 Each member computes their pairwise shared secrets:
$ss_{ij} = (\pi^{s_{ji}} E_{ji})^{r_{ij}}$
Each member broadcasts $H(ss_{ij}||E_{ij}||s_{ij}||E_{ji}||s_{ji})$
[All members verify the pairwise hash values]

Rd 3 Every member broadcasts $(g^{z_i})^{y_i}$ and ZKP$\{\tilde{y}_i\}$.
Let K_{ij} be the pairwise Dragonfly key between members i and j. Members compute
$\kappa_{ij}^{MAC} = H(K_{ij}||"MAC"), \kappa_{ij}^{KC} = H(K_{ij}||"KC")$
Members broadcast
$t_{ij}^{MAC} = HMAC(\kappa_{ij}^{MAC}, g^{y_i}||\text{ZKP}\{y_i\}||(g^{z_i})^{y_i}||\text{ZKP}\{\tilde{y}_i\})$
and $t_{ij}^{KC} = HMAC(\kappa_{ij}^{KC}, "KC"||i||j||E_{ij}||E_{ji})$
[All members verify ZKP$\{\tilde{y}_i\}$, t_{ji}^{MAC} and t_{ij}^{KC} are correct]

Dragonfly+

Rd 2 Each member computes their pairwise shared secrets:
$ss_{ij} = (\pi^{s_{ji}} E_{ji})^{r_{ij}}$
Each member broadcasts $H(ss_{ij}||E_{ij}||s_{ij}||E_{ji}||s_{ji})$
[All members verify the pairwise hash values]

Rd 3 Every member broadcasts $(g^{z_i})^{y_i}$ and ZKP$\{\tilde{y}_i\}$.
Let K_{ij} be the pairwise Dragonfly key between members i and j. Members compute
$\kappa_{ij}^{MAC} = H(K_{ij}||"MAC"), \kappa_{ij}^{KC} = H(K_{ij}||"KC")$
Members broadcast
$t_{ij}^{MAC} = HMAC(\kappa_{ij}^{MAC}, g^{y_i}||\text{ZKP}\{y_i\}||(g^{z_i})^{y_i}||\text{ZKP}\{\tilde{y}_i\})$
and $t_{ij}^{KC} = HMAC(\kappa_{ij}^{KC}, "KC"||i||j||E_{ij}||E_{ji})$
[All members verify ZKP$\{\tilde{y}_i\}$, t_{ji}^{MAC} and t_{ij}^{KC} are correct]

All members share:

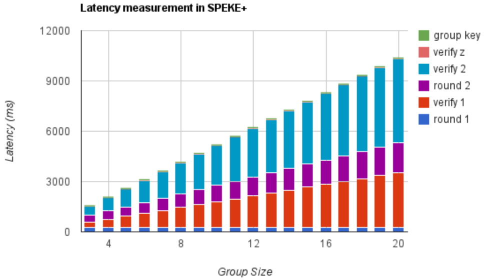
$$K = g^{y_1 \cdot y_2 + y_2 \cdot y_3 + \cdots + y_n \cdot y_1}$$

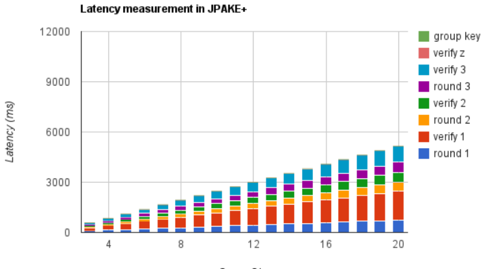
PART 3
Timings

Specifications

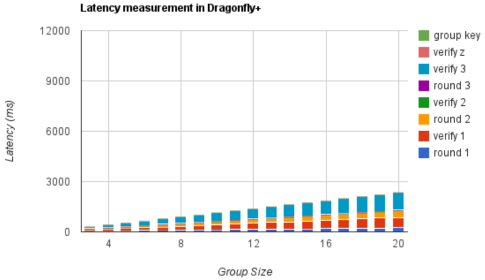
- All protocol benchmarks were implemented in Java 1.6 and run on a server (3GHz AMD processor, 6GB of RAM) running Ubuntu 12.04.
- Benchmarks measured latency, the amount of work each device would have to do in the group excluding communication.

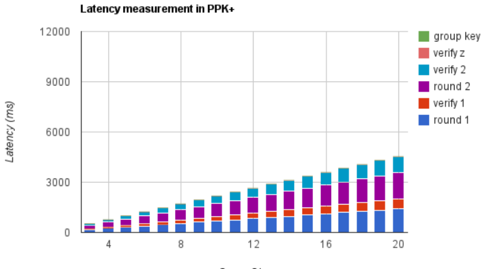
Results


Latency measurement in SPEKE+
12000
9000
6000
3000
0
Latency (ms)
4
8
12
16
20
Group Size
group key
verify z
verify 2
round 2
verify 1
round 1


Latency measurement in JPAKE+
12000
9000
6000
3000
0
Latency (ms)
4
8
12
16
20
group key
verify z
verify 3
round 3
verify 2
round 2
verify 1
round 1

Results


Latency measurement in Dragonfly+
12000
9000
6000
3000
0
Latency (ms)
4
8
12
16
20
Group Size
group key
verify z
verify 3
round 3
verify 2
round 2
verify 1
round 1


Latency measurement in PPK+
12000
9000
6000
3000
0
Latency (ms)
4
8
12
16
20
group key
verify z
verify 2
round 2
verify 1
round 1

Conclusion

Conclusion

It is possible to transfer PAKEs into GPAKEs while preserving round efficiency

SPEKE+ is very slow

J-PAKE+ is a bit slow, but proven secure (under DSDH)

PPK is faster

Dragonfly is fastest, but no security proof
(despite IEEE 802.11-2012 standard)

THANK YOU