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

Stephen Melczer, Taras Mychaskiw, and Yi Zhang

Introduction

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* of password in protocol

First Protocol: EKE (Bellovin and Merrit 1992)

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

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

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

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

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

Alice picks $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$

Bob picks $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$

J-PAKE

Setup Let $G = <g>$ be an order q subgroup of $\mathbb{Z}_p^*$
Alice picks $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$
Bob picks $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$

Rd 1 Alice sends g^{x_1}, g^{x_2} and ZKPs of x_1 and x_2 to Bob
Bob sends g^{x_3}, g^{x_4} and ZKPs of x_3 and x_4 to Alice
[Alice and Bob verify ZKPs and $g^{x_2}, g^{x_4} \neq 1$]

J-PAKE

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

Alice picks $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$

Bob picks $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$

Rd 1 Alice sends g^{x_1}, g^{x_2} and ZKPs of x_1 and x_2 to Bob

Bob sends g^{x_3}, g^{x_4} and ZKPs of x_3 and x_4 to Alice

[Alice and Bob verify ZKPs and $g^{x_2}, g^{x_4} \neq 1$]

Rd 2 Alice sends $A = g^{(x_1+x_3+x_4)x_2 \cdot s}$ and ZKP of $x_2 s$

Bob sends $B = g^{(x_1+x_2+x_3)x_4 \cdot s}$ and ZKP of $x_4 s$

J-PAKE

Setup Let $G = <g>$ be an order q subgroup of $\mathbb{Z}_p^*$
Alice picks $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$
Bob picks $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$

Rd 1 Alice sends g^{x_1}, g^{x_2} and ZKPs of x_1 and x_2 to Bob
Bob sends g^{x_3}, g^{x_4} and ZKPs of x_3 and x_4 to Alice
[Alice and Bob verify ZKPs and $g^{x_2}, g^{x_4} \neq 1$]

Rd 2 Alice sends $A = g^{(x_1+x_3+x_4)x_2 \cdot s}$ and ZKP of $x_2 s$
Bob sends $B = g^{(x_1+x_2+x_3)x_4 \cdot s}$ and ZKP of $x_4 s$

They share

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

J-PAKE

Satisfies all 4 desired properties
(under DDH and CDH assumptions)

Only two rounds of communication

No *explicit* key confirmation (only implicit)

Not patented

Dragonfly

Desciption of protocol

Security properties

PAK/PPK

Desciption of protocol

Security properties

PART 2
Extension to Group Setting

The Fairy-Ring Dance

Desciption of general procedure

How pairwise + group keys are constructed

J-PAKE+

Description of J-PAKE+

Uses 3 rounds as J-PAKE does not have explicit key confirmation

Dragonfly+

Description of Dragonfly+

PPK+

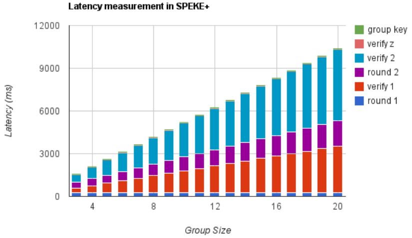
Description of PPK+

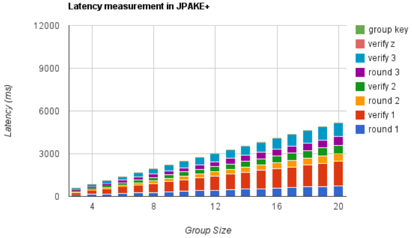
PART 3
Timings

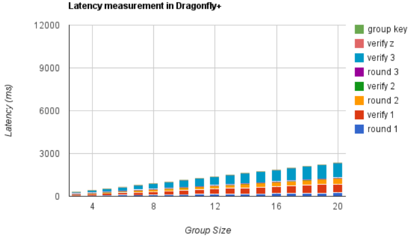
Specifications

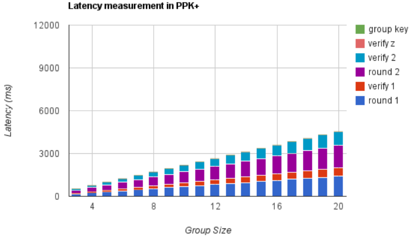
Computer info + how timings done (Java version etc.)

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 Size
group key
verify z
verify 3
round 3
verify 2
round 2
verify 1
round 1


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 Size
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 CDH)

PPK is faster but weaker security proof

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

THANK YOU