Password Authenticated Key Exchange by Juggling

A key exchange protocol without PKI

Feng Hao

Centre for Computational Science
University College London

Security Protocols Workshop '08

Outline

Password Authenticated Key Exchange

Definition

Password Authenticated Key Exchange (PAKE) studies how to establish a secure communication channel between two parties solely based on a shared low-entropy password.

A concrete example

- Alice and Bob share a four-digit code.
- Now Alice wants to send Bob a private message.
- But, eavesdropping is everywhere.
- And attackers may intercept a message, and change it at will.
- Bootstrap a high-entropy key from a low-entropy secret. Is it possible?

The PAKE problem

Research development

- In 1992, Bellovin and Merrit proposed EKE, a milestone.
- In 1996, Jablon designed SPEKE, another well-known scheme.
- In 2000, IEEE formed P1363.2 Working Group to standardize PAKE solutions. Still on-going ...

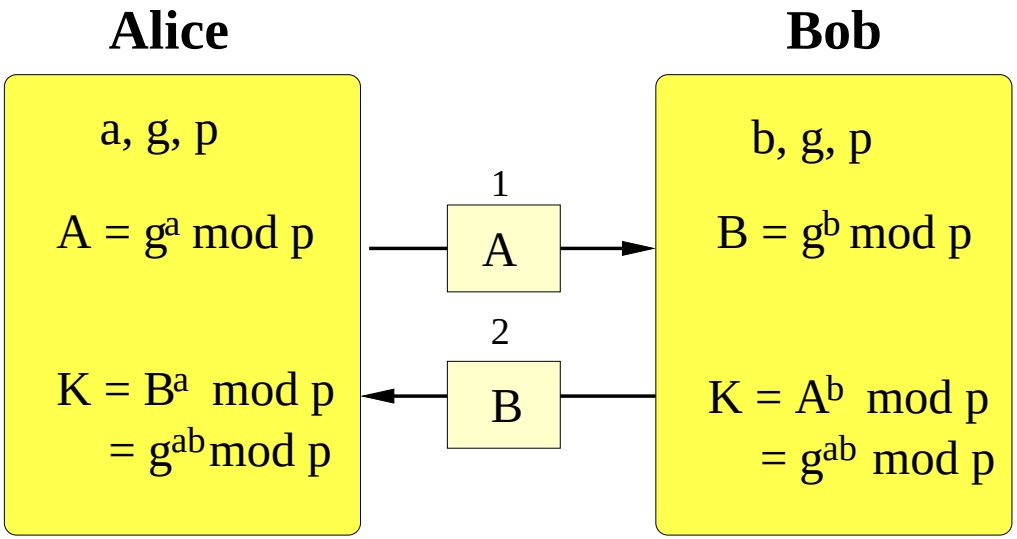
To date, the problem remains unsolved.

- Patent is one big issue; many schemes were patented.
- Also there are technical issues.

Patent is one big issue

- EKE (1992) was patented by Lucent.
- SPEKE (1996) was patented by Phoenix.
- SRP-6 (1998) was patented by Stanford.
- OPAKE (2005) was patented by DoCoMo.
- In practice, many applications have no choice but to use SSL/TLS, which relies on a PKI in place.

A Basic Diffie-Hellman protocol



- EKE modifies the protocol by encrypting A and B with a password, hence the name "Encrypted Key Exchange".
- SPEKE modifies the protocol by replacing g with a password-derived variable, say a hash of password.

Reported Weaknesses with EKE

Practical attack

- An attacker can narrow down the password range [Jaspan'96].

Theoretical analysis

- Proving the EKE security requires a strong "ideal-cipher model" assumption [Bellare et al'00]

Efficiency

- EKE uses long exponents by definition.

Reported Weaknesses with SPEKE

Practical attack

- An attacker can guess multiple passwords in one try [Zhang'04].

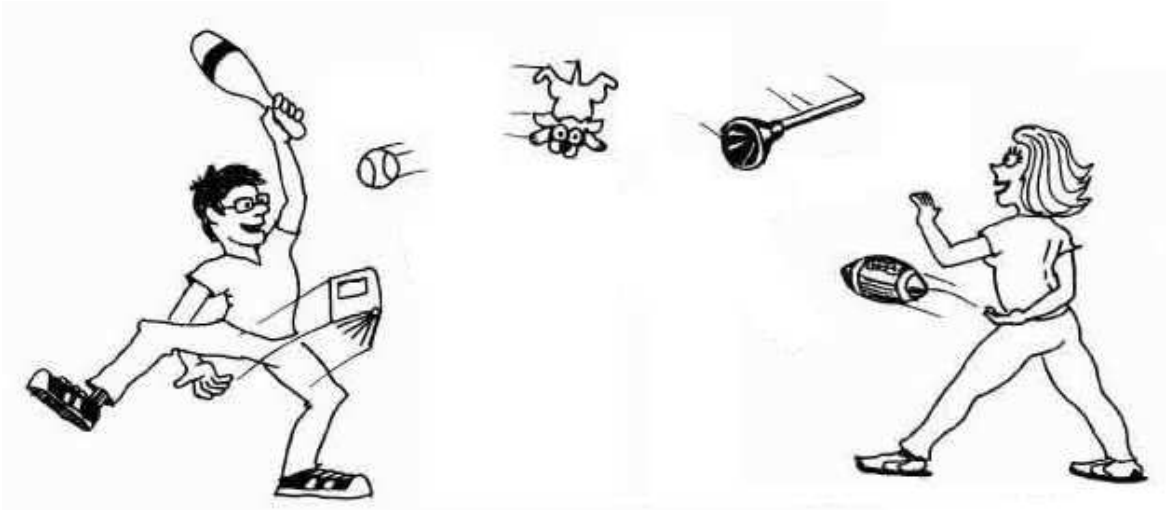
Theoretical analysis

- Proving the security of SPEKE requires relaxing the requirements – allow multiple guesses of password in on try [MacKenzie'01].

Efficiency

- SPEKE uses long exponents by definition.

A New Approach – Public Key Juggling



- A technique first devised to solve the Dining Cryptographers problem [Hao,Zielinski'06], in a multi-party case.
- We apply the juggling technique to the two-party case.

Password Authenticated Key Exchange by Juggling

Round 1:

1. Alice sends g^{x_1}, g^{x_2} and Know-Proof $\{x_1, x_2\}$.
2. Bob sends g^{x_3}, g^{x_4} and Know-Proof $\{x_3, x_4\}$.

Round 2:

1. Alice sends $\mathcal{A} = g^{(x_1+x_3+x_4)\cdot x_2 \cdot s}$ and Know-Proof $\{x_2 \cdot s\}$.
2. Bob sends $\mathcal{B} = g^{(x_1+x_2+x_3)\cdot x_4 \cdot s}$ and Know-Proof $\{x_4 \cdot s\}$.

To get a common key:

- Alice compute $K = (\mathcal{B}/g^{x_2 \cdot x_4 \cdot s})^{x_2} = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$.
- Bob computes $K = (\mathcal{A}/g^{x_2 \cdot x_4 \cdot s})^{x_4} = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$.
- In this juggling game, each public key is a ball.

Why the Knowledge Proofs in the protocol?

The sixth principle in designing robust protocols

- "Do not assume that a message you receive has a particular form (such as g^r for known r) unless you can check this." [Anderson,Needham'95]

Schnorr's signature

- A well-established technique
- Non-interactive, efficient.
- It allows one to prove knowledge of r for g^r without leaking it.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Computation cost of J-PAKE - Java, 2.33-GHz

Item	Description	No of Exp	Time (ms)
1	Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$	4	23
2	Verify KPs for $\{x_3, x_4\}$	4	24
3	Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$	2	9
4	Verify KP for $\{x_4 \cdot s\}$	2	10
5	Compute κ	2	9
	Total	**14**	**75**

- It requires 14 exponentiations, while EKE and SPEKE only 2.
- But, both EKE and SPEKE require long exponents.
- One exp is 6-7 times more expensive than in J-PAKE for a typical p, q setting.
- Overall, the cost is actually about the same.

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Security analysis of J-PAKE

Modules	Security property	Attacker type	Assumptions
Schnorr signature	leak 1-bit: whether sender knows discrete logarithm	passive/active	DL and random oracle
Password encryption	indistinguishable from random	passive/active	DDH
Session key	incomputable	passive	CDH
	incomputable	passive (know s)	CDH
	incomputable	passive (know other session keys)	CDH
	incomputable	active (if $s' \neq s$)	CDH
Key confirmation	leak nothing	passive	–
	leak 1-bit whether $s' = s$	active	CDH

Summary of J-PAKE security proofs

1. Off-line dictionary attack resistance: It does not leak any password verification information to a passive attacker.
2. Known-key security: It prevents a disclosed session key from affecting the security of other sessions.
3. Forward secrecy: It produces session keys that remain secure even when the password is later disclosed.
4. On-line dictionary attack resistance: It strictly limits an active attacker to test only one password per protocol execution.

By comparison, EKE does not fulfill 1, and SPEKE does not fulfill 4.

Comparison to the de facto standard SSL/TLS

Alice: "I know the password to Prince William's email."
Bob: "No, you don't."
Alice: "Yes, I do."
Bob: "Prove it."
Alice: "All right, I tell you." (She whispers in Bob's ear)
Bob: "Interesting. Now I'm going to tell the *Daily Mirror*."
Alice: "Oops!"

Using J-PAKE: a Zero Knowledge Proof protocol

- Reveal only one-bit: whether or not one knows the password.
- Much more resistant against phishing attacks.
- In addition, it requires no PKI deployments ⋆⋆⋆

Summary

We proposed the J-PAKE protocol

- Authenticate password? Let's play a game.
- If juggling successful, both sides obtain a session key.
- If failure, it leaks nothing more than "wrong password".
- Better security than EKE and SPEKE, with comparable efficiency.

Compare with SSL/TLS

- It prevents leaking passwords, say to a fake bank website.
- It requires no PKI developments.

Thank you!