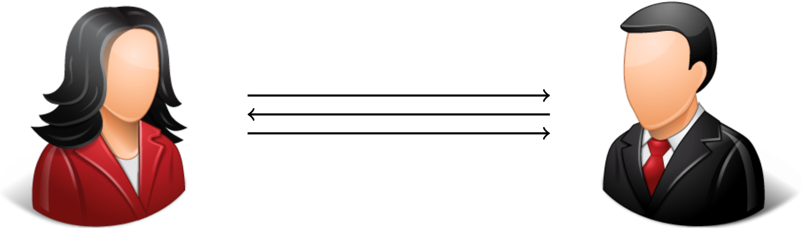
Mitigating Server Breaches in Password-Based Authentication: Secure and Efficient Solutions

Damien Vergnaud

École normale supérieure – CNRS – INRIA – PSL

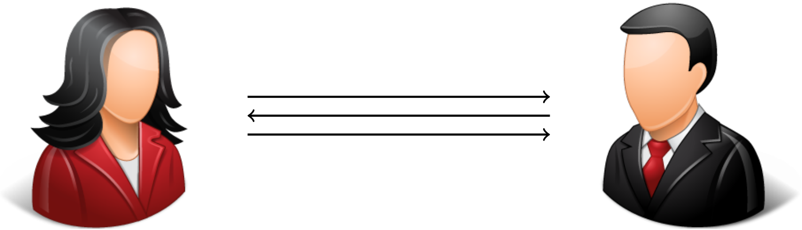
Outline of the Talk

Authenticated Key Exchange



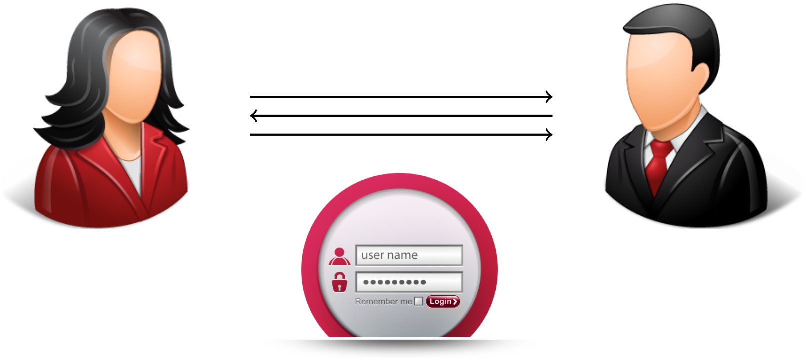
- Alice and Bob agree on a **common secret key** K
  means: public/secret key pair (1 or 2), common secret
- implicit authentication: only Alice and Bob can compute K
- semantic security: K indistinguishable from random

Authenticated Key Exchange



- Alice and Bob agree on a **common secret key** K
  means: public/secret key pair (1 or 2), common secret
- implicit authentication: only Alice and Bob can compute K
- semantic security: K indistinguishable from random

Password-Authenticated Key Exchange (PAKE)



- prove to each other that they know the password
- without disclosing any useful information about it
- get a **shared secret** out at the end.

On-Line Dictionary Attacks

- people use weak passwords
- **Example:** `RockYou.com` password breach of 32M accounts (2010)
  - Total entropy of passwords: 21.1 **bits**
  - Top 100 passwords cover 4.6% of accounts
- Unavoidable attack:
  - adversary interacts with a player, trying a password
  - **success** $\rightsquigarrow$ it has guessed the password
  - **failure** $\rightsquigarrow$ it tries again with another password

On-Line Dictionary Attacks

- people use weak passwords
- **Example:** `RockYou.com` password breach of 32M accounts (2010)
  - Total entropy of passwords: 21.1 **bits**
  - Top 100 passwords cover 4.6% of accounts
- Unavoidable attack:
  - adversary interacts with a player, trying a password
  - **success** $\rightsquigarrow$ it has guessed the password
  - **failure** $\rightsquigarrow$ it tries again with another password

Security Models

- Various security models
  - Game-based security (Bellare-Pointcheval-Rogaway, 2000)
  - Simulation-based security (Boyko-MacKenzie-Patel, 2000)
  - Universal Composability (Canetti-Halevi-Katz-Lindell-MacKenzie, 2005)
- The adversary controls **all** the communications:
  It can create, modify, transfer, alter, delete messages
- Users can participate in concurrent executions of the protocol
- **On-line dictionary attack** should be the best attack ...
  - $q_S = \#$ Active Sessions
  - $N = \#$ Dictionary
  - $\rightsquigarrow$ No adversary should win with probability greater than q_S/N!

Distributed PAKE

- inspired by multi-party computation
  (Ford and Kaliski – 2000)

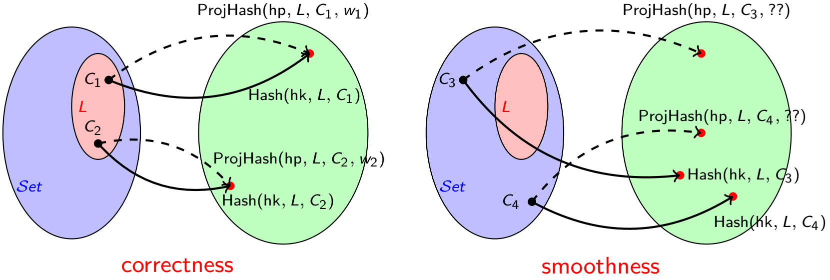
- passwords shared among two servers (or more)
  - distributed computation between client and servers
  - uses a **gateway** (with no secret)
  - ends up in the gateway and the client sharing a secret.

- time divided into distinct periods
  - servers update their sharing of the passwords
  - adversary can corrupt servers multiple times
    but only one for each period.
  - The user **does not** need to update his password

Building Block: Projective Hashing

- introduced by Cramer and Shoup (2002)
  - **Implicit designated-verifier proofs**
  - IND-CCA encryption scheme
- **Applications:**
  - Password-Authenticated Key Exchange
  - Oblivious Transfer
  - Relatively-Sound / Dual-System NIZK
  - Zero-Knowledge Arguments
  - Witness Encryption
  - ...

Smooth Projective Hash Functions (SPHF)



- HashKG(L) $\rightsquigarrow$ hk for language $L \subset \mathcal{S}et$
- ProjKG(hk, L, C) $\rightsquigarrow$ hp
- Hash(hk, L, C)
- ProjHash(hp, L, C, w)

Proof of a Diffie-Hellman tuple (Cramer-Shoup)

$\mathbb{G} = \langle g_1 \rangle = \langle g_2 \rangle$. $|\mathbb{G}| = p$

$$L = \{(g_1^r, g_2^r), r \in \mathbb{Z}_p^*\} \subset \mathbb{G}^2 = \mathcal{S}et$$

- $\mathsf{HashKG}(L) \rightsquigarrow \mathsf{hk} = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$;
- $\mathsf{ProjKG}(\mathsf{hk}, L, \perp) \rightsquigarrow \mathsf{hp} = g_1^{x_1} g_2^{x_2}$.
- $\mathsf{Hash}(\mathsf{hk}, L, C = (c_1, c_2)) \rightsquigarrow H = c_1^{x_1} \cdot c_2^{x_2} \in \mathbb{G}$.
- $\mathsf{ProjHash}(\mathsf{hp}, L, C = (g_1^r, g_2^r), w = r) \rightsquigarrow H' = \mathsf{hp}^r \in G$.

$\rightsquigarrow$ Public-Key Encryption with CCA-security: $\mathbf{CS_{pk}(m; r)}$

Proof of a Diffie-Hellman tuple (Cramer-Shoup)

$\mathbb{G} = \langle g_1 \rangle = \langle g_2 \rangle$. $|\mathbb{G}| = p$

$$L = \{(g_1^r, g_2^r), r \in \mathbb{Z}_p^*\} \subset \mathbb{G}^2 = \mathcal{S}et$$

- $\mathsf{HashKG}(L) \rightsquigarrow \mathsf{hk} = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2$;
- $\mathsf{ProjKG}(\mathsf{hk}, L, \perp) \rightsquigarrow \mathsf{hp} = g_1^{x_1} g_2^{x_2}$.
- $\mathsf{Hash}(\mathsf{hk}, L, C = (c_1, c_2)) \rightsquigarrow H = c_1^{x_1} \cdot c_2^{x_2} \in \mathbb{G}$.
- $\mathsf{ProjHash}(\mathsf{hp}, L, C = (g_1^r, g_2^r), w = r) \rightsquigarrow H' = \mathsf{hp}^r \in G$.

$\rightsquigarrow$ Public-Key Encryption with CCA-security: $\mathbf{CS_{pk}(m; r)}$

Proof of a Diffie-Hellman tuple (Cramer-Shoup)

$\mathbb{G} = \langle g_1 \rangle = \langle g_2 \rangle$. $|\mathbb{G}| = p$

$$L = \{(g_1^r, g_2^r), r \in \mathbb{Z}_p^*\} \subset \mathbb{G}^2 = \mathcal{S}et$$

- $\mathsf{HashKG}(L) \rightsquigarrow \mathsf{hk} = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$;
- $\mathsf{ProjKG}(\mathsf{hk}, L, \perp) \rightsquigarrow \mathsf{hp} = g_1^{x_1} g_2^{x_2}$.
- $\mathsf{Hash}(\mathsf{hk}, L, C = (c_1, c_2)) \rightsquigarrow H = c_1^{x_1} \cdot c_2^{x_2} \in \mathbb{G}$.
- $\mathsf{ProjHash}(\mathsf{hp}, L, C = (g_1^r, g_2^r), w = r) \rightsquigarrow H' = \mathsf{hp}^r \in G$.

$\boxed{\rightsquigarrow \text{Public-Key Encryption with CCA-security: } \mathbf{CS_{pk}(m; r)}}$

Application to PAKE

- proposed by Katz-Ostrovsky-Yung (Diffie-Hellman)
- generalized by Gennaro-Lindell

pw_A

$\xrightarrow{C_A}$

$\xleftarrow{C_B,\ \mathsf{hp}_B}$

$\xrightarrow{\mathsf{hp}_A}$

pw_B

$C_A = \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_A; r_A)$

$\mathsf{hk}_A \leftarrow \mathsf{HashKG}(L)$

$\mathsf{hp}_A \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_B)$

$C_B = \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_B; r_B)$

$\mathsf{hk}_B \leftarrow \mathsf{HashKG}(L)$

$\mathsf{hp}_B \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_A)$

$$\mathsf{Hash}(\mathsf{hk}_A, L, C_B) \cdot \mathsf{ProjHash}(\mathsf{hp}_B, L, C_A, r_A) \stackrel{?}{=} \mathsf{Hash}(\mathsf{hk}_B, L, C_A) \cdot \mathsf{ProjHash}(\mathsf{hp}_A, L, C_B, r_B)$$

Application to PAKE

- proposed by Katz-Ostrovsky-Yung (Diffie-Hellman)
- generalized by Gennaro-Lindell

pw_A

$\xrightarrow{C_A}$

$\xleftarrow{C_B,\ \mathsf{hp}_B}$

$\xrightarrow{\mathsf{hp}_A}$

pw_B

$C_A = \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_A; r_A)$
$\mathsf{hk}_A \leftarrow \mathsf{HashKG}(L)$
$\mathsf{hp}_A \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_B)$

$C_B = \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_B; r_B)$
$\mathsf{hk}_B \leftarrow \mathsf{HashKG}(L)$
$\mathsf{hp}_B \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_A)$

$$\mathsf{Hash}(\mathsf{hk}_A, L, C_B) \cdot \mathsf{ProjHash}(\mathsf{hp}_B, L, C_A, r_A) \stackrel{?}{=} \mathsf{Hash}(\mathsf{hk}_B, L, C_A) \cdot \mathsf{ProjHash}(\mathsf{hp}_A, L, C_B, r_B)$$

Application to PAKE

- proposed by Katz-Ostrovsky-Yung (Diffie-Hellman)
- generalized by Gennaro-Lindell

pw_A

$$\xrightarrow{C_A}$$

$$\xleftarrow{C_B, \mathrm{hp}_B}$$

$$\xrightarrow{\mathrm{hp}_A}$$

pw_B

$$C_A = \mathrm{CS}_{\mathrm{pk}}(\mathrm{pw}_A; r_A)$$
$$\mathrm{hk}_A \leftarrow \mathrm{HashKG}(L)$$
$$\mathrm{hp}_A \leftarrow \mathrm{ProjKG}(\mathrm{hk}, L, C_B)$$

$$C_B = \mathrm{CS}_{\mathrm{pk}}(\mathrm{pw}_B; r_B)$$
$$\mathrm{hk}_B \leftarrow \mathrm{HashKG}(L)$$
$$\mathrm{hp}_B \leftarrow \mathrm{ProjKG}(\mathrm{hk}, L, C_A)$$

$$\mathrm{Hash}(\mathrm{hk}_A, L, C_B) \cdot \mathrm{ProjHash}(\mathrm{hp}_B, L, C_A, r_A) \stackrel{?}{=} \mathrm{Hash}(\mathrm{hk}_B, L, C_A) \cdot \mathrm{ProjHash}(\mathrm{hp}_A, L, C_B, r_B)$$

Application to PAKE

- proposed by Katz-Ostrovsky-Yung (Diffie-Hellman)
- generalized by Gennaro-Lindell

pw_A

$$\xrightarrow{C_A}$$
$$\xleftarrow{C_B,\ \mathsf{hp}_B}$$
$$\xrightarrow{\mathsf{hp}_A}$$

pw_B

$$C_A = \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_A; r_A)$$
$$\mathsf{hk}_A \leftarrow \mathsf{HashKG}(L)$$
$$\mathsf{hp}_A \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_B)$$

$$C_B = \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_B; r_B)$$
$$\mathsf{hk}_B \leftarrow \mathsf{HashKG}(L)$$
$$\mathsf{hp}_B \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_A)$$

$$\mathsf{Hash}(\mathsf{hk}_A, L, C_B) \cdot \mathsf{ProjHash}(\mathsf{hp}_B, L, C_A, r_A) \stackrel{?}{=} \mathsf{Hash}(\mathsf{hk}_B, L, C_A) \cdot \mathsf{ProjHash}(\mathsf{hp}_A, L, C_B, r_B)$$

Application to PAKE

- proposed by Katz-Ostrovsky-Yung (Diffie-Hellman)
- generalized by Gennaro-Lindell

pw_A

$$\xrightarrow{C_A}$$

$$\xleftarrow{C_B,\ \mathsf{hp}_B}$$

$$\xrightarrow{\mathsf{hp}_A}$$

pw_B

$$\begin{aligned} C_A &= \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_A; r_A) \\ \mathsf{hk}_A &\leftarrow \mathsf{HashKG}(L) \\ \mathsf{hp}_A &\leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_B) \end{aligned}$$

$$\begin{aligned} C_B &= \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_B; r_B) \\ \mathsf{hk}_B &\leftarrow \mathsf{HashKG}(L) \\ \mathsf{hp}_B &\leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_A) \end{aligned}$$

$$\mathsf{Hash}(\mathsf{hk}_A, L, C_B) \cdot \mathsf{ProjHash}(\mathsf{hp}_B, L, C_A, r_A) \stackrel{?}{=} \mathsf{Hash}(\mathsf{hk}_B, L, C_A) \cdot \mathsf{ProjHash}(\mathsf{hp}_A, L, C_B, r_B)$$

Application to PAKE

- proposed by Katz-Ostrovsky-Yung (Diffie-Hellman)
- generalized by Gennaro-Lindell

pw_A

$$\xrightarrow{\quad C_A \quad}$$

$$\xleftarrow{\quad C_B, \mathsf{hp}_B \quad}$$

$$\xrightarrow{\quad \mathsf{hp}_A \quad}$$

pw_B

$$\begin{aligned} C_A &= \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_A; r_A) \\ \mathsf{hk}_A &\leftarrow \mathsf{HashKG}(L) \\ \mathsf{hp}_A &\leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_B) \end{aligned}$$

$$\begin{aligned} C_B &= \mathsf{CS}_{\mathsf{pk}}(\mathsf{pw}_B; r_B) \\ \mathsf{hk}_B &\leftarrow \mathsf{HashKG}(L) \\ \mathsf{hp}_B &\leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C_A) \end{aligned}$$

$$\mathsf{Hash}(\mathsf{hk}_A, L, C_B) \cdot \mathsf{ProjHash}(\mathsf{hp}_B, L, C_A, r_A) \stackrel{?}{=} \mathsf{Hash}(\mathsf{hk}_B, L, C_A) \cdot \mathsf{ProjHash}(\mathsf{hp}_A, L, C_B, r_B)$$

Katz-MacKenzie-Taban-Gligor DPAKE (2005/2012)

- extends and builds upon Katz-Ostrovsky-Yung PAKE
- password $\mathsf{pw} = \mathsf{pw}_0$ shared as $\mathsf{pw}_0 = \mathsf{pw}_1 \cdot \mathsf{pw}_2$ (high entropy)
- **Protocol execution.** (high level)
  - two executions of the KOY protocol
  - client $\leftrightarrow$ server A (using server B to assist with the authentication)
  - client $\leftrightarrow$ server B (using server A to assist with the authentication)
- client's work increase by a factor 2
  servers' work increase by a factor 6

Katz-MacKenzie-Taban-Gligor DPAKE (2005/2012)

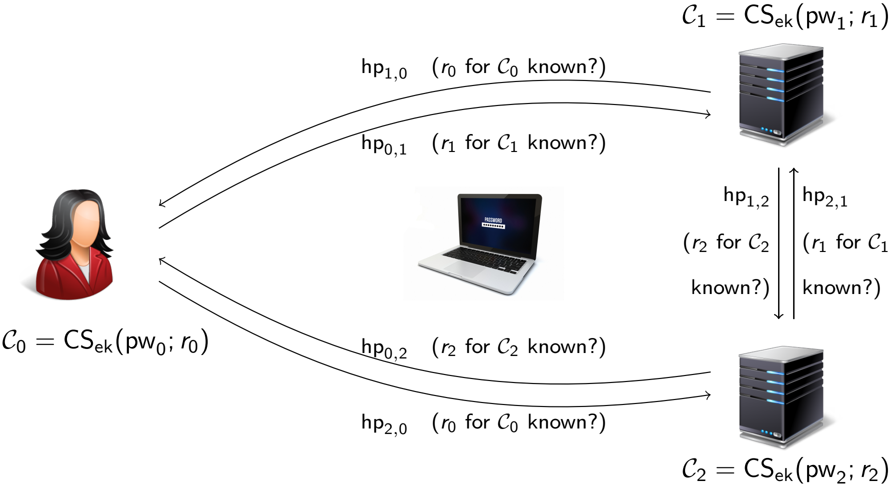
- extends and builds upon Katz-Ostrovsky-Yung PAKE
- password $\mathsf{pw} = \mathsf{pw}_0$ shared as $\mathsf{pw}_0 = \mathsf{pw}_1 \cdot \mathsf{pw}_2$ (high entropy)
- **Protocol execution.** (high level)
  - two executions of the KOY protocol
  - client $\leftrightarrow$ server A (using server B to assist with the authentication)
  - client $\leftrightarrow$ server B (using server A to assist with the authentication)
- client's work increase by a factor 2
  servers' work increase by a factor 6

Design Principle

- U owns a password pw_0 and wants to interact with G
- S_1 owns a share pw_1 of pw_0 $(\mathsf{pw}_0 = \mathsf{pw}_1 \cdot \mathsf{pw}_2)$
- S_2 owns a share pw_2 of pw_0
- G interacts with S_1 and S_2

- "three-party PAKE"
- U checks (using an SPHF) whether $\mathsf{pw}_0 = \mathsf{pw}_1 \cdot \mathsf{pw}_2$
- S_1 checks (using an SPHF) whether $\mathsf{pw}_1 = \mathsf{pw}_0/\mathsf{pw}_2$
- S_2 checks (using an SPHF) whether $\mathsf{pw}_2 = \mathsf{pw}_0/\mathsf{pw}_1$

Construction 1



Efficiency

	Ciphertext	Proj. Keys	Gateway
Client	4	4	0
Server	4	4	1

- communication complexity similar to Katz *et al.*'s scheme
- time complexity similar to Katz *et al.*'s scheme
- **linear-time** update protocol (as Katz *et al.*'s scheme)

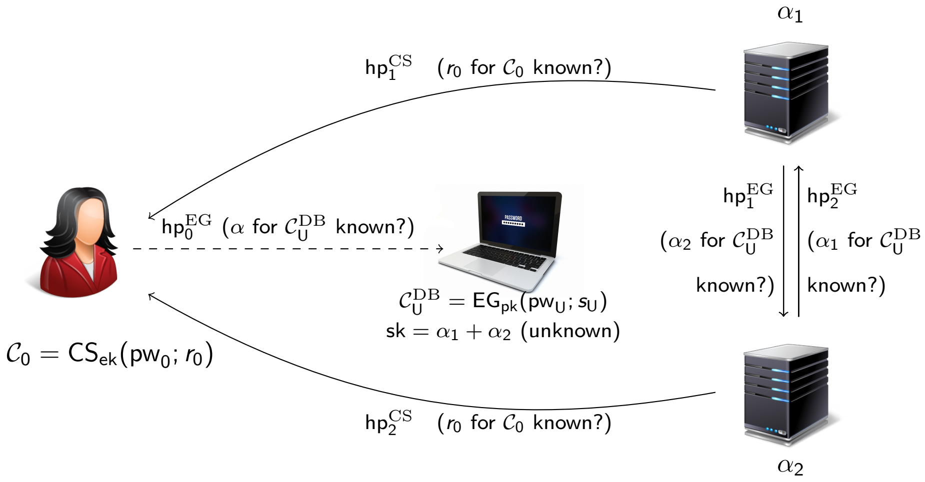
Design Principle

- U owns a password π and wants to interact with G
- G owns a public database DB of encrypted passwords
- G interacts with S_1 and S_2, each owning a share of sk

- idea similar
- only the client needs to compute a ciphertext C
  (the other ciphertext C' in DB)
- participants implicitly check (using several SPHF) that:

$$\mathsf{Dec}(C) = \mathsf{Dec}(C')$$

Construction 2



Efficiency

	Ciphertext	Proj. Keys	Gateway
Client	4	2	0
Server	0	2	1

- constant-time update protocol
- communication complexity decreased by more than 50%
  (9 group elements *vs* 20 group elements for Katz *et al.*'s scheme).
- the client performs 8 full exponentiations; each server performs 7 exponentiations
  (instead of 15 and 13 respectively for Katz *et al.*'s scheme).

Conclusion

- Two constructions of distributed PAKE
  - secure in the standard security model (without random oracles)
  - efficient using standard cryptographic libraries (do not require pairings)
- **Extensions**
  - can be generalized to the setting with n servers
  - can be adapted with SPHFs for Paillier and LWE encryption
- **Open Problems**
  - efficient construction in the Universal Composability framework

Strongly Leakage-Resilient Authenticated Key Exchange

Rongmao Chen

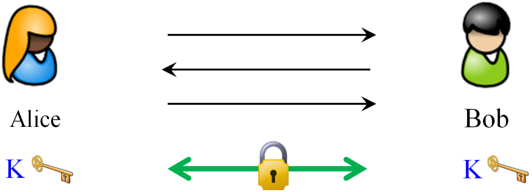
UNIVERSITY OF WOLLONGONG

RSAConference2016

San Francisco | February 29 – March 4 | Moscone Center

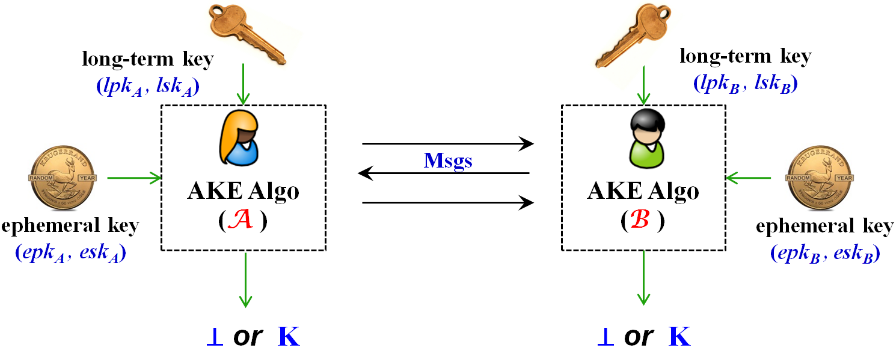
Joint work with Yi Mu, Guomin Yang, Willy Susilo and Fuchun Guo

Authenticated Key Exchange (AKE)

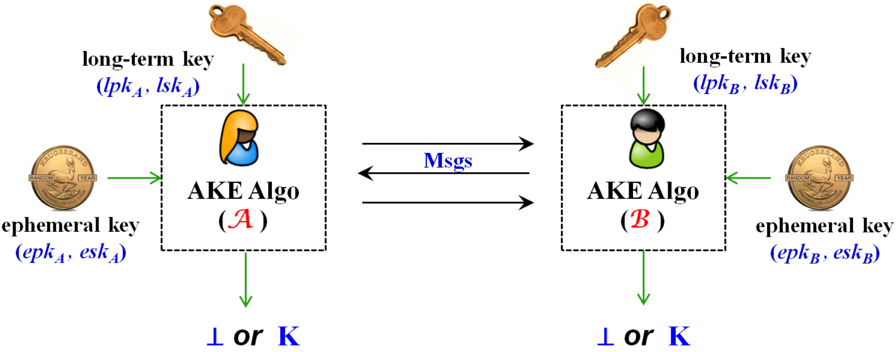


- **Truly Fundamental Cryptographic Protocol**
  - Establish a secure channel by agreeing on a common session key
  - Core in network standards, e.g., IPSec, SSL/TLS, SSH, etc
  - Practical protocols: ISO (a.k.a SIG-DH), IKE (a.k.a SIGMA), etc

A Closer Look at AKE

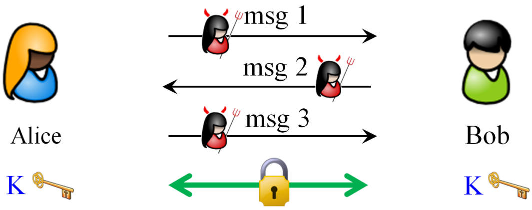


A Closer Look at AKE



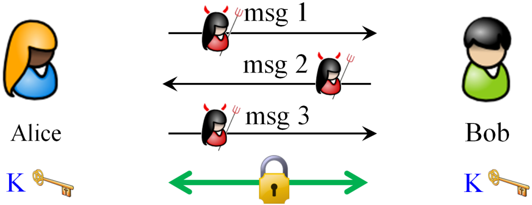
$$\mathsf{K} = \mathcal{A}(lsk_{\mathcal{A}}, esk_{\mathcal{A}}, \mathsf{Msgs}) = \mathcal{B}(lsk_{\mathcal{B}}, esk_{\mathcal{B}}, \mathsf{Msgs})$$

Conventional AKE Security Model



- **Security Notion**
  - Mutual Authentication
  - Secure Key Establishment
- **Adversarial Model**
  - BR [BR93]
  - BCK [BCK98]
  - CK [CK01]
  - eCK [LLM07]

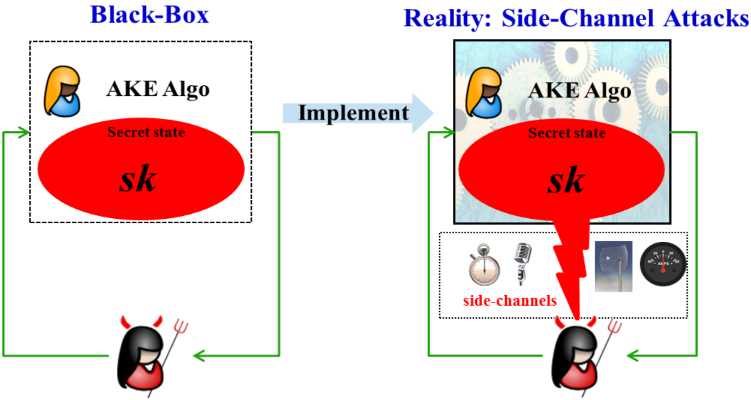
Conventional AKE Security Model



- **Security Notion**
  - Mutual Authentication
  - Secure Key Establishment
- **Adversarial Model**
  - BR [BR93]
  - BCK [BCK98]
  - CK [CK01]
  - eCK [LLM07]

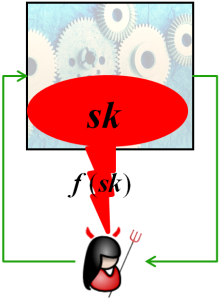
⇝ Black-Box Model

Black-Box Model vs. Reality



⇝ Physical implementations leak secret state through *side-channels*: e.g., power consumption,time, radiation, sound, heat...

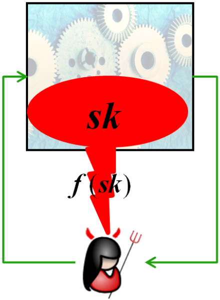
Modeling Side-Channel Attacks



- Modeled by an abstract function $f \in \mathcal{F}$ (leakage function family)
-  obtains $f(sk)$ in addition to the normal black-box interaction
- Arbitrary $\mathcal{F}$? No...(e.g.: $f(sk) = sk$ means no security!)
- Some restrictions are necessary 

RSAConference2016

Modeling Side-Channel Attacks



- Modeled by an abstract function $f \in \mathcal{F}$ (leakage function family)
-  obtains $f(sk)$ in addition to the normal black-box interaction
- Arbitrary $\mathcal{F}$? No...(e.g.: $f(sk) = sk$ means no security!)
- Some restrictions are necessary 

  Solution in one go $\rightsquigarrow$ under minimal restrictions

Prior Work

- **Modeling Leakage Resilience**
  - Relative Leakage Model [...,AGV09, NS09, KV09, DKL09]
    - $f : \{0,1\}^{|sk|} \to \{0,1\}^{\leq\lambda}, \lambda < |sk|$ (small keys, e.g.,1024 bits)
  - Bounded Retrieval Model [Dzi06, CLW06,...]
    - $f : \{0,1\}^{|sk|} \to \{0,1\}^{\leq\lambda}, \lambda < |sk|$ (increase $|sk|$ for flexibly large λ)
  - Auxiliary Input Model [...,DKL09, KV09]
    - $f : \{0,1\}^{|sk|} \to \{0,1\}^{*}$, f is computationally hard-to-invert
  - Continuous Leakage Model [...,DP08, FKPR10, JV10, BKKV10]
    - leakage happens per execution of protocol
- **Leakage-Resilient AKE**
  - CK-Based [ADW09, DHLW10]
  - eCK-Based [MO11, ASB14, ABS14]

Our Motivation

Limitations of Prior Work

- *Challenge-Dependent* Leakage
  - ⇝ Most disallowed to bypass the trivial attack [ADW09, DHLW10, MO11]
  - ⇝ After-the-Fact Leakage: requires split-state [ASB14, ABS14]
- No *Partial* Leakage on  [ADW09, DHLW10, MO11, ASB14, ABS14]
  - ⇝ Independent from ephemeral secret reveal in eCK

Our Results

■ A Strongly Leakage-Resilient AKE Security Model

AKE Models	Partial Leakage Setting				Basic Models
	Challenge-Dependent	lsk	esk	Leakage Model	
[ADW09]	$\times$	$\surd$	$\times$	Bounded-Retrieval	CK
[DHLW10]	$\times$	$\surd$	$\times$	Relative Leakage	CK
[MO11]	$\times$	$\surd$	$\times$	Relative Leakage	eCK
[ASB14]	$\surd$ (w/ split-state)	$\surd$	$\times$	Relative Leakage	eCK
CLR-eCK	$\surd$ (w/o split-state)	$\surd$	$\surd$	Relative Leakage	eCK

(CLR-eCK: *Challenge-Dependent Leakage-Resilient eCK Model*)

■ A Generic Construction with an Efficient Instantiation

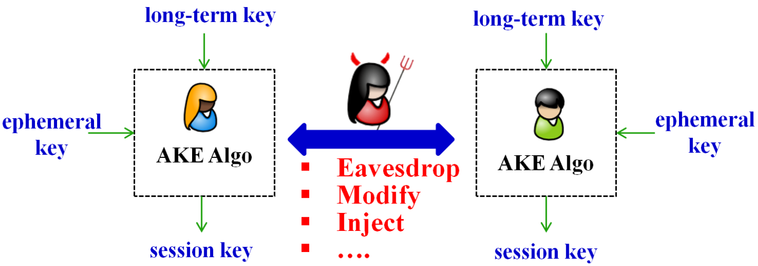
Protocols	Round	Communication	Computation	AKE Models
eSIG-DH	3	$3 \cdot \vert Cer \vert + 2 \cdot \vert \mathbb{G} \vert + 2 \cdot \vert Sig \vert$	$4 \cdot Exp + 2 \cdot Sgn + 2 \cdot Ver$	[ADW09]
Enc-DH	3	$4 \cdot \vert Cer \vert + \vert \mathbb{G} \vert + 2 \cdot \vert CT \vert$	$4 \cdot Exp + 2 \cdot Enc + 2 \cdot Dec$	[DHLW10]
MO	2	$4 \cdot \vert Cer \vert + 9 \cdot \vert \mathbb{G} \vert + 3 \cdot \vert Exk \vert$	$20 \cdot Exp$	[MO11]
π	2	$4 \cdot \vert Cer \vert + 2 \cdot \vert \mathbb{G} \vert + 2 \cdot \vert Sig \vert$	$24 \cdot Exp$	[ASB14]
Our Protocol	1	$4 \cdot \vert Cer \vert + 6 \cdot \vert \mathbb{G} \vert + 2 \cdot \vert Exk \vert$	$16 \cdot Exp$	CLR-eCK

RSAConference2016

Rest of the Talk

- **Challenge-Dependent Leakage-Resilient eCK Model**
  - Query Definition
  - Restrictions on Leakage Query
- **Our Generic Construction**
  - Building Blocks
  - Core Overview & Security Analysis
  - An DDH-Based Instantiation
- **Conclusions**

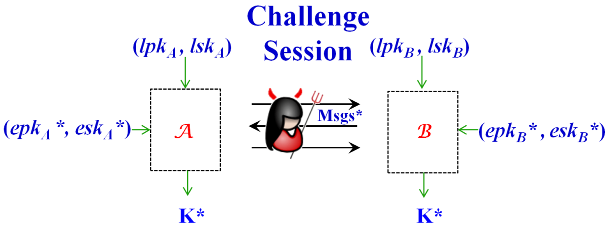
Our New Model: CLR-eCK



Queries by 

- Send: activate an instance via a network message
- Establish: register a long-term public key on behalf of a party
- Reveal: session key, long-term key, ephemeral key
- Leakage: long-term key ($f_1 \in \mathcal{F}_{\text{bbd-I}}$), ephemeral key ($f_2 \in \mathcal{F}_{\text{bbd-II}}$)

Test Query for Challenge Session



Test Query (*only once*) by  **(guess b)**

- Pick $b \xleftarrow{\$} \{0,1\}$, if $b = 1$,  is given K^*, otherwise a random key
- Challenge Session (sid^*) should be *fresh*
  - no reveal query on K^*
  - no reveal query on $< lsk_{\mathcal{A}}, esk^*_{\mathcal{A}} >$ or $< lsk_{\mathcal{B}}, esk^*_{\mathcal{B}} >$ ($\overline{\mathsf{sid}^*}$ exists)
  - no reveal query on $< lsk^*_{\mathcal{A}}, esk^*_{\mathcal{A}} >$ or $lsk^*_{\mathcal{B}}$ ($\overline{\mathsf{sid}^*}$ does not exist)
  - leakage queries satisfy the **defined restrictions**

Restrictions on Leakage Queries

Restrictions on Leakage Queries by 

- Bounded Leakage Setting
  $\mathcal{F}_{\text{bbd-I}} = \{f : \{0,1\}^{|lsk|} \to \{0,1\}^{\leq \lambda_1}\}$, where $\lambda_1 < |lsk|$
  $\mathcal{F}_{\text{bbd-II}} = \{f : \{0,1\}^{|esk|} \to \{0,1\}^{\leq \lambda_2}\}$, where $\lambda_2 < |esk|$
- Leakage Function Commitment
  1.  commits $\mathcal{F}_1 \subseteq \mathcal{F}_{\text{bbd-I}}$ (resp., $\mathcal{F}_2 \subseteq \mathcal{F}_{\text{bbd-II}}$) before revealing the corresponding esk (resp., lsk)
  2.  queries any $f_1 \in \mathcal{F}_{\text{bbd-I}}$ (resp., $f_2 \in \mathcal{F}_{\text{bbd-II}}$) before revealing the corresponding esk (resp., lsk) and thereafter can only ask $f_1 \in \mathcal{F}_1$ (resp. $f_2 \in \mathcal{F}_2$)

$$\mathsf{K}^* = \mathcal{A}(lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}, \mathsf{Msgs}^*) = \mathcal{B}(lsk_{\mathcal{B}}, esk^*_{\mathcal{B}}, \mathsf{Msgs}^*)$$

Restrictions on Leakage Queries

Restrictions on Leakage Queries by

- Bounded Leakage Setting

  $\mathcal{F}_{\text{bbd-I}} = \{f : \{0,1\}^{|lsk|} \rightarrow \{0,1\}^{\leq \lambda_1}\}$, where $\lambda_1 < |lsk|$

  $\mathcal{F}_{\text{bbd-II}} = \{f : \{0,1\}^{|esk|} \rightarrow \{0,1\}^{\leq \lambda_2}\}$, where $\lambda_2 < |esk|$

- Leakage Function Commitment

  1. commits $\mathcal{F}_1 \subseteq \mathcal{F}_{\text{bbd-I}}$ (resp., $\mathcal{F}_2 \subseteq \mathcal{F}_{\text{bbd-II}}$) before revealing the corresponding esk (resp., lsk)
  2. queries any $f_1 \in \mathcal{F}_{\text{bbd-I}}$ (resp., $f_2 \in \mathcal{F}_{\text{bbd-II}}$) before revealing the corresponding esk (resp., lsk) and thereafter can only ask $f_1 \in \mathcal{F}_1$ (resp. $f_2 \in \mathcal{F}_2$)

$$\mathsf{K}^* = \mathcal{A}(lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}, \mathsf{Msgs}^*) = \mathcal{B}(lsk_{\mathcal{B}}, esk^*_{\mathcal{B}}, \mathsf{Msgs}^*)$$

$\rightsquigarrow$ **still adaptive leakage**?

Restrictions on Leakage Queries

Restrictions on Leakage Queries by 

- Bounded Leakage Setting
  $\mathcal{F}_{\text{bbd-I}} = \{f : \{0,1\}^{|lsk|} \rightarrow \{0,1\}^{\leq \lambda_1}\}$, where $\lambda_1 < |lsk|$
  $\mathcal{F}_{\text{bbd-II}} = \{f : \{0,1\}^{|esk|} \rightarrow \{0,1\}^{\leq \lambda_2}\}$, where $\lambda_2 < |esk|$
- Leakage Function Commitment
  1.  commits $\mathcal{F}_1 \subseteq \mathcal{F}_{\text{bbd-I}}$ (resp., $\mathcal{F}_2 \subseteq \mathcal{F}_{\text{bbd-II}}$) before revealing the corresponding esk (resp., lsk)
  2.  queries any $f_1 \in \mathcal{F}_{\text{bbd-I}}$ (resp., $f_2 \in \mathcal{F}_{\text{bbd-II}}$) before revealing the corresponding esk (resp., lsk) and thereafter can only ask $f_1 \in \mathcal{F}_1$ (resp. $f_2 \in \mathcal{F}_2$)

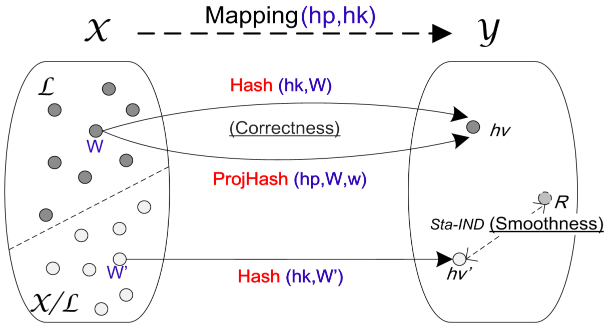
$$\mathsf{K}^* = \mathcal{A}(lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}, \mathsf{Msgs}^*) = \mathcal{B}(lsk_{\mathcal{B}}, esk^*_{\mathcal{B}}, \mathsf{Msgs}^*)$$

$\rightsquigarrow$ **still adaptive leakage**? 

The Generic Construction

Building Blocks

- Randomness Extractor Ext
- Pseudo-Random Function PRF, πPRF
- Smooth Projective Hash Function (Extension)
  SPHF=(SPHFSetup, HashKG, ProjKG, WordG, Hash, ProjHash)



RSAConference2016

Core Component Overview

$\mathcal{A}$		$\mathcal{B}$
$lsk_{\mathcal{A}} = \mathsf{hk}, lpk_{\mathcal{A}} = (\mathsf{hp}, r_{\mathcal{A}_1}, r_{\mathcal{A}_2})$		$lsk_{\mathcal{B}} = \mathsf{hk}', lpk_{\mathcal{B}} = (\mathsf{hp}', r_{\mathcal{B}_1}, r_{\mathcal{B}_2})$
$esk_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{u(k)}, t_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{t_3(k)},$		$esk_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{u(k)}, t_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{t_3(k)}$
$\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}_1})$		$\widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_{\mathcal{B}_1})$
$\widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(esk_{\mathcal{A}}, r_{\mathcal{A}_2})$		$\widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(esk_{\mathcal{B}}, r_{\mathcal{B}_2})$
$(w_{\mathcal{A}}, x) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_{\mathcal{A}_1})$		$(w_{\mathcal{B}}, y) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk_{\mathcal{B}}) + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_{\mathcal{B}_1})$
$W_{\mathcal{A}} = \mathsf{WordG}(w_{\mathcal{A}}), X = g^x$		$W_{\mathcal{B}} = \mathsf{WordG}(w_{\mathcal{B}}), Y = g^y$
	$\xrightarrow{(\widehat{B}, \widehat{A}, W_{\mathcal{A}}, X, t_{\mathcal{A}})}$	
	$\xleftarrow{(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, Y, t_{\mathcal{B}})}$	

Main Idea

- $lsk, esk \xrightarrow{\mathsf{Ext}}$ seeds for PRFs
- An additional Diffie-Hellman protocol for shared key g^{xy}
- Hash value of $W_{\mathcal{A}}, W_{\mathcal{B}}$ + $g^{xy} \xrightarrow{\mathsf{Ext}, \pi\mathsf{PRF}}$ K

Security Analysis

Theorem

The generic AKE construction is CLR-eCK-secure.

Proof Sketch

sid^*: *challenge session* chosen by , $\overline{\mathrm{sid}^*}$: *matching session* of sid^*

- **Case1:** $\overline{\mathrm{sid}^*}$ exists: either lsk or esk unknown to $\overset{\text{Ext}}{\Longrightarrow}$ $(x,y) \overset{c}{\equiv} (r_1, r_2) \overset{\$}{\leftarrow} \mathbb{Z}_p \times \mathbb{Z}_p \Longrightarrow g^{xy} \overset{c}{\equiv} r \overset{\$}{\leftarrow} \mathbb{Z}_p \overset{\text{Ext},\pi\text{PRF}}{\Longrightarrow}$ K is random
- **Case2:** $\overline{\mathrm{sid}^*}$ does not exists: lsk unknown to $\overset{\mathcal{L} \overset{c}{\equiv} \mathcal{X} \setminus \mathcal{L}}{\Longrightarrow}$ replace $W_{\mathcal{A}}/W_{\mathcal{B}}$ with $W' \in \mathcal{X} \setminus \mathcal{L} \overset{\text{SPHF}}{\Longrightarrow}$ $\mathrm{Hash}(W', lsk) \overset{\$}{\equiv} r \overset{\$}{\leftarrow} \mathcal{Y} \overset{\text{Ext},\pi\text{PRF}}{\Longrightarrow}$ K is random
- Simulation for *non-challenge session* for : Ext, PRF

RSAConference2016

DDH-Based Instantiation

Let $\mathbb{G}$ be a group of primer order p and $g_1, g_2 \in \mathbb{G}, H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$.

$$\mathcal{L}_{\mathsf{DH}} = \{(u_1, u_2) | \exists r \in \mathbb{Z}_p, s.t., u_1 = g_1^r, u_2 = g_2^r\}$$

Witness space $\mathbb{Z}_p$, $\mathcal{L}_{\mathsf{DH}} \subset \mathcal{X} = \mathbb{G}^2, \mathcal{Y} = \mathbb{G}$.

SPHF on $\mathcal{L}_{\mathsf{DH}}$

- SPHFSetup: param $= (\mathbb{G}, p, g_1, g_2)$
- HashKG: hk $= (\alpha_1, \alpha_2, \beta_1, \beta_2) \xleftarrow{\$} \mathbb{Z}_p^4$
- ProjKG: hp $= (\mathsf{hp}_1, \mathsf{hp}_2) = (g_1^{\alpha_1} g_2^{\alpha_2}, g_1^{\beta_1} g_2^{\beta_2}) \in \mathbb{G}_p^2$
- WordG$(w = r)$: $W = (g_1^r, g_2^r)$
- Hash: hv $= u_1^{\alpha_1 + d\beta_1} u_2^{\alpha_2 + d\beta_2} \quad (d = H_1(W, aux'))$
- ProjHash: hv$' = \mathsf{hp}_1^r \mathsf{hp}_2^{dr}$

Conclusions

- **A New Strongly Leakage-Resilient AKE Security Model**
  - Capture challenge-dependent leakage (w/o split state assumption)
  - Capture partial leakage on the ephemeral secret (randomness)
- **A Generic Construction with an Instantiation**
  - Secure under the new strong model w/o RO
  - Efficient in communication, computation and round complexity
- **Future Work**
  - Stronger leakage setting: Auxiliary Input, Continuous Leakage?
  - Dealing with full leakage: capturing intermediate value leakage?

THANK YOU!

rc517@uowmail.edu.au