Hash Proof Systems and Password Protocols

III – SPHF-based PAKE

David Pointcheval

CNRS, Ecole normale supérieure/PSL & INRIA

8th BIU Winter School – Key Exchange
February 2018

Intuition of PAKE with a Commitment

We denote $\mathcal{L}_{pw}$ the language of the commitments of pw

- Alice sends C_A, a commitment of pw_A, to Bob (no leakage: hiding property)
- Bob can ask to verify that $C_A \in \mathcal{L}_{pw_B}$:
  - Bob sends hp_B to Alice, and computes $\mathsf{H}_A \leftarrow \mathsf{Hash}(hk_B, C_A)$
  - Alice can compute $\mathsf{pH}_A \leftarrow \mathsf{ProjHash}(hp, C_A, w_A)$

$$\mathsf{H}_A = \mathsf{pH}_A \iff pw_A = pw_B$$

Security: If $pw_B \neq pw_A$, H_A is perfectly unpredictable to Alice (smoothness)

For a non-trivial language, the commitment must be perfectly binding
e.g., ElGamal encryption: $C_A = (g^r, h^r \times g^{pw_A})$

SPHF-based PAKE: First Attempt

$\mathcal{X} = \mathbb{G}^2$ and $\mathcal{L}_{pw} = \{(g^r, h^r \times g^{pw})\}$

- Alice sends $C_A = (u = g^r, e = h^r \times g^{pw_A})$ to Bob
- Bob generates $hk = (\alpha, \beta) \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and sends $hp \leftarrow g^\alpha h^\beta$
- Bob computes $\mathsf{H} \leftarrow u^\alpha (e/g^{pw_B})^\beta$
- Alice computes $\mathsf{pH} \leftarrow hp^r$

$$\left.\begin{array}{l} \mathsf{H} \leftarrow u^\alpha (e/g^{pw_B})^\beta \\ \mathsf{pH} \leftarrow hp^r \end{array}\right\} \mathsf{H}_A = \mathsf{pH}_A = g^{\alpha r} h^{\beta r} \iff pw_A = pw_B$$

Security: If $pw_B \neq pw_A$, H is perfectly unpredictable to Alice (smoothness)

C_A does not leak pw_A under the **DDH** assumption
From the view of pH (Reveal-query), Bob can look for pw such that $u^\alpha (e/g^{pw})^\beta = \mathsf{pH}$
$\Longrightarrow$ Off-line dictionary attack!

SPHF-based PAKE

We denote $\mathcal{L}_{pw}$ the language of the commitments of pw

- Alice sends C_A, a commitment of pw_A, to Bob (no leakage: hiding property)
- Bob can ask to verify that $C_A \in \mathcal{L}_{pw_B}$:
  - Bob sends hp_B to Alice, and computes $\mathsf{H}_A \leftarrow \mathsf{Hash}(hk_B, C_A)$
  - Alice can compute $\mathsf{pH}_A \leftarrow \mathsf{ProjHash}(hp, C_A, w_A)$

$$\mathsf{H}_A = \mathsf{pH}_A \iff pw_A = pw_B$$

Bob must also prove his knowledge of $pw_B = pw_A$ before having access to pH

- Either with an implicit proof [Gennaro–Lindell – Eurocrypt '03]
- Or with an explicit proof [Groce-Katz – CCS '10]

Outline

Outline

Outline

SPHF-based PAKE: Implicit Proof

We denote $\mathcal{L}_A/\mathcal{L}_B$ the languages of the commitments of pw_A/pw_B

- Alice sends C_A, a commitment of pw_A, to Bob
- Bob can ask to verify that $C_A \in \mathcal{L}_B$:
  - Bob sends hp_B to Alice, and computes $\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
  - Alice can compute $\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, w_A)$
- Bob sends C_B, a commitment of pw_B, to Alice
- Alice can ask to verify that $C_B \in \mathcal{L}_A$:
  - Alice sends hp_A to Bob, and computes $\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$
  - Bob can compute $\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, w_B)$
- Bob computes $K_B \leftarrow \mathsf{H}_A \oplus \mathsf{pH}_B$
- Alice computes $K_A \leftarrow \mathsf{pH}_A \oplus \mathsf{H}_B$

$$K_B = \mathsf{H}_A \oplus \mathsf{pH}_B = \mathsf{pH}_A \oplus \mathsf{H}_B = K_A \Longleftrightarrow pw_A = pw_B$$

SPHF-based PAKE: Man-In-The-Middle Attack

$\mathcal{X} = \mathbb{G}^2$ and $\mathcal{L}_{pw} = \{(g^r, h^r \times g^{pw})\}$

- Alice sends $C_A = (u_A = g^{r_A}, e_A = h^{r_A} \times g^{pw_A})$ to Bob
- Bob generates $hk_B = (\alpha_B, \beta_B) \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and sends $hp_B \leftarrow g^{\alpha_B} h^{\beta_B}$
- Bob sends $C_B = (u_B = g^{r_B}, e_B = h^{r_B} \times g^{pw_B})$ to Alice
- Alice generates $hk_A = (\alpha_A, \beta_A) \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and sends $hp_A \leftarrow g^{\alpha_A} h^{\beta_A}$
- Alice computes $K_A \leftarrow u_B^{\alpha_A} \cdot (e_B/g^{pw_A})^{\beta_A} \times hp_B^{r_A}$
- Bob computes $K_B \leftarrow hp_A^{r_B} \times u_A^{\alpha_B} \cdot (e_A/g^{pw_B})^{\beta_B}$

$$\Big\} \; K_A = K_B \Longleftrightarrow pw_A = pw_B$$

The adversary can do a man-in-the-middle attack:

- forwards everything
- excepted C_B to Alice, that is replaced by $C'_B = C_B \times (g, h)$

$$K'_A = u_B^{\alpha_A} g^{\alpha_A} \cdot (e_B/g^{pw_A})^{\beta_A} h^{\beta_A} \times hp_B^{r_A} = K_A \times g^{\alpha_A} h^{\beta_A} = K_B \times hp_A$$

SPHF-based PAKE: Man-In-The-Middle Attack

From the man-in-the-middle attack:

- the adversary can ask for a Reveal-query to Alice
- the adversary can ask for a Test-query to Bob (the session ID's are different)
- the adversary can check the relation between the keys to decide on b'

The commitment C_B must be non-malleable or confirmed to Bob

GL-PAKE

[Gennaro-Lindell – Eurocrypt '03]

Alice		Bob
$r_A \xleftarrow{\$} \$; C_A \leftarrow \mathsf{Enc}(pw_A, r_A)$	$\xrightarrow{C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}(); hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
		$\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$r_B \xleftarrow{\$} \$; C_B \leftarrow \mathsf{Enc}'(pw_B, r_B)$
$hk_A \xleftarrow{\$} \mathsf{HashKG}(); hp_A \leftarrow \mathsf{ProjKG}(hk_A)$		
$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$	$\xrightarrow{hp_A}$	$\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		$K_B \leftarrow \mathsf{pH}_B \times \mathsf{H}_A$

Which are the security properties of the encryption schemes ?

GL-PAKE: Security Proof

Send-queries to Bob: Oracle-Generated C_A with $pw_A = pw_B = pw$

Alice		Bob
$r_A \xleftarrow{\$} \$; C_A \leftarrow \mathsf{Enc}(pw, r_A)$	$\xrightarrow{C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}(); hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
		$\cancel{\mathsf{H}_A \leftarrow \mathsf{Hash}(hk_B, C_A)}$
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}(hp_B, C_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$\cancel{r_B \xleftarrow{\$} \$; C_B \leftarrow \mathsf{Enc}'(pw, r_B)}$ $C_B \leftarrow \mathsf{Enc}'(\$, \$)$
$hk_A \xleftarrow{\$} \mathsf{HashKG}(); hp_A \leftarrow \mathsf{ProjKG}(hk_A)$		
$\mathsf{H}_B \leftarrow \mathsf{Hash}(hk_A, C_B)$	$\xrightarrow{hp_A}$	$\cancel{\mathsf{pH}_B \leftarrow \mathsf{ProjHash}(hp_A, C_B, r_B)}$ $\mathsf{pH}_B \leftarrow \mathsf{H}_B$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		$\cancel{K_B \leftarrow \mathsf{pH}_B \times \mathsf{H}_A}$ $K_B \leftarrow K_A$

- Oracle-generated C_A should imply oracle-generated hp_A
- Correctness
- Oracle-generated hp_A should confirm hp_B: Correctness
- IND-CPA

GL-PAKE: Security Proof

Send-queries to Bob: Oracle-Generated C_A with $pw_A \neq pw_B$

Alice

$r_A \stackrel{\$}{\leftarrow} \$;\ C_A \leftarrow \mathsf{Enc}(pw_A, r_A)$ $\xrightarrow{\ C_A\ }$

$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$ $\xleftarrow{\ hp_B, C_B\ }$

$hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$

$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$ $\xrightarrow{\ hp_A\ }$

$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$

Bob

$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$\cancel{\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)}$

$\cancel{r_B \stackrel{\$}{\leftarrow} \$; C_B \leftarrow \mathsf{Enc}'(pw, r_B)}$ $\quad C_B \leftarrow \mathsf{Enc}'(\$, \$)$

$\cancel{\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)}$

$\cancel{K_B \leftarrow \mathsf{pH}_B \times \mathsf{H}_A}$ $\quad K_A \leftarrow \$$

- Smoothness
- IND-CPA

GL-PAKE: Security Proof

Send-queries to Bob: Non Oracle-Generated C_A

Bob

$\xrightarrow{\ C_A\ }$

$\mathsf{Dec}(C_A) \stackrel{?}{=} pw_B$: STOP&WIN

$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$\cancel{\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)}$

$\cancel{r_B \stackrel{\$}{\leftarrow} \$; C_B \leftarrow \mathsf{Enc}'(pw, r_B)}$ $\quad C_B \leftarrow \mathsf{Enc}'(\$, \$)$ $\xleftarrow{\ hp_B, C_B\ }$

$\xrightarrow{\ hp_A\ }$ $\cancel{\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)}$

$\cancel{K_B \leftarrow \mathsf{pH}_B \times \mathsf{H}_A}$ $\quad K_B \leftarrow \$$

- The adversary must encrypt the correct password: password-guessing probability
- Smoothness
- IND-CPA

GL-PAKE: Security Proof

Send-queries to Alice: Oracle-Generated C_B

Alice

$r_A \stackrel{\$}{\leftarrow} \$;\ C_A \leftarrow \mathsf{Enc}(pw_A, r_A)$ $\xrightarrow{\ C_A\ }$

$\cancel{\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)}$ $\xleftarrow{\ hp_B, C_B\ }$

$hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$ $\xrightarrow{\ hp_A\ }$

$\cancel{\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)}$

$K_A \stackrel{\$}{\leftarrow} \$$ $\quad \cancel{K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A}$

Oracle-Generated C_A

Bob

$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$C_B \leftarrow \mathsf{Enc}'(\$, \$)$

If $pw_A = pw_B$, $K_B \leftarrow K_A$

If $pw_A \neq pw_B$, $K_B \leftarrow \$$

Non Oracle-Generated C_A

Bob

$\xrightarrow{\ C_A\ }$ $\mathsf{Dec}(C_A) \stackrel{?}{=} pw_B$: STOP&WIN

$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$\xleftarrow{\ hp_B, C_B\ }$ $C_B \leftarrow \mathsf{Enc}'(\$, \$)$

$\xrightarrow{\ hp_A\ }$ $K_B \leftarrow \$$

Smoothness

GL-PAKE: Security Proof

Send-queries to Alice: Non Oracle-Generated C_B

Alice

$r_A \xleftarrow{\$} \$;\ C_A \leftarrow \mathsf{Enc}(pw_A, r_A)$ $\xrightarrow{C_A}$

$\cancel{pH_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)}$ $\xleftarrow{hp_B, C_B}$

$hk_A \xleftarrow{\$} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$ $\xrightarrow{hp_A}$

$\mathsf{Dec}'(C_B) \stackrel{?}{=} pw_A$: STOP&WIN

$\cancel{H_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)}$

$K_A \xleftarrow{\$} \$$ $\quad \cancel{K_A \leftarrow H_B \times pH_A}$

- The adversary must encrypt the correct password: password-guessing probability
- Smoothness

GL-PAKE: Security Proof

Oracle-Generated C_B — Oracle-Generated C_A

Alice

$C_A \leftarrow \mathsf{Enc}(\$, \$)$ $\quad \cancel{r_A \xleftarrow{\$} \$; C_A \leftarrow \mathsf{Enc}(pw_A, r_A)}$ $\xrightarrow{C_A}$

$hk_A \xleftarrow{\$} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$ $\xleftarrow{hp_B, C_B}$

$K_A \leftarrow \$$ $\xrightarrow{hp_A}$

Bob

$hk_B \xleftarrow{\$} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$C_B \leftarrow \mathsf{Enc}'(\$, \$)$

$\cancel{\text{If } pw_A = pw_B}$ If compatible oracles, $K_B \leftarrow K_A$

$\cancel{\text{If } pw_A \neq pw_B}$ If incompatible oracles, $K_B \leftarrow \$$

Non Oracle-Generated C_B — Non Oracle-Generated C_A

Alice

$C_A \leftarrow \mathsf{Enc}(\$, \$)$ $\quad \cancel{r_A \xleftarrow{\$} \$; C_A \leftarrow \mathsf{Enc}(pw_A, r_A)}$ $\xrightarrow{C_A}$

$hk_A \xleftarrow{\$} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$ $\xleftarrow{hp_B, C_B}$

$\mathsf{Dec}'(C_B)$ in $\mathcal{P}$ $\quad \cancel{\mathsf{Dec}'(C_B) \stackrel{?}{=} pw_A : \text{STOP\&WIN}}$ $\xrightarrow{hp_A}$

$K_A \leftarrow \$$

Bob

$\cancel{\mathsf{Dec}(C_A) \stackrel{?}{=} pw_B : \text{STOP\&WIN}}$ $\quad \mathsf{Dec}(C_A)$ in $\mathcal{P}$

$hk_B \xleftarrow{\$} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$C_B \leftarrow \mathsf{Enc}'(\$, \$)$

$K_B \leftarrow \$$

- IND-CCA + No abort anymore: difference if the guesses are correct

GL-PAKE: Security Proof

To be more precise, in the final game

- The Execute-queries just work as Send-queries with oracle-generated flows
- The actual passwords are not set at the beginning, but randomly chosen at the end
- WIN = a random password (with Player ID) is in $\mathcal{P}$: the probability is q_S/N

Encryption schemes:

- $(\mathsf{Enc}, \mathsf{Dec})$: SPHF-friendly L-IND-CCA encryption scheme $\ell = (A, B, vk)$
  $\Longrightarrow$ where vk is the verification key of a OT-Signature
  $\Longrightarrow$ Labeled Cramer-Shoup Encryption
- $(\mathsf{Enc}', \mathsf{Dec}')$: SPHF-friendly IND-CPA encryption scheme
  $\Longrightarrow$ ElGamal Encryption
- (C_A, hp_B, C_B, hp_A) signed by A: OT-signature (sk, vk)
  $\Longrightarrow$ an oracle-generated C_A implies the same oracle-generated hp_A,
  and confirms the received (hp_B, C_B)

Labeled Cramer-Shoup Ciphertext Languages

Cramer-Shoup Encryption Scheme is an L-IND-CCA PKE:

$$C = (u_1 = g_1^r, u_2 = g_2^r, e = h^r m, v = (cd^t)^r) \text{ with } t = \mathcal{H}(\ell, u_1, u_2, e)$$

C is a CS ciphertext of pw iff $(u_1, u_2, e/pw, v)$ is an r-th power of (g_1, g_2, h, cd^t)

$$\mathsf{HashKG}() : hk = (\alpha, \beta, \gamma, \delta) \stackrel{\$}{\leftarrow} \mathbb{Z}_q^4 \qquad \mathsf{ProjKG}(hk, C) : hp = g_1^\alpha g_2^\beta h^\gamma (cd^t)^\delta$$

$$\mathsf{Hash}(hk, C) : \mathsf{H} = u_1^\alpha u_2^\beta (e/pw)^\gamma v^\delta \qquad \mathsf{ProjHash}(hp, C, r) : \mathsf{pH} = hp^r$$

This is not a CS-SPHF, hence the GL relaxation [Gennaro-Lindell – Eurocrypt '03]

GL-PAKE: Complete Protocol

[Gennaro-Lindell – Eurocrypt '03]

Alice		Bob
$(sk, vk) \stackrel{\$}{\leftarrow} \mathsf{SignKG}(); \ell = (A, B, vk)$ $r_A \stackrel{\$}{\leftarrow} \$; C_A \leftarrow \mathsf{CS}^\ell(pw_A, r_A)$	$\xrightarrow{vk, C_A}$	
		$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_B \leftarrow \mathsf{ProjKG}(hk_B, C_A)$ $\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$ $r_B \stackrel{\$}{\leftarrow} \$; C_B \leftarrow \mathsf{EG}(pw_B, r_B)$
	$\xleftarrow{hp_B, C_B}$	
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$ $hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_A \leftarrow \mathsf{ProjKG}(hk_A, C_B)$ $\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$ $\Sigma \leftarrow \mathsf{Sign}(sk, (C_A, hp_B, C_B, hp_A))$	$\xrightarrow{hp_A, \Sigma}$	
		$\mathsf{Verif}(vk, (C_A, hp_B, C_B, hp_A), \Sigma)?$ $\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$ $K_B \leftarrow \mathsf{H}_A \times \mathsf{pH}_B$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		

A key confirmation can be added to the third flow: Explicit Authentication of Alice

Outline

SPHF-based PAKE: Explicit Proof

We denote $\mathcal{L}_A/\mathcal{L}_B$ the languages of the commitments C of pw_A/pw_B

- Alice sends C_A, a commitment of pw_A with random coins r_A, to Bob
- Bob can ask to verify that $C_A \in \mathcal{L}_B$:
  - Bob sends hp_B to Alice, and computes $\mathrm{H}_A \leftarrow \mathrm{Hash}_B(hk_B, C_A)$
  - Alice can compute $\mathrm{pH}_A \leftarrow \mathrm{ProjHash}_A(hp, C_A, r_A)$
- Alice parses pH_A as $r'_B \| K_A$
- Bob parses H_A as $r_B \| K_B$
- Bob sends C_B, a commitment of pw_B with random coins r_B, to Alice
- Alice can recompute the commitment C'_B of pw_A with random coins r'_B and check whether $C'_B \stackrel{?}{=} C_B$

For a non-trivial language, the commitment C_A must be perfectly binding

To avoid false positive on $C'_B \stackrel{?}{=} C_B$, the commitment C_B must be perfectly binding

e.g., Public-Key Encryption Scheme

GK-PAKE

[Groce-Katz – CCS '10]

Alice		Bob
$r_A \stackrel{\$}{\leftarrow} \$; C_A \leftarrow \mathrm{Enc}(pw_A, r_A)$	$\xrightarrow{C_A}$	$hk_B \stackrel{\$}{\leftarrow} \mathrm{HashKG}(); hp_B \leftarrow \mathrm{ProjKG}(hk_B)$
		$r_B \| K_B = \mathrm{H}_A \leftarrow \mathrm{Hash}(hk_B, C_A)$
$r'_B \| K_A = \mathrm{pH}_A \leftarrow \mathrm{ProjHash}(hp_B, C_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$C_B \leftarrow \mathrm{Enc}'(pw_B, r_B)$
$C'_B \leftarrow \mathrm{Enc}'(pw_A, r'_B)$		
$C'_B \stackrel{?}{=} C_B$		

Which are the security properties of the encryption schemes ?

GK-PAKE: Security Proof

Send-query to Alice: Oracle-Generated C_B with $pw_B = pw_A = pw$

Alice		Bob
$r_A \stackrel{\$}{\leftarrow} \$; C_A \leftarrow \mathrm{Enc}(pw, r_A)$	$\xrightarrow{C_A}$	$hk_B \stackrel{\$}{\leftarrow} \mathrm{HashKG}(); hp_B \leftarrow \mathrm{ProjKG}(hk_B)$
		$r_B \| K_B \leftarrow \mathrm{Hash}(hk_B, C_A)$
$K_A \leftarrow K_B$ $\cancel{r'_B \| K_A \leftarrow \mathrm{ProjHash}(hp_B, C_A, r_A)}$	$\xleftarrow{hp_B, C_B}$	$C_B \leftarrow \mathrm{Enc}'(pw, r_B)$
$C'_B \leftarrow C_B$ $\cancel{C'_B \leftarrow \mathrm{Enc}'(pw, r'_B)}$		
OK $\cancel{C'_B \stackrel{?}{=} C_B}$		

- C_B must be specific to this execution
- Oracle-Generated C_B must imply Oracle-Generated hp_B
- Correctness

GK-PAKE: Security Proof

Send-query to Alice: Oracle-Generated C_B with $pw_B \neq pw_A$

Alice

$r_A \xleftarrow{\$} \$;\ C_A \leftarrow \mathsf{Enc}(pw_A, r_A)$ $\xrightarrow{C_A}$

$\cancel{r'_B \| K_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)}$ $\xleftarrow{hp_B, C_B}$

$\cancel{C'_B \leftarrow \mathsf{Enc}'(pw_A, r'_B)}$

NOK $\cancel{C'_B \stackrel{?}{=} C_B}$

Bob

$hk_B \xleftarrow{\$} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$r_B \| K_B \leftarrow \mathsf{Hash}_B(hk_B, C_A)$

$C_B \leftarrow \mathsf{Enc}'(pw_B, r_B)$

- $pw_A \neq pw_B \Longrightarrow C'_B \neq C_B$

GK-PAKE: Security Proof

Send-query to Alice: Non Oracle-Generated C_B

Alice

$r_A \xleftarrow{\$} \$;\ C_A \leftarrow \mathsf{Enc}(pw_A, r_A)$ $\xrightarrow{C_A}$

$\cancel{r'_B \| K_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)}$ $\xleftarrow{hp_B, C_B}$

$\mathsf{Dec}'(C_B) \stackrel{?}{=} pw_A$: STOP&WIN $\cancel{C'_B \leftarrow \mathsf{Enc}'(pw_A, r'_B)}$

NOK $\cancel{C'_B \stackrel{?}{=} C_B}$

- C_B specific to this execution, and non-malleable
- The adversary must encrypt the correct password: password-guessing probability
- $\mathsf{Dec}'(C_B) \neq pw_A \Longrightarrow C'_B \neq C_B$

GK-PAKE: Security Proof

Send-query to Alice: Oracle-Generated C_B

Alice

$C_A \leftarrow \mathsf{Enc}(\$, \$)$ $\cancel{r_A \xleftarrow{\$} \$; C_A \leftarrow \mathsf{Enc}(pw_A, r_A)}$ $\xrightarrow{C_A}$

If $pw_A = pw_B$, $K_A \leftarrow K_B$, OK $\xleftarrow{hp_B, C_B}$

If $pw_A \neq pw_B$, NOK

Bob

$hk_B \xleftarrow{\$} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$r_B \| K_B \leftarrow \mathsf{Hash}_B(hk_B, C_A)$

$C_B \leftarrow \mathsf{Enc}'(pw_P, r_B)$

Send-query to Alice: Non Oracle-Generated C_B

Alice

$C_A \leftarrow \mathsf{Enc}(\$, \$)$ $\cancel{r_A \xleftarrow{\$} \$; C_A \leftarrow \mathsf{Enc}(pw_A, r_A)}$ $\xrightarrow{C_A}$

$\mathsf{Dec}'(C_B) \stackrel{?}{=} pw_A$: STOP&WIN $\xleftarrow{hp_B, C_B}$

NOK

IND-CPA

GK-PAKE: Security Proof

Send-query to Bob: Oracle-Generated C_A

Alice		Bob
$C_A \leftarrow \mathsf{Enc}(\$, \$)$	$\xrightarrow{C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}()$; $hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
		$\cancel{r_B \| K_B \leftarrow \mathsf{Hash}_B(hk_B, C_A)}$ $K_B \xleftarrow{\$} \$$
If $pw_A = pw_B$, $K_A \leftarrow K_B$, OK	$\xleftarrow{hp_B, C_B}$	$\cancel{C_B \leftarrow \mathsf{Enc}'(pw_B, r_B)}$ $C_B \leftarrow \mathsf{Enc}'(\$, \$)$
If $pw_A \neq pw_B$, NOK		

- Correctness + IND-CCA

GK-PAKE: Security Proof

Send-query to Bob: Non Oracle-Generated C_A

	Bob
$\xrightarrow{C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}()$; $hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
	$\mathsf{Dec}(C_A) \stackrel{?}{=} pw_B$: STOP&WIN
	$\cancel{r_B \| K_B \leftarrow \mathsf{Hash}_B(hk_B, C_A)}$ $K_B \xleftarrow{\$} \$$
$\xleftarrow{hp_B, C_B}$	$\cancel{C_B \leftarrow \mathsf{Enc}'(pw_B, r_B)}$ $C_B \leftarrow \mathsf{Enc}'(\$, \$)$

- The adversary must encrypt the correct password: password-guessing probability
- Smoothness + IND-CCA

GK-PAKE: Security Proof

Oracle-Generated C_B / Oracle-Generated C_A

Alice		Bob
$C_A \leftarrow \mathsf{Enc}(\$, \$)$	$\xrightarrow{C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}()$; $hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
		$K_B \xleftarrow{\$} \$$
If compatible oracles $\cancel{\text{If } pw_A = pw_B}$, $K_A \leftarrow K_B$, OK	$\xleftarrow{hp_B, C_B}$	$C_B \leftarrow \mathsf{Enc}'(\$, \$)$
If incompatible oracles $\cancel{\text{If } pw_A \neq pw_B}$, NOK		

Non Oracle-Generated C_B / Non Oracle-Generated C_A

Alice		Bob
$C_A \leftarrow \mathsf{Enc}(\$, \$)$	$\xrightarrow{C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}()$; $hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
		$\cancel{\mathsf{Dec}(C_A) \stackrel{?}{=} pw_B : \text{STOP\&WIN}}$ $\mathsf{Dec}(C_A)$ in $\mathcal{P}$
		$K_B \xleftarrow{\$} \$$
$\mathsf{Dec}'(C_B)$ in $\mathcal{P}$ $\cancel{\mathsf{Dec}'(C_B) \stackrel{?}{=} pw_A : \text{STOP\&WIN}}$	$\xleftarrow{hp_B, C_B}$	$C_B \leftarrow \mathsf{Enc}'(\$, \$)$
NOK		

- No abort anymore: difference if the guesses are correct

GK-PAKE: Security Proof

To be more precise, in the final game

- The Execute-queries just work as Send-queries with oracle-generated flows
- The actual passwords are not set at the beginning, but randomly chosen at the end
- WIN = a random password (with Player ID) is in $\mathcal{P}$: the probability is q_S/N

Encryption schemes:

- $(\mathsf{Enc}, \mathsf{Dec})$: SPHF-friendly IND-CPA encryption scheme
  $\Longrightarrow$ ElGamal Encryption
- $(\mathsf{Enc}', \mathsf{Dec}')$: L-IND-CCA encryption scheme: $\ell' = (A, B, C_A, hp_B)$
  $\Longrightarrow$ this makes C_B specific to this execution because of C_A
  $\Longrightarrow$ an oracle-generated C_B implies the same oracle-generated hp_B
  $\Longrightarrow$ Labeled Cramer-Shoup Encryption

GK-PAKE

[Groce-Katz – CCS '10]

- Alice generates and sends $C_A = (u \leftarrow g^{r_A}, e \leftarrow h^{r_A} g^{pw_A}) \in \mathbb{G}^2$
- Bob
  - generates $hk_B = (\alpha, \beta)$ and $hp_B = g^\alpha h^\beta$
  - computes $r_B \| K_B = KDF(u^\alpha \cdot (e/g^{pw_B})^\beta)$
  - generates $C_B = (u_1 = g_1^{r_B}, u_2 = g_2^{r_B}, e = h^{r_B} g^{pw_B}, v = (cd^t)^{r_B})$, with $t = \mathcal{H}(A, B, C_A, hp_B, u_1, u_2, e)$
  - sends $(hp_B, C_B) \in \mathbb{G}^5$
- Alice
  - computes $r'_B \| K_A = KDF(hp_B^{r_A})$
  - generates $C'_B = (u'_1 = g_1^{r'_B}, u'_2 = g_2^{r'_B}, e = h^{r'_B} g^{pw_A}, v' = (cd^{t'})^{r'_B}$, with $t' = \mathcal{H}(A, B, C_A, hp_B, u'_1, u'_2, e')$
  - aborts if $C'_B \neq C_B$

A key confirmation can be added to the second flow: Explicit Authentication of Bob

Outline

Better Efficiency

IND-PCA

Security proofs: WIN = a random password (with Player ID) is in $\mathcal{P}$

- One either decrypts every C into pw and checks whether $pw \in \mathcal{P}$ or not
  $\Longrightarrow$ need of decryption oracle
- Or one checks for every $pw \in \mathcal{P}$ whether C encrypts pw for some ciphertext C
  $\Longrightarrow$ need of plaintext-checking oracle

The previous proofs work for IND-PCA encryption schemes, instead of IND-CCA

KV-SPHF

Number of flows in GL-PAKE: 3 flows because hp depends on/after χ
With hp possibly sent before χ: 1-round protocol with KV-SPHF

IND-PCA Encryption from KEM

$$
\begin{array}{rl}
\mathsf{KeyGen}(): & (sk, pk) \stackrel{\$}{\leftarrow} \mathsf{EncapsKG}() \\
 & hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}(), hp \leftarrow \mathsf{ProjKG}(hk) \\
 & sk' = (sk, hk), pk' = (pk, hp) \\
\mathsf{Enc}(pk' = (pk, hp), m): & (K, c) \stackrel{\$}{\leftarrow} \mathsf{Encaps}(pk, r), e \leftarrow m + K \\
 & \chi = (K, c), w = r \\
 & t = \mathcal{H}(c, e), v = \mathsf{ProjHash}(hp, \chi, t, w) \\
 & C = (c, e, v) \\
\mathsf{Dec}(sk' = (sk, hk), C = (c, e, v)): & K \leftarrow \mathsf{Decaps}(sk, c), m \leftarrow e - K \\
 & \chi = (K, c), t \leftarrow \mathcal{H}(c, e), v' \leftarrow \mathsf{Hash}(hk, \chi, t) \\
 & \text{If } (v' \neq v) \Longrightarrow \text{Reject} \\
 & \text{Else Return } m
\end{array}
$$

IND-PCA Security Proof

$$
\begin{array}{lcl}
m_0, m_1 \in \mathbb{G} & \xleftarrow{pk' = (pk, hp)} & (sk, pk) \leftarrow \mathsf{EncapsKG}(), hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}(), hp \leftarrow \mathsf{ProjKG}(hk) \\
 & \xrightarrow{m_0, m_1} & b \stackrel{\$}{\leftarrow} \{0, 1\} \\
 & & \cancel{\chi = (K, c) \stackrel{\$}{\leftarrow} \mathsf{Encaps}(pk, w)} \quad \chi = (\$, c), (K, c) \stackrel{\$}{\leftarrow} \mathsf{Encaps}(pk, \$) \\
 & & \cancel{e \leftarrow K + m_b} \quad e \stackrel{\$}{\leftarrow} \mathbb{G} \\
 & & t = \mathcal{H}(c, e);\ \cancel{v \leftarrow \mathsf{ProjHash}(hp, \chi, t, w)} \quad v \leftarrow \mathsf{Hash}(hk, \chi, t) \\
b' \in \{0, 1\} & \xleftarrow{C} & C \leftarrow (c, e, v) \\
 & \xrightarrow{b'} & b' \stackrel{?}{=} b
\end{array}
$$

$$
\begin{array}{lll}
 & PCA(sk', m', C'): & \cancel{K' \leftarrow \mathsf{Decaps}(sk, c'), m' \leftarrow e' - K'} \\
\cancel{sk' = (sk, hk)} \quad sk' = (hk) & & \chi' \leftarrow (e' - m', c'), t' \leftarrow \mathcal{H}(c', e') \qquad C' \neq C \Longrightarrow (\chi', t') \neq (\chi, t) \\
C' = (c', e', v') & & v'' \leftarrow \mathsf{Hash}(hk, \chi', t') \\
 & & v'' \neq v' \ \cancel{\text{or } m'' \neq m'} \qquad \Longrightarrow \text{Reject}
\end{array}
$$

Smoothness (soundness): $v'' = v' \Longrightarrow e' - m'$ valid key $\Longrightarrow m'$ valid plaintext
+ 2-Universal Smoothness (simulation-soundness)
Correctness + Indistinguishability/Hard Subset Membership $\Longrightarrow \Pr[b' = b] = \frac{1}{2}$

SCS: Short Cramer-Shoup

[Abdalla-Benhamouda-P. – PKC '15]

$$\begin{aligned}
\mathsf{KeyGen}(): \quad & sk = (s, hk_1 = (x_1, x_2), hk_2 = (y_1, y_2)) \stackrel{\$}{\leftarrow} \mathbb{Z}_q^5 \\
& pk = (g_2 \leftarrow g_1^s, hp_1 \leftarrow g_1^{x_1} g_2^{x_2}, hp_2 \leftarrow g_1^{y_1} g_2^{y_2}) \\
\mathsf{Enc}(pk, m): \quad & r \stackrel{\$}{\leftarrow} \mathbb{Z}_q;\ u = g_1^r, e \leftarrow g_2^r \times m \\
& v = (hp_1 \times hp_2{}^t)^r, \text{ with } t \leftarrow \mathcal{H}(u, e) \\
\mathsf{Dec}(sk, (u, e, v)): \quad & m = e/u^s, v \stackrel{?}{=} u^{x_1 + tx_2} \times (e/m)^{x_2 + ty_2}, \text{ with } t \leftarrow \mathcal{H}(u, e)
\end{aligned}$$

Theorem (IND-PCA Security)

This SCS encryption scheme is IND-PCA under the DDH assumption

GK-SPOKE (Simple Password-Only Key Exchange)

[Abdalla-Benhamouda-P. – PKC '15]

- Alice generates and sends $C_A = (u \leftarrow g^{r_A}, e \leftarrow h^{r_A} g^{pw_A}) \in \mathbb{G}^2$
- Bob
  - generates $hk_B = (\alpha, \beta)$ and $hp_B = g^\alpha h^\beta$
  - computes $r_B \| K_B = KDF(u^\alpha \cdot (e/g^{pw_B})^\beta)$
  - generates $C_B = (u = g_1^{r_B}, e = g_2^{r_B} g^{pw_B}, v = (cd^t)^{r_B})$, with $t = \mathcal{H}(A, B, C_A, hp_B, u, e)$
  - sends $(hp_B, C_B) \in \mathbb{G}^4$
- Alice
  - computes $r'_B \| K_A = KDF(hp_B^{r_A})$
  - generates $C'_B = (u' = g_1^{r'_B}, e' = g_2^{r'_B} g^{pw_A}, v' = (cd^{t'})^{r'_B}$, with $t' = \mathcal{H}(A, B, C_A, hp_B, u', e')$
  - aborts if $C'_B \neq C_B$

Instead of 7 group elements, only 6 group elements with a 2-flow protocol

GL-PAKE: Reminder of the Idea

[Gennaro-Lindell – Eurocrypt '03]

Alice

- Alice sends C_A, a commitment of pw_A, to Bob
- Bob can ask to verify that $C_A \in \mathcal{L}_B$ (language of commitments of pw_B):
  - Bob sends hp_B to Alice, and computes $\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
  - Alice can compute $\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, w_A)$

Bob

- Bob sends C_B, a commitment of pw_B, to Alice
- Alice can ask to verify that $C_B \in \mathcal{L}_A$ (language of commitments of pw_A):
  - Alice sends hp_A to Bob, and computes $\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$
  - Bob can compute $\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, w_B)$

$$K_B = \mathsf{H}_A \oplus \mathsf{pH}_B = \mathsf{pH}_A \oplus \mathsf{H}_B = K_A \iff pw_A = pw_B$$

KV-PAKE: Katz-Vaikuntanathan's Idea

[Katz-Vaikuntanathan – TCC '11]

Both are sent in parallel:

- Alice sends C_A, hp_A to Bob
- Bob sends C_B, hp_B to Alice

Upon reception of the partner's flow:

- Alice computes $\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, w_A)$ and $\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$
- Bob computes $\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, w_B)$ and $\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$

Then

$$K_B = \mathsf{H}_A \oplus \mathsf{pH}_B = \mathsf{pH}_A \oplus \mathsf{H}_B = K_A$$

KV Smoothness

[Katz-Vaikuntanathan – TCC '11]

KV-Smoothness

$\forall f \rightarrow \mathcal{X} \backslash \mathcal{L}$, with the probability space $hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}()$, $hp \leftarrow \mathsf{ProjKG}(hk)$

$$\{(hp, \mathsf{H}) \,|\, \mathsf{H} \leftarrow \mathsf{Hash}(hk, f(hp))\} \approx \{(hp, \mathsf{H}) \,|\, \mathsf{H} \stackrel{\$}{\leftarrow} \Pi\}$$

Given hp, no adversary can find $\chi \in \mathcal{X} \backslash \mathcal{L}$ for which it can distinguish $\mathsf{Hash}(hk, \chi)$

KV-SPHF for SCS Ciphertexts of m [Abdalla-Benhamouda-P. – PKC '15]

$$c = (u = g^r, e = h^r m, v = (cd^t)^r) \text{ with } t = \mathcal{H}(u, e)$$

$$hk = (\alpha, \beta, \gamma, \delta) \qquad hp = (hp_1 \leftarrow g^\alpha h^\gamma c^\delta, hp_2 \leftarrow g^\beta d^\delta)$$

$$\mathsf{Hash}(hk, c) = u^{\alpha + t\beta} (e/m)^\gamma v^\delta = (hp_1 hp_2^t)^r = \mathsf{ProjHash}(hp, c, r)$$

KV-SPOKE (Simple Password-Only Key Exchange)

[Abdalla-Benhamouda-P. – PKC '15]

- Alice generates
  - $hk_A = (\alpha, \beta, \gamma, \delta)$ and $hp_A = (hp_1 \leftarrow g^\alpha h^\gamma c^\delta, hp_2 \leftarrow g^\beta d^\delta)$
  - $C_A = (u \leftarrow g^{r_A}, e \leftarrow h^{r_A} g^{pw_A}, v \leftarrow (cd^{t_A})^{r_A})$ for $t_A = \mathcal{H}(A, B, u, e, hp_A)$
- Alice sends $C_A \in \mathbb{G}^3$ and $hp_A \in \mathbb{G}^2$
- Alice receives $C_B = (u', e', v') \in \mathbb{G}^3$ and $hp_B = (hp'_1, hp'_2) \in \mathbb{G}^2$ from Bob
- Alice computes
  - $t_B = \mathcal{H}(B, A, u', e', hp_B)$
  - $\mathsf{H}_B = u'^{\alpha + t_B \beta} (e'/pw_A)^\gamma v'^\delta$
  - $\mathsf{pH}_A = (hp'_1 hp'^{t_A}_2)^{r_A}$
  - $K_A = \mathsf{pH}_A \times \mathsf{H}_B$

Only 5 group elements sent by each player in a 2-simultaneously flow protocol

Outline

Outline

SPHF-Based PAKE: Protocol

Alice		Bob
$hk_A \xleftarrow{\$} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$	$\xrightarrow{hp_A, C_A}$	$hk_B \xleftarrow{\$} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
$\ell_A = (A, B, hp_A), r_A \xleftarrow{\$} \$, C_A \leftarrow \mathsf{Enc}^{\ell_A}(pw_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$\ell_B = (B, A, hp_B), r_B \xleftarrow{\$} \$; C_B \leftarrow \mathsf{Enc}^{\ell'}(pw_B, r_B)$
$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$		$\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$		$\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		$K_B \leftarrow \mathsf{H}_A \times \mathsf{pH}_B$

UC-secure against static corruptions

SPHF-Based PAKE: Simulation

Alice		Bob
$hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_A \leftarrow \mathsf{ProjKG}(hk_A)$	$\xrightarrow{hp_A, C_A}$	$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
$\ell_A = (A, B, hp_A), r_A \stackrel{\$}{\leftarrow} \$, C_A \leftarrow \mathsf{Enc}^{\ell_A}(pw_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$\ell_B = (B, A, hp_B), r_B \stackrel{\$}{\leftarrow} \$; C_B \leftarrow \mathsf{Enc}^{\ell'}(pw_B, r_B)$
$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$		$\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$		$\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		$K_B \leftarrow \mathsf{H}_A \times \mathsf{pH}_B$

`NewSession`: for U with U'
- $hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp \leftarrow \mathsf{ProjKG}(hk)$
- $\ell = (U, U', hp), r \stackrel{\$}{\leftarrow} \$, C \leftarrow \mathsf{Enc}^{\ell}(pw, r)$

Flow hp', C'
- Oracle-Generated from U': $hk', hp' \leftarrow \mathsf{ProjKG}(hk'), C' \leftarrow \mathsf{Enc}^{\ell'}(pw', r')$
  - $\mathsf{H} \leftarrow \mathsf{Hash}(hk, C')$
  - $\cancel{\mathsf{pH} \leftarrow \mathsf{ProjHash}(hp', C, r)}$ $\mathsf{pH} \leftarrow \mathsf{Hash}(hk', C)$
  - $K \leftarrow \mathsf{H} \times \mathsf{pH}$

Passwords known for corrupted players

- Non Oracle-Generated: $pw' \leftarrow \mathsf{Dec}^{\ell'}(C')$ and `TestPwd`(pw')
  - $\cancel{\mathsf{H} \leftarrow \mathsf{Hash}(hk, C')}$ $\mathsf{H} \stackrel{\$}{\leftarrow} \$$ if incorrect guess
  - $\mathsf{pH} \leftarrow \mathsf{ProjHash}(hp', C, r)$
  - $\cancel{K \leftarrow \mathsf{H} \times \mathsf{pH}}$ $K \stackrel{\$}{\leftarrow} \$$ if incorrect guess

SPHF-Based PAKE: Simulation

Alice		Bob
$hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_A \leftarrow \mathsf{ProjKG}(hk_A)$	$\xrightarrow{hp_A, C_A}$	$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
$\ell_A = (A, B, hp_A), r_A \stackrel{\$}{\leftarrow} \$, C_A \leftarrow \mathsf{Enc}^{\ell_A}(pw_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$\ell_B = (B, A, hp_B), r_B \stackrel{\$}{\leftarrow} \$; C_B \leftarrow \mathsf{Enc}^{\ell'}(pw_B, r_B)$
$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$		$\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$		$\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		$K_B \leftarrow \mathsf{H}_A \times \mathsf{pH}_B$

Passwords known for corrupted players

- `NewSession`: for U with U'
  - $hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp \leftarrow \mathsf{ProjKG}(hk)$
  - $\ell = (U, U', hp)$, $\cancel{r \stackrel{\$}{\leftarrow} \$, C \leftarrow \mathsf{Enc}^{\ell}(pw, r)}$ $C \leftarrow \mathsf{Enc}^{\ell}(\$, \$)$
- Flow hp', C'
  - Oracle-Generated from U': $hk', hp' \leftarrow \mathsf{ProjKG}(hk'), C' \leftarrow \mathsf{Enc}^{\ell'}(pw', r')$
    - $\cancel{\mathsf{H} \leftarrow \mathsf{Hash}(hk, C')}$ $\mathsf{H} \leftarrow \$$
    - $\cancel{\mathsf{pH} \leftarrow \mathsf{Hash}(hk', C)}$ $\mathsf{pH} \leftarrow \mathsf{H}'$, if $pw' = pw$; $\mathsf{pH} \stackrel{\$}{\leftarrow} \$$, if $pw' \neq pw$
    - $\cancel{K \leftarrow \mathsf{H} \times \mathsf{pH}}$ Same \$ as compatible partner, independent \$ if incompatible partner
  - Non Oracle-Generated: $pw' \leftarrow \mathsf{Dec}^{\ell'}(C')$ and `TestPwd`(pw')
    - $K \stackrel{\$}{\leftarrow} \$$ if incorrect guess / possibly chosen by the adversary otherwise

SPHF-Based PAKE: Simulation

Alice		Bob
$hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_A \leftarrow \mathsf{ProjKG}(hk_A)$	$\xrightarrow{hp_A, C_A}$	$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp_B \leftarrow \mathsf{ProjKG}(hk_B)$
$\ell_A = (A, B, hp_A), r_A \stackrel{\$}{\leftarrow} \$, C_A \leftarrow \mathsf{Enc}^{\ell_A}(pw_A, r_A)$	$\xleftarrow{hp_B, C_B}$	$\ell_B = (B, A, hp_B), r_B \stackrel{\$}{\leftarrow} \$; C_B \leftarrow \mathsf{Enc}^{\ell'}(pw_B, r_B)$
$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$		$\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$
$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$		$\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$
$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$		$K_B \leftarrow \mathsf{H}_A \times \mathsf{pH}_B$

- `NewSession`: for U with U'
  - $hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}(); hp \leftarrow \mathsf{ProjKG}(hk)$
  - $\ell = (U, U', hp), C \leftarrow \mathsf{Enc}^{\ell}(\$, \$)$
- Flow hp', C'
  - Oracle-Generated from U': $hk', hp' \leftarrow \mathsf{ProjKG}(hk'), C' \leftarrow \mathsf{Enc}^{\ell'}(pw', r')$
    - ~~Same \$ as compatible partner, independent \$ if incompatible partner~~ `NewKey`
  - Non Oracle-Generated: $pw' \leftarrow \mathsf{Dec}^{\ell'}(C')$ and `TestPwd`(pw')
    - ~~$K \stackrel{\$}{\leftarrow} \$$ if incorrect guess / possibly chosen by the adversary otherwise~~ `NewKey`

Outline

SPHF-Based PAKE: Protocol

Alice

$hk_A \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_A \leftarrow \mathsf{ProjKG}(hk_A)$

$\ell_A = (A, B, hp_A),\ r_A \stackrel{\$}{\leftarrow} \$,\ C_A \leftarrow \mathsf{Com}^{\ell_A}(pw_A, r_A)$

$\xrightarrow{hp_A,\ C_A}$

Bob

$hk_B \stackrel{\$}{\leftarrow} \mathsf{HashKG}();\ hp_B \leftarrow \mathsf{ProjKG}(hk_B)$

$\ell_B = (B, A, hp_B),\ r_B \stackrel{\$}{\leftarrow} \$;\ C_B \leftarrow \mathsf{Com}^{\ell'}(pw_B, r_B)$

$\xleftarrow{hp_B,\ C_B}$

Alice

$\mathsf{H}_B \leftarrow \mathsf{Hash}_A(hk_A, C_B)$

$\mathsf{pH}_A \leftarrow \mathsf{ProjHash}_A(hp_B, C_A, r_A)$

$K_A \leftarrow \mathsf{H}_B \times \mathsf{pH}_A$

Bob

$\mathsf{H}_A \leftarrow \mathsf{Hash}_B(hk_B, C_A)$

$\mathsf{pH}_B \leftarrow \mathsf{ProjHash}_B(hp_A, C_B, r_B)$

$K_B \leftarrow \mathsf{H}_A \times \mathsf{pH}_B$

With Com $=$ Enc, if r required for ProjHash(hp, C, r):
$\Longrightarrow$ no security against adaptive corruptions
Extractable and equivocable commitment (i.e. UC-secure) and SPHF-friendly:
$\Longrightarrow$ security against adaptive corruptions

SPHF-Friendly Commitments

- Based on Cramer-Shoup (extractability) and Pedersen (equivocability)
  Inspired from the Canetti-Fischlin commitment [Canetti-Fischlin – Crypto '01]
  $\Longrightarrow$ Commitment size linear in mk^2 [Abdalla-Chevalier-P. – Crypto '09]
- Improvement with Haralambiev (equivocability) [Haralambiev – PhD Thesis '11]
  $\Longrightarrow$ Commitment size linear in mk [Abdalla-Benhamouda-Blazy-Chevalier-P. – Asiacrypt '13]
- SPHF-Friendly variant of FLM commitment [Fischlin-Libert-Manulis – Asiacrypt '11]
  $\Longrightarrow$ Commitment size linear in k [Blazy-Chevalier – Asiacrypt '16]

m = length of the password $\qquad k$ = security parameter

Outline

Conclusion

In the line of the KOY protocol [Katz-Ostrovsky-Yung – Crypto '01]
the GL methodology widely used for PAKE [Gennaro–Lindell – Eurocrypt '03]

- BPR-secure protocols
- UC-secure protocols for static corruptions [Canetti-Halevi-Katz-Lindell-MacKenzie – Eurocrypt '05]
- UC-secure protocols for adaptive corruptions [Abdalla-Chevalier-P. – Crypto 09]
- One-Round protocols (BPR and UC) [Katz-Vaikuntanathan – TCC '11]
- UC-secure protocols for adaptive corruptions without erasures [Abdalla-Benhamouda-P. – PKC '17]

Equivalently, SPHF can be used for Oblivious Transfer