**AKE**

AKE: Authenticated Key Exchange
- allows two players to agree on a common key
- authentication of partners
Diffie-Hellman

\[ X = g^x \]
\[ Y = g^y \]
\[ Y^x = SK = X^y \]

With signed flows, authentication can be provided
PAKE

Usual authentication means: **PKI**
- allows signatures
- requires certificates
- Not realistic in practice

**PAKE**: Password-Authenticated Key Exchange
- authentication means: a short **password**
- unavoidable attack: **on-line dictionary attack**
  (one test-password per active execution)
BAC: Basic Access Control

In 1998: passports with digital information, but **no security** to protect access

In 2004:

- encrypted communication with the reader
- **access control**: BAC
- authentication means: MRZ (Machine Readable Zone)
  - at most 72 bits of entropy
  - but actually approx. 40 bits
  - exhaustive search is fast: **password**
BAC: Basic Access Control

Symmetric **Enc** and **Mac** keys derived from the **pw**

Random \( r_P, k_P \)

\[
C_P \leftarrow \text{Enc}_{pw}(r_P, r_R, k_P)
\]

\[
M_P \leftarrow \text{Mac}_{pw}(C_P)
\]

\[
K \leftarrow k_P \oplus k_R
\]

Random \( r_R, k_R \)

\[
C_R \leftarrow \text{Enc}_{pw}(r_R, r_P, k_R)
\]

\[
M_R \leftarrow \text{Mac}_{pw}(C_R)
\]
BAC: Basic Access Control

Symmetric **Enc** and **Mac** keys derived from the **pw**

Random $r_P, k_P$

$C_P \leftarrow \text{Enc}_{pw}(r_P, r_R, k_P)$

$M_P \leftarrow \text{Mac}_{pw}(C_P)$

$K \leftarrow k_P \oplus k_R$

Random $r_R, k_R$

$C_R \leftarrow \text{Enc}_{pw}(r_R, r_P, k_R)$

$M_R \leftarrow \text{Mac}_{pw}(C_R)$

**Off-line dictionary attack**: Mac verification! To be avoided…

David Pointcheval
First security model: **Indistinguishability of session keys**

[Bellare-Pointcheval-Rogaway EC00]

Two players $A$ and $B$ and an adversary $\mathcal{A}$

The adversary $\mathcal{A}$ can concurrently make

- $A$ and $B$ play honestly: passive attack ($\text{Execute}$-queries)
- an execution with $A$ or $B$: active attack ($\text{Send}$-queries)
- $A$ or $B$ reveal their password: corruption of the password ($\text{Corrupt}$-query)
- $A$ or $B$ reveal their session key: missuses of the session key ($\text{Reveal}$-query)
- a test on any (fresh) session key, but once ($\text{Test}$-query)
  - the answer is either the real ($b=1$) or random ($b=0$) session key
  - the adversary outputs a guess $b'$ for $b$

$$\text{Adv}(\mathcal{A}) = \Pr[b'=1|b=1] - \Pr[b'=1|b=0]$$

should be upper-bounded by $q_{\text{send}} / \#\text{Dic} + \epsilon$.
EKE Family

EKE: Encrypted Key Exchange

\[ X' = E_{pw}(g^x) \]

\[ Y' = E_{pw}(g^y) \]

Quite efficient in theory but requires an ideal cipher onto G

- BPR-security

Patent with priority date October 2nd, 1991 (Expired)

Issue: How to build an efficient block-cipher \( E_k: G \rightarrow G \) ?

Efficient variant: SPAKE

for Simple Password-Authenticated Key Exchange
**SPAKE**

**SPAKE: Simple Password-Authenticated Key Exchange**

\[ X' = g^x U^{pw} \]

\[ Y' = g^y V^{pw} \]

\[ K = H(X', Y', g^{xy}) \]

BPR-secure in the ROM: **indistinguishability of the session key** with advantage bounded by \( q/#\text{Dic} + \epsilon \), after \( q \) active sessions

no more ideal function onto a group structure

- just a random bit-string

Quite efficient

- **1 group element** in each direction
- **4 exponentions** on each side
SPEKE

**SPEKE: Simple Password Exponential Key Exchange** [Jablon 96]

\[ H(pw)^x \]

\[ H(pw)^y \]

- quite efficient in theory
- security analysis in ROM and \( G \subset \mathbb{Z}_p^* \) [MacKenzie 01]
- but requires a hash function \( H: \{0,1\}^* \rightarrow G \)

Patent with priority date April 17th, 1996
Projective Hashing

[Cramer-Shoup C98-EC02]

\[ W \] \rightarrow \text{Hash} \rightarrow H \quad \text{unpredictable}

KG \rightarrow \text{hk}
Projective Hashing

[Cramer-Shoup C98-EC02]

Useful language:

\[ R_{pw}(C,r) = 1 \text{ iff } C = Enc(pw,r) \]

such that \( R(W,w) = 1 \)
KOY/GL Framework

\[ C_1 = E_1(pw_c, r_1) \]

\[ C_2 = E_2(pw_S, r_2), \quad hp_2 = ProjKG(hk_2, C_1) \]

\[ hp_1 = ProjKG(hk_1, C_2) \]

Language: valid ciphertexts of the password

First construction secure in the standard model
KOY/GL Framework

\[ C_1 = E_1(pw_c, r_1) \]
\[ C_2 = E_2(pw_S, r_2), \quad hp_2 = \text{ProjKG}(hk_2, C_1) \]
\[ hp_1 = \text{ProjKG}(hk_1, C_2) \]

KOY: \( E_1 = E_2 \)
- Cramer-Shoup encryption

GL: \( E_1 = E_2 \)
- non-malleable commitment
- instantiated with IND-CCA encryption

\( hk = (\alpha, \beta, \gamma, \lambda) \)
\[ C = (u = g_1^r, v = g_2^r, e = h^r, pw, w = (cd^\varepsilon)^r) \quad H = u^\alpha \cdot v^\beta \cdot (e/pw)^\gamma \cdot w^\lambda \]
\[ hp = g_1^\alpha \cdot g_2^\beta \cdot h^\gamma \cdot (cd^\varepsilon)^\lambda \quad H' = hp^r \]

\( C_1 = C_2 = 4 \) group elements
\( hp_1 = hp_2 = 1 \) group element

3 flows and 10 group elements + OT-Signature
Improvements (1)

\[ C_1 = E_1(\text{pw}_c, r_1) \]

\[ C_2 = E_2(\text{pw}_S, r_2), \quad h p_2 = \text{ProjKG}(h k_2, C_1) \]

\[ h p_1 = \text{ProjKG}(h k_1, C_2) \]

\[ \begin{align*}
E_1 \text{ or } E_2 & \text{ IND-CCA encryption} \\
E_2 \text{ or } E_1 & \text{ IND-CPA encryption}
\end{align*} \]

[Canetti-Halevi-Katz-Lindell-MacKenzie EC05]

\[ E_2 \text{ ElGamal: } C_2 = (u=g^r, e=h^r, \text{pw}) \quad H = u^\alpha (e/\text{pw})^\beta \]

\[ h k_1 = (\alpha, \beta) \quad h p_1 = g^\alpha h^\beta \quad H' = h p^r \]

\[ \text{hp}_1 \text{ independent of } C_2 \quad \text{IND-CPA} \]
Improvements (1)

- \( C_1 = E_1(pwc, r_1), \) \( hp_1 = \text{ProjKG}(hk_1) \)
- \( C_2 = E_2(pw_s, r_2), \) \( hp_2 = \text{ProjKG}(hk_2, C_1) \)

\( E_1 \) or \( E_2 \) IND-CCA encryption

\( E_2 \) or \( E_1 \) IND-CPA encryption

[Canetti-Halevi-Katz-Lindell-MacKenzie EC05]

**ElGamal:**

- \( C_2 = (u = gr, \ e = hr \ \text{pw}) \)
- \( H = u^\alpha (e/\text{pw})^\beta \)
- \( H' = hp^r \)
- \( hp_1 \) independent of \( C_2 \)
- \( \text{IND-CPA} \)

\( C_2 = 2 \) group elements

\( hp_1 = 1 \) group element

2 flows and no more OT-Signature
Improvements (2)

\[ C_1 = E_1(p_{w_c}, r_1), \quad hp_1 = \text{ProjKG}(hk_1) \]
\[ C_2 = E_2(p_{w_S}, r_2), \quad hp_2 = \text{ProjKG}(hk_2, C_1) \]

- **E_2** IND-CPA encryption
- **E_1** IND-PCA encryption

Plaintext-Checking Attack

[Okamoto-Pointcheval CTRSA01]

**E_1** Cramer-Shoup Variant: \( C = (u=g^r, e=h^r g^{pw}, w=(cd^\varepsilon)^r) \)

\[ hk = (\alpha, \beta, \gamma) \]
\[ hp = g^\alpha h^\beta (cd^\varepsilon)^\gamma \]
\[ H = u^\alpha (e/g^{pw})^\beta w^\gamma \]
\[ H' = hp^r \]

IND-PCA

\[ C_1 = 3 \text{ group elements} \]
\[ hp_2 = 1 \text{ group element} \]

2 flows and 7 group elements
**SPOKE**

**[Abdalla-BenHamouda-Pointcheval PKC15]**

**SPOKE-GL**

Random $r$, $\alpha'$, $\beta'$

\[
\begin{align*}
(u & = g^r, \; e = h^r g^{\text{pw}_a}, \; w = (cd^e)^r), \; hp' = g^{\alpha'} h^{\beta'} \\
(u' & = g^{r'}, \; e' = h^{r'} g^{\text{pw}_b}), \; hp = g^{\alpha} h^{\beta} (cd^e)^\gamma
\end{align*}
\]

Random $r'$, $\alpha$, $\beta$

\[
hp^r \times u'^\alpha (e'/g^{\text{pw}_a})^{\beta'} = \text{SK} = u^{\alpha} (e/g^{\text{pw}_b})^{\beta} w^\gamma \times hp'^{r'}
\]

**Properties**

- Secure in the BPR setting under the sole DDH assumption

- **Very efficient GL-PAKE:**
  - 2 flows and 7 group elements
**GK Paradigm**

- One ciphertext of the password
- A hash proof on it to derive random coins to re-encrypt the password
- This ciphertext can be checked

\[
\begin{align*}
C_1 &= E_1(pw_c, r_1) \\
C_2 &= E_2(pws, H), \ h p_2 = ProjKG(hk_2, C_1)
\end{align*}
\]

Random \( r' \)  \( (u' = g^{r'}, e' = h^{r'} g^{pw_a}) \)

\( (K, r) \leftarrow \text{Hash} \)

\( (u = g^r, e = h^r g^{pw_b}, w = (cd^e)r), \ hp' = g^{\alpha'} h^\beta \)

Random \( \alpha', \beta' \)

\( (K, r) \leftarrow \text{ProjHash} \)

Check \( (u, e, w) \)

\[ hp' r' = (K, r) = u'^{\alpha'} (e' / g^{pw_b})^\beta' \]

**Properties**

- Secure in the BPR setting under the sole DDH assumption
- The most efficient PAKE: 2 flows and 6 group elements
Conclusion

- Password-Authenticated Key Exchange is now efficient
  - In **Random Oracle Model**: EKE-like
    - 1 group element to send in each direction
    - 4 exponentiations for each player
  - In the **Standard Model**: GK-like
    - 2 and 4 group elements to be sent respectively
    - 8 and 10 exponentiations to be computed respectively
- Other variants:
  - one-round PAKE (2 simultaneous flows)
  - security in the Universal Composability framework