Password-Authenticated Key Exchange

David Pointcheval

Privacy and Contactless Services
May 27th, 2015

AKE

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

Diffie-Hellman

$$X = g^x$$

→

$$Y = g^y$$

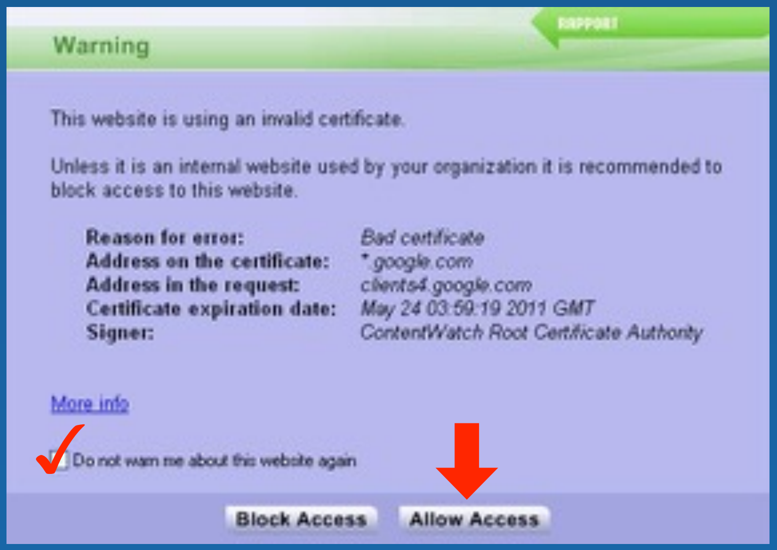
←

$$Y^x = \mathbf{SK} = X^y$$

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**

P		R
Random r_P, k_P	r_P →	Random r_R, k_R
		$C_R \leftarrow \mathsf{Enc}_{pw}(r_R, r_P, k_R)$
		$M_R \leftarrow \mathsf{Mac}_{pw}(C_R)$
	← C_R, M_R	
$C_P \leftarrow \mathsf{Enc}_{pw}(r_P, r_R, k_P)$		
$M_P \leftarrow \mathsf{Mac}_{pw}(C_P)$		
	C_P, M_P →	

$$K \leftarrow k_P \oplus k_R$$

BAC: Basic Access Control

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

Random r_P, k_P

$\xrightarrow{\quad r_P \quad}$

Random r_R, k_R

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

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

$\xleftarrow{\quad C_R,\, M_R \quad}$

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

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

$\xrightarrow{\quad C_P,\, M_P \quad}$

$K \leftarrow k_P \oplus k_R$

Off-line dictionary attack: Mac verification!

To be avoided…

PAKE

- First security model: **Indistinguishability of session keys**

[Bellare-Pointcheval-Rogaway EC00]

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

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

should be upper-bounded by $q_{\mathbf{send}}$ / #Dic + ε

EKE Family

- **EKE: Encrypted Key Exchange** [Bellovin-Merritt S&P92]

$$X' = \mathbf{E}_{\mathrm{pw}}(g^x) \longrightarrow$$

$$Y' = \mathbf{E}_{\mathrm{pw}}(g^y) \longleftarrow$$

- Quite efficient in theory but requires in ***ideal cipher onto* G**
  - BPR-security [Bellare-Pointcheval-Rogaway EC00] [Bresson-Chevassut-Pointcheval CCS03]
  - Patent with priority date October 2nd, 1991 (***Expired***)
  - **Issue**: How to build an efficient bock-cipher $\mathbf{E}_k$: **G** → **G** ?
- Efficient variant: **SPAKE** [Abdalla-Pointcheval CTRSA05]
  - for *Simple Password-Authenticated Key Exchange*

SPAKE

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

$$X' = g^x U^{\text{pw}}$$

$$\longrightarrow$$

$$Y' = g^y V^{\text{pw}}$$

$$\longleftarrow$$

$$K = H(X', Y', g^{xy})$$

- BPR-secure in the ROM: **indistinguishability of the session key** with advantage bounded by q/#Dic + ε, 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]

$$H(pw)^x \longrightarrow$$

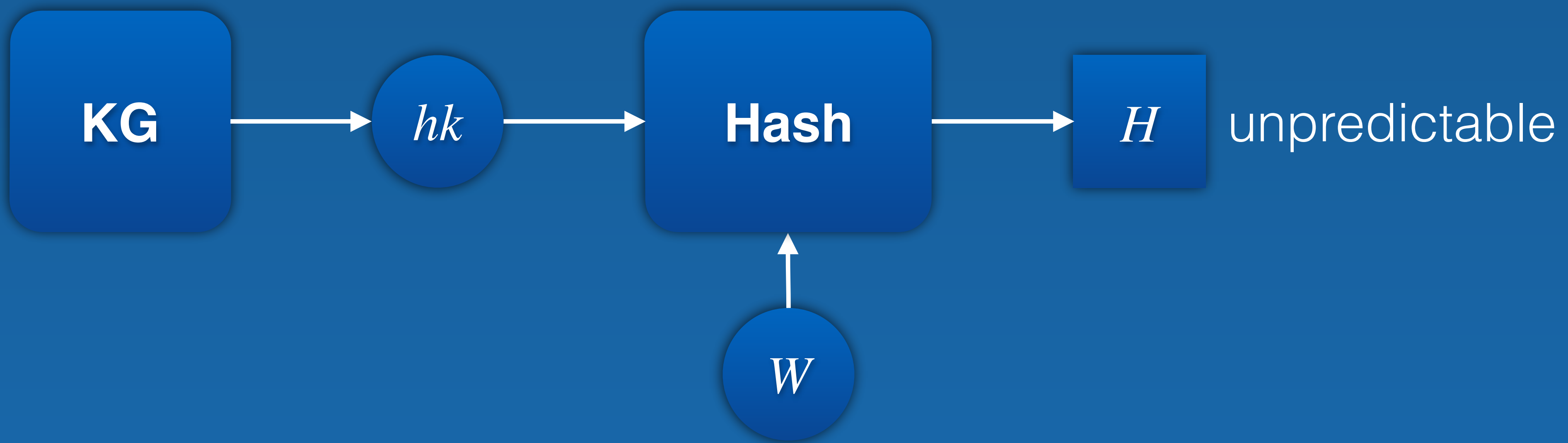
$$\longleftarrow H(pw)^y$$

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

  - Patent with priority date April 17th, 1996

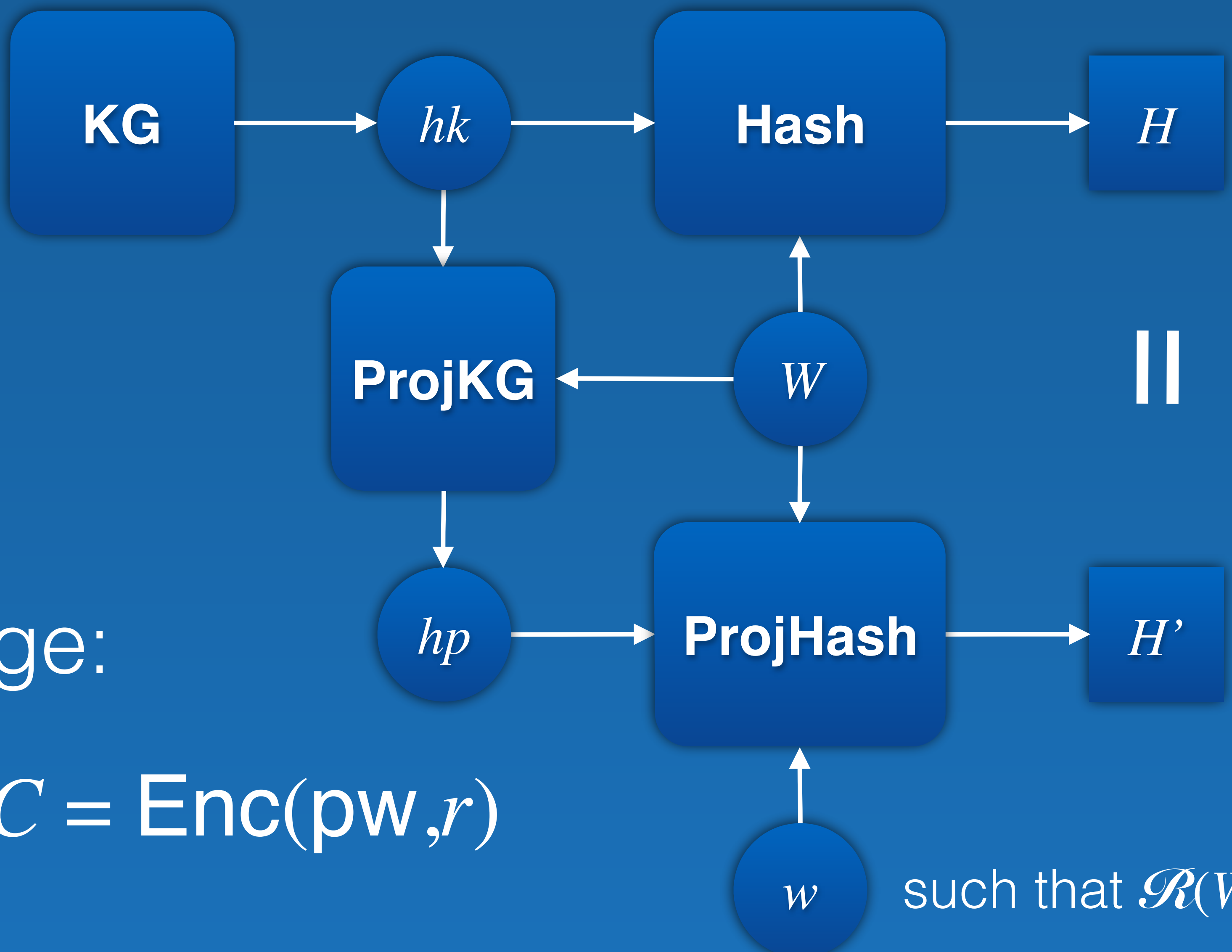
Projective Hashing

[Cramer-Shoup C98-EC02]



Projective Hashing

[Cramer-Shoup C98-EC02]



such that $\mathcal{R}(W,w)=1$

Useful language:

$\mathcal{R}_{\text{pw}}(C,r)=1$ iff $C = \text{Enc}(\text{pw},r)$

KOY/GL Framework

[Katz-Ostrovsky-Yung C01/Gennaro-Lindell EC03] (Simplified)

$C_1 = \mathbf{E}_1(pw_C, r_1)$ →

$C_2 = \mathbf{E}_2(pw_S, r_2)$, $hp_2 = \mathbf{ProjKG}(hk_2, C_1)$ ←

$hp_1 = \mathbf{ProjKG}(hk_1, C_2)$ →

Langage: valid ciphertexts of the password

$$\mathbf{Hash}(hk_1, C_2) \times \mathbf{ProjHash}(hp_2, C_1, r_1) = \mathbf{ProjHash}(hp_1, C_2, r_2) \times \mathbf{Hash}(hk_2, C_1)$$

First construction secure in the standard model

KOY/GL Framework

$C_1 = \mathbf{E}_1(pw_C, r_1)$ →

← $C_2 = \mathbf{E}_2(pw_S, r_2)$, $hp_2 = \mathbf{ProjKG}(hk_2, C_1)$

$hp_1 = \mathbf{ProjKG}(hk_1, C_2)$ →

- **KOY: $\mathbf{E}_1 = \mathbf{E}_2$**
  - Cramer-Shoup encryption
- **GL: $\mathbf{E}_1 = \mathbf{E}_2$**
  - non-malleable commitment
  - instantiated with IND-CCA encryption

$hk = (\alpha, \beta, \gamma, \lambda)$

$C = (u = g_1^r, v = g_2^r, e = h^r\, pw, w = (cd^\varepsilon)^r)$ $H = u^\alpha v^\beta (e/pw)^\gamma w^\lambda$

$hp = g_1^\alpha g_2^\beta h^\gamma (cd^\varepsilon)^\lambda$ $H' = hp^r$

$C_1 = C_2 = 4$ group elements

$hp_1 = hp_2 = 1$ group element

3 flows and 10 group elements + OT-Signature

Improvements (1)

$C_1 = \mathbf{E}_1(pw_C, r_1)$ →

← $C_2 = \mathbf{E}_2(pw_S, r_2)$, $hp_2 = \mathbf{ProjKG}(hk_2, C_1)$

$hp_1 = \mathbf{ProjKG}(hk_1, C_2)$ →

- $\mathbf{E}_1$ or $\mathbf{E}_2$ IND-CCA encryption
- $\mathbf{E}_2$ or $\mathbf{E}_1$ IND-CPA encryption

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

$\mathbf{E}_2$ ElGamal: $C_2 = (u=g^r, e=h^r\,\mathrm{pw})$

$H = u^\alpha (e/\mathrm{pw})^\beta$

$hk_1 = (\alpha,\beta)$ $\quad hp_1 = g^\alpha h^\beta$

$H' = hp^r$

hp_1 independent of C_2

IND-CPA

Improvements (1)

$C_1 = \mathbf{E}_1(pw_C, r_1),\ hp_1 = \mathbf{ProjKG}(hk_1)$ →

← $C_2 = \mathbf{E}_2(pw_S, r_2),\ hp_2 = \mathbf{ProjKG}(hk_2, C_1)$

- $\mathbf{E}_1$ or $\mathbf{E}_2$ IND-CCA encryption
- $\mathbf{E}_2$ or $\mathbf{E}_1$ IND-CPA encryption

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

$\mathbf{E}_2$ ElGamal: $C_2 = (u=g^r, e=h^r\,\mathrm{pw})$ $H = u^\alpha (e/\mathrm{pw})^\beta$

$hk_1 = (\alpha,\beta)$ $hp_1 = g^\alpha h^\beta$ $H' = hp^r$

hp_1 independent of C_2 IND-CPA

C_2 = 2 group elements

hp_1 = 1 group element

2 flows and no more OT-Signature

Improvements (2)

$C_1 = \mathbf{E}_1(pw_C, r_1)$, $hp_1 = \mathbf{ProjKG}(hk_1)$ →

← $C_2 = \mathbf{E}_2(pw_S, r_2)$, $hp_2 = \mathbf{ProjKG}(hk_2, C_1)$

- $\mathbf{E}_2$ IND-CPA encryption
- $\mathbf{E}_1$ IND-PCA encryption
  - Plaintext-Checking Attack

[Okamoto-Pointcheval CTRSA01]

$\mathbf{E}_1$ Cramer-Shoup Variant: $C = (u=g^r, e=h^r g^{pw}, w=(cd^\varepsilon)^r)$

$hk = (\alpha, \beta, \gamma)$ $\qquad hp = g^\alpha h^\beta (cd^\varepsilon)^\gamma$

$H = u^\alpha (e/g^{pw})^\beta w^\gamma$ $\qquad H' = hp^r$ $\qquad$ IND-PCA

C_1 = 3 group elements

hp_2 = 1 group element

2 flows and 7 group elements

SPOKE

[Abdalla-BenHamouda-Pointcheval PKC15]

- **SPOKE-GL**

Random r, α', β' $(u = g^r,\ e = h^r g^{pw_a},\ w = (cd^\varepsilon)^r),\ hp' = g^{\alpha'} h^{\beta'}$ →

← $(u' = g^{r'},\ e' = h^{r'} g^{pw_b}),\ hp = g^\alpha h^\beta (cd^\varepsilon)^\gamma$ Random r', α, β

$$hp^r \times u'^{\alpha'} (e'/g^{pw_a})^{\beta'} = SK = u^\alpha (e/g^{pw_b})^\beta w^\gamma \times hp'^{r'}$$

- **Properties**
  - Secure in the BPR setting under the sole DDH assumption
  - **Very efficient GL-PAKE:**
    2 flows and 7 group elements

SPOKE

- **GK Paradigm** [Groce-Katz CCS10]
  - One ciphertext of the password
  - A hash proof on it to derive random coins to re-encrypt the password
  - this ciphertext can be checked

$C_1 = \mathbf{E}_1(pw_C, r_1)$ →

← $C_2 = \mathbf{E}_2(pw_S, H)$, $hp_2 = \mathbf{ProjKG}(hk_2, C_1)$

Random r' — $(u'=g^{r'}, e'=h^{r'} g^{pw_a})$ → Random α', β'

$(K,r) \leftarrow$ Hash ← $(u=g^r, e=h^r g^{pw_b}, w=(cd^{\varepsilon})^r), hp' = g^{\alpha'} h^{\beta}$ — $(K,r) \leftarrow$ ProjHash

Check (u,e,w) $\quad hp'^{r'} = (K,r) = u'^{\alpha'} (e'/g^{pw_b})^{\beta'}$

- **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