

Intercambio de claves de Diffie–Hellman

Clase 14

<https://apuntes.indcpa.com>

Versión 0.1, mayo 2025

- Hasta 1970, toda la criptografía utilizada era de clave simétrica.
- Esto no era una gran limitación dado que el uso principal era militar, donde “canales seguros” de intercambio de clave físicos existen.
- Esta es una gran limitación para uso civil.
- Vamos ahora a introducir brevemente el intercambio de claves (KEX, “key exchange”) de Diffie y Hellman.
- Lo hacemos por motivos informacionales e históricos, al ser el primer protocolo de PKC considerado seguro.
- No vamos a intentar demostrar su seguridad en detalle, porque los modelos de seguridad de KEX pueden ser muy complicados.

$\text{Exp}_{\text{PAKE}, \mu, \ell, \mathcal{A}}^{\text{AKE}}:$ $\text{pp}_{\text{AKE}} \leftarrow \text{AKE.Setup}$ For $i \in [\mu]$: $(pk_i, sk_i) \leftarrow \text{AKE.Gen}(\text{pp}_{\text{AKE}}, P_i);$ $crp_i := \text{false}$ //Corruption variable PKList := $\{pk_i\}_{i \in [\mu]}$; $b \leftarrow \{0, 1\}$ For $(i, s) \in [\mu] \times [\ell]$: $\text{var}_i^s := (\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s) := (0, 0, 0, 0);$ $\text{Aflag}_i^s := \text{false}$ //Whether Pid_i^s is corrupted when π_i^s accepts $T_i^s := \text{false}; k\text{Rev}_i^s := \text{false}$ // Test, Key Reveal variables $\text{stRev}_i^s := \text{false}; \text{FirstAcc}_i^s := 0$ // State Reveal & First Acceptance variables $b^* \leftarrow \mathcal{A}^{\text{PAKE}^{(1)}}(\text{pp}_{\text{AKE}}, \text{PKList})$ Winsub := false Winsub := true, If $\exists (i, s) \in [\mu] \times [\ell]$ s.t. (1) $\Psi_i^s = \text{accept}$ // π_i^s is τ-accepted (2) $\text{Aflag}_i^s = \text{false}$ // P_i is $\hat{\tau}$-corrupted with $j := \text{Pid}_i^s$ and $\hat{\tau} > \tau$ (3) (3.1) $\vee$ (3.2) $\vee$ (3.3). Let $j := \text{Pid}_i^s$ (3.1) $\exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$ (3.2) $\exists t \in [\ell], (j', t') \in [\mu] \times [\ell]$ with $(j, t) \neq (j', t')$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge \text{Partner}(\pi_j^t \leftarrow \pi_{j'}^{t'})$ (3.3) $\exists t \in [\ell], (i', s') \in [\mu] \times [\ell]$ with $(i, s) \neq (i', s')$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge \text{Partner}(\pi_{i'}^{s'} \leftarrow \pi_j^t)$ //Replay attacks Winsub := false If $b^* = b$: Winsub := true; Return 1 Else: Return 0 $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$: //Checking whether $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$ If π_i^s sent the first message and $k_i^s = K(\pi_i^s, \pi_j^t) \neq \emptyset$: Return 1 If π_i^s received the first message and $k_i^s = K(\pi_j^t, \pi_i^s) \neq \emptyset$: Return 1 Return 0 $\pi_i^s(\text{msg}, j)$: // π_i^s executes AKE according to the protocol specification If $\text{Pid}_i^s = \emptyset$: $\text{Pid}_i^s := j$ If $\text{Pid}_i^s = j$: π_i^s receives msg and uses $\text{res}, \text{var}_i^s$ to generate the next message msg' of AKE, and updates $(\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)$: If $\text{msg} = \top$: π_i^s generates the first message msg' as initiator; If msg is the last message of AKE: $\text{msg}' := \emptyset$; Return msg' If $\text{Pid}_i^s \neq j$: Return $\perp$ $\mathcal{C}_{\text{AKE}}(\text{query})$: If $\text{query} = \text{RegisterCorrupt}(u, pk_u)$: If $u \in [\mu]$: Return $\perp$ PKList := $\text{PKList} \cup \{pk_u\}$ $crp_u := \text{true}$ Return PKList	$\mathcal{C}_{\text{AKE}}(\text{query})$: If $\text{query} = \text{Send}(i, s, j, \text{msg})$: If $\Psi_i^s = \text{accept}$: Return $\perp$ $\text{msg}' \leftarrow \pi_i^s(\text{msg}, j)$ If $\Psi_i^s = \text{accept}$: If $crp_j = \text{true}$: $\text{Aflag}_i^s := \text{true}$; // Determine whether π_i^s accepts before its partner If $crp_j = \text{false} \wedge \exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$: If $\Psi_j^t \neq \text{accept}$: $\text{FirstAcc}_i^s := \text{true}; \text{FirstAcc}_j^t := \text{false}$ If $\Psi_j^t = \text{accept}$: $\text{FirstAcc}_i^s := \text{false}; \text{FirstAcc}_j^t := \text{true}$ Return msg' If $\text{query} = \text{Corrupt}(i)$: If $i \notin [\mu]$: Return $\perp$ For $s \in [\ell]$: If $\text{FirstAcc}_i^s = \text{false} \wedge \text{stRev}_i^s = \text{true}$: If $T_i^s = \text{true}$: Return $\perp$; //avoid TA6 If $\exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$: If $T_j^t = \text{true}$: Return $\perp$; //avoid TA7 $crp_i := \text{true}$ Return sk_i If $\text{query} = \text{SessionKeyReveal}(i, s)$: If $\Psi_i^s \neq \text{accept}$: Return $\perp$ If $T_i^s = \text{true}$: Return $\perp$ //avoid TA2 Let $j := \text{Pid}_i^s$ If $\exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$: If $T_j^t = \text{true}$: Return $\perp$ //avoid TA4 $k\text{Rev}_i^s := \text{true}$; Return k_i^s If $\text{query} = \text{StateReveal}(i, s)$: If $\text{FirstAcc}_i^s = \text{false} \wedge crp_i = \text{true}$: If $T_i^s = \text{true}$: Return $\perp$; //avoid TA6 Let $j := \text{Pid}_i^s$ If $\exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$: If $T_j^t = \text{true}$: Return $\perp$; //avoid TA7 $\text{stRev}_i^s := \text{true}$; Return st_i^s If $\text{query} = \text{Test}(i, s)$: If $\Psi_i^s \neq \text{accept} \vee \text{Aflag}_i^s = \text{true} \vee k\text{Rev}_i^s = \text{true}$ $\vee T_i^s = \text{true}$: Return $\perp$ //avoid TA1, TA2, TA3 If $\text{FirstAcc}_i^s = \text{false}$: If $crp_i = \text{true} \wedge \text{stRev}_i^s = \text{true}$: Return $\perp$ //avoid TA6 Let $j := \text{Pid}_i^s$ If $\exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$: If $k\text{Rev}_j^t = \text{true} \vee T_j^t = \text{true}$: Return $\perp$ //avoid TA4, TA5 If $\exists t \in [\ell]$ s.t. $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$: If $\text{FirstAcc}_j^t = \text{false} \wedge crp_j = \text{true}$ $\wedge \text{stRev}_j^t = \text{true}$: Return $\perp$ //avoid TA7 $T_i^s := \text{true}; k_0 := k_i^s; k_1 \leftrightarrow k_2$; Return k_0
--	---

Fig. 5. The security experiments $\text{Exp}_{\text{PAKE}, \mu, \ell, \mathcal{A}}^{\text{AKE}}$ (both without red text) and $\text{Exp}_{\text{PAKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}}$ (with red text). The list of trivial attacks is given in Table 2.

Modelo de seguridad de intercambio de clave autenticado usado en Han *et al.*, CRYPTO 2021.

Preliminares matematicos II

Definition (grupo abeliano)

Sea G un conjunto, $\times$ una operación binaria $\times: G \times G \rightarrow G$. Decimos que $\mathbb{G} = (G, \times)$ es un grupo abeliano si

1. $\times$ es asociativa: $(a \times b) \times c = a \times (b \times c), \quad \forall a, b, c \in G$.
2. Existe un elemento neutro $e \in G$ tal que $a \times e = e \times a = a, \quad \forall a \in G$.
3. Cada elemento a tiene un inverso a^{-1} , tal que $a \times a^{-1} = a^{-1} \times a = e, \quad \forall a \in G$.
4. $\times$ es conmutativa: $a \times b = b \times a, \forall a, b \in G$.

Si $|G| < \infty$, decimos que $\mathbb{G}$ es finito, y que $|G|$ es el orden de $\mathbb{G}$. A menudo abusamos notación y escribimos $a \in \mathbb{G}$ en lugar de $a \in G$.

Example

- $(\mathbb{Z}, +)$ es un grupo abeliano
- $(\mathbb{Z}, \cdot)$ **no es** un grupo abeliano. $(\mathbb{Q}, \cdot)$ tampoco. $(\mathbb{Q} \setminus \{0\}, \cdot)$ lo es. $(\mathbb{R} \setminus \{0\}, \cdot)$ también.
- Sea $\mathbb{Z}_n$ el conjunto de los enteros “modulo n ”. $(\mathbb{Z}_n, +)$ es un grupo abeliano.

Comentario

En un grupo podemos “exponenciar”: si $a \in \mathbb{G}$, $a^n := a \times a \times \cdots \times a$ (n veces). Las comunes reglas valen:

- $a^n \times a^m = a^{n+m},$
- $(a^n)^m = a^{nm} = a^{mn} = (a^m)^n,$
- $a^1 \cdot a^{-1} = a^0 = e.$

*En el caso de un grupo donde usamos “+”, como $(\mathbb{Z}, +)$, el exponente es un factor:
 $a + \cdots + a$ (n veces) $= n \times a.$*

Definition (grupo cíclico)

Un grupo $\mathbb{G} = (G, \times)$ se dice cíclico si existe $g \in G$ tal que el grupo $\langle g \rangle$ “generado” g ,

$$\langle g \rangle := (\{g^n \mid n \in \mathbb{Z}\}, \times),$$

es $\mathbb{G}$, o sea $\mathbb{G} = \langle g \rangle$.

Example

$(\mathbb{Z}, +)$ es cíclico, y generado por ± 1 .

Lemma

Dado un grupo $\mathbb{G} = (G, \times)$ de orden primo $|G| = p$, $\mathbb{G}$ es cíclico y generado por cualquier elemento no neutro.

Example

$(\mathbb{Z}_p, +)$ es cíclico. Cualquier $1 \leq g < p$ lo genera. Por ejemplo, sea $p = 3$.

1 es un generador		2 es un generador	
1	mod 3 = 1	2	mod 3 = 2
1 + 1	mod 3 = 2	2 + 2	mod 3 = 4 mod 3 = 1
1 + 1 + 1	mod 3 = 3	2 + 2 + 2	mod 3 = 6 mod 3 = 0
1 + 1 + 1 + 1	mod 3 = 4 mod 3 = 1	2 + 2 + 2 + 2	mod 3 = 8 mod 3 = 2

Vamos a definir tres problemas matemáticos relacionados, y un postulado.

Definition (DLOG, informal)

Sea $\mathbb{G}$ un grupo cíclico. Dados $g, h \in \mathbb{G}$, el problema del logaritmo discreto (DLOG, “discrete logarithm”) requiere recuperar $x \in \mathbb{Z}$ tal que $g^x = h$, o sea “ $x = \log_g h$ ”.

Comentario

DLOG es fácil de resolver en $(\mathbb{Z}, +)$ y $(\mathbb{R} \setminus \{0\}, \times)$:

- $(\mathbb{Z}, +)$: recordando que “ g^x ” es $x \cdot g$, $x \cdot g = h \iff x = h/g$.
- $(\mathbb{R} \setminus \{0\}, \times)$: $g^x = h \iff x = \log_g h$, donde $\log$ es el logaritmo “comun”.

Definition (CDH, informal)

Sea $\mathbb{G}$ un grupo cíclico finito con generador g . Sean $x \xleftarrow{\$} [|\mathbb{G}|]$, $y \xleftarrow{\$} [|\mathbb{G}|]$ muestreados independientemente. Dados (g, g^x, g^y) , calcular $g^{xy} = (g^x)^y = (g^y)^x$.

Comentario

CDH no puede ser mas difícil de DLOG (o sea, resolver DLOG $\Rightarrow$ resolver CDH). Supongamos que $\mathcal{O}$ es un oráculo que calcula $(g, g^x) \mapsto x$. Dados (g, g^x, g^y) , calculamos $(g^y)^{\mathcal{O}(g, g^x)} = (g^y)^x = g^{xy}$.

Lemma

Sea $\mathbb{G} = \langle g \rangle$ un grupo finito. Dada la variable aleatoria $z \sim U([|\mathbb{G}|])$, g^z es una variable aleatoria $g^z \sim U(\mathbb{G})$.

Definition (DDH)

Sea $\mathbb{G} = \langle g \rangle$ un grupo cíclico finito. Para simplificar notación, supongamos que g es un “parámetro publico”, o sea un elemento fijo y disponible públicamente. Definimos el juego Exp^{DDH} .

$\text{Exp}^{\text{DDH}}(\mathcal{A}, b)$
1: $x, y \xleftarrow{\$} [G]$
2: if $b = 0$: $z \leftarrow x \cdot y$
3: else : $z \xleftarrow{\$} [G]$
4: $b' \leftarrow \mathcal{A}(g^x, g^y, g^z)$
5: return b'

Decimos que $\mathbb{G}$ oroga seguridad (ε, t) -DDH si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{DDH}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{DDH}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{DDH}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Finalmente introducimos un nuevo postulado:

Postulado

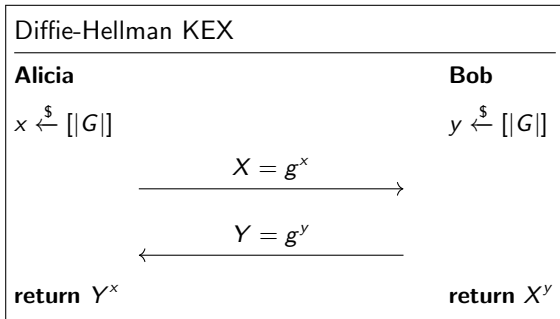
Dado $\lambda > 0$, existe un grupo cíclico finito $\mathbb{G}$ que otorga λ -bits de seguridad DDH.

Comentario

Por motivos criptoanalíticos, es generalmente deseable que $\mathbb{G}$ tenga orden primo.

Intercambio de claves de Diffie-Hellman

Supongamos que dos partes Alicia y Bob se ponen de acuerdo públicamente sobre un grupo cíclico finito $\mathbb{G}$ y sobre un generador g , tal que $\mathbb{G} = \langle g \rangle$.

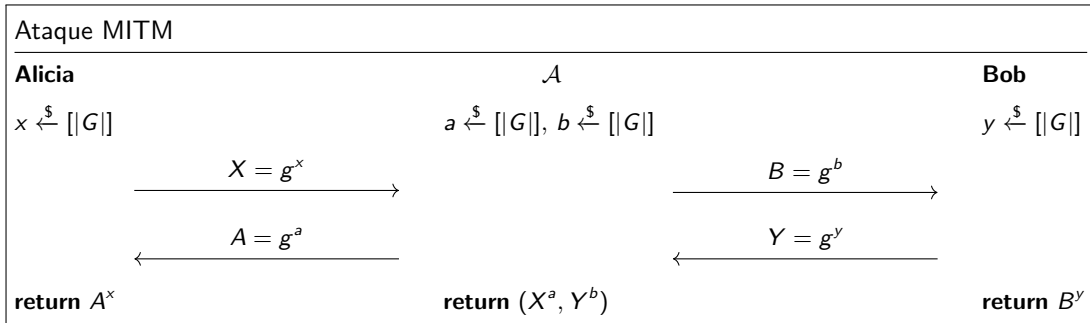


- Notamos que $X^y = (g^x)^y = g^{xy} = g^{yx} = (g^y)^x = Y^x$.
 - ▶ Alicia y Bob retornan el mismo elemento $g^{xy} \in \mathbb{G}$.
- Podemos definir un PRG $H: \mathbb{G} \rightarrow \{0, 1\}^\ell$, y utilizar $H(g^{xy})$ como clave de SKE.

Definir la seguridad del KEX puede ser complicado. Supongamos un adversario que se limita a observar pasivamente la comunicacion (el “transcript”) (g^x, g^y) .

- Si CDH es difícil en $\mathbb{G}$, sabemos que calcular g^{xy} es difícil.
- Es esto suficiente a garantizar que DH es seguro frente a un adversario “pasivo” (que solo observa trafico de red)? CDH no excluye que “mitad de los bits” de g^{xy} sean aprendibles.
- De aqui la idea de usar DDH: si (g^x, g^y, g^{xy}) es indistinguible de (g^x, g^y, g^z) con z aleatorio, $\mathcal{A}$ aprende “nada” sobre xy observando (g^x, g^y) .

- Notamos también que $\mathcal{A}$ podría no limitarse simplemente a observar tráfico, y también interceptarlo si tiene control sobre la red.
- Este tipo de ataque se llama machine-in-the-middle (MITM).



- En este caso $\mathcal{A}$ podría fingir de ser Bob a Alice, y Alicia a Bob.

- Este tipo de tactica no funciona si Alice y Bob intercambian claves secretas de persona, y utilizan exclusivamente criptografia simetrica.
- En el setting de criptografia de clave publica, defendernos de este tipo de tactica requiere nuevos instrumentos, para construir una “infraestructura de claves publicas” (PKI, public-key infrastructure).
 - ▶ En lugar de iniciar intercambio de claves en el momento, usaremos esquemas de cifrado de clave publica (PKE, “public-key encryption”), donde Bob primero publica una clave pk, y Alicia cifra usando pk.
 - ▶ La clave de Bob va a ser “certificada” como autentica usando un sistema de firma digital.
 - ▶ Con estas tecnicas, podremos verificar la identidad de un poseedor de clave publica, evitando que $\mathcal{A}$ nos engañe en cifrar hacia una clave que no le pertenece a Bob.

Empezamos entonces definiendo y construyendo PKE.