Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 13 → 16

Portland State University
Lecturer: Fang Song

Mar. 1, 2018

Draft note. Version: March 1, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) number theory, TDPs, RSA
- Group theory, Diffie-Hellman, ElGamal
- Hybrid Encryption

We talked about the factoring and RSA problem last time, which provides a candidate for the "magic" box that Diffie and Hellman envisioned: a trapdoor one-way permutation. Although the direct use of it in public-key encryption (PubKE), e.g., plain-RSA, is not CPA-secure, we show how to constrcut a CPA-secure PubKE from a TDP and a hard-core predicate of it.

In this class, we introduce another important contribution in Diffie-Hellman's seminal paper: a key-exchange protocol, and equally important the problem used in their construction (related to finding discrete logarithm). Later people realized that the DH key-exchange protocol can be adapted to public-key encryption schemes fairly easily, and the underlying problem is becoming another basic assumption for building PKC. To get a hold on it, we need a bit of group theory.

Group theory 101 in 15 mins

Definition 1. A group is a set G endowed with an (abstract) operation, denote by $\circ : G \times G \to G$, satisfying

- (Associativity) For every $a, b, c \in G$, $a \circ (b \circ c) = (a \circ b) \circ c$.
- (Identity) There exists a $u \in G$, such that for every $a \in G$, $a \circ u = a$.
- (Inverse) For every $a \in G$, there exists $a' \in G$ such that $a \circ a' = u$.

If in addition for every $a, b \in G$, $a \circ b = b \circ a$, we call G an abelian group. We usually write $g^k = g \circ \ldots \circ g$ (k times) for an integer $k \geq 1$.

When $N = p$ is prime, $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ is of special interest to us. $\mathbb{Z}_p$ can be generated by one element[1].

Two examples we have seen so far:

- $\mathbb{Z}_N, +$ mod N: $u = 0$.
- $\mathbb{Z}_N^*, \cdot$ mod N: $u = 1$. This group with $N = pq$, p and q prime, a central object in RSA.

[1] EX. consider $\mathbb{Z}_7^*$. Compute 3^i mod 7 and 2^i mod 7 for $i = 0, 1, 2, \ldots, 6$. Can you find other generators?

Definition 2. Let G be a group with size n. Suppose there exists an $g \in G$ such that $g, g^2, \ldots, g^n$ are all distinct (hence cover all of G). Then G is called a *cyclic* group, and g is called a generator. We denote $\langle g \rangle$ the cyclic group generated by g.

$(\mathbb{Z}_N, + \bmod N)$ is also a cyclic group, and 1 is a generator.

Theorem 3. *$\mathbb{Z}_p^*$ is a cyclic group with $\cdot \bmod p$ for any prime p.*

Later on, we will just use a generic cyclic group $G = \langle g \rangle$ with a generator g.

PKC based on discrete logarithm

Let $G = \langle g \rangle$ with size[2] $|G| = q$. Recall $\mathbb{Z}_q := \{0, \ldots, q-1\}$.

[2] It is preferrable to let q be prime, since it is believed the discrete logarithm problem is hardest in prime-order groups.

Discrete logarithm and DH key-exchange

Consider the function

$$F^{\mathsf{GEXP}} : \begin{array}{l} \mathbb{Z}_q \to G \\ x \mapsto g^x . \end{array}$$

Given $y = g^x$, x is called the discrete logarithm of y wrt g, and we denote it $x := \log_g y$. We state explicitly some crucial observations:

- g^x is uniformly random in G if we pick uniform $x \leftarrow \mathbb{Z}_q$.
- F^{GEXP} is bijective (in fact an isomorphism: we can identify $\mathbb{Z}_q$ and G by the mapping $x \leftrightarrow g^x$).

The discrete logarithm problem is to find $x := \log_q y$ given $(G, g, y \in G)$. It is believed to be a hard problem despite many years of efforts. The best algorithm (on a classical computer) runs in time $\sim 2^{n^{1/3} \log n^{2/3}}$. We formulate it as the DL assumption. Let GS be an algorithm that on input 1^n, generates a tuple (G, q, g), where $G = \langle g \rangle$ with size $|G| = q$.

The Discrete-Logarithm assumption

$F^{\mathsf{GEXP}}(x) := g^x$ is a *one-way* function (a permutation if we identify $\mathbb{Z}_q$ with G), for some GS.

Diffie-Hellman Key Exchange. Diffie-Hellman proposed a protocol for sharing a secret key via public communitcation.

Intuitively, since DL is hard to compute, any eavesdropper who intercepts h_A and h_B cannot compute x or y, and hence cannot compute $g^{xy} = h_A^y = h_B^x$. But there is a catch. The eavesdropper doesnot need to compute x or y, all it wants is to compute g^{xy} from h_A and

1. Alice generates $(G, q, g) \leftarrow \mathsf{GS}(1^n)$.
2. Alice picks uniform $x \leftarrow \mathbb{Z}_q$ and computes $h_A := g^x$.
3. Alice sends (G, q, g, h_A) to Bob.
4. Bob picks uniform $y \leftarrow \mathbb{Z}_q$ and computes $h_B := g^y$. Send h_B to Alice.
5. Alice computes $k_A := h_B^x$ and Bob computes $k_B := h_A^y$.

Figure 1: The Diffie-Hellman key-exchange protocol

h_B, and this is not immediately rule out by the DL assumption. In fact, similar to the situation of factoring and RSA assumptions, DL assumption is not immediately useful, and we need variants of it.

Diffie-Hellman assumptions

1. Challenger CH generates $(G, q, g) \leftarrow \mathsf{GS}(1^n)$.
2. CH chooses uniform $x_1, x_2 \leftarrow \mathbb{Z}_q$, and computes $h_1 := g^{x_1}$ and $h_2 := g^{x_2}$.
3. $\mathcal{A}$ is given (G, q, g, h_1, h_2), and $\mathcal{A}$ outputs $h \in G$.
4. The output of the game is defined to be 1 if $h = g^{x_1 x_2}$ and 0 otherwise.

Figure 2: The computational Diffie-Hellman game $\mathsf{CDH}_{\mathcal{A},\mathsf{GS}}(n)$

Computational Diffie-Hellman assumption (CDH).

Definition 4. We say that the CDH problem is hard relative to GS if for all PPT $\mathcal{A}$, $\Pr[\mathsf{CDH}_{\mathcal{A},\mathsf{GS}}(n) = 1] \leq \mathrm{negl}(n)$.

The CDH assumption
there exists a GS relative to which the **CDH** problem is hard.

CDH assumption implies that in the DHKE protocol, computing $k = g^{xy}$ from $h_A = g^x$ and $h_B = g^y$ is unfeasible. But we might also want the key k to look random. CDH does not immediately guarantees that. We need to strengthen the assumption one step further.

Decisional Diffie-Hellman assumption (DDH). The DDH assumption says that g^{xy} not only is hard to compute, but it also looks just like a random group element, even if g^x and g^y are known to the adversary. Formally the following two distributions should be hard to distinguish.

- $(G, q, g) \leftarrow \mathsf{GS}(1^n)$, $x_1, x_2 \leftarrow \mathbb{Z}_q$, and output $(G, q, g, g^{x_1}, g^{x_2}, g^{x_1 x_2})$;

- $(G, q, g) \leftarrow \mathsf{GS}(1^n)$, $x_1, x_2, x_3 \leftarrow \mathbb{Z}_q$, and output $(G, q, g, g^{x_1}, g^{x_2}, g^{x_3})$; i.e., g^{x_3} is an independent random group element.

  For any PPT distinguisher D,

$$|\Pr[D(G, q, g, g^{x_1}, g^{x_2}, g^{x_1 x_2}) = 1] - \Pr[D(G, q, g, g^{x_1}, g^{x_2}, g^{x_3})]| \leq \mathrm{negl}(n)\,.$$

The DDH assumption

For random $x_1 \leftarrow \mathbb{Z}_q, x_2 \leftarrow \mathbb{Z}_q$ and $h \leftarrow G$, the following two distributions are computationally indistinguishable

$$\{(g^{x_1}, g^{x_2}, g^{x_1 x_2})\} \approx_c \{(g^{x_1}, g^{x_2}, h)\}\,.$$

Relations between DL, CDH, DDH. Clearly $DDH \leq CDH \leq DL$. Namely if one can solve DL, then CDH is simple. Likewise, if one can solve CDH, then DDH becomes easy as well. But the reverse directions are unknown. Therefore DDH appears to be the easiest and DL seems to be the hardest. As a result, DDH assumption is the strongest (assuming a potentially easier problem being hard).

! The choice of cyclic group G matters. There are many *easy* groups. Read [KL: 8.3.3] for more discussion.

Theorem 5. *DH Key-Exchange is "secure"*[3] *assuming the DDH assumption.*

[3] Read [KL: Def.10.1] for a formal definition of security

ElGamal PubKE

Can we use DL related assumptions to build PubKE? Unfortunately, we do not know how to construct a TDP out of DL/CDH/DDH. However, it turns out that the DH key-exchange protocol can be easily adapted to a PubKE known as the *El Gamal* scheme.

Let $\mathcal{G}$ be as above. Construct $\Pi = (G, E, D)$

- G: run $(G, q, g) \leftarrow \mathsf{GS}(1^n)$, choose uniform $x \leftarrow \mathbb{Z}_q$ and compute $h := g^x$.

$$pk := (G, q, g, h), \quad sk := (G, q, g, x)\,.$$

  Message space is group elements of G.

- E: on input pk and message $m \in G$, choose uniform $y \leftarrow \mathbb{Z}_q$ and output

$$c := (c_1 = g^y, c_2 = h^y \cdot m)\,.$$

- D: on input sk and ciphertext $c = (c_1, c_2)$, output

$$\hat{m} := c_2 / c_1^x$$

Figure 3: The El Gamal PubKE scheme

Correctness of the El Gamal

$$\hat{m} = \frac{c_2}{c_1^x} = \frac{h^y \cdot m}{(g^y)^x} = \frac{g^{xy} \cdot m}{g^{xy}} = m\,.$$

To gain some intuition about the security, note that the critical information that an adversary obtains is $(h = g^x, g^y, g^{xy} \cdot m)$. But under DDH assumption, g^{xy} would be indistinguishable from an independent uniform h'. Then the encryption is essentially one-time-pad in group G with fresh "key" for each message. Read KL book for the detailed security proof.

Theorem 6 ([KL: Thm. 11.18])**.** *Under DDH assumption, El Gamal is CPA-secure.*

Hybrid Encryption

All PubKE schemes we have seen so far rely on problems in number theory. They are usually much slower than private-key encryption schemes. Encrypting every message with a PubKE, especially when they are long, is inefficient in practice. A common practice in the real world is to combine PubKE with PrivKE as a *hybrid encryption* scheme. Basically, Alice (sender) uses Bob's (receiver) public key to encrypt a priv-key (session-key), and uses PrivKE to encrypt the actual message m. Bob would first decryt using secret-key in PubKE to recover the session key with which he recovers the message.

Let $\Pi^{pub} = (G^{pub}, E^{pub}, D^{pub})$ be a PubKE and $\Pi^{priv} = (G^{priv}, E^{priv}, D^{priv})$ be a PrivKE. Construct **PubKE** $\Pi = (G, E, D)$

- $G = G^{pub}$: run $(pk, sk) \leftarrow G^{pub}(1^n)$.
- E: on input pk and message m, run $k \leftarrow G^{priv}(1^n)$. Output ciphertext

$$c := (c_1 = E^{pub}_{pk}(k), c_2 = E^{priv}_{k}(m))\,.$$

- D: on input sk and ciphertext $c = (c_1, c_2)$, compute $k := D^{pub}_{sk}(c_1)$ and output

$$m := D^{priv}_{k}(c_2)\,.$$

Figure 4: Hybrid Encryption

Notice Π is a PubKE, but benefits from the efficiency advantage of a PrivK.

Theorem 7 (KL-Thm. 11.12)**.** *If Π^{pub} is CPA-secure and Π^{priv} is computationally secret, then Π is CPA-secure.*

Proof idea. A hybrid argument bridging computational indistinguishability.

$$\begin{array}{ccc} \langle E^{pub}_{pk}(k), E^{priv}_{k}(m_0)\rangle & \overset{\text{3: transitivity}}{\longleftrightarrow} & \langle E^{pub}_{pk}(k), E^{priv}_{k}(m_1)\rangle \\ \text{1: Security of } \Pi^{pub} \updownarrow & & \updownarrow \text{ 1': Security of } \Pi^{pub} \\ \langle E^{pub}_{pk}(0^n), E^{priv}_{k}(m_0)\rangle & \overset{\text{2: Security of } \Pi^{priv}}{\longleftrightarrow} & \langle E^{pub}_{pk}(0^n), E^{priv}_{k}(m_1)\rangle \end{array}$$

□

The direct implementation of this idea requires two operations: pick a sym-key, and then encrypt it under the public-key scheme. We can merge two steps, which gives the primitive called *key-encapsulation* mechanism (KEM). Then we call the symmetric-key scheme a *data-encapsulation* mechanism (DEM), and we obtain the generic KEM/DEM hybrid encryption.

$$\begin{aligned} pk &\to \textbf{K-ENCAPS}(c_1, k); \\ (k, m) &\to \textbf{D-ENCAPS}c_2\,. \end{aligned}$$

Read [KL: Sect.11.4] if you are interested in the details.