**Disclaimer**. Draft note. No guarantee on completeness nor soundness. Read with caution, and shoot me an email at <a href="mailto:fsong@pdx.edu">fsong@pdx.edu</a> for corrections/comments (they are always welcome!)

**Last time**. Public-key revolution. **Today**. Public-key encryption

## 1 Discret-Logarithm and CDH/DDH assumptions

Recall two multiplicative groups we discussed last time.

 $\mathbb{Z}_N^* = \{a \in \{1, ..., N-1\} : \gcd(a, N) = 1\}$  under multiplication modulo N. We introduced the Euler function  $\phi(N) := |\mathbb{Z}_N^*|$  to be the order of  $\mathbb{Z}_N^*$ . This group is related to the important RSA assumption, and we will come back to it later.

Another group is  $\mathbb{Z}_p^* := \{1, ..., p-1\}$  under multiplication modulo a prime p. It is a special case of an interesting family of groups called the *cyclic* groups.

Let *G* be a finite group with order |G| = q. For arbitrary  $g \in G$ , consider

$$\langle g \rangle := \{ g^0, g^1, \ldots \},$$

where 
$$g^k := \underbrace{g \circ g \circ \dots \circ g}_{k \text{ times}} = [g \cdot g \cdot \dots \cdot g \text{ mod } p].$$

**Fact 1.** For any  $g \in G$ ,  $g^q = 1$ .

Let the order of g, denoted ord(g), be the smallest i such that  $g^i = 1$ .

**Fact 2.**  $\langle g \rangle = \{g^0, g^1, \dots, g^{i-1}\}$  is a subgroup of G.

**Definition 3.** If  $G = \langle g \rangle$  for some  $g \in G$ . Then we say G is *cyclic*, and g is called a generator of G.

**Fact 4.** *If* p *is prime,*  $\mathbb{Z}_p^*$  *is a cyclic group of order* p-1.

**Example 5.** Consider  $\mathbb{Z}_7^*$ . It is cyclic by the fact above.

$$\langle 2 \rangle = \{1, 2, 4\}$$
 2 is not a generator

$$\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\}$$
 3 is a generator

We state explicitly some crucial observations:

- $g^x$  is uniformly random in G if we pick uniform  $x \leftarrow \mathbb{Z}_q := \{0, ..., q-1\}$ .
- $f: \mathbb{Z}_q \to G$  defined by  $f(x) := g^x$  is a permutation on G.

Now we define the famous *discrete-logarithm* problem. Although the problem is well-defined for any cyclic group, we would like to work in some group so that the problem is hard to solve. Therefore we introduce a group generating algorithm GS and define the group-related computational problems relative to GS. On a security parameter  $1^n$ ,  $GS(1^n)$  outputs a triple (G, q, g) where G is a cyclic group of order g (g has length g), and g is a generator of G.

**Definition 6** (KL-Def.8.62). We say that the DL problem is hard relative to GS if for all PPT  $\mathscr{A}$ ,  $\Pr[\mathsf{DLog}_{\mathscr{A},\mathscr{G}}(n)=1] \leq \operatorname{negl}(n)$ .

Portland State U, Winter 2017

FEBRUARY 23

**FS NOTE**: Draw DL diagram

- 1. Challenger *CH* generates  $(G, q, g) \leftarrow GS(1^n)$ .
- 2. CH chooses a uniform  $h \in G$ . [TS: EX. How to do this?]
- 3.  $\mathscr{A}$  is given (G, q, g, h), and  $\mathscr{A}$  outputs  $x \in \mathbb{Z}_q$ .
- 4. The output of the game is defined to be 1 if  $g^x = h$  and 0 otherwise.

Figure 1: The discrete-logarithm game  $\mathsf{DLog}_{\mathscr{A}\mathscr{Q}}(n)$ 

#### The Discrete-Logarithm assumption

there exists a GS relative to which the **DL** problem is hard. In other words, there exists a GS, such that  $f(x) := g^x$  is a *one-way* permutation.

**FS NOTE**: Draw CDH diagram

- 1. Challenger *CH* generates  $(G, q, g) \leftarrow GS(1^n)$ .
- 2. CH chooses a uniform  $x_1, x_2 \leftarrow \mathbb{Z}_q$ , and computes  $h_1 := g^{x_1}$  and  $h_2 := g^{x_2}$ .
- 3.  $\mathscr{A}$  is given  $(G, q, g, h_1, h_2)$ , and  $\mathscr{A}$  outputs  $h \in G$ .
- 4. The output of the game is defined to be 1 if  $h = g^{x_1x_2}$  and 0 otherwise.

Figure 2: The computational Diffie-Hellman game  $CDH_{\mathcal{A}\mathcal{G}}(n)$ 

The DL problem seems a hard problem despite many years of efforts. The best algorithm (on a classical computer) runs in time  $\sim 2^{n^{1/3}\log n^{2/3}}$ . The DL problem is not immediately useful for cryptography. Instead, we introduce the following related problems.

**Definition 7.** We say that the CDH problem is hard relative to GS if for all PPT  $\mathscr{A}$ ,  $\Pr[\mathsf{CDH}_{\mathscr{A},\mathscr{G}}(n) = 1] \leq \operatorname{negl}(n)$ .

#### The CDH assumption

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

Another variant is called the Decisional DH problem.

**Definition 8** (KL-Def.8.63). We say that the DDH problem is hard relative to GS if for all PPT  $\mathcal{A}$ ,  $\Pr[\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)=1] \leq \mathsf{negl}(n)$ .

FEBRUARY 23

**FS NOTE**: Draw DDH diagram

- 1. Challenger *CH* generates  $(G, q, g) \leftarrow \mathsf{GS}(1^n)$ .
- 2. CH chooses a uniform  $x_1, x_2 \leftarrow \mathbb{Z}_q$ , and computes  $h_1 := g^{x_1}$  and  $h_2 := g^{x_2}$ .
- 3. *CH* picks uniform  $b \leftarrow \{0,1\}$ , if b = 0, let  $h = g^{x_1x_2}$  otherwise pick uniform  $h \leftarrow G$ .
- 4.  $\mathscr{A}$  is given  $(G, q, g, (h_1, h_2, h))$ , and  $\mathscr{A}$  outputs  $b' \in \{0, 1\}$ .
- 5. The output of the game is defined to be 1 if b' = b and 0 otherwise.

Figure 3: The Decisional Diffie-Hellman game  $DDH_{\mathcal{A},\mathcal{G}}(n)$ 

#### The DDH assumption

there exists a GS relative to which the **DDH** problem is hard. In other words, 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  $(g^{x_1}, g^{x_2}, g^{x_1x_2}) \approx_c (g^{x_1}, g^{x_2}, h)$ .

**Relations between DL, CDH, DDH.** Clearly  $DDH \le CDH \le DL$ . Namely if one can solve DL, then CDH is simple. [TS: EX. Why?] Likewise, if one can solve CDH, then DDH becomes easy as well. [TS: EX. Why?] 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 is hard).

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

Now we are finally ready to instantiate the Diffie-Hellman key-exchange. [TS: Ask class about abstract version.]

Clearly  $k_A = k_B = g^{xy}$ . What an eavesdropper can see, however, are only  $h_A$  and  $h_B$ . By the CDH assumption, computing  $h = g^{xy}$  from  $(h_A, h_B)$  is infeasible for any PPT adversary. We usually need a stronger security where the final k is indistinguishable from a uniformly random key. Luckily, the DDH assumption exactly gives us that  $^1$ .

[ Read [KL: Definition 10.1 & Theorem 10.3] for a formal security definition of key-exchange and a proof that DDH implies security of the DHKE. ]

Note that DHKE is vulnerable to *man-in-the-middle* attacks. See your homework problem.

 $<sup>^{1}</sup>$ More precisely,  $g^{xy}$  is indist. from a uniformly random group element. Alice and Bob will need to apply some *key-derivation*, e.g., a "nice" hash function, to get uniform bit strings.

FEBRUARY 23

**FS NOTE**: Draw DHKE protocol

- 1. Alice generates  $(G, q, g) \leftarrow GS(1^n)$ .
- 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 4: The Diffie-Hellman key-exchange protocol

## 2 Public-key encryption

We just finished one of major contributions of the seminar work by Diffie-Hellman, which was inspired by Merkle's idea of exchanging a secret key through insecure channels. DH was able to improve and instantiate this idea based on DL and achieves an exponential gap between the computational costs of the honest users and that of an adversary. Another important *conceptual* contribution of Diffie-Hellman is that they envisioned *public-key* encryption and authentication (i.e. digital signature), assuming existence of some magic box. Let's dicuss PubKE first.

**Definition 9.** A public-key encryption scheme is a triple of PPT algorithms (G, E, D) such that

- 1.  $G: (pk, sk) \leftarrow G(1^n)$ . pk: public-key, sk: secret-key.
- 2. E: on input pk and m,  $c \leftarrow E_{nk}(m)$ .
- 3. *D*: takes sk and c, and computes  $m := D_{sk}(c)$  (assuming D is always deterministic).

Correctness requirement:  $D_{sk}(E_{pk}(m)) = m$  for any m except with negligible probability.

Did you notice the main distinction from a private-key encryption scheme? G generates a pair of keys (pk, sk): one for encryption and the other one for decryption. (That's why people aslo call them asymmetric encrytion and symmetric encrytion respectively)

The standard security notion for PubKE will be CPA security, the key idea of which we should be familiar with by now.

## 2.1 CPA-security

Given a PubKE scheme  $\Pi = (G, E, D)$  and an adversary  $\mathcal{A}$ , consider

**Definition 10.** A public-key encryption scheme  $\Pi$  is CPA-secure if for any PPT  $\mathcal{A}$ ,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathscr{A},\Pi}(n) = 1] \le \frac{1}{2} + \operatorname{negl}(n).$$

FS NOTE: Draw CPA diagram

- 1. CH runs  $(pk, sk) \leftarrow G(1^n)$ .
- 2.  $\mathscr{A}$  is given pk, and output  $(m_0, m_1)$  with  $|m_0| = |m_1|$ .
- 3. *CH* picks uniform bit  $b \leftarrow \{0,1\}$  and computes  $c \leftarrow E_{pk}(m_b)$ . Send this challenge ciphertext c to  $\mathcal{A}$ .
- 4.  $\mathscr{A}$  outputs b'.  $\mathscr{A}$  succeeds if b' = b, and let  $\mathsf{PubK}^\mathsf{cpa}_{\mathscr{A},\Pi}(n) = 1$  in this case. Otherwise define  $\mathsf{PubK}^{\mathsf{cpa}}_{\mathscr{A},\Pi}(n) = 0$ .

Figure 5: The CPA indistinguishability game  $PubK_{\mathscr{A},\Pi}^{cpa}(n)$ 

Where is the CPA part? Once pk is given,  $\mathscr{A}$  can just implement the encryption oracle  $E_{nk}$  on its own. Hence in the public-key setting, there makes no difference between an eavesdropper and an CPA-adversary, in contrast to the private-key setting (computational secrecy is strictly weaker than CPA-security).

Again, CPA-security necessarily needs randomized encryption.

**Theorem 11** (KL-Thm.11.4). *No determinstic public-key encryption is CPA-secure.* Multiple encryptions. We do not formally discuss CPA-security for encrypting multiple messages [KL: Definition 11.5] other than stating the important fact

1-bit encryption is complete for public-key CPA-security.

#### **Constructing CPA-secure PubKE** 3

How do we construct a CPA-secure PubKE? Diffie-Hellman already suggested an approach using an imaginary "magic" box. Let's formalize more precisely the "magic" box they had in mind: trapdoor one-way permutations.

## **Trapdoor one-way permutations**

**FS NOTE**: Draw diagram

**Definition 12.** A trapdoor one-way permutation (TDP) is a triple of poly-time algorithms

- G:  $(pk, sk) \leftarrow G(1^n)$ . pk is called a public key and sk is called a secret key (or trapdoor sometimes denoted as td).
- F: deterministic algorithm  $y = F_{pk}(x)$  and  $F_{pk}(\cdot)$  is a permutation on domain X.
- *I*: deterministic inversion algorithm  $x = I_{sk}(y)$ .

FEBRUARY 23

Portland State U, Winter 2017

and it satisfies the *correctness* and *one-wayness* conditions:

- Correctness:  $I_{sk}(f_{pk}(x)) = x$  for all  $x \in X$  except with negligible probability (over choice of (pk, sk)).
- One-way (without knowing sk):  $F_{pk}$  is one-way, namely for any PPT  $\mathcal{A}$ , it holds that

$$\Pr\left[x'=x:(pk,sk)\leftarrow G(1^n),x\leftarrow X,y=F_{pk}(x),x'\leftarrow \mathcal{A}(pk,y)\right]\leq \operatorname{negl}(n).$$

We often abuse notation and denote a TDP just by F.

**Implementing DH approach verbatim**. Construct PubKE  $\Pi = (G, E, D)$  from a TDP (G, F, I):

- G = G:  $(pk, sk) \leftarrow G(1^n)$ .
- $E: c = E_{pk}(m) := F_{pk}(m)$ .
- $D: m = D_{sk}(c) := I_{sk}(c)$ .

You should realize immediately that this CANNOT be CPA-secure, because it is deterministic! The fact is DH didn't fully realize the importance of randomized encryption in the beginning, and this has to wait till the seminar work<sup>2</sup> by another two pioneers in PubKC, Shafi Goldwasser and Silvio Micali.

### 3.2 PKE from TDP + hard-core predicate

A correct idea of designing a CPA-secure PubKE from a TDP is combining a *hard-core* predicate of  $F_{pk}$ . Recall a hard-core predicate  $hc: X \to \{0,1\}$  of F is an efficiently computatable function such that

$$\Pr[b' = b : x \leftarrow X, b := \mathsf{hc}(x), b' \leftarrow \mathcal{A}(pk, F_{pk}(x))] \le \mathsf{negl}(n).$$

Assume we have (F, hc) being a TDP and a hard-core predicate of F. We will give candidates based on number-theoretical assumptions  $(RSA)^3$ . We propose the following PubKE scheme for single-bit messages.

Given (G, F, I) a TDP, and hc a hard-core predicate of it, construct  $\Pi = (G, E, D)$  for encryting one-bit messages  $m \in \{0, 1\}$ 

- *G*: (same as in TDP)  $(pk, sk) \leftarrow G(1^n)$
- E: on input pk and  $m \in \{0,1\}$ , pick random  $r \leftarrow X$  and output  $c \leftarrow E_{pk}(m) := (F_{pk}(r), hc(r) \oplus m)$ .
- *D*: given sk and  $c = (c_1, c_2)$ ,  $m = D_{sk}(c) := c_2 \oplus hc(I_{sk}(c_1))$ .

**Theorem 13** (variant of KL-Thm. 11.33 & 13.5). Π *is CPA-secure*.

*Proof idea.* Distinguishing encryption of 0 and encryption of 1 is equivalent to predicting hc(r) from  $F_{pk}(r)$ , which is not feasible since hc is a hard-core predicate of F.

<sup>&</sup>lt;sup>2</sup>Goldwasser, Shafi, and Silvio Micali. "Probabilistic encryption." Journal of computer and system sciences 28.2 (1984): 270-299. http://www.sciencedirect.com/science/article/pii/0022000084900709.

<sup>&</sup>lt;sup>3</sup>In fact, for any TDP, there exists a related TDP for which there is a hard-care predicate.

Intro to Cryptography Portland State U, Winter 2017

# **Lecture 11**

February 23

[ EX. Finish the proof ]

Page 7

**Instructor: Fang Song**