Lecture 12: Public-Key Encyrption Schemes

Notes by Yael Kalai

MIT - 6.5620
Lecture 12 (October 15, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Recap:

Last class we transitioned to public-key cryptography!

- Presented the Diffie-Hellman Key Agreement protocol.
- Defined public-key encryption scheme.
- Presented the El-Gamal encryption scheme.

Today

- Prove the security of the El-Gamal encryption scheme, and its variants.
- Define trapdoor (injective) one-way functions.
- Construct a public-key encryption scheme from any (injective) one-way trapdoor function.

The El-Gamal Encryption Scheme

- Let $G = G_\lambda$ be a group of prime order q of size roughly 2^λ, and let $g \in G$ be a generator. The message space $\mathcal{M}_\lambda$ is G.
- $\mathsf{Gen}(1^\lambda)$:
  1. Choose at random $s \leftarrow \mathbb{Z}_q$.
  2. Set $\mathsf{pk} = g^s$ and $\mathsf{sk} = s$.
  3. Output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$:
  1. Choose at random $r \leftarrow \mathbb{Z}_q$.
  2. Output $\mathsf{ct} = (g^r, \mathsf{pk}^r \cdot m)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:
  1. Parse $\mathsf{ct} = (R, \mathsf{ct}')$.

2. Output $m = \mathsf{ct}' \cdot (R^{-\mathsf{sk}})$.

Theorem 1. *This scheme is* CPA *secure under the* DDH *assumption.*

Recall that the DDH assumption w.r.t. a group G prime order q with generator $g \in G$, states that

$$(g^a, g^b, g^{a \cdot b}) \approx (g^a, g^b, g^c)$$

where $a, b, c \leftarrow \mathbb{Z}_q$.

Recall the definition of CPA security for public-key encryption:

Definition 2. A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is said to be CPA-secure if for every poly-size $\mathcal{A}$ there exists a negligible μ such that for every $\lambda \in \mathbb{N}$ the adversary $\mathcal{A}$ wins the game below with probability at most $\frac{1}{2} + \mu(\lambda)$:

1. The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ chooses $m_0, m_1 \in \mathcal{M}_\lambda$ and send these messages to the challenger.
3. The challenger samples a random bit $b \leftarrow \{0,1\}$ and computes $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$. and sends ct_b to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b' (as a guess for b).

$\mathcal{A}$ wins if and only if $b' = b$.

Proof of Theorem 1 Fix any poly-size adversary $\mathcal{A}$ and suppose that $\mathcal{A}$ wins in the CPA security game (Definition 2) with non-negligible advantage $\epsilon : \mathbb{N} \to [0,1]$. We construct a poly-size adversary $\mathcal{B}$ that breaks the DDH assumption. Given (g^a, g^b, g^c), $\mathcal{B}$ emulates the CPA security game with $\mathcal{A}$, as follows:

1. Send $\mathsf{pk} = (g, g^a)$ to $\mathcal{A}$.
2. Emulate $\mathcal{A}(\mathsf{pk})$ to compute $m_0, m_1 \in G$.
3. Sample a random $\beta \leftarrow \{0,1\}$, and send to $\mathcal{A}$ the ciphertext $(g^b, g^c \cdot m_\beta)$.
4. Compute the guess β' computed by $\mathcal{A}$.
5. If $\beta = \beta'$ then output 1 (indicating that the input (g^a, g^b, g^c) is a DDH tuple) and otherwise output 0.

Note that if (g^a, g^b, g^c) is a DDH tuple then this is a perfect simulation of the CPA game and hence, by our assumption

$$\Pr[\mathcal{B}(g^a, g^b, g^c) = 1 \mid (g^a, g^b, g^c) \in \mathsf{DDH}] \geq \frac{1}{2} + \epsilon(\lambda)$$

On the other hand,

$$\Pr_{a,b,c \leftarrow [q]}[\mathcal{B}(g^a, g^b, g^c) = 1] = \frac{1}{2}$$

even if $\mathcal{B}$ is all powerful, since g^c is a perfect mask of m_β.

□

Basically, the above proof shows that if we run the Diffie-Helman key agreement protocol polynomially many times $t = \text{poly}(\lambda)$, with the same first message g^a and with independent randomly chosen second messages $g^{b_1}, \ldots, g^{b_t}$, then,

$$(g^a, g^{b_1}, \ldots, g^{b_t}, g^{a \cdot b_1}, \ldots, g^{a \cdot b_t}) \approx (g^a, g^{b_1}, \ldots, g^{b_t}, g^{c_1}, \ldots, g^{c_t})$$

where $c_1, \ldots, c_t \leftarrow [q]$.

This can be proven via a standard hybrid argument (which if you squint, you can see that this is basically what is going on inside the CPA-security proof above). The point is that given (g^a, g^b, g^c) we can efficiently simulate the tuple

$$\left(g^{b_1}, \ldots, g^{b_{i-1}}, g^{a \cdot b_1}, \ldots, g^{a \cdot b_{i-1}}\right)$$

and the tuple

$$\left(g^{b_{i+1}}, \ldots, g^{b_t}, g^{c_{i+1}}, \ldots, g^{c_t}\right).$$

An Alternative Formulation of the El-Gamal Encryption Scheme

In the El-Gamal encryption scheme we presented above, the message space is G and we use g^{ab} as a one time pad by multiplying it with the message $m \in G$. Alternatively, we can think of the message space as being $\{0,1\}^\ell$ and defining the encryption algorithm to be:

$$\mathsf{Enc}(\mathsf{pk}, m) = (g^r, H(pk^r) \oplus m)$$

where $H : G \to \{0,1\}^\ell$.

Why is this better? First, $\mathcal{M} = \{0,1\}^\ell$ is a more natural message space. Second, if we use an appropriate H we can rely on the weaker CDH assumption (as opposed to the stronger DDH assumption).

For example, we can have H be a hardcore predicate! Recall that by Goldreich-Levin's theorem we can take H to be the randomized predicate that on input x and randomness r outputs

$$H(x; r) = x \cdot r = \sum_{i=1}^{\lambda} x_i r_i \bmod 2.$$

Where we think of G as embedded in $\{0,1\}^\lambda$, and the resulting message space is $\mathcal{M} = \{0,1\}$.

In practice, H is taken to output many bits (say λ bits), and the assumption is that H "behaves like a truly random function." This is formalized via the *random oracle model* (ROM) described below.

Random Oracle Model

The random oracle model was introduced by Bellare and Rogaway [1]. The idea is to analyze the security of the cryptosystem assuming that the underlying hash function H is a truly random function and that the adversary only has black-box access to H (i.e., it can feed inputs and obtain outputs).

Note that if H was indeed a random oracle then the above variant of the El-Gamal encryption scheme would be secure under the CDH assumption, since if the adversary cannot compute pk^r then $H(\mathsf{pk}^r)$ looks truly random and hence can be safely used as a one-time pad.

We emphasize that this random oracle model is an *ideal* model, which is not consistent with reality! Hash functions are *not* random functions, and the adversary is given more than black-box access to the function, since it is given a succinct description of H that allows him to compute the function efficiently. Indeed, there are known counter examples, due to Canetti, Goldreich and Halevi [2], of schemes that are secure in the random oracle model and are insecure when the random oracle is instantiated with *any* hash function.

Nevertheless, the random oracle model has proven to be extremely useful! It is a great proxy for analyzing the security of cryptosystems, and so far we have not seen examples of real-world schemes that were proven secure in the random oracle model, and were broken in practice where the hash function was taken to be an off-the-shelf hash function, such as SHA256.

Constructing Public-Key Encryption from General Assumptions

So far we have seen how to construct a public-key encryption scheme from a very specific assumption, which may or may not be secure! Diffie and Hellman, in their seminal paper [3], proposed to use trapdoor (injective) one-way functions to construct a public-key encryption.

Trapdoor One-Way Functions

Definition 3. A trapdoor (injective) one-way function family consists of a PPT key generation algorithm Gen, that takes as input 1^λ and outputs a (public) hash key hk together with an associated (secret) trapdoor key td, and a family of functions $F_{\mathsf{hk}} : \mathcal{D}_{\mathsf{hk}} \to \mathcal{R}_{\mathsf{hk}}$. The following properties are required to hold:

- **Efficient computable:** There is a poly-time algorithm $\mathcal{A}$ such that for every $(\mathsf{hk}, \mathsf{td}) \in \mathsf{Gen}(1^\lambda)$ and for every $x \in \mathcal{D}_{\mathsf{hk}}$, $\mathcal{A}(\mathsf{hk}, x) = F_{\mathsf{hk}}(x)$.
- **Efficiently sampleable:** There is a PPT algorithm $\mathcal{A}$ such that for any $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda)$, and any $x \in \mathcal{D}_{\mathsf{hk}}$,
$$\Pr[\mathcal{A}(\mathsf{hk}) = x] = \frac{1}{|\mathcal{D}_{\mathsf{hk}}|}$$
(I.e., $\mathcal{A}$ outputs a uniformly random sample from the domain $\mathcal{D}_{\mathsf{hk}}$)
- **Trapdoor invertible:** There is a poly-time inversion algorithm $\mathcal{I}$ such that for any $(\mathsf{hk}, \mathsf{td}) \in \mathsf{Gen}(1^\lambda)$ and any $x \in \mathcal{D}_{\mathsf{hk}}$
$$\mathcal{I}(\mathsf{td}, F_{\mathsf{hk}}(x)) = x$$
- **One-way:** For every polys-size $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,
$$\Pr[\mathcal{A}(\mathsf{hk}, F_{\mathsf{hk}}(x)) = x] \le \mu(\lambda)$$
where the probability is over $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda)$ and over $x \leftarrow \mathcal{D}_{\mathsf{hk}}$.

Public-key Encryption from Trapdoor One-Way Functions

Intuitively, to build a public-key encryption scheme from a trapdoor one-way function, we should take the public key to be hk and the secret key to be td. However, it is not clear what to do next. We will rely on the Goldreich-Levin theorem that there exists a hardcore predicate $\mathsf{HCP}(x, r) = x \cdot r$ such that
$$(\mathsf{hk}, F_{\mathsf{hk}}(x), r, \mathsf{HCP}(x, r) \approx (\mathsf{hk}, F_{\mathsf{hk}}(x), r, U)$$
where $U \leftarrow \{0, 1\}$.

Remark. There seems to be a mismatch here since on the one hand $x \in \mathcal{D}_{\mathsf{hk}}$ and on the other hand HCP assumes that $x \in \{0, 1\}^\lambda$. However, we argue that this mismatch is only syntactic and can be corrected easily. Basically, all we need to do is embed $\mathcal{D}_{\mathsf{hk}}$ in $\{0, 1\}^\ell$, for an appropriate $\ell(\lambda) = \text{poly}(\lambda)$.

Alternatively, the fact that $x \leftarrow \mathcal{D}_{\mathsf{hk}}$ is efficiently sampleable implies that one can think of F_{hk} as having the domain $\{0, 1\}^\lambda$, and on input $r \in \{0, 1\}^\lambda$ it samples $x \leftarrow \mathcal{D}_{\mathsf{hk}}$ using randomness r and outputs $F_{\mathsf{hk}}(x)$.[1] Thus, from now on we assume that $\mathcal{D}_{\mathsf{hk}} = \{0, 1\}^\lambda$.[2]

[1] We will also need to assume that the transformation from $r \in \{0, 1\}^\lambda$ to x is injective. This can be achieved by taking a shorter λ and using a PRG. We omit the details here.

[2] We note that we will only be interested in how F_{hk} behaves on random elements $x \leftarrow \mathcal{D}_{\mathsf{hk}}$ hence this change of domain does not have an affect.

In what follows, given a trapdoor one-way function family, defined by Gen_F and $\{F_{\mathsf{hk}}\}$, we construct the following public-key encryption scheme:

- Gen: Takes as input 1^λ and does the following:
  1. Sample $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}_F(1^\lambda)$.
  2. Set $\mathsf{pk} = \mathsf{hk}$ and $\mathsf{sk} = \mathsf{td}$.
  3. Output $(\mathsf{pk}, \mathsf{sk})$.
- Enc: takes as input the public key hk and a message $m \in \{0,1\}$, and does the following:
  1. Sample a random $x \leftarrow \{0,1\}^\lambda$.
  2. Compute $y = F_{\mathsf{hk}}(x)$.
  3. Choose a random $r \leftarrow \{0,1\}^\lambda$, and let $b = \mathsf{HCP}(r, x)$.
  4. Output $(y, r, b \oplus m)$.
- Dec: On input a secret key td and a ciphertext $\mathsf{ct} = (y, r, c)$, does the following:
  1. Compute $x = \mathcal{I}(\mathsf{td}, y)$.
  2. Compute $b = \mathsf{HCP}(r, x)$.
  3. Output $m = b \oplus c$.

Theorem 4. *The public-key encryptions scheme above is* CPA*-secure*

Proof. Suppose for the sake of contradiction that there exists a poly-size adversary $\mathcal{A}$ and a non-negligible ϵ such that $\mathcal{A}$ wins in the CPA game with probability ϵ. We construct a poly-size adversary that inverts the one-way function. To this end, it suffices to construct a poly-size $\mathcal{B}$ for which there exists a non-negligible ϵ such that for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{B}(\mathsf{hk}, F_{\mathsf{hk}}(x), r) = \mathsf{HCP}(x, r)] \geq \epsilon(\lambda),$$

where the probability is over $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}_F(1^\lambda)$ and over $x, r \leftarrow \{0,1\}^\lambda$.

$\mathcal{B}(\mathsf{hk}, F_{\mathsf{hk}}(x), r)$ emulates the adversary $\mathcal{A}$ in the CPA game, as follows:

1. Send $\mathsf{pk} = \mathsf{hk}$ to $\mathcal{A}$.
2. Emulate $\mathcal{A}(\mathsf{hk})$ to obtain $m_0, m_1 \in \{0,1\}$.
3. Choose at random $b \leftarrow \{0,1\}$.
4. For each $d \in \{0,1\}$ (which should be thought of as a guess for $\mathsf{HCP}(x, r)$),

(a) Let $\mathsf{ct}_d = (F_{\mathsf{hk}}(x), r, d \oplus m_b)$.

(b) Emulate $\mathcal{A}$ on input ct_d to obtain b'_d.

5. If $b'_0 = b'_1$ then output a random guess $b' \leftarrow b$.

6. Otherwise output d such that $b'_d = b$.

Note that for $d = \mathsf{HCP}(x, r)$ it holds that

$$\Pr[b'_d = b] \geq \frac{1}{2} + \epsilon(\lambda)$$

whereas for a random $d \leftarrow \{0, 1\}$,

$$\Pr[b'_d = b] = \frac{1}{2}.$$

therefore $\mathcal{B}$ outputs $\mathsf{HCP}(x, r)$ with a non-negligible advantage, as desired.

□

Next class we will see examples of trapdoor one-way function families.

References

[1] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93)*, pages 62–73, Fairfax, Virginia, USA, 1993. ACM.

[2] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC '98)*, pages 209–218, Dallas, Texas, USA, 1998. ACM.

[3] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.