Lecture 8: Public Key Encryption II

MIT - 6.5620
Lecture 7 (October 5, 2026)

> ***Warning:*** This document is a rough draft and may contain bugs. Please feel free to email us with corrections.

Recap:

Last class we transitioned to public-key cryptography!

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

Today

- Present the El-Gamal encryption scheme and prove its security.
- Define trapdoor (injective) one-way functions, and look at constructions.

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 (Theorem 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

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

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 [1], 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 $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A}(\mathsf{hk}, F_{\mathsf{hk}}(x)) = x] \leq \mu(\lambda)$$

  where the probability is over $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda)$ and over $x \leftarrow \mathcal{D}_{\mathsf{hk}}$.

Warmup Construction: Rabin's Function

As a warmup we present Rabin's function family [3]. It is not quite what we want, since it is not injective. We will later elaborate on how one can convert this trapdoor family into an injective one.

The Construction

- $\mathsf{Gen}(1^\lambda)$:
  1. Choose two random primes $p, q \leftarrow \{0,1\}^\lambda$.
  2. Compute $N = p \cdot q$.
  3. Output $\mathsf{hk} = N$ and $\mathsf{td} = (p, q)$.
- $F_N(x) : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, where

$$F_N(x) = x^2 \bmod N.$$

Definition 4. Factoring is said to be hard if for every poly-size $\mathcal{A}$ there exists a negligible μ such that

$$\Pr[\mathcal{A}(N) = (p, q)] \leq \mu(\lambda)$$

where the probability is over randomly chosen primes $p, q \leftarrow \{0,1\}^\lambda$ and where $N = p \cdot q$.

Claim 1. Rabin's function family is a trapdoor (non-injective) one-way function family (where the inverstion algorithm finds all preimages) assuming the hardness of Factoring.

Proof. We prove that this function family satisfies all the conditions from **??** except the injectiveness.

- **Efficiently computable:** It is easy to see that given $\mathsf{hk} = N$ and given any $x \in \mathbb{Z}_N^*$ it is easy to compute $F_N(x) = x^2 \bmod N$.

- **Efficiently sampleable:** One can efficiently sample an element from $\mathbb{Z}_N^*$ uniformly at random, by sampling a random element in $x \leftarrow \{1, \ldots, N-1\}$, and checking that $\gcd(x, N) = 1$.[1] If this is not the case then given x one can efficiently compute p, q such that $N = p \cdot q$, and hence sample a random element $x \leftarrow \mathbb{Z}_N^*$ by sampling a random element $a \leftarrow \mathbb{Z}_p^*$ and a random element $b \leftarrow \mathbb{Z}_q^*$, and using the CRT theorem to compute $x \in \mathbb{Z}_N^*$ such that

$$x = a \bmod p \quad \wedge \quad x = b \bmod q$$

[1] Note that gcd is an efficiently computable function, via Euclid's the extended gcd algorithm.

- **Trapdoor invertible:** Given $\mathsf{td} = (p, q)$ and given $y = F_N(x) = x^2 \bmod N$, we will use the Chinese Remainder Theorem (CRT), compute all four square-roots of y modulo N, as follows:

  1. Compute $a \in \mathbb{Z}_p^*$ such that $a^2 = y \bmod p$.

     If $p = 3 \bmod 4$ then this can be done by setting $a = y^{\frac{p+1}{4}}$. Note that

     $$a^2 = y^{\frac{p+1}{2}} = y^{\frac{p-1}{2}} \cdot y = y,$$

     where the latter equality follows from the fact that y is a square, and hence if $g \in \mathbb{Z}_p^*$ is a generator then $y = g^{2b}$, for some $b \in \mathbb{N}$, and hence $y^{\frac{p-1}{2}} = g^{b \cdot (p-1)} = 1$.

     Remark. Note that we rely on the fact that we know the order of the group $\mathbb{Z}_p^*$, which is $p - 1$. In the above, we also rely on the fact that $\frac{p+1}{4} \in \mathbb{N}$, which holds only if $p = 3 \bmod 4$. We can remove the restriction that $p = 3 \bmod 4$, at the price of making the algorithm for finding square-roots more complicated. For the sake of completeness we present the general square-root finding algorithm for $\mathbb{Z}_p^*$ at the end of these notes.

     We will later see that it is hard to compute square roots modulo N, assuming the hardness of Factoring, since computing the order of the group $\mathbb{Z}_N^*$ is as hard as Factoring.

     Note that α and $p - \alpha$ are the only two square roots of y modulo p. This follows from the fact that p is prime and hence $\mathsf{GF}[p]$ is a field.

  2. Compute b such that $b^2 = y \bmod q$.

     This is done in a similar manner, and as above, β and $q - \beta$ are the only two square roots of y modulo q.

  3. For every one of the four elements

     $$(a, b), (a, q-b), (p-a, b), (p-a, p-b) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$$

     output the corresponding elements $c_1, c_2, c_3, c_4 \in \mathbb{Z}_N^*$.

4. **One-way:** We will show that if the one-way condition does not hold then we can break the hardness of Factoring assumption. Specifically, suppose that there exists a poly-size $\mathcal{A}$ and a non-negligible ϵ such that

$$\Pr[\mathcal{A}(N, y) = x : \ x^2 = y \bmod N] \geq \epsilon(\lambda)$$

where the probability is over randomly chosen primes $p, q \leftarrow \{0,1\}^\lambda$, and setting $N = p \cdot q$, and over a randomly chosen $r \leftarrow \mathbb{Z}_N^*$ and setting $y = r^2 \bmod N$.

We will construct a poly-size algorithm $\mathcal{B}$ that breaks the hardness of Factoring. Given $N = p \cdot q$, the algorithm $\mathcal{B}$ does the following:

(a) Choose a random element $r \leftarrow \mathbb{Z}_N^*$.
(b) Compute $y = r^2 \bmod N$.
(c) Compute $x = \mathcal{A}(N, y)$.
(d) Compute $d = \gcd(N, x + r)$.
(e) If $d = 1$ then abort, and otherwise output $\{d, N/d\}$.

We next argue that $\mathcal{B}$ successfully factors N with probability $\frac{\epsilon(\lambda)}{2}$. To this end denote by Succ the event that $\mathcal{A}$ is successful at finding a preimage.

$$\Pr[\mathcal{B}(N) = (p, q)] =$$
$$\Pr[\mathcal{B}(N) = (p, q) | \ \mathsf{Succ}] \cdot \Pr[\mathsf{Succ}] + \Pr[\mathcal{B}(N) = (p, q) | \ \neg\mathsf{Succ}] \cdot \Pr[\neg\mathsf{Succ}] \geq$$
$$\Pr[\mathcal{B}(N) = (p, q) | \ \mathsf{Succ}] \cdot \Pr[\mathsf{Succ}] \geq$$
$$\Pr[\mathcal{B}(N) = (p, q) | \ \mathsf{Succ}] \cdot \epsilon(\lambda) =$$
$$\frac{1}{2} \cdot \epsilon(\lambda)$$

where the last equation follows from the fact that y has four preimages modulo N:

$$(a, b), (a, q - b), (p - a, b), (p - a, q - b),$$

and $\mathcal{A}$ has no information about the preimage r chosen by $\mathcal{B}$. Hence, the output would be identical if first $\mathcal{A}$ would output a preimage x and then $\mathcal{B}$ would choose a random preimage r. Thus, with probability $1/2$ the two primages will be equal modulo one of the primes and not equal modulo the other, in which case indeed $\gcd(N, x + r)$ will output one of the primes.

□

Rabin's construction is very nice but it is not injective, and hence it is not clear how it can be used to construct a public-key encryption scheme. However, note that we can easily make it injective by

restricting its domain to be:

$$\mathbb{QR}_N = \{x^2 : \ x \in \mathbb{Z}_N^*\}.$$

Namely, we can define

$$F_N : \mathbb{QR}_N \to \mathbb{QR}_N$$

which turns out to be a permutation, if $p, q \equiv 3 \bmod 4$!

Terminology: An integer N of the form $p \cdot q$, where p and q are primes such that $p = 3 \bmod 4$ and $q = 3 \bmod 4$, is called a Blum integer.

The reason F_N is a permutation, assuming N is a Blum integer, is that $x \in \mathbb{QR}_N$ if and only if $x \in \mathbb{QR}_p$ and $x \in \mathbb{QR}_q$. Recall that every element in $c \in \mathbb{QR}^N$ has 4 square roots in $\mathbb{Z}_N^*$, and these roots are of the form in the CRT representation: $\{(\pm a, \pm b)\}$ for some $a \in \mathbb{Z}_p^*$ and $b \in \mathbb{Z}_q^*$. The fact that $p = 3 \bmod 4$ and $p = 3 \bmod 4$ implies that $-1 \notin \mathbb{QR}_p$ and $-1 \notin \mathbb{QR}_q$, and hence exactly one of these four roots is in $\mathbb{QR}_N$.

This construction turned out to be extremely useful, and has led to the Goldwasser-Micali encryption scheme [2], which was the first public-key encryption scheme that was proven to be secure! We will look at this encryption scheme in the next lecture.

RSA Construction of a Trapdoor Permutation

We now present the RSA function, due to Rivest, Shamir, and Adleman [4]. The RSA trapdoor function came before Rabin's function, but it is convenient to think of it as a modification of Rabin's function, that converts it from a trapdoor function into a trapdoor permutation.

Recall that Rabin's function $F_N(x) = x^2 \bmod N$ had the problem that each image has 4 preimages. This results from the fact that in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ every square has two square roots, and thus by the CRT theorem, in $\mathbb{Z}_N^*$ each square has four square roots.

There is a natural fix to this! Instead of taking $F_N(x) = x^2 \bmod N$, define it to be

$$F_{N,e}(x) = x^e \bmod N$$

where e is any element that is co-prime to $p-1$ and $q-1$. The point is that if e is co-prime to $p-1$ and $q-1$ then $F_{N,e} : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ is a permutation. The reason is that if e is co-prime to $p-1$ then there exists d such that $d \cdot e = 1 \bmod (p-1)$ and hence if

$$x_1^e = x_2^e \bmod N$$

then

$$x_1^e = x_2^e \bmod p \quad \wedge \quad x_1^e = x_2^e \bmod q.$$

Note that

$$(x_1^e)^d \bmod p = {x_1}^{1+c\cdot(p-1)} \bmod p = x_1,$$

and the same holds for q and for x_2. This implies that

$$x_1 = x_2 \bmod p \quad \wedge \quad x_1 = x_2 \bmod q$$

which by the CRT theorem implies that $x_1 = x_2$.

One way of choosing such e is setting $e = 3$ and choosing random p, q such that $p \bmod 3 = 2$ and $q \bmod 3 = 2$. This ensures that 3 is co-prime to $p-1$ and $q-1$.

This is exactly the RSA trapdoor permutation! The next question to ask is whether this is still one-way assuming the hardness of Factoring? The sad answer is that we don't know. Instead we make a new assumption, referred to as the RSA assumption.

The RSA Assumption: For every poly-size $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A}((N,e), F_{N,e}(x)) = x] \leq \mu(\lambda)$$

where the probability is over randomly chosen $p, q \leftarrow \{0,1\}^\lambda$ such that $p \bmod 3 = 2$ and $q \bmod 3 = 2$, and over $x \leftarrow \mathbb{Z}_N^*$ where $N = p \cdot q$ and $e = 3$.

Under this assumption the one-wayness holds by definition!

References

[1] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[2] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[3] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Cambridge, MA, January 1979. Technical Report.

[4] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.