Lecture 13: Public-Key Encryption Schemes (Cont.)

Notes by Yael Kalai

MIT - 6.5620

Lecture 13 (October 20, 2025)

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

Recap:

In the last lecture, we saw how to construct a CPA-secure public-key encryption scheme from any trapdoor injective one-way function family.

Today:

We will see constructions of trapdoor one-way functions from number theoretic assumptions.

First, let us recall the definition of trapdoor one-way functions and how they can be used to construct a public-key encryption scheme.

Definition 1. 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:

- **Efficiently 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 $\mathcal{D}$ such that for any $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda)$,

  $$\mathcal{D}(\mathsf{hk}) \equiv U_{\mathcal{D}_{\mathsf{hk}}}$$

  Namely $\mathcal{D}(\mathsf{hk})$ outputs an element in $\mathcal{D}_{\mathsf{hk}}$ whose distribution is statistically close to the uniform distribution over 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 $x \leftarrow \mathcal{D}_{\mathsf{hk}}$.

The corresponding public-key encryption scheme is defined as follows, where $\mathsf{HCP}(x; r)$ is any hardcore predicate such as the Goldreich Levin (randomized) hardcore predicate

$$\mathsf{HCP}(x; r) = x \cdot r.$$

- $\mathsf{Gen}(1^\lambda)$ outputs $(\mathsf{pk}, \mathsf{sk}) = (\mathsf{hk}, \mathsf{td})$.
- $\mathsf{Enc}(\mathsf{hk}, m) = (F_{\mathsf{hk}}(x), r, \mathsf{HCP}(x; r) \oplus m)$.
- $\mathsf{Dec}(\mathsf{td}, y, r, c)$ uses td to obtain x which is the inverse of y, and then use (x, r) to unmask c and obtain $m = \mathsf{HCP}(x; r) \oplus c$.

Correctness follows from the fact that F_{hk} is injective, and the inversion algorithm finds the same x that is used to mask the message m. Security relies on the fact that by definition of a hardcore predicate,

$$(\mathsf{hk}, F_{\mathsf{hk}}(x), r, \mathsf{HCP}(x, r) \approx (\mathsf{hk}, F_{\mathsf{hk}}(x), r, U)$$

where $U \leftarrow \{0, 1\}$.

Number-Theory Review

The constructions we will see are based on number theory, but this time over the group

$$\mathbb{Z}_N^* = \{x : \ \mathsf{GCD}(x, N) = 1\}.$$

where N is a product of two large primes $p, q \leftarrow \{0, 1\}^\lambda$. Note that

$$|\mathbb{Z}_N^*| = N - p - q + 1 = (p - 1)(q - 1).$$

What makes this group very different from $\mathbb{Z}_p^*$ where p is prime, is that $\mathbb{Z}_N^*$ is a group of unknown order (assuming the hardness of Factoring).

*The Chinese Remainder Theorem (*CRT*)* the CRT theorem asserts that for $N = p \cdot q$, the group $\mathbb{Z}_N^*$ is isomorphic to the group $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$, with the isomorphism given by

$$x \to (x \bmod p, x \bmod q).$$

Namely, $c = a \cdot b \bmod N$ if and only if

$$c = a \cdot b \bmod p \quad \wedge \quad c = a \cdot b \bmod q.$$

Furthermore, for any $a \in \mathbb{Z}_p^*$ and $b \in \mathbb{Z}_q^*$, there is a unique *efficiently computable* $c \in \mathbb{Z}_N^*$ such that

$$c = a \bmod p \quad \wedge \quad c = b \bmod q.$$

Uniqueness follows from the fact that if $x_1, x_2 \in \mathbb{Z}_N^*$ satisfy that

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

then

$$x_1 - x_2 \bmod p = 0 \quad \wedge \quad x_1 - x_2 \bmod q = 0$$

which by the fact that p, q are co-prime implies that

$$x_1 - x_2 = 0 \bmod N,$$

as desired.

Claim 1. Given any primes p, q and any $a, b \in \mathbb{N}$ that are co-prime to p and q, respectively, one can efficiently compute $c \in \mathbb{Z}_N^*$, for $N = p \cdot q$, such that

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

Proof. Given any primes p, q and any $a, b \in \mathbb{N}$, co-prime to p and q, respectively, compute $c \in \mathbb{Z}_N^*$, as follows.

- Compute p^{-1} which is the inverse of p modulo q.
- Compute q^{-1} which is the inverse of q modulo p.
- Output $c = q \cdot q^{-1} \cdot a + p \cdot p^{-1} \cdot b \bmod N$.

.

Remark. The notation above is a bit misleading since $p \cdot p^{-1} \neq 1 \bmod N$, and similarly for q. Moreover, neither p nor q have an inverse modulo N. Crucially, the inverses p^{-1} and q^{-1} were defined over q and p, respectively.

It is easy to see that

$$c \bmod p = a \quad \wedge \quad c \bmod q = b,$$

as desired. □

Notation We denote by (a, b) the CRT representation of an element $c \in \mathbb{Z}_N^*$. Namely (a, b) refers to the unique $c \in \mathbb{Z}_N^*$ such that

$$c \bmod p = a \quad \wedge \quad c \bmod q = b,$$

Warmup Construction: Rabin's Function

As a warmup we present Rabin's function family [2]. 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 2. 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 2. 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 Definition 1 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

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

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

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.

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

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 [1], which was the first public-key encryption scheme that was proven to be secure!

The Goldwasser-Micali Public-Key Encryption Scheme

- $\mathsf{Gen}(1^\lambda)$:
  1. Sample random primes $p, q \leftarrow \{0,1\}^\lambda$, such that
  $$(p \bmod 4 = 3) \quad \wedge \quad (q \bmod 4 = 3).$$
  This choice of p, q ensures that (-1) is not a quadratic residue modulo p or modulo q.
  This is the case since let $x \in \mathbb{Z}_p^*$ be any generator then $-1 = x^{\frac{p-1}{2}}$. Note that this would be a quadratic residue only if $\frac{p-1}{2}$ was even modulo $p-1$, i.e., if there exists $a \in \mathbb{N}$ such that
  $$2a = \frac{p-1}{2} \bmod (p-1)$$
  which holds only if $p = 1 \bmod 4$.
  2. Let $N = p \cdot q$.
  3. Output $\mathsf{pk} = N$ and $\mathsf{sk} = (p, q)$.
- $\mathsf{Enc}(\mathsf{pk}, m)$:
  1. Sample a random $x \leftarrow \mathbb{Z}_N^*$.
  2. Compute $y = x^2 \bmod N$.
  3. Output $\mathsf{ct} = (-1)^m \cdot y$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:
  1. Output $m = 0$ if $y \in \mathbb{QR}_N$ and output $m = 1$ otherwise.

Security follows from the assumption that it is secure! Namely, the assumption is that it is hard to distinguish random elements in $\mathbb{QR}_N$ from random elements in $\mathbb{Z}_N^*$ that are not quadratic residues modulo p or q.

Remark. It turns out that it is easy to distinguish random elements in $\mathbb{QR}_N$ from random elements in $\mathbb{Z}_N^* \setminus \mathbb{QR}_N$. But we believe that it is hard to distinguish between random elements in $\mathbb{QR}_N$ from random elements in $\{x \in \mathbb{Z}_N^* : (x \bmod p) \notin \mathbb{QR}_p \ \wedge \ (x \bmod q) \notin \mathbb{QR}_q\}$. This is called the Quadratic Residue assumption and is precisely what is needed in order to prover the security of the Goldwasser-Micali encryption scheme.

RSA Construction of a Trapdoor Permutation

We now present the RSA function, due to Rivest, Shamir, and Adleman [3]. 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

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

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

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

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!

Remark. The original proposed RSA encryption scheme worked with message space $\mathcal{M}_\lambda = \mathbb{Z}_N^*$, and the encryption algorithm was defined to be

$$\mathsf{Enc}((N,e), m) := m^e \bmod N.$$

However, this is clearly insecure, since it is deterministic! We can make it secure by using the trapdoor function with a hardcore predicate, as we saw in the last lecture. Namely, let $\mathcal{M} = \{0,1\}$ and

$$\mathsf{Enc}((N,e), m) = (F_{N,e}(x), r, \mathsf{HCP}(x;r) \oplus m).$$

We can take HCP to be the Goldreich-Levin hardcore predicate. But, it turns out that the least-significant bit $P(x) := x \bmod 2$ is a hardcore predicate for RSA. Namley, if there exists a poly-size adversary that guesses the least-significant bit of x with non-negligible advantage given $x^e \bmod N$, then there is a poly-size adversary that breaks the RSA Assumption with non-negligible probability.

So, one can use the following encryption algorithm:

$$\mathsf{Enc}((N,e), m) = (F_{N,e}(x), (x \bmod 2) \oplus m).$$

Algorithm for finding square-roots in $\mathbb{Z}_p^*$

We next show how to find square-roots in the group $\mathbb{Z}_p^*$ for *any* prime p. In general, any prime p can be written as $p - 1 = 2^s \cdot k$, where $k \in \mathbb{N}$ is an odd integer.

The algorithm: Given a prime $p = 2^s \cdot k$ and a quadratic residue $y \in \mathbb{Z}_p^*$ do the following:

If

$$y^k = 1 \bmod p,$$

then the square-root of y can be computed as in the case where $p = 3 \bmod 4$, by computing

$$a = y^{\frac{k+1}{2}}.$$

Note that

$$a^2 = y^k \cdot y = y.$$

More generally, the algorithm will find z such that $y^k \cdot z^2 = 1$, and set

$$a = y^{\frac{k+1}{2}} \cdot z,$$

where

$$a^2 = y^{k+1} \cdot z^2 = y.$$

Finding such an element z is a bit complicated, and is done as follows:

1. Find any element $w \in \mathbb{Z}_p^*$ which is not a quadratic residue. This can be done by sampling a random $w \leftarrow \mathbb{Z}_p^*$ and checking if $w^{\frac{p-1}{2}} = -1$. If this is the case then w is not a quadratic residue. Otherwise, try again. Since half of the elements are not quadratic residues we expect to find one in constant time.
2. Let $W = w^k \bmod p$.
3. Set $a = 0$.
4. Note that the facts that $a = 0$ and y is a quadratic residue, imply that
$$\left(y^k \cdot W^{2a}\right)^{2^{s-1}} = \left(y^k\right)^{2^{s-1}} = 1$$
5. Find the minimal $s' \leq s-1$ such that
$$\left(y^k \cdot W^{2a}\right)^{2^{s'}} = 1$$
6. If $s' = 0$ then output $z = W^a$.
7. Otherwise, let $a := a + 2^{s-s'-1}$ and go back to Item 5.

We next argue that s' always decreases by at least one in each step, so that the algorithm terminates after at most $s \leq \log p$ steps. Namely, we claim that

$$\left(y^k \cdot W^{2\cdot\left(a+2^{s-s'-1}\right)}\right)^{2^{s'-1}} = 1$$

This follows from the following calculations:

$$\left(y^k \cdot W^{2\cdot\left(a+2^{s-s'-1}\right)}\right)^{2^{s'-1}} =$$
$$\left(y^k \cdot W^{2a}\right)^{2^{s'-1}} \cdot W^{2^{s-1}} =$$
$$(-1) \cdot (w^k)^{2^{s-1}} = (-1) \cdot (-1) = 1,$$

as needed.

References

[1] 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.

[2] 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.

[3] 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.