Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 12

Portland State University
Lecturer: Fang Song

Feb. 15, 2018

Draft note. Version: February 19, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) Computational indist., Public key revolution;
- Review of number theory
- Trapdoor one-way permutations
- Factoring and RSA

Trapdoor one-way permutations

Recall Diffie-Hellman envisioned public-key encryption and digital signature via a imaginary "magic" function. This is formalized as a trapdoor one-way permutation in modern terminology.

Definition 1. 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_{pk}: deterministic algorithm $y = F_{pk}(x)$ and $F_{pk}(\cdot)$ is a permutation on domain X.
- F_{sk}: deterministic inversion algorithm $x = F_{sk}(y)$.

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

- Correctness: $F_{sk}(F_{pk}(x)) = x$ 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

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

How do we construct such a TDP? Diffie-Hellman didn't know one in their 1976 paper, and had to wait another year till RSA found one. So far we are most successful with number-theoretic problems.

Number theory 101 in 30 mins

Divisibility and prime numbers

- Integers $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$. $\|a\|$ denotes its length (i.e., number of digits) in binary representation.
- Natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- k divides N, $k|n$, if n is a multiple of k.
- Prime number $p \geq 2$: only divisors are 1 and p. Otherwise, call it a *composite* number. [1]

How do we measure complexity of integer arithmetic? We count the number of basic operations as a function of the length $\|a\|$ (say in binary representation)[2].

[1] There are infinitely many prime numbers; and they behave much like random numbers (of course there is no randomness). Testing prime is efficient by both randomized and deterministic algorithms.

[2] For example, adding two n-bit numbers takes linear time; multiplying them takes $O(n^2)$ by high-school algorithm.

Modular arithmetic

Let a, N be integers ($N \geq 2$). By the division procedure, we can write

$$a = qN + r\,,$$

we call q the quotient, and r the remainder[3]

[3] Ex. $a = 15, N = 7$ and $15 = 2 \cdot 7 + 1$.

For integers a, b, n, we write

$$a = b \bmod N\,,$$

if a and b have them same remainder when divided by N. N is called the Modulus.

Let $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$. We define two operations mod addition $+ \bmod N$ and mod-multiplication $\cdot \bmod N$. For addition $+$, every $a \in \mathbb{Z}_N$ has an unique (additive) inverse in $b \in \mathbb{Z}_N$ such that $a + b = 0 \bmod N$.

But if we care about multiplication $\cdot$, it is not always possible to find a' such that $aa' = 1 \bmod N$[4]. To characterize which elements in $\mathbb{Z}_N$, we need the notion of the greatest common divisor and co-prime numbers.

[4] Consider $\mathbb{Z}_6$. $2 + 4 = 0 \mod 6$. But $2 \cdot 1 = 2 \mod 6$, $2 \cdot 2 = 4 \bmod 6$, $2 \cdot 3 = 0 \bmod 6$, $2 \cdot 4 = 2 \bmod 6$, $2 \cdot 5 = 4 \bmod 6$.

Greatest common divisor $gcd(a, b)$: the largest integer that is a divisor of both a and b[5]. Euclid's algorithm can compute $gcd(a, b)$ efficiently (i.e., poly in $\|a\|$ and $\|b\|$). We say a, b co-prime (aka relatively prime) if $gcd(a, b) = 1$.

[5] Ex. $gcd(10, 14) = 2$.

Theorem 2. *$a \in \mathbb{Z}_N$ has an inverse, i.e., $a' \in \mathbb{Z}_N$ such that $aa' = 1 \bmod N$, iff. $gcd(a, N) = 1$.*

Let $\mathbb{Z}_N^* := \{a \in \mathbb{Z}_N : gcd(a, N) = 1\}$ be the set of numbers co-prime to N[6]. Euler's function $\phi(N) := |\mathbb{Z}_N^*|$.

[6] Ex. $\mathbb{Z}_6^* = \{1, 5\}$

Modular Exponentiation. We will work with exponentiation modulo a large Modulus N:

$$a^b \bmod N = \underbrace{a \cdot a \ldots a}_{b \text{ times}} \bmod N \,,$$

for $a \in \mathbb{Z}_N$ and $b > 0$ a positive integer. Repeated squaring algorithm computes $a^b \bmod N$ in polynomial time.

Theorem 3 (Euler's theorem)**.** *If $N \geq 2$ and $a \in \mathbb{Z}_N^*$, then $a^{\phi(n)} = 1 \bmod n$.*

PKC based on factoring

The factoring problem and assumption

Now we introduce the famous problems and assumptions related to integer factorization.

Define $F^{\mathsf{MULT}}(p,q) = p \cdot q$, where p, q are n-bit prime numbers. The problem of factorization is to invert F^{MULT} (on random p and q). The study of factoring has a long history and yet the best factoring algorithm known still requires running time $\sim \exp(n^{1/3} \log n^{2/3})$ based on general number field sieve.

The Factoring assumption
F^{MULT} is a one-way function.

F^{MULT} is not immediately useful for public-key crypto. We introduce a related problem.

The RSA problem and assumption

Consider group $\mathbb{Z}_N^*$. Define F^{RSA} as follows:

- G: $(N, p, q) \leftarrow G(1^n)$; $N = pq$ where p, q are n-bit prime; $e, d > 0$, and $gcd(e, \phi(N)) = 1$, $ed = 1 \mod \phi(N)$. Let $pk = (N, e)$, and $sk = (N, d)$.
- $F^{\mathsf{RSA}}_{pk} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, $x \mapsto x^e \bmod N$.
- $F^{\mathsf{RSA}}_{sk} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, $y \mapsto y^d \bmod N$.

  Observe that
- F^{RSA}_{pk} is a permutation on $\mathbb{Z}_N^*$.
- $F^{\mathsf{RSA}}_{sk} = (F^{\mathsf{RSA}}_{pk})^{-1}$: $(x^e)^d \overset{WHY?}{=} x^{ed \mod \phi(N)} = 1 \bmod N$.[7]

  Then f_e is a permutation on $\mathbb{Z}_N^*$[8]. The inverse permutation is actually $f_d(y) := y^d \bmod N$, i.e., the same function with a different exponent, where $ed = 1 \bmod \phi(N)$ [KL: Corollary 8.22].

$$(x^e)^d = x^{ed} \overset{WHY?}{=} x^{ed \bmod \phi(N)} = x \bmod N \,.$$

The RSA problem is inverting F^{RSA}_{pk}, i.e. computing e-th root modulo N.

[7] F^{RSA}_{pk} and F^{RSA}_{sk} are in fact the same function (modular exponentiation) with different exponents (e, d), where $ed = 1 \bmod \phi(N)$.

[8] Verify on your own

The RSA assumption
F^{RSA} is a one-way trapdoor permutation.

Relationship btween RSA and factoring. It is not hard to see that RSA $\leq$ Factoring[9]. Does hardness of factoring imply hardness of RSA? This remains an open question. We do know that finding d from N, e is as hard as factoring N. In your homework, you will show that computing $\phi(N)$ is as hard as factoring N as well.

[9] If we can factor N to get (p, q), then we can compute $\phi(N) = (p-1)(q-1)$. Hence computing the trapdoor $d = e^{-1} \bmod \phi(N)$ becomes easy.

CPA encryption from RSA

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 $\mathsf{hc} : X \to \{0, 1\}$ of F is an efficiently computatable function such that

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

Assume we have (F, hc) being a TDP and a hard-core predicate of F[10]. We propose the following PubKE scheme for single-bit messages.

[10] For any TDP, there exists a related TDP for which there is a hard-care predicate. As a concrete instantiation, the function of the least significant bit $\mathsf{lsb}(x)$ is a hard-core predicate for $F^{\mathsf{RSA}}_{pk}(x)$, assuming F^{RSA}_{pk} is one-way.

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 $c = (c_1, c_2)$, $m = D_{sk}(c) := c_2 \oplus \mathsf{hc}(I_{sk}(c_1))$.

Theorem 4 (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 $\mathsf{hc}(r)$ from $F_{pk}(r)$, which is not feasible since hc is a hard-core predicate of F. $\square$