CIS 5560

Cryptography
Lecture 16

Course website:

pratyushmishra.com/classes/cis-5560-s24/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Recap of Last Lecture(s)

- Public Key Encryption
  - Definition of IND-CPA
- ElGamal Encryption
  - Version with message space = $\mathbb{G}$
  - Version with arbitrary message space

Today’s Lecture

- Public Key Encryption from **Trapdoor OWFs**
  - RSA Encryption
    - Arithmetic modulo composites
    - Factoring

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

- Gen(): randomized alg. outputs a key pair (pk, sk)
- Enc(pk, m): randomized alg. that takes $m \in \mathscr{M}$ and outputs $c \in \mathscr{C}$
- Dec(sk, c): deterministic alg. that takes $c \in \mathscr{C}$ and outputs $m \in \mathscr{M} \cup \{ \perp \}$

Correctness: $\forall$(pk, sk) output by Gen(), $\forall m \in \mathscr{M}$, Dec(sk, Enc(pk, m)) $= m$

Security: IND-CPA for PKE

For all PPT adversaries $\mathscr{A}$, the following holds:

$$\Pr\left[b = \mathscr{A}(\mathsf{Enc}(\mathsf{pk}, m_b)) \;\middle|\; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n) \\ \text{Sample } b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathscr{A}(\mathsf{pk}) \end{array} \right] \leq \mathsf{negl}(n)$$

Construction of PKE:
Trapdoor Functions

Trapdoor functions (TDF)

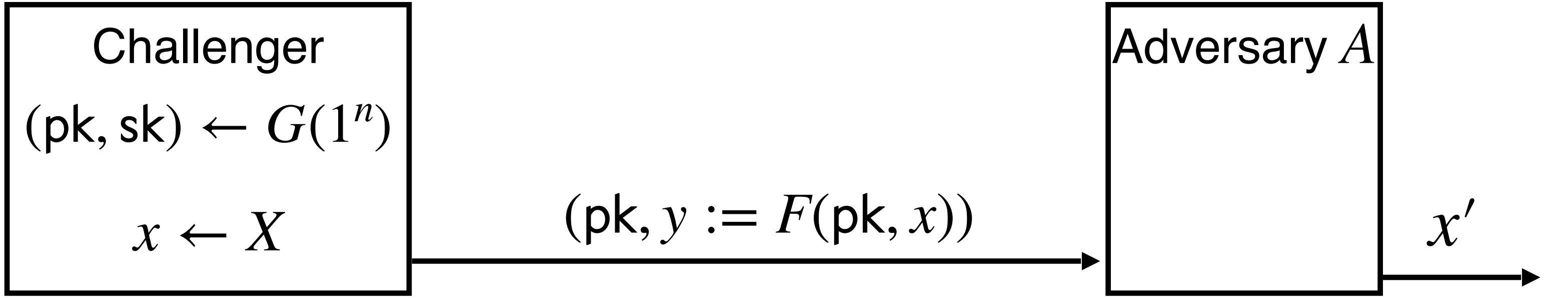
<u>Def</u>: A **trapdoor function** for input space X and output space Y is a triple of efficient algorithms (G, F, F^{-1})

- $G(1^n)$: randomized algorithm that outputs a key pair $(\mathsf{pk}, \mathsf{sk})$
- $F(\mathsf{pk}, \cdot\,)$: deterministic algorithm that computes $f : X \to Y$
- $F^{-1}(\mathsf{sk}, \cdot\,)$: defines a function $Y \to X$ that inverts $F(\mathsf{pk}, \cdot\,)$

More precisely: $\forall (\mathsf{pk}, \mathsf{sk}) \leftarrow G(1^n),\ \forall x \in X,\quad F^{-1}(\mathsf{sk}, F(\mathsf{pk}, x)) = x$

Secure Trapdoor Functions (TDFs)

A TDF (G, F, F^{-1}) is secure if F_{pk} is a one-way function:



Def: (G, F, F^{-1}) is a **secure TDF** if for all efficient A:

$$\Pr\left[F(\mathsf{pk}, x) = F(\mathsf{pk}, x') \;\middle|\; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow G(1^n) \\ x \leftarrow X \\ x' \leftarrow A(\mathsf{pk}, F(\mathsf{pk}, x)) \end{array} \right] = \mathsf{negl}(n)$$

Construction: PKE from TDFs

PKE from Secure TDFs: Attempt 1

- (G, F, F^{-1}): secure TDF $X \to Y$

Gen(1^n):	Enc(pk, m):	Dec(sk, c):
1. Output (pk, sk) $\leftarrow G(1^n)$.	1. Output $c \leftarrow F(\text{pk}, m)$.	1. Output $m := F^{-1}(\text{sk}, c)$.

Q: Is this secure?

A: No! Entirely deterministic → cannot achieve IND-CPA!

PKE from Secure TDFs

- (G, F, F^{-1}): secure TDF $\quad X \to Y$
- $(\mathsf{Gen}, \mathsf{Enc}_s, \mathsf{Dec}_\mathsf{s})$: symmetric AE defined over $(\mathscr{K}, \mathscr{M}, \mathscr{C})$
- $H : X \to \mathscr{K}$: a hash function (like the one in Hashed ElGamal)

$\mathsf{Gen}(1^n)$:

1. Output $(\mathsf{pk}, \mathsf{sk}) \leftarrow G(1^n)$.

$\mathsf{Enc}(\mathsf{pk}, m)$:

1. Sample $x \leftarrow X$.
2. Compute key $k \leftarrow H(x)$
3. Output $(y \leftarrow F(\mathsf{pk}, x), c \leftarrow \mathsf{Enc}_s(k, m))$

$\mathsf{Dec}(\mathsf{sk}, (y, c))$:

1. Compute $x := F^{-1}(\mathsf{sk}, y)$.
2. Compute key $k \leftarrow H(x)$
3. Output $\mathsf{Dec}_s(k, c)$

In pictures:

$$\underbrace{\boxed{F(\mathsf{pk}, x)}}_{\text{header}}\underbrace{\boxed{\qquad\qquad\mathsf{Enc}_s(H(x), m)\qquad\qquad}}_{\text{body}}$$

<u>Security Theorem</u>:

If (G, F, F^{-1}) is a secure TDF,

$(\mathsf{Gen}, \mathsf{Enc}_s, \mathsf{Dec}_s)$ is an AE scheme, and

$H : X \to \mathscr{K}$ is a "random oracle"

then $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure.

Review: Arithmetic modulo composites

Review: arithmetic mod composites

Let $N = pq$ where p, q are prime

$$\mathbb{Z}_N = \{0,1,2,\ldots,N-1\}; \quad \mathbb{Z}_n^* = \{\text{ invertible elements in } \mathbb{Z}_N\}$$

<u>Facts</u>:

- $x \in \mathbb{Z}_N$ is invertible $\Leftrightarrow$ $\gcd(x, N) = 1$
- Number of elements in $\mathbb{Z}_N^*$ is $\varphi(N) = (p-1)(q-1) = N - p - q + 1$

<u>Euler's thm</u>: $\forall x \in \mathbb{Z}_N^* : x^{\varphi(N)} = 1$

Modular e-th roots

We know how to solve modular **linear** equations:

$ax + b = 0$ in $\mathbb{Z}_N$ Solution: $x = -b \cdot a^{-1}$ in $\mathbb{Z}_N$

(inverses are fast even for N composite)

What about higher degree polynomials?

Example: Let $N = pq$ for two primes p, q.

Given an arbitrary $y \in \mathbb{Z}_N$, can we find x such that

$y = x^e \mod N$?

Answering these questions requires the factorization of N

(as far as we know)

The factoring problem

Gauss (1805): *"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic."*

Best known alg. (NFS): run time $2^{O(\sqrt[3]{n})}$ for n-bit integer

Current world record: **RSA-768** (232 digits)

- Work: two years on hundreds of machines
- Factoring a 1024-bit integer: about 1000 times harder

⇒ likely possible this decade

Key lengths

Security of public key system should be comparable to security of symmetric cipher:

Cipher key-size	RSA Modulus size
80 bits	1024 bits
128 bits	3072 bits
256 bits (AES)	**15360** bits

Construction of Trapdoor Functions

Big question:
can we use hardness of computing e-th roots to construct a secure TDF?

Secure TDFs from e-th roots

Gen(1^n):

1. Sample primes $p, q \sim 1024$ bits
2. Set $N = pq$
3. Sample e, d s.t. $e = d^{-1} \mod \varphi(N)$
4. Set sk $= (p, q, d)$ and pk $:= (N, e)$
5. Output (pk, sk).

$F(\mathsf{pk} = (N, e), x)$:

1. Output $x^e \mod N$.

Dec(sk $= (p, q, d), y)$:

1. Output $x := y^d \mod N$.

Correctness: $\forall (\mathsf{pk}, \mathsf{sk}) \leftarrow G(1^n)$, $\forall x \in X$, $F^{-1}(\mathsf{sk}, F(\mathsf{pk}, x)) = x$?

$$F_{\mathsf{sk}}^{-1}(F_{\mathsf{pk}}(x)) = (x^e)^d \equiv x^{1 \mod \varphi(N)} \equiv x^{1 + k\varphi(N)} \equiv x \mod N$$

This is called the RSA Trapdoor permutation

First published: Scientific American, Aug. 1977.

Very widely used:

- SSL/TLS: certificates and key-exchange
- Secure e-mail and file systems

… many others

Secure TDFs from e-th roots

Gen(1^n):

1. Sample primes $p, q \sim 1024$ bits
2. Set $N = pq$
3. Sample e, d s.t. $e = d^{-1} \mod \varphi(N)$
4. Set sk $= (p, q, d)$ and pk $:= (N, e)$
5. Output (pk, sk).

$F($pk $= (N, e), x)$:

1. Output $x^e \mod N$.

Dec(sk $= (p, q, d), y)$:

1. Output $x := y^d \mod N$.

Security?

By "assumption"

RSA assumption: Roughly, computing e-th roots is hard

$$\Pr\left[A(\mathsf{pk}, x^e \mod N) = x \;\middle|\; \begin{array}{r}(\mathsf{pk} = (N, d), \mathsf{sk} = (p, q, e)) \leftarrow G(1^n) \\ x \leftarrow X\end{array}\right] = \mathsf{negl}(n)$$

The RSA TDF is actually a trapdoor *permutation*

Is the RSA assumption plausible?

To invert the RSA one-way func. (without d) attacker must compute

$$x \quad \text{from} \quad c = x^e \mod N.$$

How hard is computing e-th roots modulo N ??

Best known algorithm:

- Step 1: factor N (hard)
- Step 2: compute e-th roots modulo p and q (easy)

Shortcuts?

Must one factor N in order to compute e-th roots?

To prove no shortcut exists we needa reduction:

- Efficient algorithm for e-th roots mod N

  $\Rightarrow$ efficient algorithm for factoring N.
- Oldest problem in public key cryptography.

Some evidence no reduction exists: (BV'98)

- "Algebraic" reduction $\Rightarrow$ factoring is easy.

Textbook RSA is insecure

Textbook RSA encryption:

- public key: **(N,e)** Encrypt: $\mathbf{c} \longleftarrow \mathbf{m^e}$ (in Z_N)
- secret key: **(N,d)** Decrypt: $\mathbf{c^d} \longrightarrow \mathbf{m}$

Insecure cryptosystem !!

- Is not semantically secure and many attacks exist

⇒ The RSA trapdoor permutation is not an encryption scheme !

RSA in practice

How **not** to improve RSA's performance

To speed up RSA decryption use small private key d ($d \approx 2^{128}$)

$$c^d = m \pmod{N}$$

Wiener'87: if $d < N^{0.25}$ then RSA is insecure.

BD'98: if $d < N^{0.292}$ then RSA is insecure (open: $d < N^{0.5}$)

<u>Insecure:</u> priv. key d can be found from (N,e)

RSA With Low public exponent

To speed up RSA encryption use a small e: $c = m^e \pmod{N}$

- Minimum value: **$e=3$** ($\gcd(e, \varphi(N)) = 1$)
- Recommended value: **$e=65537=2^{16}+1$**

Encryption: 17 multiplications

<u>Asymmetry of RSA:</u> fast enc. / slow dec.

- ElGamal (next module): approx. same time for both.

Further reading

- A Computational Introduction to Number Theory and Algebra,
  V. Shoup, 2008 (V2), Chapter 1-4, 11, 12

  Available at **//shoup.net/ntb/ntb-v2.pdf**