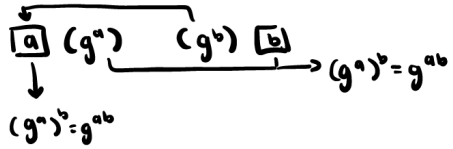
Lesson 10 Lecture Notes

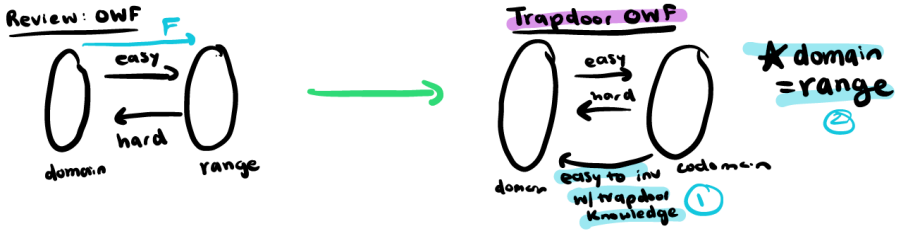
CS555

1 Recap

1.1 Diffie-Hellman Key Exchange Construction



1.2 One-Way Functions (OWF) vs Trapdoor OWF



1.3 Trapdoor Function Definition

A function family $F = \{F_n\}_{n\in\mathbb{N}}$ with F_n itself a collection of functions $F_n = \{F_i : \{0,1\}^n \to \{0,1\}^{m(n)}\}_{i\in I_n}$ is a trapdoor OWF if it is:

1. Easy to sample function index with a trapdoor.
2. Easy to compute $F_i(x)$ given i and x.
3. Easy to compute inverse of $F_i(x)$ given trapdoor.
4. One-way for every PPT adversary A when all but the trapdoor is revealed.

2 Trapdoor Permutation to IND-Secure Public-Key Encryption

2.1 Construction Attempt 1

- Gen: Random index i with trapdoor t_i, release i.
- Enc: Output $c = F_i(m)$.
- Dec: Output $F_i^{-1}(c)$ using trapdoor t_i.

Issue: Having an OWF does not guarantee all bits are secret. Some input bits may not be hardcore, breaking IND-CPA security (you could use that bit to distinguish between input messages)

2.2 Construction Attempt 2

- Gen: Same as above.
- Enc: Pick random r. Output $c = (F_i(r), \text{HCB}(r) \oplus m)$. [Note, this means you can only encrypt one bit here]
- Dec: Recover r from $F_i(r)$ using trapdoor $\to$ compute $\text{HCB}(r)$, XOR with m.

This achieves IND-CPA security via hybrid argument.

3 Trapdoor Permutation Candidates

- RSA
- Rabin / Blum-Williams

3.1 RSA Review

- $N = pq$, product of two large primes.
- $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N \mid \gcd(a, N) = 1\}$ is a group under multiplication mod N.
- Inverses exist and are easy to compute.
- $\varphi(N) = (p-1)(q-1) = \text{ord}(\mathbb{Z}_N^*)$.
- Given (p, q), factoring N is easy. Without, factoring is hard.

3.2 RSA Trapdoor construction

Let $e \in \mathbb{Z}$ with $\gcd(e, \phi(N)) = 1$. Then you have the following trapdoor permutation

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

with trapdoor d such that $ed \equiv 1 \pmod{\phi(N)}$.

- Proof: $(x^e)^d \equiv x^{ed} \equiv x^{k\phi(n)+1} \equiv 1 \cdot x \equiv x \pmod{n}$
- given N, e and $x^e \pmod{N}$, it is hard to compute x
- if factoring is easy, RSA is broken

3.3 Chinese Remainder Theorem and RSA Inversion

- **CRT**

$$\begin{aligned} x &\equiv r_p r_q \pmod{pq} \\ x \equiv r_p \pmod{p}, &\quad x \equiv r_q \pmod{q} \\ x = k_p p + r_p \pmod{q}, &\quad x = k_q q + r_q \pmod{p} \end{aligned}$$

- **Finding Modular Inverse of e:** We want to find d such that

$$ed \equiv 1 \pmod{\phi(n)}.$$

Let:

$$\phi(n) = (p-1)(q-1)$$

Then:

$$\begin{aligned} e \equiv r_{ep} \pmod{p-1}, &\quad e \equiv r_{eq} \pmod{q-1} \\ d \equiv r_{ap} \pmod{p-1}, &\quad e \equiv r_{dp} \pmod{q-1} \end{aligned}$$

So:

$$ed \equiv r_{ep} \cdot r_{dp} \equiv 1 \pmod{p-1} \quad e \equiv r_{eq} \cdot r_{dp} \pmod{q-1}$$

  - To reverse, you only need to do one of these. We know finding the inverse with a prime modulus is easy → break down until you hit a prime modulus

3.4 Hardcore Bits for RSA

- Goldreich-Levin bit: $GL(r; r') = \langle r, r' \rangle \bmod 2$.
- Least significant bit (LSB).
- Most significant bit (MSB): $\text{HALF}_N(r) = 1$ iff $r < \frac{N}{2}$
  - here any bit of r is hardcore

4 RSA Encryption

- $\text{Gen}(1^n)$: $N = pq$, choose (e, d) with $ed \equiv 1 \pmod{\phi(N)}$. Public key (N, e), secret key d.
- $\text{Enc}(pk, b)$: For bit b, pick random $r \in \mathbb{Z}_n^*$, output $c = (r^e \pmod{N}, \text{LSB}(r) \oplus b)$.
- $\text{Dec}(sk, c)$: Recover r via RSA inversion, compute $\text{HCB}(r)$, XOR with ciphertext.

Note: Impractical since only one bit encrypted at a time.
Theoretical Solution:

- Goldreich-Levin (GL): extract one hardcore bit via inner product

$$\mathrm{GL}(r', r) = \langle r', r \rangle \bmod 2$$

- Alternative: use a random matrix $A \in \{0,1\}^{d \times n}$ with rows $r_1, r_2, \ldots, r_d$

$$Ar \Rightarrow \text{gives } d \text{ hardcore bits}$$

- Note: always generating random A is costly in practice.

Practical Solution:

- Use a function f that encodes $\mathrm{LSB}(r)$ into many bits.
- Encryption: compute $f(\mathrm{LSB}(r)) \oplus m$

5 Quadratic Residues Modulo p

- For prime p, exactly half of $\mathbb{Z}_p^*$ are squares.
- Legendre symbol:

$$\left(\frac{x}{p}\right) = \begin{cases} 1 & \text{if } x \text{ is a square mod } p \\ -1 & \text{if } x \text{ is not a square mod } p \\ 0 & \text{if } x \equiv 0 \pmod{p} \end{cases}$$

- We have

$$\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} \pmod{p}$$

- Easy to compute square roots mod p
  - when $p \equiv 3 \pmod 4$:

$$\sqrt{x} \equiv \pm x^{\frac{p+1}{4}} \pmod{p}$$

  - Proof

$$\left(\pm x^{\frac{p+1}{4}}\right)^2 \equiv x^{\frac{p+1}{2}} = x \cdot x^{\frac{p-1}{2}} \equiv x \pmod{p}$$

6 Quadratic Residues Modulo $N = pq$

- x is a square mod N iff x is square mod p and mod q.
- Jacobi symbol:

$$\left(\frac{x}{N}\right) = \left(\frac{x}{p}\right)\left(\frac{x}{q}\right)$$

  - Jacobi is $+1$ if both are squares OR if both are non-square
  - We can find $\left(\frac{x}{N}\right)$ in polynomial time without knowing (p, q)
  - Given N no PPT can distinguish quadratic residues mod N and quadratic non-residues mod N without knowing p, q.
  - Suppose we know primes p and q. We can find the square root y of x by first having

$$x \equiv y_p^2 \pmod{p}, \quad x \equiv y_q^2 \pmod{q}$$

    Then we can reconstruct the square root modulo N as:

$$y = C_p y_p + C_q y_q$$

    where:

$$C_p \equiv 1 \pmod{p}, \quad C_p \equiv 0 \pmod{q}$$

    OR

$$C_q \equiv 0 \pmod{p}, \quad C_q \equiv 1 \pmod{q}$$

7 Factoring via Square Roots

If an algorithm can compute square roots mod N, then we can feed it $x = z^2 \pmod N$ for a random $z \in \mathbb{Z}_N^*$. With probability $1/2$ we can factor N by computing $\gcd(z - y, N)$.