COMP 8920: Cryptography	Winter 2025

Lecture 24 — April 3, 2025

Prof. Curtis Bright — *Scribe: Mahzabin Chowdhury*

Semantically Secure RSA

We make RSA semantically secure by introducing randomness into the cryptosystem, adding a random oracle $G : \mathbb{Z}_2^k \to \mathbb{Z}_2^m$ into the public key. Let $\mathcal{P} = \mathbb{Z}_2^m$, $\mathcal{C} = \mathbb{Z}_2^k \times \mathbb{Z}_2^m$, and define

$$e_k(x) = (r^b \bmod n, G(r) \oplus x) \tag{1}$$

where $(y_1, y_2) \in \mathbb{Z}_2^k \times \mathbb{Z}_2^m$ for random $r \in \mathbb{Z}_2^k$ and

$$d_k((y_1, y_2)) = G(y_1^a \bmod n) \oplus y_2 \tag{2}$$

This works, since $d_k(y_1, y_2)$ equals

$$\begin{aligned} G((r^b \bmod n)^a \bmod n) \oplus y_2 &= G(r^{ab} \bmod n) \oplus G(r) \oplus x && (3)\\ &= G(r) \oplus G(r) \oplus x && (4)\\ &= x && (5) \end{aligned}$$

since $r^{ab} = r$.

An informal argument why this is semantically secure (i.e., the distinguishing problem can't be solved with probability more than $\frac{1}{2}$) is that in order to determine any information about x we must determine the mask $G(r)$. Any partial information about r is useless because G is a random oracle; the only way to compute $G(r)$ is to determine r. Under the assumption that RSA is secure, this augmented cryptosystem is semantically secure. The main drawback is data expansion: m bits of plaintext expand to $m + k$ bits of ciphertext.

The Discrete Log Problem

Say G is a group, $\alpha \in G$ of order n, and define $\langle \alpha \rangle = \{\alpha^i : 0 \le i \le n-1\}$ to be the cyclic group generated by α. For instance $G = \mathbb{Z}_p^*$ where p is prime, and α is a primitive element of $\mathbb{Z}_p^*$, i.e., $\langle \alpha \rangle = \mathbb{Z}_p^*$.

The discrete log problem is: given $\beta \in \langle \alpha \rangle$ to determine the value of i for which $\beta = \alpha^i$, i.e., compute $i = \log_\alpha(\beta)$, the discrete log of β base α.

Example: take $p = 2579$ and $\alpha = 2$, a primitive element in $\mathbb{Z}_p^*$. What is $\log_2(949)$ in $\mathbb{Z}_p^*$?

In contrast to logs over the reals, computing logs in $\mathbb{Z}_p^*$ seems difficult in general. The naive strategy would be to compute $2^2, 2^3, 2^4, \ldots, 2^{p-2} \pmod p$ until 949 is reached. In the worst case, this uses at most p evaluations of $\alpha \bmod p$. Since each multiplication mod$p$ is $O((\log p)^2)$ bit operations, this uses $O(p(\log p)^2)$ bit operations, which is $O(2^{\log p}(\log p)^2)$. In contrast to logarithms over the reals, computing discrete logs in $\mathbb{Z}_p^*$ is generally difficult.

The naive strategy would be to compute

$$2^2, 2^3, 2^4, \ldots, 2^{p-2} \mod p$$

until 949 is reached. In the worst case, this requires at most p evaluations of powers modulo p.

Each multiplication modulo p takes $O((\log p)^2)$ bit operations, so the total cost is:

$$O(p(\log p)^2) = O(2^{\log_2 p} \cdot (\log p)^2),$$

which is exponential time in $\log p$.

ElGamal Cryptosystem

The ElGamal cryptosystem is based on the difficulty of the discrete logarithm problem.

Suppose:

- p is a prime
- α is a primitive element of $\mathbb{Z}_p^*$
- Let $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$
- Let the keyspace be $\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \mod p\}$.

Public key: (p, α, β)

Private key: $a = \log_\alpha \beta$

Encryption

To encrypt a message x, choose a random $k \in \mathbb{Z}_{p-1}$ and compute:

$$\text{Enc}_k(x) = (\alpha^k \mod p,\ x \cdot \beta^k \mod p)$$

Let:

$$(y_1, y_2) = (\alpha^k,\ x \cdot \beta^k) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$$

Decryption

To decrypt (y_1, y_2), compute:

$$x = y_2 \cdot (y_1^a)^{-1} \mod p$$

The encryption "masks" x by multiplying it with β^k, a random-looking element. Eve knows β, but not k, and would need to solve:

$$k = \log_\alpha(\alpha^k)$$

which is presumed hard.

However, Bob can compute β^k without knowing k:

$$(\alpha^k)^a \equiv \alpha^{ak} \equiv (\alpha^a)^k \equiv \beta^k \mod p$$

Once β^k is computed, its inverse modulo p, $(\beta^k)^{-1}$, is easy to find using the Euclidean algorithm.

Eve would need to compute $a = \log_\alpha \beta$, which is presumed to be a hard discrete log problem.

Example

Let:

$$p = 2579, \quad \alpha = 2, \quad \beta = 949$$

Alice wants to send message $x = 1299$. She picks a random $k = 853$ and computes:

$$y_1 = 2^{853} \mod 2579 = 435$$

$$\beta^k = 949^{853} \mod 2579 = 2396$$

$$y_2 = 1299 \cdot 2396 \mod 2579 = 2396$$

So the ciphertext is:

$$(y_1, y_2) = (435, 2396)$$

Bob's private key is $a = 765$. He computes:

$$x = 2396 \cdot (435^{765})^{-1} \mod 2579$$

$$435^{765} \mod 2579 = 2424, \quad \text{and } 2424^{-1} \mod 2579 = 1980$$

$$x = 2396 \cdot 1980 \mod 2579 = 1299$$

Security Consideration

To be secure, p should have at least 2048 bits, and $p-1$ should have at least one large prime factor.

A common approach is to choose p of the form:

$$p = 2q + 1$$

where q is also prime. Such primes are called **safe primes**.

It is conjectured that there are infinitely many safe primes, and the number of safe primes in the interval $[1, n]$ is approximately:

$$\frac{1.32}{(\ln n)^2}$$

Thus, if p is 2048 bits long, you might need to try about 1.5 million candidate values before finding a safe prime.