CS555 Cryptography Notes

Lecture 9: September 23, 2025

1 Discrete Log Assumption

$\mathbb{Z}_N^*$ is a group under multiplication where $|\mathbb{Z}_N^*| = \phi(N)$ if $\gcd(a, N) = 1$.

Order of $\mathbb{Z}_N^*$ = Euler's totient function $\phi(N)$. **Traditional Exponentiation:** $a \to a^2 \to a^3 \to \ldots \to a^n$: $O(n)$.
Required: $a \to a^2 \to a^4 \to \ldots \to a^{2^n}$: $O(\log n)$.

Discrete Log Problem: Given a generator g and $h \in \mathbb{Z}_p^*$, find $x \in \mathbb{Z}_{p-1}$ such that $h = g^x \bmod p$.

1.1 Questions

1. Is DLP hard for random p? Could it be easy for some p?
2. Given g: is the problem hard for all g?
3. Given p and g: is the problem hard for all h?

Theorem 1. *If there is a ppt algorithm A such that*

$$\Pr[A(p, g, g^x \bmod p) = x] > 1/\mathit{poly}(\log p)$$

for random generator g of $\mathbb{Z}_p^*$ and random p, then there is a ppt algorithm that solves DLP.

Given $g \in \mathbb{Z}_{p-1}$, then there is a ppt algorithm B such that $B(p, g, g^x \bmod p) = x$ for g, x. That is, worst case is the same as average case. Implies that answer for Q2 and Q3 is the same as all in replaced by random (minimal method of study exponent: $O(p)$ through all exponents).

1.2 Baby Step - Giant Step Algorithm

Input: p, g, h where $h \in \{1, 2, \ldots, p-1\}$
$w = \lceil \sqrt{p} \rceil$
$x \equiv k_2 + iy$ where $i \in [0, \sqrt{p} - 1]$, $k \in [0, \sqrt{p}]$ and $x \in [0, p-1]$

Compute: $g, g^2, \ldots, g^{\sqrt{p}}$ for $O(\sqrt{p}\log p)$. We will find g^k, we compute $g^k, g^{2k}, g^{3k}, \ldots, g^{\sqrt{p}k}$ in for each one, we find the inverse (which is easy to do) and check if it is in the set $\{g, g^2, \ldots, g^{\sqrt{p}}\}$. Once we did such k and i, we find the decomposition.

Computational DH Assumption:

For $x, y \in \mathbb{Z}_{p-1} : A(p, g, g^x, g^y) = g^{xy}$

We need to select "a safe" prime: A prime p is called a Sophie-Germain prime if $p = 2q + 1$ is also prime. In this case, p is called a safe prime.

Safe primes are maximally hard for Pohlig-Hellman algorithm, which runs based on the Chinese Remainder Theorem.

It is unknown that there exist sufficiently dense number of safe primes. From simulation, it seems that there are c/n^2 safe primes of constant c.

$$F_n = \sum_{n,p,g} \text{ s.t. } F_{n,p,g}(x) = (p, g, g^x \bmod p)$$

is a one-way permutation family: n is the input length (bits).

1.3 Diffie-Hellman / El Gamal Encryption

Gen(1^n)**:** Generate an n-bit prime p and generator g of $\mathbb{Z}_p^*$. Choose a random number $x \in \mathbb{Z}_{p-1}$. Let $pk = (p, g, g^x)$ and let $sk = x$.

Enc(pk, m) where $m \in \mathbb{Z}_p^*$: Generate random $y \in \mathbb{Z}_{p-1}$ and output $(g^y, g^{xy} \cdot m)$.

Dec$(sk = x, c)$**:** Compute g^{xy} using g^x and g^y and divide 2^{nd} component to retrieve m.
This is just the high-level idea. How to implement it and make it IND-CPA secure?

Sufficient and necessary condition to test if g is a generator: Let $\Sigma_1, \ldots, \Sigma_k$ be the prime factors of $p-1$. Then g is a generator of $\mathbb{Z}_p^*$ if and only if $g^{(p-1)/\Sigma_i} \neq 1 \bmod p$ for all i.

OPEN: Test if g is a generator without prime factorization of $p - 1$.

OPEN: Deterministically come up with g.

Answer to the 2nd question: This isn't IND-CPA It leaks some information: