# Algorithmic Number Theory and Public-key Cryptography

## Discrete-log based cryptography

Jean-Sébastien Coron

University of Luxembourg

April 20, 2025

# Summary

- Algorithmic number theory.
  - Generators of $\mathbb{Z}_p$
  - The discrete-log problem
- Discrete-log based cryptosystems
  - Diffie-Hellmann key exchange
  - ElGamal encryption: security proof
  - Schnorr signature scheme

# Groups

- Definitions
  - A group $G$ is *finite* if $|G|$ is finite. The number of elements in a finite group is called its *order*.
  - A group $G$ is *cyclic* if there is an element $g \in G$ such that for each $h \in G$ there is an integer $i$ such that $h = g^i$. Such an element $g$ is called a generator of $G$.
  - Let $G$ be a finite group and $a \in G$. The *order* of $a$ is defined to be the least positive integer $t$ such that $a^t = 1$.
- Facts
  - Let $G$ be finite group and $a \in G$. The order of $a$ divides the order of $G$.
  - Let $G$ be a cyclic group of order $n$ and $d|n$, then $G$ has exactly $\phi(d)$ elements of order $d$. In particular, $G$ has $\phi(n)$ generators.

# The multiplicative group $\mathbb{Z}_p^*$

- Let $p$ be a prime integer.
    - The set $\mathbb{Z}_p^*$ is the set of integers modulo $p$ which are invertible modulo $p$.
    - The set $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$ for the operation of multiplication modulo $p$.
- Generators of $\mathbb{Z}_p^*$ :
    - There exists $g \in \mathbb{Z}_p^*$ such that any $h \in \mathbb{Z}_p^*$ can be uniquely written as $h = g^x \mod p$ with $0 \leq x < p-1$.
    - The integer $x$ is called the *discrete logarithm* of $h$ to the base $g$, and denoted $\log_g h$.

# Finding a generator of $\mathbb{Z}_p^*$

- Finding a generator of $\mathbb{Z}_p^*$ for prime $p$.
  - The factorization of $p-1$ is needed. Otherwise, no efficient algorithm is known.
  - Factoring is hard, but it is possible to generate $p$ such that the factorization of $p-1$ is known.
- Generator of $\mathbb{Z}_p^*$
  - $g \in \mathbb{Z}_p^*$ is a generator of $\mathbb{Z}_p^*$ if and only if $g^{(p-1)/q} \neq 1 \mod p$ for each prime factor $q$ of $p-1$.
  - There are $\phi(p-1)$ generators of $\mathbb{Z}_p^*$

# Finding a generator

- Let $q_1, \ldots q_r$ be the prime factors of $p-1$
  - 1) Generate a random $g \in \mathbb{Z}_p^*$
  - 2) For $i = 1$ to $r$ do
    - Compute $\alpha_i = g^{(p-1)/q_i} \mod p$
    - If $\alpha_i = 1 \mod p$, go back to step 1.
  - 3) Output $g$ as a generator of $\mathbb{Z}_p^*$
- Complexity:
  - There are $\phi(p-1)$ generators of $\mathbb{Z}_p^*$.
  - A random $g \in \mathbb{Z}_p^*$ is a generator with probability $\phi(p-1)/(p-1)$.
  - If $p - 1 = 2 \cdot q$ for prime $q$, then $\phi(p-1) = q - 1$ and this probability is $\simeq 1/2$.

# Safe prime $p$

- Safe prime $p$: both $p$ and $q = (p-1)/2$ are primes.
  - Generate a random prime $p$.
  - Test if $q = (p-1)/2$ is prime. Otherwise, generate another $p$.
- Finding a generator $g$ for $\mathbb{Z}_p^*$
  - Generate a random $g \in \mathbb{Z}_p^*$ with $g \neq \pm 1$
  - Check that $g^q \neq 1 \mod p$. Otherwise, generate another $g$.
  - Complexity: there are $\phi(p-1) = q-1$ generators, therefore $g$ is a generator with probability $\simeq 1/2$.
- Finding a generator $g$ of the subgroup $G$ of order $q$.
  - Generate a random $h \in \mathbb{Z}_p^*$. Let $g = h^2$. Then $g$ must be of order $q$. If $g \neq 1$, then $g$ is a generator of $G$.

# Subgroup of $\mathbb{Z}_p^*$

- We want to work in a prime-order subgroup of $\mathbb{Z}_p^*$
  - Generate $p, q$ such that $p - 1 = 2 \cdot q$ and $p, q$ are prime
  - Find a generator $g$ of $\mathbb{Z}_p^*$
  - Then $g' = g^2 \mod p$ is a generator of a subgroup $G$ of $\mathbb{Z}_p^*$ of prime order $q$.

# Discrete logarithm

- Let $g$ be a generator of $\mathbb{Z}_p^*$
  - For all $a \in \mathbb{Z}_p^*$, $a$ can be written uniquely as $a = g^x \bmod p$ for $0 \le x < p - 1$.
  - The integer $x$ is called the *discrete logarithm* of $a$ to the base $g$, and denoted $\log_g a$.
- Computing discrete logarithms in $\mathbb{Z}_p^*$
  - Hard problem: no efficient algorithm is known for large $p$.
  - Brute force: enumerate all possible $x$. Complexity $\mathcal{O}(p)$.
  - Baby step/giant step method: complexity $\mathcal{O}(\sqrt{p})$.

# Baby step/giant step method

- Given $a = g^x \bmod p$ where $0 \leq x < p - 1$, we wish to compute $x$.
- Let $m = \lfloor \sqrt{p} \rfloor$. Build a table:

$$L = \{\, (g^i \bmod p, i) \,|\, 0 \leq i < m \}$$

  and sort $L$ according to the first component $g^i \bmod p$.
  - Size: $\mathcal{O}(\sqrt{p} \log p)$. Time: $\mathcal{O}(\sqrt{p} \log^2 p)$.
- Compute the sequence of values $a \cdot g^{-j \cdot m} \bmod p$, until a collision with $g^i$ is found in the table $L$, which gives:

$$a \cdot g^{-j \cdot m} = g^i \bmod p \Rightarrow a = g^{j \cdot m + i} \bmod p \Rightarrow x = j \cdot m + i$$

- Time: $\mathcal{O}(\sqrt{p} \log^2 p)$. Memory: $\mathcal{O}(\sqrt{p} \log p)$

# Discrete Logarithms in groups of order $q^e$

- Let $p$ be a prime and $g$ a generator of a subgroup of $\mathbb{Z}_p^*$ of order $q^e$ for some $q$, where $e > 1$.
- Given $a = g^x \bmod p$ for $0 \le x < q^e$, we wish to compute $x$.
- We write $x = u \cdot q + v$ where $0 \le v < q$ and $0 \le u < q^{e-1}$
  - $a^{q^{e-1}} = \left(g^{q^{e-1}}\right)^x = \left(g^{q^{e-1}}\right)^v \bmod p$
  - We compute $v$ by using the previous method in the subgroup of order $q$ generated by $g^{q^{e-1}}$
- $a \cdot g^{-v} = (g^q)^u$ so we compute $u$ recursively, in the subgroup of order $q^{e-1}$ generated by $g^q$.
- Time complexity $\mathcal{O}(e \cdot \sqrt{q} \cdot \log^2 p)$

# Discrete Logarithms in $\mathbb{Z}_p^*$

- Let $p$ be a prime and we know the factorization

$$p - 1 = \prod_{i=1}^{r} q_i^{e_i}$$

- Given $a = g^x \bmod p$ for $0 \le x < p - 1$ where $g$ is a generator of $\mathbb{Z}_p^*$, we wish to compute $x$.
- For $1 \le i \le r$ we have:

$$a^{(p-1)/q_i^{e_i}} = \left(g^{(p-1)/q_i^{e_i}}\right)^x = \left(g^{(p-1)/q_i^{e_i}}\right)^{x \bmod q_i^{e_i}} \bmod p$$

- We compute $x_i = x \bmod q_i^{e_i}$ for all $1 \le i \le r$ by using the previous method in the subgroup generated by $g^{(p-1)/q_i^{e_i}}$
- Using CRT we find $x$ from the $x_i$'s.
- Complexity $\mathcal{O}(\sqrt{q} \cdot \log^k p)$, where $q = \max q_i$
- The hardness of computing discrete logarithms in $\mathbb{Z}_p^*$ is determined by the size of the largest prime factor of $p - 1$.
  - In general we work in a subgroup of $\mathbb{Z}_p^*$ of prime order.

# Diffie-Hellman protocol

- Enables Alice and Bob to establish a shared secret key that nobody else can compute, without having talked to each other before.
- Key generation
  - Let $p$ a prime integer, and let $g$ be a generator of $\mathbb{Z}_p^*$. $p$ and $g$ are public.
  - Alice generates a random $x$ and publishes $X = g^x \bmod p$. She keeps $x$ secret.
  - Bob generates a random $y$ and publishes $Y = g^y \bmod p$. He keeps $y$ secret.

# Diffie-Hellman protocol

- Key establishment
  - Alice sends $X$ to Bob. Bob sends $Y$ to Alice.
  - Alice computes $K_a = Y^x \bmod p$
  - Bob computes $K_b = X^y \bmod p$

$$K_a = Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y = K_b$$

- Alice and Bob now share the same key $K = K_a = K_b$
  - Without knowing $x$ or $y$, the adversary is unable to compute $K$.
  - Computing $g^{xy}$ from $g^x$ and $g^y$ is called the *Diffie-Hellman problem*, for which no efficient algorithm is known.
  - The best known algorithm for solving the Diffie-Hellman problem is to compute the discrete logarithm of $g^x$ or $g^y$.

# El-Gamal encryption

- Key generation
  - Let $G$ be a subgroup of $\mathbb{Z}_p^*$ of prime order $q$ and $g$ a generator of $G$.
  - Let $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$. Let $h = g^x \bmod p$.
  - Public-key : $(g, h)$. Private-key : $x$
- Encryption of $m \in G$ :
  - Let $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$
  - Output $c = (g^r, h^r \cdot m)$
- Decryption of $c = (c_1, c_2)$
  - Output $m = c_2/(c_1^x) \bmod p$

# Security of El-Gamal

- To recover $m$ from $(g^r, h^r \cdot m)$
  - One must find $h^r$ from $(g, g^r, h = g^x)$
- Computational Diffie-Hellman problem (CDH) :
  - Given $(g, g^a, g^b)$, find $g^{ab}$
  - No efficient algorithm is known.
  - Best algorithm is finding the discrete-log
- However, attacker may already have some information about the plaintext !

# Semantic security

- Indistinguishability of encryption (IND-CPA)
  - The attacker receives $pk$
  - The attacker outputs two messages $m_0, m_1$
  - The attacker receives encryption of $m_\beta$ for random bit $\beta$.
  - The attacker outputs a "guess" $\beta'$ of $\beta$
- Adversary's advantage :
  - $\mathrm{Adv} = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|$
  - A scheme is IND-CPA secure if the advantage of any computationally bounded adversary is a negligible function of the security parameter.
  - This means that the adversary's success probability is not better than flipping a coin.

# Proof of security

- Reductionist proof :
  - If there is an attacker who can break IND-CPA with non-negligible probability,
  - then we can use this attacker to solve DDH with non-negligible probability
- The Decision Diffie-Hellmann problem (DDH) :
  - Given $(g, g^a, g^b, z)$ where $z = g^{ab}$ if $\gamma = 1$ and $z \overset{R}{\leftarrow} G$ if $\gamma = 0$, where $\gamma$ is random bit, find $\gamma$.
  - $\text{Adv}_{DDH} = |\Pr[\gamma' = \gamma] - \frac{1}{2}|$
  - No efficient algorithm known when $G$ is a prime-order subgroup of $\mathbb{Z}_p^*$.

# Proof of security

- We get $(g, g^a, g^b, z)$ and must determine if $z = g^{ab}$
  - We give $pk = (g, h = g^a = g^x)$ to the adversary
  - $sk = a = x$ is unknown.
  - Adversary sends $m_0, m_1$
  - We send $c = (g^b = g^r, z \cdot m_\beta)$ for random bit $\beta$
  - Adversary outputs $\beta'$ and we output $\gamma' = 1$ (corresponding to $z = g^{ab}$) if $\beta' = \beta$ and 0 otherwise.

# Analysis

- If $\gamma = 0$, then $z$ is random in $G$
  - Adversary gets no information about $\beta$, because $m_\beta$ is perfectly masked by a random.
  - Therefore $\Pr[\beta' = \beta | \gamma = 0] = 1/2$
  - $\Pr[\gamma' = \gamma | \gamma = 0] = 1/2$
- If $\gamma = 1$, then $z = g^{ab} = g^{rx} = h^r$ where $h = g^x$.
  - $c$ is a legitimate El-Gamal ciphertext.
  - Therefore the attacker wins ($\beta' = \beta$) with probability $1/2 \pm \mathrm{Adv}_A$
  - We can take wlog $\Pr[\beta' = \beta | \gamma = 1] = 1/2 + \mathrm{Adv}_A$
  - Therefore $\Pr[\gamma' = \gamma | \gamma = 1] = 1/2 + \mathrm{Adv}_A$

- We have:
  - $\Pr[\gamma' = \gamma | \gamma = 0] = 1/2$
  - $\Pr[\gamma' = \gamma | \gamma = 1] = 1/2 + \mathsf{Adv}_A$

$$\begin{aligned}
\Pr[\gamma' = \gamma] &= \Pr[\gamma' = \gamma | \gamma = 0] \cdot \Pr[\gamma = 0] + \\
&\quad \Pr[\gamma' = \gamma | \gamma = 1] \cdot \Pr[\gamma = 1] \\
\Pr[\gamma' = \gamma] &= \frac{1}{2} \cdot \frac{1}{2} + \left(\frac{1}{2} + \mathsf{Adv}_A\right) \cdot \frac{1}{2} \\
\Pr[\gamma' = \gamma] &= \frac{1}{2} + \frac{\mathsf{Adv}_A}{2}
\end{aligned}$$

- Therefore:

$$\mathsf{Adv}_{DDH} = \left| \Pr[\gamma' = \gamma] - \frac{1}{2} \right| = \frac{\mathsf{Adv}_A}{2}$$

# Security of El-Gamal

- $\text{Adv}_{DDH} = \frac{\text{Adv}_A}{2}$
  - From an adversary running in time $t_A$ with advantage $\text{Adv}_A$, we can construct a DDH solver running in time $t_A + \mathcal{O}(k^2)$ with advantage $\frac{\text{Adv}_A}{2}$.
  - where $k$ is the security parameter.
- El-Gamal is IND-CPA under the DDH assumption
  - Conversely, if no algorithm can solve DDH in time $t$ with advantage $> \varepsilon$, no adversary can break El-Gamal in time $t - \mathcal{O}(k)$ with advantage $> 2 \cdot \varepsilon$

# Chosen-ciphertext attack

- El-Gamal is not chosen-ciphertext secure
  - Given $c = (g^r, h^r \cdot m)$ where $pk = (g, h)$
  - Ask for the decryption of $c' = (g^{r+1}, h^{r+1} \cdot m)$ and recover $m$.
- The Cramer-Shoup encryption scheme (1998)
  - Can be seen as extension of El-Gamal.
  - Chosen-ciphertext secure (IND-CCA) without random oracle.

# The Cramer-Shoup cryptosystem

- Key generation
  - Let $G$ a group of prime order $q$
  - Generate random $g_1, g_2 \in G$ and randoms $x_1, x_2, y_1, y_2, z \in \mathbb{Z}_q$
  - Let $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^z$
  - Let $H$ be a hash function
  - $pk = (g_1, g_2, c, d, h, H)$ and $sk = (x_1, x_2, y_1, y_2, z)$
- Encryption of $m \in G$
  - Generate a random $r \in \mathbb{Z}_q$
  - $C = (g_1^r,\ g_2^r,\ h^r m,\ c^r d^{r\alpha})$
  - where $\alpha = H(g_1^r, g_2^r, h^r m)$

# The Cramer-Shoup cryptosystem

- Decryption of $C = (u_1, u_2, e, v)$
  - Compute $\alpha = H(u_1, u_2, v)$ and test if :

$$u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha} = v$$

  - Output "reject" if the condition does not hold.
  - Otherwise, output :

$$m = e/(u_1)^z$$

- INC-CCA security
  - Cramer-Shoup is secure secure against adaptive chosen ciphertext attack
  - under the decisional Diffie-Hellman assumption,
  - without the random oracle model.
- Decision Diffie-Hellman problem:
  - Given $(g, g^x, g^y, z)$ where $z = g^{xy}$ if $b = 0$ and $z \leftarrow G$ if $b = 1$, where $b \leftarrow \{0, 1\}$, guess $b$.

# The Schnorr signature scheme

- Key generation:
  - Let $G$ be a group of order $q$ and let $g$ be a generator. Generate a private key $x \leftarrow \mathbb{Z}_q$
  - The public key is $y = g^x \mod p$
- Signature generation of $m$
  - Generate a random $k$ in $\mathbb{Z}_q$
  - Let $r = g^k$, $e = H(m \| r)$ and $s = (k - xe) \mod q$
  - Signature is $(s, e)$.
- Signature verification of $(s, e)$
  - Let $r_v = g^s y^e \mod p$ and $e_v = H(M \| r_v)$
  - Check that $e_v = e$.

- Security of Schnorr signatures
  - Provably secure against existential forgery in a chosen message attack
  - in the random oracle model under the discrete-log assumption
  - using the "Forking lemma" (Pointcheval and Stern, 1996)