Cryptosystems Based on Discrete Logarithmic Problem

→ **Diffie-Hellman Key Exchange:** Asymmetric Scheme used for Secure Shell, Transport Layer Security and IP Security based on the Discrete Logarithmic Problem

↳ Basic Idea: For prime p, exponentiation in $\mathbb{Z}_p^*$ is a one-way f^n and commutative $\Rightarrow$ $K \equiv (\alpha^x)^y \equiv (\alpha^y)^x \bmod p$ (joint secret)

Alice and Bob want to share secret key. ⟶ Set-up Protocol + Main Protocol

→ **Diffie-Hellman Set-up Protocol:**

1. Choose large prime p
2. Choose $\alpha \in \{2, 3, \dots, p-2\}$
3. Publish p and α — Domain Parameters.

We assume Alice and Bob possess either α or p → generate joint secret key k

→ **Diffie-Hellman Key Exchange:**

Alice		Bob
choose $a = k_{pr,A} \in \{2,3,\dots,p-2\}$		Choose $b = k_{pr,B} \in \{2,3,\dots,p-2\}$
compute $A = k_{pub,A} \equiv \alpha^a \bmod p$		Compute $B = k_{pub,B} \equiv \alpha^b \bmod p$
	$k_{pub,A} = A$ ⟶	
	⟵ $k_{pub,B} = B$	
$K_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$		$K_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$

* K_{AB} can establish secure comm. for symmetric algorithms.

Proof: Alice computes $B^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \bmod p$

Bob computes $A^b \equiv (\alpha^a)^b \equiv \alpha^{ab} \bmod p$ → They share K_{AB}

→ Similarities with RSA:
- ↳ p is a large prime ⟶ 2048+ bits.
- ↳ α is primitive
- ↳ K_{AB} has same bit length as p → use first 128 for symmetric alg. or use hash f^n
- ↳ private keys a and b computed using PRNG
- ↳ public keys A & B computed using square & multiply

→ Protocol can be generalized to groups of elliptic curves ⟶ Elgamal Encryption

→ Theorem: Set $\mathbb{Z}_n^* = \{ i \mid i \in [\mathbb{Z}_n \setminus \{0\}] \wedge \gcd(i,n) = 1 \}$ → Commutative group under $\cdot$ mod n.
↳ Identity Element is $e = 1$ ↳ a^{-1} computed using Euclidean Algorithm.

→ We care about FINITE STRUCTURES

→ Finite Group: Group $(G, \circ)$ is finite if it has a finite number of elements $|G|$ — cardinality

Ex: $(\mathbb{Z}_n, +)$ → $|\mathbb{Z}_n| = n$; $(\mathbb{Z}_n^*, \cdot)$ → $|\mathbb{Z}_n^*| = \Phi(n)$

→ **Def^n of Group:** A group is a set of elements G together with operation $\circ$ that combines two operands from G
- ↳ $\circ$ is closed : $\forall a,b \in G\ \exists c \in G\ (a \circ b = c)$
- ↳ $\circ$ is associative : $\forall a,b,c \in G\ (a \circ (b \circ c) = (a \circ b) \circ c)$
- ↳ Identity Element : $\forall a \in G\ (1 \circ a = a \circ 1 = a)$
- ↳ Inverse of a w.r.t $\circ$: $\forall a \in G\ \exists a^{-1} \in G\ (a \circ a^{-1} = 1)$
- ↳ $\circ$ is commutative : $\forall a,b \in G\ (a \circ b = b \circ a)$

Ex: $(\mathbb{Z}, +)$ is a group; $(\mathbb{Z} \setminus \{0\}, \cdot)$ is NOT a group ...

→ Order of an Element: Order of $a \in G$ of $(G, \circ)$, $\text{ord}(a)$, = smallest $k \in \mathbb{Z}^+$ s.t. $a^k = \underbrace{a \circ a \circ \dots \circ a}_{k \text{ times}} = 1$ (identity)

Ex: $a = 3$, $\mathbb{Z}_{11}^*$ ⟶
- $a^1 = 3$
- $a^2 = a \cdot a = 9$
- $a^3 = a \cdot a \cdot a = 27 \equiv 5 \bmod 11$
- $a^4 = a^3 \cdot a = 5 \cdot 3 = 15 \equiv 4 \bmod 11$
- $a^5 = a^4 \cdot a = 4 \cdot 3 = 12 \equiv 1 \bmod 11$

cycle repeats as we continue; ∴ $\text{ord}(a) = 5$

→ Cyclic Group: Group G which contains α with max order $\text{ord}(\alpha) = |G|$ ⟶ G is cyclic. (α: primitive / generator.)
generator in 1 group is not necessarily a generator for another.

→ Theorem: $\forall p$ prime ($(\mathbb{Z}_p^*, \cdot)$ is commutative finite cyclic group) → DHKE

→ Theorem: Let G be a finite cyclic group. $\forall a \in G\ (a^{|G|} = 1 \wedge \text{ord}(a) \mid |G|)$ → in cyclic groups only element orders that divide group cardinality exist

→ Theorem: Let G be a finite cyclic group. Then:
1. Number of primitive elements of G is $\phi(|G|)$
2. If $|G|$ is prime $\Rightarrow \forall (a \neq 1) \in G$ (a is primitive)

Special Cyclic Subgroup case: Order is prime. If $|H| = q \Rightarrow$ all non-1 elements have order q

→ Cyclic Subgroup Theorem: Let $(G, \circ)$ be a cyclic group. $\forall a \in G$ with $\text{ord}(a) = s$ is a primitive element of a cyclic subgroup of s elements
↳ To check if H subgroup of G, check all properties of G → any element of cyclic group is generator of cyclic subgroup

→ Lagrange's Theorem: Let H be a subgroup of G. $|H| \mid |G|$

→ Theorem: G is a finite cyclic group of order n where α is generator. $\forall k \mid n\ \exists$ exactly 1 cyclic subgroup H (H has order k) — generated by $\alpha^{n/k}$
H consists elements $a \in G$ s.t. $a^k = 1$ ⟶ no other subgroups.
enables to construct subgroup from any cyclic group ⟶ need generator and cardinality n.

→ **Discrete Logarithmic Problem (DLP) in $\mathbb{Z}_p^*$:** Given finite cyclic group $\mathbb{Z}_p^*$ of $|\mathbb{Z}_p^*| = p-1$, and primitive $\alpha \in \mathbb{Z}_p^*$ and another element $\beta \in \mathbb{Z}_p^*$ ⟶ Find $x \in \{ y \in \mathbb{Z} \mid 1 \leq y \leq p-1 \}$ s.t. $\alpha^x \equiv \beta \bmod p$
↳ $x = \log_\alpha \beta \bmod p$ ⟶ very difficult for large parameters & prime p. ∴ $\alpha^x \equiv \beta \bmod p$ is easy to compute ⟶ OWF

Desireable for prime cardinality to prevent Pohlig-Hellman attack

→ Generalized DLP: Given finite cyclic group $(G, \circ)$, $|G| = n$, primitive $\alpha \in G$ and element $\beta \in G$ $\Rightarrow$ Find $x \in \{1, 2, \dots, n\}$ s.t. $\beta = \overbrace{\alpha \circ \alpha \circ \dots \circ \alpha}^{x \text{ times}} = \alpha^x$

some cyclic groups in DLP cannot be used if computing x is easy → difficulty determines security

→ Ex of DLPs:
1. $(\mathbb{Z}_p, \cdot)$
2. Cyclic group formed by elliptic curve
3. $(GF(2^m), \cdot)$ ⟶ longer bit lengths required
4. Hyperelliptic curves

→ There are different practical ways of solving DLP ⟶ generic and non-generic algorithms

→ Generic Algorithms. Methods that only use group operation $\circ$ and no other information ⟶ works with any group
- Running Time depends on size of cyclic group: BFS, Baby step giant step, Pollard's Rho
- Running Time depends on size of prime factors of cardinality: Pohlig-Hellman

→ Brute Force Search: Most naive & costly to compute $\log_\alpha \beta$ ⟶ compute powers of α until $\alpha^x = \beta$ $O(|G|)$
↳ To prevent attack, $|G|$ must be large

→ Shank's Baby Step Giant Step: Time-memory tradeoff reducing runtime of BFS at cost of extra storage
↳ write $x = \log_\alpha \beta$ as $x = x_g m + x_b$ for $0 \leq x_g, x_b < m$ ⟶ m chosen $m \approx \sqrt{|G|}$
∴ $\beta = \alpha^x = \alpha^{x_g m + x_b}$ ⟶ $\beta \cdot (a^{-m})^{x_g} = \alpha^{x_b}$

1. Find x_g and x_b separately (ex: divide & conquer)
2. Baby Step: Compute & store all α^{x_b} for $0 \leq x_b < m$ — $O(\sqrt{|G|})$ time & space complexity
3. Giant Step: Check for all $0 \leq x_g < m$ for $\beta \cdot (\alpha^{-m})^{x_g} \stackrel{?}{=} \alpha^{x_b}$ for some α^{x_b}

$\beta \cdot (\alpha^{-m})^{x_{g,0}} = \alpha^{x_{b,0}}$ for some $(x_{g,0}, x_{b,0})$ ⟶ $x = x_{g,0} m + x_{b,0}$ — $O(\sqrt{|G|})$

→ Pollard's Rho: $O(\sqrt{|G|})$ runtime but better space complexity (Probabilistic using Birthday Paradox)
↳ Idea: Pseudorandomly generate group elements of form: $\alpha^i \cdot \beta^j$. WE KEEP TRACK OF i & j.
Continue until collision: $\alpha^{i_1} \cdot \beta^{j_1} = \alpha^{i_2} \cdot \beta^{j_2}$
If we subst $\beta = \alpha^x$ & compare exponents ⟶ $i_1 + x j_1 \equiv i_2 + x j_2 \bmod |G|$
↳ Best for Elliptic Curve Groups $O(\sqrt{|G|})$ $\Rightarrow x \equiv \frac{i_2 - i_1}{j_2 - j_1} \bmod |G|$

→ Pohlig-Hellman Algorithm: Based on Chinese Remainder Theorem ⇒ Exploits factorization on $|G| = p_1^{e_1} \cdot \ldots \cdot p_\ell^{e_\ell}$ → compute small $x_i \equiv x \bmod p_i^{e_i}$ in prime factorization of $|G|$. Use CRT to compute x from all x_i computed using Rho or BSGS

→ Non-generic algorithm: Exploit specific properties of Cyclic Group ⟶ powerful attacks

→ Index Calculus Method: Efficient alg. for computing DLP in $\mathbb{Z}_p^*$ and $GF(2^m)^*$ ⟶ subexponential running time
↳ depends on property: large amt. of elements in G expressed as products of subset of G ⟶ requires large bit length

→ Let's say Goal of adversary: compute shared K_{AB}
↳ Knows α and p, $A = k_{pub,A}$, $B = k_{pub,B}$ ⟶ can they compute $K = \alpha^{ab}$ from α, p, A and B

→ Generalized DHP: Given finite cycle group G of $|G| = n$, primitive $\alpha \in G$ and $A = \alpha^a$, $B = \alpha^b$. Find α^{ab} in G

→ Basic DHKE is not secure ⟶ Trudy can modify or send false messages with man-in-the-middle attack

→ General Approach of solving DHP. Consider DHP in $\mathbb{Z}_p^*$. Compute $a = k_{pr,A}$ by solving $a \equiv \log_\alpha A \bmod p$ 2. Compute $K_{AB} = B^a \bmod p$. ⟶ infeasible since p is huge
↳ In theory, method of solving DHP w/o DLP (analogous to RSA)

→ To ensure security, make sure DLP cannot be solved using Index Calculus and can withstand all attacks

→ **Elgamal Encryption Scheme:** Extension of DHKE based on DLP and DHP over $\mathbb{Z}_p^*$ where p is prime.
↳ Both parties first perform DHKE to derive shared k_M. Assume α and p generated.
New Idea: encrypt x as $y \equiv x \cdot k_M \bmod p$

→ Phases. Classical DHKE, Message Encrypt, Decrypt ⟶ k_M used to mask x; Bob reverses encrypt w/ inverse mask
- Classical DHKE: Bob computes $k_{pr} = d$ & $\beta = k_{pub}$; Alice computes new k_{pr} and k_{pub} denoted by i and k_E

→ Property of Cyclic Groups: $\forall k_M \in \mathbb{Z}_p^*$ (x maps to y if 2 values multiplied)
if k_M randomly drawn from $\mathbb{Z}_p^*$ ⇒ every y is equally likely

→ In protocol: all parameters have bit length of $\lceil \log_2 p \rceil$ ⇒ ciphertext (k_E, y) twice as long as msg

Elgamal Encryption Protocol

Alice		Bob
		choose large prime p
		$\alpha \in \mathbb{Z}_p^*$
		Choose $k_{pr} = d \in \{2, \dots, p-2\}$
		$\beta \equiv \alpha^d \bmod p$
	⟵ $k_{pub} = (p, \alpha, \beta)$	
Choose $i \in \{2, \dots, p-2\}$		
$k_E \equiv \alpha^i \bmod p$		
$k_M \equiv \beta^i \bmod p$ (mask)		
$y \equiv x \cdot k_M \bmod p$	(k_E, y) ⟶	
		$k_M \equiv k_E^d \bmod p$ (mask)
		$x \equiv y \cdot k_M^{-1} \bmod p$

→ Correctness Proof of Elgamal Encryption Protocol

WTS $d_{k_{pr}}(k_E, y) = x$:

$$d_{k_{pr}}(k_E, y) \equiv y \cdot (k_M)^{-1} \bmod p$$
$$\equiv \left[x \cdot k_M\right]\left(k_E^d\right)^{-1} \bmod p$$
$$\equiv \left[x \cdot (\alpha^m)^i\right]\left[(\alpha^i)^d\right]^{-1} \bmod p$$
$$\equiv x \cdot \alpha^{d \cdot i - d \cdot i} \bmod p$$
$$\equiv x \bmod p$$

→ EES is not computationally feasible due to large p even if it is a probabilistic encryption scheme

→ Key Generation: Should be at least 2048 bits → k_{pr} generated by TRNG, k_{pub} requires efficient exponentiation

→ Encryption: 2 modular exponentiation + modular multiplication (operands have bit length of $\lceil \log_2 p \rceil$)

→ Decryption: Main Steps: $k_M \equiv k^d \bmod p$ using SAM and EEA to get k_M^{-1} → $k_M^{-1} \equiv (k_E^d)^{-1} \bmod p$
$\equiv (k_E^d)^{-1} k_E^{p-1} \bmod p$
$\equiv k_E^{p-d-1} \bmod p$

One modular mult → $K \equiv y \cdot k_M^{-1} \bmod p$

→ There are Passive and Active Attacks against EES

→ Passive Attacks: Recovering x from $p, \alpha, \beta = \alpha^d, k_E = \alpha^i, y = x \cdot \beta^i$ → security relies on DHP and DLP

Say Oscar was able to solve DLP. Then he can attack in 2 ways.

1. Recover x by finding d: $d \equiv \log_\alpha \beta \bmod p$ → $x \equiv y \cdot (k_E^d)^{-1} \bmod p$
2. Recover x by finding i: $i \equiv \log_\alpha k \bmod p$ → $x \equiv y \cdot (\beta^i)^{-1} \bmod p$

DLP

Contrastingly, use Index Calculus