

# Discrete Logarithm and Encryption Schemes

## Cryptology (1)

---

Martin Stanek

2025

KI FMFI UK Bratislava

# Discrete logarithm problem

- Given a finite group  $(G, \cdot)$  and elements  $g, y \in G$ . Compute  $x \in \mathbb{Z}$  such that  $g^x = y$ .
- usually cyclic (sub)groups with generator  $g$
- DLOG is easy/hard depending on the group  $(G, \cdot)$
- Easy:
  - $(\mathbb{Z}_n, +)$  – DLOG by solving congruence  $gx \equiv y \pmod{n}$
- Hard:
  - $(\mathbb{Z}_p^*, \cdot)$  for prime  $p$ ; usually with  $g$  generating a subgroup of large prime order  $q$
  - Elliptic curve groups (various curve types over finite fields)

## Example of DLOG in $(\mathbb{Z}_{11}^*, \cdot)$

- $g = 5$ :  $\log_5 9 = 4$ ;  $\log_5 7$  does not exist

$x$	0	1	2	3	4	5	6	7	8	9	10
$5^x \bmod 11$	1	5	3	4	9	1	5	3	4	9	1

- $g = 7$ :  $\log_7 2 = 3$ ;  $\log_7 10 = 5$

$x$	0	1	2	3	4	5	6	7	8	9	10
$7^x \bmod 11$	1	7	5	2	3	10	4	6	9	8	1

# Solving “hard” instances of DLOG

- $(\mathbb{Z}_p^*, \cdot)$ 
  - Specific algorithms, for example when  $p - 1$  lacks large prime factor
  - General algorithm: Number Field Sieve for DLOG – complexity as GNFS for factorization ( $\Rightarrow$  equal key length)
- Elliptic curves
  - specific algorithms, e.g., for supersingular curves over small field
  - elliptic curves should be chosen carefully
- Generic algorithms
  - work for any cyclic group
  - the best algorithms for some groups, such as some elliptic curve groups
  - complexity  $O(n^{1/2})$ , for  $n = |G|$
  - algorithms: baby-step/giant-step, Pollard’s  $\rho$ , Pohlig-Hellman

# Equivalent key lengths

- NIST Recommendations (SP 800-57 part 1 rev. 5) (2020)
- other estimates are compared at [Keylength.com](https://keylength.com)

symmetric	modular (subgroup)	elliptic curves
80	1024 (160)	160
112	2048 (224)	224
128	3072 (256)	256
192	7680 (384)	384
256	15360 (512)	512

## Selection of the base is irrelevant for DLOG

- $g, h$  - generators of  $G$ ,  $|G| = n$
- $y$  - input
- if  $\log_h(\cdot)$  can be computed efficiently, then  $\log_g(\cdot)$  can be computed:
  1. compute  $a, b$ :  $h^a = g, h^b = y$
  2.  $g^{ba^{-1}} = (h^a)^{ba^{-1}} = h^b = y$ , where the inverse is computed mod  $n$
- since  $g, h$  are generators, the inverse  $a^{-1}$  mod  $n$  must exist
- For some constructions, e.g. ElGamal digital signature scheme, it is important to choose the generator carefully (there are strong and weak ones)!

# How to choose a generator of $(\mathbb{Z}_p^*, \cdot)$

- generator of  $(\mathbb{Z}_p^*, \cdot)$ 
  - assume  $p = 2q + 1$  for a prime number  $q$  ( $p$  is called a “safe” prime)
  - $|\mathbb{Z}_p^*| = p - 1$ , thus any element has order in  $\{1, 2, q, p - 1\}$
  - there are  $\varphi(p - 1) = \varphi(2)\varphi(q) = q - 1$  generators
  - the probability of a random element being a generator is  $\frac{q-1}{p-1} = \frac{q-1}{2q} \approx 50\%$
  - testing:  $g \notin \{1, -1\}$  is a generator  $\Leftrightarrow g^q \bmod p \neq 1$
- generator of a subgroup
  - assume a prime  $q \mid (p - 1)$
  - choose random  $h$  and compute  $g = h^{(p-1)/q} \bmod p$ ; if  $g = 1$  choose again
  - trivially  $g^q \equiv 1 \pmod{p}$  (FLT), so we have  $\text{ord}(g) \mid q$
  - since  $\text{ord}(g) > 1$ , it follows  $\text{ord}(g) = q$
  - useful for working in smaller subgroup (shorter exponents are used)

## Security of the last bit(s) of DLOG in $(\mathbb{Z}_p^*, \cdot)$

- let  $g$  be a generator of  $(\mathbb{Z}_p^*, \cdot)$
- we can write  $p = 2^s \cdot t + 1$  for  $s \geq 1$  and some odd  $t$
- input:  $y \in \mathbb{Z}_p^*$
- let  $x = \log_g y$  ( $g^x \bmod p = y$ )
- we use the binary representation of  $x = (x_l \dots x_1 x_0)_2 = 2^l x_l + \dots + 2x_1 + x_0$
- compute:

$$y^{\frac{p-1}{2}} \equiv g^{\frac{x(p-1)}{2}} \equiv g^{x_0 \frac{p-1}{2}} \equiv \begin{cases} 1 & \text{if } x_0 = 0 \\ -1 & \text{if } x_0 = 1 \end{cases} \pmod{p}$$

- $x_0$  can be found

... cont.

- we can continue for  $s$  bits
- let us assume that  $x_0, \dots, x_{i-1}$  are known ( $i < s$ )
- compute:

$$\begin{aligned} \left( y \cdot g^{-\left(x_0 + \dots + 2^{i-1}x_{i-1}\right)} \right)^{\frac{p-1}{2^{i+1}}} &\equiv g^{\left(2^i x_i + \dots + 2^l x_l\right) \frac{p-1}{2^{i+1}}} \\ &\equiv g^{x_i \frac{p-1}{2}} \equiv \begin{cases} 1 & \text{if } x_i = 0 \\ -1 & \text{if } x_i = 1 \end{cases} \pmod{p} \end{aligned}$$

- cannot be extended for more than  $s$  bits
- we can limit the “damage” to a single bit by choosing a safe prime

# Encryption schemes

---

# ElGamal encryption scheme

- ElGamal (1985)
  - originally, a default algorithm in GPG (still an option in GPG)
- Initialization:
  1. choose a large random prime  $p$ , and a generator  $g$  of  $(\mathbb{Z}_p^*, \cdot)$
  2. choose a random  $x \in \{1, \dots, p - 2\}$
  3.  $y = g^x \bmod p$
  - public key:  $y, p, g$  (the values  $p, g$  can be shared among group of users)
  - private key:  $x$

# ElGamal – encryption and decryption

- Encryption (plaintext  $m \in \mathbb{Z}_p^*$ ):

$$(r, s) = (g^k \bmod p, y^k \cdot m \bmod p), \quad \text{for random } k \in \mathbb{Z}_{p-1}$$

- Decryption (ciphertext  $(r, s)$ , computation mod  $p$ ):

$$s \cdot r^{-x} = y^k \cdot m \cdot r^{-x} = g^{xk} \cdot g^{-xk} \cdot m = m$$

- encryption: two exponentiations; decryption: single exponentiation
  - $r = g^k$  and  $y^k$  can be precomputed
- randomized encryption: 1 plaintext maps to  $\approx p$  ciphertexts
- security of the private key: DLOG problem
- knowledge of  $k$  allows to decrypt without  $x$ :  $s \cdot y^{-k} = m$ 
  - computing  $k$  from  $r$ : DLOG problem

- Reusing  $k$ :  $m_1 \mapsto (r, s_1)$ ,  $m_2 \mapsto (r, s_2)$ , we can compute  $\frac{s_1}{s_2} = \frac{m_1}{m_2}$
- Homomorphic property for multiplication:
  - encryptions of two plaintexts  $m_1, m_2$ :  
 $m_1 \mapsto (r_1, s_1) = (g^{k_1}, y^{k_1} \cdot m_1)$ ,  $m_2 \mapsto (r_2, s_2) = (g^{k_2}, y^{k_2} \cdot m_2)$
  - multiplying the ciphertexts:  $(r_1 \cdot r_2, s_1 \cdot s_2) = (g^{k_1+k_2}, y^{k_1+k_2} \cdot (m_1 \cdot m_2))$
- Simple malleability:  $(r, s) \mapsto (r, s \cdot m')$  changes the plaintext from  $m$  to  $m \cdot m'$
- Blinding (CCA):
  - access to a CCA oracle
  - How to decrypt  $(r, s)$  if the oracle won't decrypt this message?
  - use  $(rg^c, sy^c \cdot m')$  for a random value  $c$  and  $m'$
  - after decryption we get a message  $m \cdot m'$ , so  $m$  can be recovered easily

- Computational Diffie-Hellman problem (CDH):
  - compute  $g^{ab}$  given  $g, g^a, g^b$  for random generator  $g$ , and random  $a, b$
  - DLOG  $\Rightarrow$  CDH (opposite direction is open in general)
- ElGamal decryption without the private key  $\Leftrightarrow$  CDH
  - [ $\Leftarrow$ ] use CDH to compute  $g^{xk}$  from  $r = g^k$  and  $y = g^x$ ; then the plaintext can be computed:  $m = s \cdot (g^{xk})^{-1}$
  - [ $\Rightarrow$ ] input:  $g^a, g^b$ 
    - set  $y = (g^a)^{-1}, r = g^b$  and  $s = g^c$  for a random  $c$
    - use the decryption oracle for  $y$  and  $(r, s)$  to get the value  $m = s \cdot r^a = g^{c+ab}$
    - finally, divide  $m$  by  $s$ :  $m \cdot s^{-1} = g^{c+ab} \cdot g^{-c} = g^{ab}$

# What is a quadratic residue?

- $a \in \mathbb{Z}_n^*$  is called a *quadratic residue* modulo  $n$  if there exists an integer  $b$  such that  $b^2 \equiv a \pmod{n}$
- otherwise  $a$  is called a quadratic nonresidue modulo  $n$
- $\text{QR}_n$  – the set of all quadratic residues modulo  $n$
- $\text{QNR}_n$  – the set of all quadratic nonresidues modulo  $n$
- trivially  $\text{QR}_n \cup \text{QNR}_n = \mathbb{Z}_n^*$
- it is easy to test quadratic residuity modulo a prime number:

**Euler's criterion.** Let  $p > 2$  be a prime and  $a \in \mathbb{Z}_p^*$ . Then  $a \in \text{QR}_p \Leftrightarrow a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ .

## Semantic “insecurity” of ElGamal

- we can test the parity of  $k$  (it is the last bit of discrete logarithm of  $r$ )
- another view: for a generator  $g$  we have  $r \in \text{QR}_p \Leftrightarrow k$  is even
- for even  $k$ :  $s \in \text{QR}_p \Leftrightarrow m \in \text{QR}_p$
- for odd  $k$ :
  - if  $y \in \text{QR}_p$ :  $s \in \text{QR}_p \Leftrightarrow m \in \text{QR}_p$
  - if  $y \in \text{QNR}_p$ :  $s \in \text{QR}_p \Leftrightarrow m \in \text{QNR}_p$
- we can compute “something” about  $m$  from the ciphertext and  $y$

# How to achieve semantic security

- use a subgroup  $\text{QR}_p$  for a safe prime  $p = 2q + 1$  (or a general cyclic group of some prime order) **and** assume the hardness of a DDH problem in this group
- DDH (Decisional Diffie-Hellman) problem: efficiently distinguish triplets  $(g^a, g^b, g^{ab})$  and  $(g^a, g^b, g^c)$  where  $c$  is random
- there are groups where CDH seems to be hard and DDH is easy
  - $(\mathbb{Z}_p^*, \cdot)$ , elliptic-curve groups with pairing

## Some variants of ElGamal scheme

- ElGamal in a general cyclic group:
  - $|G| = q$  (for prime  $q$ ) with generator  $g$
  - private key:  $x \in \mathbb{Z}_q^*$ ; public key  $y = g^x$
  - encryption of  $m \in G$ :  $(r, s) = (g^k, m \cdot y^k)$  for random  $k \in \mathbb{Z}_q^*$
  - decryption of  $(r, s)$ :  $s \cdot r^{-x} = m \cdot y^k \cdot g^{-kx} = m$
- ElGamal with a hash function:
  - overcoming the group encoding problem ( $m \in G$ )
  - encryption  $m \in \{0, 1\}^l$ :  
 $(r, s) = (g^k, m \oplus H(y^k))$  for random  $k \in \mathbb{Z}_q^*$  and suitable  $H$  and  $l$
  - security depends on CDH and properties of  $H$
  - still malleable

# Elliptic curves

---

# Elliptic curves – introduction

- we start with elliptic curves over real numbers
- Weierstrass equation ( $a, b \in \mathbb{R}$ ):  $y^2 = x^3 + ax + b$
- we are interested in non-singular curves, i.e.  $4a^3 + 27b^2 \neq 0$
- non-singular:  $x^3 + ax + b$  has no repeated roots
- points:  $E = \{(x, y) \mid y^2 = x^3 + ax + b\} \cup \{0\}$ ,  
where 0 is an identity element (point at infinity)
- group  $(E, +)$  uses a commutative “addition”:
  - notation:  $P = (x_P, y_P)$ ,  $\bar{P} = (x_P, -y_P)$
  - $P + \bar{P} = 0$
  - $P + P = R = (x_R, y_R)$  such that the line  $P\bar{R}$  is a tangent in  $P$
  - $P + Q = R = (x_R, y_R)$  such that  $\bar{R}$ ,  $P$  and  $Q$  are collinear

## Elliptic curves – addition formulas

- $P = (x_P, y_P), Q = (x_Q, y_Q)$
- case 1:  $P + (-P) = (x_P, y_P) + (x_P, -y_P) = 0$
- case 2 and case 3:  $P + Q = (x_R, y_R)$

$$x_R = \lambda^2 - x_P - x_Q$$

$$y_R = \lambda(x_P - x_R) - y_P$$

$$\lambda = \begin{cases} (3x_P^2 + a)(2y_P)^{-1} & P = Q \\ (y_Q - y_P)(x_Q - x_P)^{-1} & x_P \neq x_Q \end{cases}$$

# Elliptic curves over finite field

- $\text{GF}(p) = (\mathbb{Z}_p, +, \cdot)$ , for prime  $p > 3$ 
  - other finite fields can be used, e.g.  $\text{GF}(2^n)$ , with different forms, conditions and addition formulas
- $E = \{(x, y) \mid y^2 = x^3 + ax + b \pmod{p}\} \cup \{0\}$ ,  
for  $a, b \in \mathbb{Z}_p$  satisfying  $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$
- addition of points still “works”  $(\pmod{p})$ , i.e.  $(E, +)$  is an abelian group
- no geometric interpretation anymore
- Hasse’s theorem:  $| |E| - p - 1 | \leq 2\sqrt{p}$ 
  - counting the exact number of points: Schoof-Elkies-Atkin algorithm with  $O(\log^4 p)$  operations in  $\mathbb{Z}_p$
- remark: a point  $P = (x_P, y_P)$  can be uniquely represented by  $x_P$  and the sign of  $y_P$

## Real world examples: NIST P-256 curve

- prime:  $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

- the curve:

$$y^2 = x^3 - 3x +$$

41058363725152142129326129780047268409

11444101599372554835256314039467401291

- number of points (prime):

11579208921035624876269744694940757352999

6955224135760342422259061068512044369

- critique: Failures in NIST's ECC standards (Bernstein, Lange, 2016)

## Real world examples: Curve25519

- prime:  $p = 2^{255} - 19$
- the curve:  $y^2 = x^3 + 486662x^2 + x$
- number of points  $8 \cdot p_1$  for a prime

$$p_1 = 2^{252} + 27742317777372353535851937790883648493$$

- Montgomery form
  - different addition formulas, it can be translated into Weierstrass form
- used (along other curves) in various applications (OpenSSH, Signal, Threema, etc.)
- *equivalent* curve Ed25519 standardized for a signature scheme
  - FIPS 186-5, see also NIST SP 800-186

# DLOG in elliptic curve groups

- $(E, +)$  – elliptic curve group
- point  $P \in E$
- $kP = \underbrace{P + P + \dots + P}_k$ , for an integer  $k \geq 0$
- DLOG: given a point  $kP$ , compute  $k$
- CDH: given  $aP$  and  $bP$ , compute  $(ab)P$

## EC version of ElGamal scheme

- $(E, +)$  – elliptic curve group
- $G \in E$  – generator of some subgroup of  $E$ ,  $\text{ord}(G) = q$  (prime)
- private key: random  $x \in \mathbb{Z}_q$
- public key:  $Y = xG$
- Encryption of  $M \in E$ :  $(R, S) = (kG, kY + M)$  for random  $k \in \mathbb{Z}_q$
- Decryption of  $(R, S) \in E \times E$ :

$$S - xR = (kY + M) - xR = (kx)G + M - (kx)G = M$$

- group encoding

## Exercises

1. Consider group  $(P_n, \circ)$ , where  $P_n$  is the set of permutations on  $\mathbb{Z}_n$ , and  $\circ$  is a composition operator. How hard is DLOG problem for this group?
2. Prove the Euler's criterion.

Let  $p > 2$  be a prime and  $a \in \mathbb{Z}_p^*$ . Then  $a \in \text{QR}_p \Leftrightarrow a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ .

3. Let  $(E, +)$  be an elliptic curve group of prime order  $p$ . Let  $G, F$  be generators with unknown  $\log_G F$ . Show that  $h : \mathbb{Z}_p \times \mathbb{Z}_p \rightarrow E$  defined as  $h(a, b) = aG + bF$  is collision resistant.