Discrete logarithm and related schemes

Martin Stanek

Department of Computer Science
Comenius University
stanek@dcs.fmph.uniba.sk

Cryptology 1 (2020/21)

Content

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 are used
- DLOG is easy/hard depending on the group G
- 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 various finite fields)

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

- let $p = 11$
- case 1: $g = 5$

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

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

- case 2: $g = 7$

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

  - $\log_7 2 = 3$; $\log_7 10 = 5$

Solving “hard” instances of DLOG

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

Equivalent key lengths

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

NIST Recommendations (SP 800-57 part 1 rev. 5) (2020)

various methods are compared at www.keylength.com

Selection of the base is irrelevant for DLOG

- g, h – generators of G, $|G| = n$
- y – input
- if DLOG w.r.t. the base h can be computed efficiently, then DLOG w.r.t the base g 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 $(q - 1)/(p - 1) = (q - 1)/(2q) \approx 50\%$
  - 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 $\mathrm{ord}(g) \mid q$
  - since $\mathrm{ord}(g) > 1$, it follows $\mathrm{ord}(g) = q$
  - 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 $p = 2^s t + 1$ for $s \geq 1$ and some odd t
- input: $y \in \mathbb{Z}_p^*$
- let x be the DLOG of y, i.e. $g^x \bmod p = y$
- we use the binary representation of $x = (x_l \ldots x_1 x_0)_2 = 2^l x_l + \ldots + 2x_1 + x_0$
- compute:

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

- 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 (mod p):

$$\left(y \cdot g^{-(x_0+\ldots+2^{i-1}x_{i-1})}\right)^{(p-1)/2^{i+1}} \equiv g^{(2^i x_i+\ldots+2^l x_l)(p-1)/2^{i+1}}$$

$$\equiv g^{x_i(p-1)/2} \equiv \begin{cases} 1 & \text{if } x_i = 0 \\ -1 & \text{if } x_i = 1 \end{cases}$$

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

ElGamal encryption scheme

- ElGamal (1985)
  - example: originally, a default algorithm in GPG (still an option for public-key encryption in GPG)
  - variants exists (what (sub)groups are used)
- Initialization:
  1. choose a large random prime p, and a generator g of $(\mathbb{Z}_p^*, \cdot)$
  2. choose a random $x \in \{1, \ldots, p-2\}$
  3. $y = g^x \bmod p$

  - public key: y, p, g (the values p, g can be shared)
  - 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

Remarks

- Reusing k:
  - $m_1 \mapsto (r, s_1)$, $m_2 \mapsto (r, s_2)$, we can compute $s_1/s_2 = m_1/m_2$
- Homomorphic property:
  - 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

ElGamal – security and CDH

- 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: $m \cdot s^{-1} = g^{c+ab} \cdot g^{-c} = g^{ab}$

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 $r \in QR_p \iff k$ is even
- for even k: $s \in QR_p \iff m \in QR_p$
- for odd k:
  - if $y \in QR_p$: $s \in QR_p \iff m \in QR_p$
  - if $y \in QNR_p$: $s \in QR_p \iff m \in QNR_p$
- we can compute "something" about m from the ciphertext and y
- how to achieve semantic security:
  - use a subgroup 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 (e.g. $(\mathbb{Z}_p^*, \cdot)$, 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

Cramer–Shoup scheme

- group G, $|G| = q$ for a prime q
- H cryptographic hash function (outputs from $\mathbb{Z}_q$)
- generators $g_1, g_2 \in G$
- private key: random $x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{Z}_q^*$
- public key: g_1, g_2, $h = g_1^{x_1} g_2^{x_2}$, $c = g_1^{y_1} g_2^{y_2}$, $d = g_1^{z_1} g_2^{z_2}$
- Encryption ($m \in G$): $(u_1, u_2, w, \pi) = (g_1^r, g_2^r, h^r \cdot m, (cd^{\alpha})^r)$, where $\alpha = H(g_1^r \,||\, g_2^r \,||\, h^r \cdot m)$, and r is a random element from $\mathbb{Z}_q$
- Decryption of a ciphertext (u_1, u_2, w, π):
  1. $\alpha = H(u \,||\, v \,||\, w)$
  2. if $u_1^{y_1 + \alpha z_1} u_2^{y_2 + \alpha z_2} \neq \pi$ then reject: "invalid ciphertext"
  3. compute $w/(u_1^{x_1} u_2^{x_2}) = h^r \cdot m/(g_1^{x_1 r} g_2^{x_2 r}) = (g_1^{x_1} g_2^{x_2})^r \cdot m/(g_1^{x_1 r} g_2^{x_2 r}) = m$
- π is a non-interactive proof that $\log_{g_1} u_1 = \log_{g_2} u_2$

Cramer–Shoup scheme (2)

- IND-CCA2 secure scheme
- assumptions: DDH and H is collision resistant hash function

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 $\sim x^3 + ax + b$ 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: $P = (x_P, y_P)$, $\overline{P} = (x_P, -y_P)$
  - $P + \overline{P} = 0$
  - $P + P = R = (x_R, y_R)$ such that the line $P\overline{R}$ is a tangent in P
  - $P + Q = R = (x_R, y_R)$ such that $\overline{R}$, 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

- $GF(p) = (\mathbb{Z}_p, +, \cdot)$, for prime $p > 3$
  - other finite fields can be used, e.g. $GF(2^n)$, with different forms, conditions and addition formulas
- $E = \{(x, y) \mid y^2 = x^3 + ax + b \bmod p\} \cup \{0\}$, for $a, b \in \mathbb{Z}_p$ satisfying $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$
- addition of points still “works” (mod 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's algorithm with $O(\log^5 p)$ operations in $\mathbb{Z}_p$ or improved version 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 (1): NIST P-256 curve

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

$$y^2 = x^3 - 3x + 41058363725152142129326129780047268409 \\ 114441015993725554835256314039467401291$$

- number of points (prime):

$$115792089210356248762697446949407573529996955224135760342422259061068512044369$$

Real world examples (2): NIST P-384 curve

- prime: $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
- the curve:

$$y^2 = x^3 - 3x + \begin{array}{l} 2758019355995970587784901184038904809305 \\ 6905856361568521428707301988689241309860 \\ 865136260764883745107765439761230575 \end{array}$$

- number of points (prime):

$$\begin{array}{l} 394020061963944792122790401001436138050797392704654466679 4 \\ 690527962765939911326356939895630815229491355443365394264 3 \end{array}$$

- required for TOP SECRET classification (NSA – Commercial National Security Algorithm Suite, 2015)
- critique: Failures in NIST's ECC standards (Bernstein, Lange, 2016)

Real world examples (3): 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)
- non-standard curve
- used (along other curves) in various applications (OpenSSH, Signal, Threema, etc.)

DLOG in elliptic curve groups

- $(E, +)$ – elliptic curve group
- point $P \in E$
- $kP = \underbrace{P + P + \ldots + P}_{k}$, 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, $\mathrm{ord}(G) = q$ (prime)
- private key: random $x \in \mathbb{Z}_q$
- public key: $Y = xG$
- Encryption ($M \in E$): $(R, S) = (kG, kY + M)$ for random $k \in \mathbb{Z}_q$
- Decryption ($(R, S) \in E \times E$):

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

- group encoding