Lecture Notes: Discrete Logarithm & Elliptic Curves

Course: INFO-F-537 Cryptanalysis (Topic 5)
Document compiled by Luca Volonterio

Based on: *INFOF537-DiscreteLog* & *INFOF537-EllipticCurves*

January 5, 2026

Abstract

This document covers **Topic 5** for the oral exam. It analyzes the Discrete Logarithm Problem (DLP) across different groups. It contrasts generic algorithms (like Pollard's Rho), which apply to all groups, with specific algorithms (Index Calculus) that apply only to finite fields. This distinction explains why Elliptic Curve Cryptography (ECC) allows for much smaller key sizes than RSA or Diffie-Hellman over $\mathbb{Z}_p^*$.

Contents

1 The Discrete Logarithm Problem (DLP)

1.1 Definition

Given a cyclic group G of order q with generator g, and an element $A \in G$:

- **Problem:** Find an integer $x \in [0, q-1]$ such that
$$g^x = A. \tag{1}$$
- **Notation:** $x = \log_g A$ (the discrete logarithm).

1.2 Security Reductions (Hierarchy of Hardness)

The security of Diffie-Hellman–type protocols is usually expressed in terms of three related problems in the same group G of order q with generator g.

1. **DLP (Discrete Log Problem):** Given g and g^x, find x.
2. **CDH (Computational Diffie-Hellman):** Given g, g^a, g^b, compute g^{ab}.
3. **DDH (Decisional Diffie-Hellman):** Given g, g^a, g^b, Z, distinguish whether $Z = g^{ab}$ or Z is uniform in G.

There are standard reductions:
$$\text{DLP} \;\Rightarrow\; \text{CDH} \;\Rightarrow\; \text{DDH}.$$

If one can solve DLP efficiently, then CDH and DDH are also easy, but the converse implications are not known to hold in general.

Exam Tip: Random Self-Reducibility. The DLP is random self-reducible: if there is an algorithm that solves *random* DLP instances in a given group with non-negligible probability, then there is a reduction that solves *any* fixed instance with essentially the same complexity. There are therefore no "weak keys" in a DLP group in the same sense as badly generated RSA moduli.

2 Generic Algorithms (Group-Independent)

These algorithms work in any finite group where the DLP is defined, including $\mathbb{Z}_p^*$ and elliptic curve groups. They set a baseline for the security level.

2.1 Baby-Step Giant-Step (BSGS)

A time–memory trade-off algorithm:

- Let $m = \lceil \sqrt{q} \rceil$ and write $x = im + j$ with $0 \le i, j < m$.
- From $g^x = A$ obtain
$$g^{im+j} = A \quad \Rightarrow \quad (g^m)^i = A \cdot g^{-j}.$$
- **Baby steps:** Precompute and store $A \cdot g^{-j}$ for all $0 \le j < m$ in a hash table.
- **Giant steps:** For $0 \le i < m$, compute $(g^m)^i$ and look for a match in the table.
- **Complexity:** Time $O(\sqrt{q})$, memory $O(\sqrt{q})$. The memory cost makes this impractical in large groups.

2.2 Pollard's ρ Algorithm

The standard generic attack on elliptic-curve DLP:

- Builds a pseudo-random walk in the group using a partition and an iteration function f.
- Looks for a collision of the form
$$g^{a_i} A^{b_i} = g^{a_j} A^{b_j}$$
and solves the resulting linear equation for $\log_g A$ modulo q.
- **Complexity:** Expected time $O(\sqrt{q})$ with essentially constant memory $O(1)$.

2.3 Pohlig–Hellman Algorithm

This algorithm reduces the DLP in a group of order q to DLPs in smaller prime-power order subgroups.

- If $q = \prod_i p_i^{e_i}$, then the DLP modulo q can be reduced to DLPs modulo each $p_i^{e_i}$.
- The solutions modulo $p_i^{e_i}$ are then recombined using the Chinese Remainder Theorem.
- **Implication:** The complexity is dominated by the largest prime factor of q.
- **Security requirement:** The group order must contain at least one large prime factor (e.g. q or its largest prime factor should be of size $\approx 2^{256}$ in modern ECC).

3 Index Calculus in $\mathbb{Z}_p^*$

Index Calculus is a sub-exponential algorithm for DLP in groups where elements can be factored over a small factor base, such as $\mathbb{Z}_p^*$. It does not generalize to generic elliptic curve groups.

3.1 High-Level Mechanism

1. **Relation collection:** Find many exponents k such that
$$g^k \equiv \prod_i p_i^{e_i} \pmod{p},$$
where the p_i are small primes from a fixed factor base.
2. **Linear algebra:** Taking logarithms yields a linear system in the unknowns $\log_g p_i$, which can be solved over $\mathbb{Z}_{p-1}$.
3. **Individual logarithms:** Use the precomputed $\log_g p_i$ to express a target A in terms of the factor base and compute $\log_g A$.

3.2 Impact on Key Sizes

Index Calculus runs in sub-exponential time in p (often expressed with an $L_p[1/3]$ complexity), so to reach a given security level, the modulus p (and hence key sizes) must be large. In contrast, no comparable Index Calculus-type algorithm is known for generic elliptic curves, so the best known attack remains Pollard's ρ with $O(\sqrt{q})$ complexity. As a result, 256-bit elliptic-curve groups can offer roughly 2^{128} security, comparable to 3072-bit RSA or finite-field Diffie–Hellman.

Exam Tip: Why ECC uses shorter keys. For RSA and classical Diffie–Hellman over $\mathbb{Z}_p^*$, sub-exponential algorithms (Number Field Sieve, Index Calculus) force p to be large. For ECC, only generic $O(\sqrt{q})$ attacks are known, so q need only be large enough to resist these generic attacks, giving much shorter keys for the same security level.

4 Elliptic Curves and ECC

4.1 Weierstrass Form

An elliptic curve E over a field K (with appropriate non-singularity conditions) is given by the set of points $(x, y) \in K^2$ satisfying

$$y^2 = x^3 + ax + b \tag{2}$$

together with a distinguished point at infinity $\mathcal{O}$, which acts as the neutral element of the group law.

4.2 Group Law (Chord-and-Tangent)

The points on E form an abelian group under a geometrically defined addition:

1. For distinct points P and Q, draw the line through P and Q; it intersects the curve at a third point R'. Define $P + Q$ to be the reflection of R' across the x-axis.
2. For doubling, the line is the tangent at P and meets the curve at a point R'; again set $2P$ to be the reflection of R' across the x-axis.
3. The point at infinity $\mathcal{O}$ serves as the identity; the inverse of (x, y) is $(x, -y)$.

4.3 Elliptic Curves over Finite Fields

For cryptography one works over finite fields, typically $\mathbb{F}_p$ with p prime and $p > 3$. An elliptic curve is then

$$E : y^2 \equiv x^3 + ax + b \pmod{p}$$

together with the point at infinity. The same group law formulas extend to this setting, and the set of points $E(\mathbb{F}_p)$ forms a finite abelian group.

4.4 Number of Points, Hasse's Theorem and Cofactor

The number of points on E over $\mathbb{F}_p$ is denoted $\#E(\mathbb{F}_p)$. Hasse's theorem states:

$$|\#E(\mathbb{F}_p) - (p + 1)| \leq 2\sqrt{p}.$$

It is common to write

$$\#E(\mathbb{F}_p) = hq,$$

where q is the largest prime factor of $\#E(\mathbb{F}_p)$ and h is the cofactor. For a security level of s bits against generic attacks, one needs $q \geq 2^{2s}$ so that $\sqrt{q}$ is of order 2^s.

Exam Tip: Cofactor and subgroup choice. In practice the protocol works in the subgroup of order q generated by a base point G. The cofactor h should be small and must be taken into account in implementations to avoid small-subgroup attacks.

4.5 ECC Keys and Scalar Multiplication

To instantiate ECC:

- Choose a finite field $\mathbb{F}_p$ and an elliptic curve E over it.
- Choose a base point $G \in E(\mathbb{F}_p)$ of large prime order q.
- **Private key:** An integer $d \in \{1, \ldots, q - 1\}$.
- **Public key:** The point $Q = [d]G$ obtained by scalar multiplication.

The hardness of the elliptic-curve discrete logarithm problem (ECDLP) is the assumption that, given (G, Q), it is infeasible to recover d.

5 Important Curve Families and Examples

5.1 Weierstrass Curve NIST P-256

A widely deployed Weierstrass curve is NIST P-256:

- Defined over a prime field $\mathbb{F}_p$ with $p \approx 2^{256}$.
- Equation of the form
$$y^2 = x^3 + ax + b \pmod{p}$$
with specific constants $a = -3$ and a fixed b as specified in FIPS 186.
- The group $E(\mathbb{F}_p)$ has prime order, so the cofactor $h = 1$.
- A fixed base point G of order $q = \#E(\mathbb{F}_p)$ is specified together with its coordinates.

5.2 Montgomery Curves and Curve25519

A Montgomery curve over a field K is given by

$$By^2 = x^3 + Ax^2 + x$$

with parameters $A, B \in K$ and $B \neq 0$. Such a curve is birationally equivalent to a Weierstrass curve, and over finite fields a Weierstrass curve can be converted to a Montgomery form only when the group order is divisible by 4.

Curve25519 is a Montgomery curve used for Diffie–Hellman key exchange (X25519):

- Defined over $\mathbb{F}_p$ with $p = 2^{255} - 19$.
- Has the form
$$y^2 = x^3 + 486662x^2 + x \pmod{p}.$$
- The group order is of the form $8q$ with a cofactor $h = 8$ and a large prime factor $q \approx 2^{252}$.
- Implementations typically use only the x-coordinate for the key exchange.

5.3 Edwards and Twisted Edwards Curves

An *Edwards curve* over a field K is given by

$$x^2 + y^2 = 1 + dx^2y^2$$

for some constant $d \in K$, with the neutral element at $(0, 1)$. Such curves are birationally equivalent to Weierstrass curves, but the equivalence in finite fields requires that the group order is divisible by 4. A key advantage is that Edwards curves admit complete addition formulas with no exceptional cases.

A *twisted Edwards curve* is defined by

$$ax^2 + y^2 = 1 + dx^2y^2$$

for constants $a, d \in K$ with $a \neq 0$. Twisted Edwards curves are birationally equivalent to Montgomery curves and inherit the same advantages regarding complete formulas and efficient, uniform implementations.

5.4 Example Curves: Ed448 and Ed25519

Two important Edwards-type curves for signatures are:

- **Ed448-Goldilocks:** An Edwards curve over a field of size approximately 2^{448}, with group order $\#E = 4q$ for a large prime q and cofactor $h = 4$. This curve is used with EdDSA to provide a high security level.
- **Ed25519:** A twisted Edwards curve over the same field as Curve25519, with equation
  $$x^2 + y^2 = 1 + dx^2y^2 \pmod p$$
  for a specific d, group order of the form $8q$ with cofactor $h = 8$, and a base point G of order q. Ed25519 can be obtained from Curve25519 via a birational map.

6 Signature Schemes: ECDSA vs EdDSA

6.1 ECDSA (Elliptic Curve Digital Signature Algorithm)

ECDSA is the standard EC-based analogue of DSA used in TLS, many PKI deployments, and Bitcoin.

- **Signing:**
  1. Pick a fresh random nonce $k \in \{1, \dots, q-1\}$.
  2. Compute $R = [k]G$ and let r be the x-coordinate reduced modulo q.
  3. Compute
     $$s = k^{-1}(\text{hash}(m) + d \cdot r) \bmod q.$$
  4. The signature is (r, s).
- **Verification:** Check that r, s are in range, compute
  $$u_1 = \text{hash}(m)\, s^{-1} \bmod q, \quad u_2 = r\, s^{-1} \bmod q,$$
  and then $X = [u_1]G + [u_2]Q$. If the x-coordinate of X (reduced modulo q) equals r, the signature is valid.

Nonce Reuse Vulnerability

If the same nonce k is ever reused to sign two different messages, the two signatures (r, s_1) and (r, s_2) satisfy
$$s_1 - s_2 \equiv k^{-1}(\text{hash}(m_1) - \text{hash}(m_2)) \pmod q,$$
which allows recovery of k and then of the private key d. Randomness failures in ECDSA are therefore catastrophic.

6.2 EdDSA (Edwards-Curve Digital Signature Algorithm)

EdDSA is a modern, deterministic Schnorr-like signature scheme defined over (twisted) Edwards curves, such as Ed25519 or Ed448.

Key Generation

- Start from a random seed a (the secret key material).
- Compute hash(a) and split it into:
  - A *secret scalar* used as the effective private key.
  - An *ephemeral derivation key* used to derive nonces.
- Set the public key $A = [\text{secret scalar}]\, G$.

Signing

Given a message m:

1. Derive a deterministic nonce
$$k = \text{hash}(\text{ephemeral derivation key} \parallel m).$$
2. Compute $R = [k]G$.
3. Compute the challenge
$$e = \text{hash}(R \parallel A \parallel m).$$
4. Compute the response
$$s = k + e \cdot (\text{secret scalar}) \bmod q.$$
5. The signature is (R, s).

Verification

Given (R, s) and public key A:

1. Recompute
$$e = \text{hash}(R \parallel A \parallel m).$$
2. Check whether
$$[s]G \overset{?}{=} R + [e]A.$$

6.3 Summary: ECDSA vs EdDSA

Aspect	ECDSA	EdDSA
Underlying curve	Usually Weierstrass (e.g. P-256)	(Twisted) Edwards (e.g. Ed25519, Ed448)
Nonce	Random, per-signature	Deterministic from key & message
Nonce failure impact	Catastrophic key recovery	Eliminated by construction
Formulas	Have exceptional cases	Complete, uniform formulas
Determinism	Probabilistic signatures	Deterministic signatures
Structure	DSA-style	Schnorr-style

Exam Tip: Implementation robustness. EdDSA's deterministic nonces and complete addition formulas on Edwards curves significantly reduce side-channel and RNG-related pitfalls compared to ECDSA, which is notoriously fragile with respect to nonce generation.

7 Self-Assessment Questions

Level 1: Concepts

Q: What is the Discrete Logarithm Problem?
Answer: Given a generator g and a value A in a finite cyclic group, find the integer x such that $g^x = A$.

Q: Why are elliptic-curve keys much shorter than RSA keys for the same security?
Answer: Because for finite fields and RSA there exist sub-exponential index-calculus-type attacks, much larger moduli are needed. For generic elliptic curves, only $O(\sqrt{q})$ attacks like Pollard's ρ are known, so a 256-bit group order already yields around 128-bit security.

Level 2: Algorithms

Q: How does the Pohlig–Hellman algorithm affect the choice of group parameters?
Answer: It shows that the complexity of DLP depends on the largest prime factor of the group order. Therefore one must choose groups where the order is prime or has a very large prime factor and avoid smooth group orders.

Q: What is the main difference between Baby-Step Giant-Step and Pollard's Rho?
Answer: Both run in expected time $O(\sqrt{q})$, but BSGS requires $O(\sqrt{q})$ memory, whereas Pollard's ρ uses essentially constant memory. This makes Pollard's ρ the practical generic attack on large groups.

Level 3: Protocol Security

Q: In ECDSA, what happens if the random nonce k is reused for two signatures?
Answer: The private key can be immediately recovered from the two signatures because the equations relating k, the hashes, and the signatures become a simple linear system that reveals k and then the private key.

Q: Why is EdDSA considered safer than ECDSA regarding implementation?
Answer: EdDSA derives the nonce deterministically from the secret key material and the message, eliminating dependence on external randomness. Moreover, Edwards curves have complete addition formulas, reducing the risk of side-channel leaks and edge-case bugs in implementations.