Elliptic Curve Cryptosystem

DSC 4110/6111: Modern Cryptography and AI/ML Security

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

August 12, 2026

Table of Content

Elliptic Curve Cryptosystem

Why Elliptic Curves?

We have already seen two major public-key approaches:

- **RSA:** security related to the hardness of factoring

$$N = pq.$$

- **DH / ElGamal:** security based on the **Discrete Logarithm Problem (DLP)**

$$y = g^x \quad \Longrightarrow \quad x \text{ is hard to recover.}$$

- **ECC:** uses a different mathematical group where the corresponding problem is the **Elliptic-Curve DLP (ECDLP)**

$$Q = xP \quad \Longrightarrow \quad x \text{ is hard to recover.}$$

Main Advantage

ECC provides comparable classical security with **much smaller keys**, leading to lower storage, bandwidth, and computational cost.

ECC: Security per Key Bit

For approximately comparable classical security:

Security Level	RSA	ECC
≈ 80 bits	1024 bits	160 bits
≈ 128 bits	3072 bits	256 bits
≈ 192 bits	7680 bits	384 bits
≈ 256 bits	15360 bits	512 bits

Thus, ECC can provide similar classical security with significantly smaller public keys and signatures.

- Less storage and communication bandwidth
- Faster key operations in many applications
- Particularly useful for constrained devices

Important

The advantage comes from the different hardness properties of the underlying mathematical problems, not from ECC using a fundamentally different public-key paradigm.

What Is an Elliptic Curve?

An elliptic curve over a finite field $\mathbb{F}_q$ is commonly written as

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

where $a, b \in \mathbb{F}_q$.
The curve must be **non-singular**, requiring

$$\boxed{4a^3 + 27b^2 \neq 0}.$$

This ensures that the curve has no cusps or self-intersections.
The set of curve points is

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ is the **point at infinity**.

Cryptographic Importance

The points of $E(\mathbb{F}_q)$, together with $\mathcal{O}$, form an **abelian group** under elliptic-curve point addition.

The Elliptic-Curve Group

The points on an elliptic curve, together with the point at infinity $\mathcal{O}$, form an abelian group under **point addition**.

- **Identity:**
$$P + \mathcal{O} = P.$$
- **Inverse:** For $P = (x, y)$,
$$-P = (x, -y), \qquad P + (-P) = \mathcal{O}.$$
- **Point Addition:** For two points P, Q on the curve, a geometric/algebraic operation defines another point
$$P + Q = R.$$
- **Scalar Multiplication:** Repeated point addition gives
$$\boxed{kP = \underbrace{P + \cdots + P}_{k \text{ times}}}.$$

Cryptographic Core

ECC cryptographic operations are built primarily from **scalar multiplication**. Computing $Q = kP$ is efficient, while recovering k from (P, Q) is believed to be hard.

Elliptic-Curve Discrete Logarithm Problem

Let G be an elliptic-curve group with a publicly known point P.
Given

$$P, \qquad Q = kP,$$

it is easy to compute Q when k is known.
However, given (P, Q), recovering

$$\boxed{k}$$

is believed to be computationally hard.
This is the **Elliptic-Curve Discrete Logarithm Problem (ECDLP)**.

Compare with the ordinary DLP:

$$\boxed{Q = g^k \quad \longleftrightarrow \quad Q = kP}$$

- Ordinary DLP: repeated multiplication in a multiplicative group.
- ECDLP: repeated point addition in an elliptic-curve group.

Cryptographic Assumption

The security of many ECC schemes relies on the assumed hardness of solving ECDLP for appropriately chosen elliptic-curve groups.

ECC Key Generation

Public parameters are fixed first:

$$\boxed{E,\ G,\ n}$$

where E is the elliptic curve, G is a publicly known base point, and n is the order of G.

1. Choose a random private key
$$d \xleftarrow{\$} \mathbb{Z}_n^*.$$
2. Compute the public key using scalar multiplication
$$\boxed{Q = dG}.$$
3. The resulting key pair is
$$\boxed{sk = d, \qquad pk = Q}.$$

The private key d is kept secret, while Q can be publicly distributed.

Security

Given G and $Q = dG$, recovering the private key d requires solving the Elliptic-Curve Discrete Logarithm Problem.

Elliptic-Curve Diffie–Hellman (ECDH)

ECDH is the elliptic-curve analogue of the Diffie–Hellman key agreement protocol.
Public parameters:

$$\boxed{E,\ G,\ n}$$

where G is the base point of order n.
Alice chooses

$$a \xleftarrow{\$} \mathbb{Z}_n^*, \qquad Q_A = aG.$$

Bob chooses

$$b \xleftarrow{\$} \mathbb{Z}_n^*, \qquad Q_B = bG.$$

They exchange Q_A and Q_B.
Alice computes

$$K_A = aQ_B = a(bG) = abG.$$

Bob computes

$$K_B = bQ_A = b(aG) = abG.$$

Therefore,

$$\boxed{K_A = K_B = abG}.$$

Connection with DHKE

$$g^a \quad g^b \quad g^{ab} \longrightarrow aG \quad bG \quad abG$$

ECDH: Security Intuition

An eavesdropper observes only the public values

$$Q_A = aG, \qquad Q_B = bG.$$

To compute the shared secret

$$abG,$$

the attacker would need to solve a computationally hard problem related to the elliptic-curve discrete logarithm.

$$\boxed{G,\ aG,\ bG \quad \not\Rightarrow \quad abG \text{ efficiently}}$$

Thus ECDH provides a shared secret without transmitting the secret itself.

Important

As with ordinary DHKE, ECDH by itself does **not authenticate** the public keys. Authentication, for example through digital signatures and certificates, is required to prevent a man-in-the-middle attack.

Elliptic-Curve Digital Signature Algorithm (ECDSA)

Elliptic-Curve Digital Signature: ECDSA

ECDSA provides digital signatures using the elliptic-curve group. It plays a role similar to RSA and DSA signatures.
Public parameters:

$$\boxed{E,\ G,\ n,\ H}$$

where G has order n and H is a cryptographic hash function.

- **Key Generation:** choose

$$d \xleftarrow{\$} \mathbb{Z}_n^*, \qquad Q = dG.$$

$$\boxed{sk = d, \qquad pk = Q}$$

- **Signing:** use the secret d and a fresh random nonce k to generate a signature (r, s).
- **Verification:** use the public key Q to check whether the signature corresponds to the message.

Security

ECDSA security relies on the hardness of problems related to the **Elliptic-Curve Discrete Logarithm Problem (ECDLP)**.

ECDSA: Signing and Verification

Let

$$z = H(m).$$

Sign using $sk = d$:
Choose fresh

$$k \xleftarrow{\$} \mathbb{Z}_n^*.$$

Compute

$$R = kG, \qquad r = x_R \bmod n.$$

Then compute

$$s = k^{-1}(z + rd) \bmod n.$$

The signature is

$$\boxed{\sigma = (r, s)}.$$

Verify using $pk = Q$:
Compute

$$w = s^{-1} \bmod n,$$

$$u_1 = zw, \qquad u_2 = rw.$$

Compute

$$X = u_1 G + u_2 Q.$$

Accept iff

$$\boxed{x_X \bmod n = r}.$$

ECDSA: Correctness Intuition

During signing,

$$s = k^{-1}(z + rd) \pmod{n}.$$

Therefore,

$$ks \equiv z + rd \pmod{n}.$$

During verification,

$$w = s^{-1}, \qquad u_1 = zw, \qquad u_2 = rw.$$

Since $Q = dG$,

$$\begin{aligned} X &= u_1 G + u_2 Q \\ &= zwG + rw(dG) \\ &= (z + rd)wG \\ &= kswG \\ &= kG \\ &= R. \end{aligned}$$

Therefore,

$$x_X = x_R$$

and hence

$$\boxed{x_X \bmod n = r}.$$

ECC in Practical Cryptographic Systems

Elliptic-curve cryptography provides two major functionalities:

- **Key Agreement:** ECDH

$$Q_A = aG, \qquad Q_B = bG$$
$$K = abG.$$

- **Digital Signature:** ECDSA

$$\mathrm{Sign}_{sk}(m) \to \sigma, \qquad \mathrm{Verify}_{pk}(m, \sigma).$$

- **Modern Signature Family:** EdDSA
  - Designed for efficient and robust elliptic-curve signatures.
  - Ed25519 is a widely deployed instance.

Important practical examples:

X25519 $\to$ ECDH key agreement

Ed25519 $\to$ EdDSA digital signature

Practical View

ECC is used primarily for **key agreement and digital signatures**, while symmetric cryptography is used for efficient bulk data encryption.

Important ECC Constructions

Construction	Purpose	Example
ECDH	Key Agreement	X25519
ECDSA	Digital Signature	secp256k1, P-256
EdDSA	Digital Signature	Ed25519, Ed448

These constructions use elliptic-curve scalar multiplication as their fundamental operation:

$$\boxed{Q = kG}.$$

Their security is based on the hardness of elliptic-curve discrete-logarithm-related problems.

RSA vs. ECC

	RSA	**ECC**
Security assumption	Integer Factorization	ECDLP
Typical security	3072-bit RSA $\approx$ 128-bit	256-bit ECC $\approx$ 128-bit
Key size	Large	Much smaller
Computation	Relatively expensive	Generally more efficient
Storage / Bandwidth	Higher	Lower
Digital Signature	RSA-PSS	ECDSA / EdDSA
Key Agreement	RSA key transport	ECDH
Deployment	Widely deployed, mature	Widely deployed, especially modern systems

Main Difference

ECC achieves comparable classical security with much smaller keys, making it attractive for bandwidth-, storage-, and computation-constrained systems.

Quantum Threat to Classical Public-Key Cryptography

The security of the major classical public-key schemes we have seen relies on problems that are vulnerable to quantum algorithms.

RSA $\longrightarrow$ Integer Factorization

DH / ElGamal $\longrightarrow$ DLP

ECC / ECDH / ECDSA $\longrightarrow$ ECDLP

A sufficiently powerful quantum computer can use **Shor's algorithm** to solve:

Factoring and Discrete Logarithms

in polynomial time.

Therefore,

RSA, DH, ElGamal and ECC are not post-quantum secure

The Problem

From Classical Cryptography to Post-Quantum Cryptography

RSA
DH / ElGamal
ECC

$\xrightarrow{\text{Shor's Algorithm}}$

Quantum Threat

This motivates the search for new public-key constructions whose underlying mathematical problems are believed to remain hard even for quantum computers.

Classical Public-Key Cryptography $\longrightarrow$ Post-Quantum Cryptography (PQC)

- Keep using **classical computers**.
- Replace vulnerable mathematical assumptions.
- Seek security against both classical and quantum attackers.

Next

We now look at public-key constructions based on mathematical problems that are believed to resist known quantum algorithms.

Transition to Post-Quantum Cryptography

We have seen that classical public-key cryptography can efficiently provide key establishment and digital signatures.

RSA and ECC provide efficient classical public-key cryptography, but neither survives a sufficiently powerful quantum computer.

$$\boxed{\text{Classical PKC} \longrightarrow \text{Quantum Threat} \longrightarrow \text{PQC}}$$

- **Classical PKC:** RSA, DH, ElGamal, ECC
- **Quantum Threat:** Shor's algorithm
- **PQC:** New public-key constructions designed to remain secure against quantum attacks

The key question now is:

$$\boxed{\text{What mathematical problems remain hard for quantum computers?}}$$

IISER THIRUVANANTHAPURAM

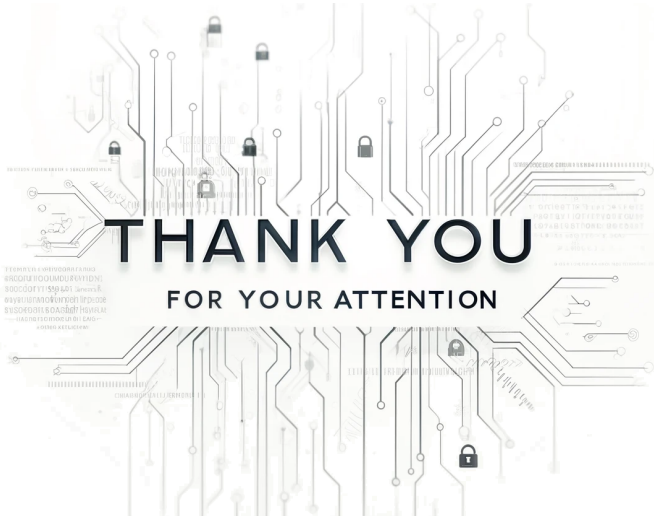
References

Textbook:

1. Douglas R. Stinson and Maura Paterson. *Cryptography: Theory and Practice*. 4th Edition. Chapman & Hall/CRC Press, 2019. ISBN: 9780367148591.

Acknowledgment:

- The figures in this work were generated with the assistance of ChatGPT (OpenAI).



Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com
Course webpage: https://laltu-sardar.github.io/courses/26_crypto.html.