NTNU | Norwegian University of Science and Technology

DISCRETE LOG PUBLIC KEY CRYPTOSYSTEMS

TTM4135 - Lecture 9

Tjerand Silde

02.02.2026

Motivation

- Discrete log cryptosystems are currently the main alternative to the RSA cryptosystem and the integer factorization hardness assumption
- Discrete log ciphers are widely deployed and standardized in practice
- When implemented on elliptic curves, discrete log cryptosystems are often more efficient than RSA due to much smaller parameters

Outline

Diffie–Hellman Key Exchange
- Protocol
- Properties

ElGamal Cryptosystem
- Algorithm
- Security

Elliptic Curves

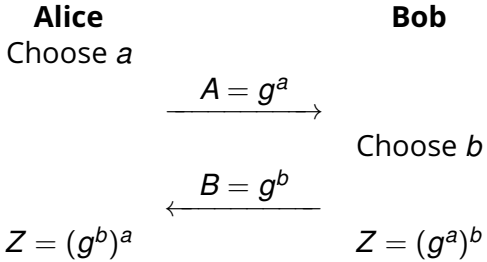
Elliptic Curve Cryptography

Contents

Diffie–Hellman Key Exchange

- Two users, Alice and Bob, want to agree upon a shared secret using only public communication over potentially untrusted networks
- Public knowledge: generator g of a multiplicative group G of order t
- Alice and Bob select random values a and b resp. where $0 < a, b < t$
- Alice sends $A = g^a$ to Bob (*over an insecure channel*)
- Bob sends $B = g^b$ to Alice (*over an insecure channel*)
- Alice and Bob both compute secret key $Z = g^{ab}$
- Originally the group G used was $\mathbb{Z}_p^*$ for some large prime p
- Today it is common to use an elliptic curve group for the group G

Diffie-Hellman Protocol



The shared secret value Z can be used to compute a key for, say, AES. This is done using a *key derivation function* based on a public hash function.

Example in $\mathbb{Z}_p^*$

Public knowledge is $p = 181, g = 2$:

- Alice selects random $a = 50$
- Bob selects random $b = 33$
- Alice sends $g^{50} \bmod 181 = 116$ to Bob
- Bob sends $g^{33} \bmod 181 = 30$ to Alice
- Alice computes $Z = 30^{50} \bmod 181$
- Bob computes $Z = 116^{33} \bmod 181$
- Common secret is $Z = 49$

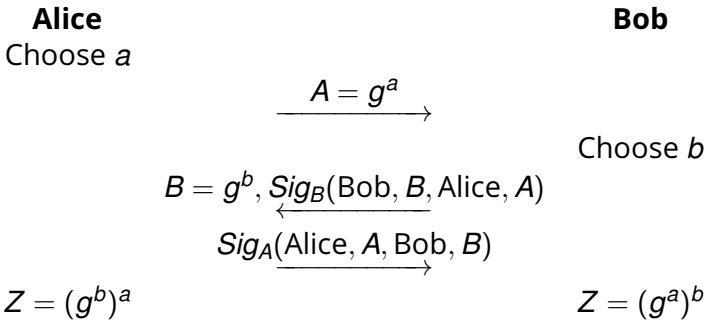
Security of Diffie–Hellman

- An attacker who can find discrete logarithms in G can break the protocol:
  1. intercept the value g^a and take the discrete log to obtain a
  2. compute g^{ab} in the same way as B and obtain the value Z
- It is unknown whether a better way exists to break the protocol than by computing discrete logarithms (we have proofs in ideal security models)
- The problem of finding $Z = g^{ab}$ from knowledge of g, g^a and g^b is known as the *(computational) Diffie–Hellman problem*

Authenticated Diffie–Hellman

- In the basic Diffie–Hellman protocol (as described above), the messages between Alice and Bob are not properly authenticated
- Neither Alice nor Bob knows who the secret Z is shared with unless the messages are authenticated and tied to the parties who communicate
- This allows a *man-in-the-middle* attack, where the adversary sets up two keys, one with Alice and one with Bob, and relays messages between them
- Authentication can be added in different ways, for example by adding *digital signatures* (see next lecture for how to construct digital signatures)

Authenticated Diffie-Hellman



- $Sig_A(m)$ is a digital signature on message m by Alice
  $Sig_B(m)$ is a digital signature on message m by Bob
- Assume that both parties can verify each other's signature

Static and Ephemeral Diffie–Hellman

- The Diffie–Hellman protocol described above uses *ephemeral keys*: keys which are used once and then discarded.
- In *static* Diffie–Hellman, Alice chooses a long-term private key x_A with corresponding public key $y_A = g^{x_A}$ for use with all parties.
- Similarly, Bob chooses a long-term private key x_B with corresponding public key $y_B = g^{x_B}$ for use with all parties.
- Now Alice and Bob can find a shared secret $S = g^{x_A x_B}$ just by looking up (or knowing beforehand) each others' public keys.
- The secret S is static: it stays the same long-term, until Alice and Bob change their public keys. We can also combine static and ephemeral keys.

Contents

ElGamal Cryptosystem

- Proposed by Taher ElGamal in 1985
- Turns the Diffie–Hellman protocol into an encryption scheme
- Here we look at original version of the scheme where the group G is $\mathbb{Z}_p^*$
- Alice combines her ephemeral private key with Bob's long-term public key

ElGamal Key Generation

- Select a prime p and a generator g of $\mathbb{Z}_p^*$.
- Select a long term private key x where $1 < x < p - 1$.
- Compute $y = g^x \bmod p$. The public key is (p, g, y).
- Often p and g are shared between all users in some system.

Encryption and Decryption

Encryption The public key is $K_E = (p, g, y)$

1. For any message M where $0 < M < p$
2. Sample random k where $1 < k < p - 1$
3. $C = E(M, K_E) = (g^k \bmod p, M \cdot y^k \bmod p)$

Decryption The private key is $K_D = x$ with $y = g^x \bmod p$

1. Let the ciphertext be $C = (C_1, C_2)$
2. $D(C, K_D) = C_2 \cdot C_1^{-x} \bmod p = M$

Why it works

- We can think of k as a one time mask of the message
- We are combining the ephemeral secret k of the sender with the static secret x and the public key y of the receiver
- $C_2 \cdot C_1^{-x} = (M \cdot y^k) \cdot (g^k)^{-x} = M \cdot (g^x)^k \cdot (g^{-x})^k \bmod p = M$

Powers of integers modulo 19

Table 2.7 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Stallings book, Table 2.7

Example

Key generation Choose prime $p = 181$ and generator $g = 2$

- Private key of A is $x = 50$
- Public key is $p = 181, g = 2, y = 116$

Encryption Sender wants to send $M = 97$

- Sender chooses random $k = 31$
- Ciphertext is $(98, 173) = (C_1, C_2)$

Decryption A receives (C_1, C_2) and recovers M by:

- $C_1^x = 98^{50} \bmod p = 138$
- $M = C_2 \times (C_1^x)^{-1} \bmod p$
  $= 173 \times 138^{-1} \bmod 181 = 97$

Security of ElGamal

- If an attacker can solve the discrete log problem, the system can be broken by determining the private key x from $g^x \bmod p$.
- It is quite possible for many users to share the same p and g values.
- There is no need for padding as in RSA – each ciphertext is randomized.
- The ElGamal cryptosystem has a proof of security in a suitable model subject to the difficulty of the so-called *decision Diffie–Hellman problem*.

Discrete Log Problem and Factoring

Solving the discrete log problem over $\mathbb{Z}_p$ is comparable to the difficulty of factoring n, where n is the product of two primes, i.e. if the number of bits in n is the same as the number of bits in p.

Hence, the discrete log cryptosystems modulo p offer the same level of security as the RSA algorithm. But we can do better...

Contents

What are elliptic curves?

- Elliptic curves are algebraic structures formed from cubic equations.
- An example is the set of all (x, y) pairs which satisfy the equation:

$$y^2 = x^3 + ax + b \bmod p$$

- This example is a curve over the field $\mathbb{Z}_p$ but elliptic curves can be defined over any field.
- Once an identity element is added, a binary operation (like addition or multiplication) can be defined on these points like any other group.
- With this operation the elliptic curve points form an *elliptic curve group*.

An Example

- Recall $y^2 = x^3 + ax + b \bmod p$. We will call this curve $E_p(a, b)$.
- Let $a = b = 1$, and $p = 23$, so $E_{23}(1, 1) : y^2 = x^3 + x + 1 \bmod 23$.
- We can show that $x = 9$ and $y = 7$ satisfies the curve equation:
  - $7^2 = 49 \equiv 3 \bmod 23$.
  - $9^3 + 9 + 1 \equiv 3 \bmod 23$.
- Therefore, the point $P = (9, 7)$ is on the elliptic curve $E_{23}(1, 1)$.

Other points on $E_{23}(1, 1)$

Table 10.1 Points (other than O) on the Elliptic Curve $E_{23}(1, 1)$

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

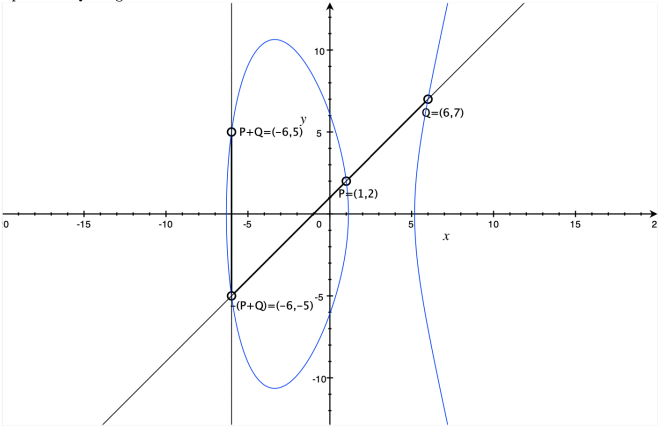
Stallings book, Table 10.1

Elliptic Curve Computations

- The elliptic curve group operation could be denoted by any symbol, but by convention is it usually called elliptic curve *addition* with a "+".
- We write $P + Q = R$ to show the group operation on curve points P and Q with result R, which will be a new point satisfying the curve equation.
- The *elliptic curve discrete log problem* is to find the value of k, given a point P and a generator G so that $P = [k]G = G + G + \ldots + G$ (k times).
- Efficient computation of elliptic curve scalar multiplication can use the *double-and-add algorithm*, by replacing multiplication in the square and multiply algorithm with addition.

Elliptic Curve – Visual Representation

Example: Let $E : y^2 = x^3 - 34x + 37$ be defined over $\mathbb{Q}$, $P = (1, 2)$ and $Q = (6, 7)$. We will compute $P + Q$ using the above formulas.



https://www.umsl.edu/~siegelj/information_theory/projects/elliptic_curves_group_law.pdf

Elliptic Curve Addition

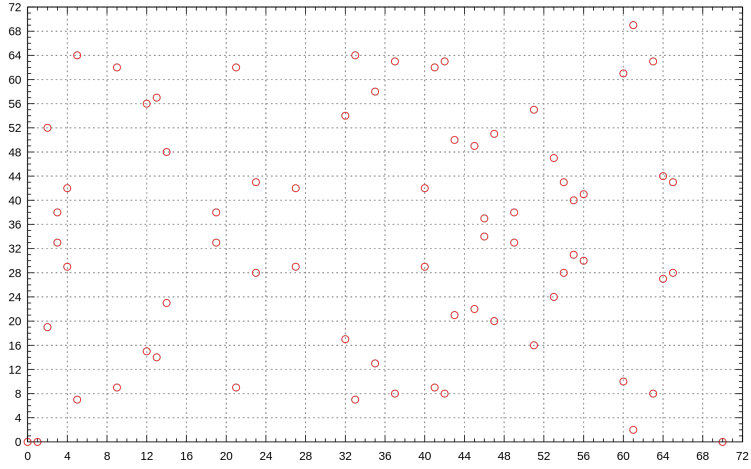
Given two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$,
we compute the sum $P + Q$ in the following way:

1. If $P = \mathcal{O}$, output Q. If $Q = \mathcal{O}$, output P.
2. If $x_1 = x_2$ and $y_2 = -y_1$, then output $\mathcal{O}$.
3. Otherwise, let $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = -y_1 - \lambda \cdot (x_3 - x_1)$, where

$$\lambda = \begin{cases} \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \\ \frac{y_1 - y_2}{x_1 - x_2} & \text{otherwise,} \end{cases}$$

and output $R = (x_3, y_3)$.

Elliptic Curve – Visual Representation



Representation of $y^2 = x^3 - x$ over the finite field $\mathbb{F}_{71}$

https://en.wikipedia.org/wiki/Elliptic_curve

Elliptic Curve Representations

There are several different ways of representing elliptic curves which use different point formats and different ways to computing the group operation:

Short Weierstrass form is the most common format as shown until now

Montgomery form allows a fixed time elliptic curve multiplication which is useful to avoid timing side-channel attacks

Edwards form allows for faster group operation computations

It is common to shift between representations to optimize performance

Choosing Elliptic Curves

- A new elliptic curve can be generated at any time, but common applications usually use a few standard curves
- Various predefined sets of curves exist such as the set of NIST curves contained in the standard SP 800-186 (2023)
- Desirable that standard curves are generated in a way to ensure there are no hidden special properties but researchers have disputed this for some

Example – NIST Curve P-256

$$
\begin{aligned}
p &= 115792089210356248762697446949407573530086143415290314195533631308867097853951 \\
n &= 115792089210356248762697446949407573529996955224135760342422259061068512044369 \\
s &= \texttt{c49d360886e704936a6678e1139d26b7819f7e90} \\
c &= \texttt{7efba1662985be9403cb055c75d4f7e0ce8d84a9c5114abcaf3177680104fa0d} \\
b &= \texttt{5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b} \\
G_x &= \texttt{6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296} \\
G_y &= \texttt{4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5}
\end{aligned}
$$

- Curve of n points (256-bits) over $\mathbb{Z}_p$ with generator (G_x, G_y) and equation: $y^2 = x^3 - 3x + b \bmod p$ (all NIST-curves has $a = -3 \bmod p$ for efficiency).
- Parameter s is the seed for the random generation and c is the output of a SHA-1 hash generated from s. b is then set so that $b^2 \cdot c \equiv -27 \bmod p$.

Curve 25519

- A curve allowing very fast computations and uniform addition formulas
- Proposed by Bernstein in 2005 and is now a NIST recommended curve
- The equation is $y^2 = x^3 + 486662x^2 + x$ which is in Montgomery form
- The field is the integers modulo p, where $p = 2^{255} - 19$ is prime

Contents

Discrete Logarithms on Elliptic Curves

- The discrete logarithm in elliptic curve groups is to find k, given a point P and a generator G, so that $P = [k]G = G + G + \ldots + G$ (k times).
- This is the same as in $\mathbb{Z}_p^*$ but with elliptic curve point addition as the group operation instead of modular multiplication.
- The best known algorithms for solving the elliptic curve discrete log problem are *exponential* in the length of the parameters;
  - Sub-exponential algorithms exist for factoring and finite field discrete log .
- Consequently elliptic curve implementations use much smaller keys.
- Compared with RSA the relative advantage of elliptic curve cryptography will *increase* at higher security levels.

Comparing Strength of Elliptic Curve Cryptography

Symmetric key length	RSA modulus length or length of p in $\mathbb{Z}_p^*$	Elliptic curve group size
80	1024	160
128	3072	256
192	7680	384
256	15360	512

- Brute force search of 128-bit key for AES takes roughly same computational effort as factorization of 3072-bit RSA modulus or for taking discrete logarithms in $\mathbb{Z}_p^*$ with p of 3072-bits or on an elliptic curve with elements of size 256 bits.
- NIST SP 800-57 Part 1, Recommendations for Key Management (2020)

Elliptic Curve Cryptography

- Most cryptosystems based on discrete logarithms can be constructed with elliptic curves as well as $\mathbb{Z}_p^*$. Smaller parameters but more complex.
- In particular, Diffie–Hellman key exchange and ElGamal encryption can directly be translated to elliptic curves to improve the efficiency.
- Elliptic curve cryptography is widely deployed in TLS 1.3 and elsewhere.
- We can also build signatures schemes from elliptic curve assumptions.

Questions?