Public Key Cryptography

Lecture 12

Elliptic Curve Cryptography

Index

Elliptic Curves - Motivation

- **Problem:**
  Asymmetric schemes like RSA and ElGamal require exponentiations in integer rings and fields with parameters of more than 1000 bits.
  High computational effort on CPUs with 32-bit or 64-bit arithmetic.
  Large parameter sizes critical for storage on small and embedded ones.
- **Motivation:**
  Smaller field sizes providing equivalent security are desirable.
- **Solution:**
  Elliptic Curve Cryptography uses a group of points (instead of integers) for cryptographic schemes with coefficient sizes of 160-256 bits, reducing significantly the computational effort.

Elliptic Curves

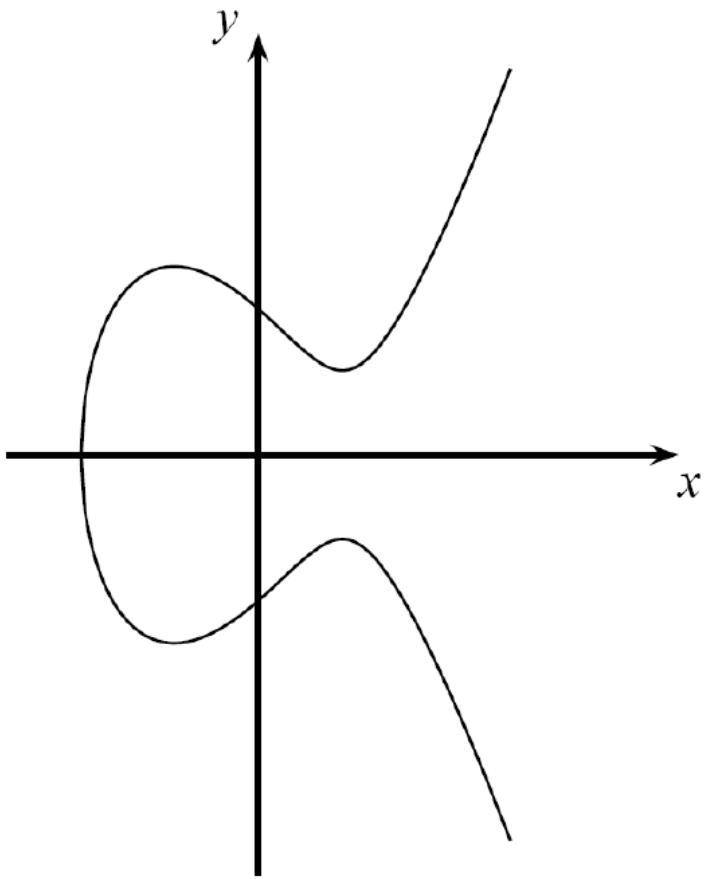
- Elliptic curves are polynomials that define points based on the (simplified) Weierstrass equation:

$$y^2 = x^3 + ax + b$$

  for parameters a, b that specify the exact shape of the curve.
- Elliptic curves cannot just be defined over the real numbers $\mathbb{R}$, but over many other types of finite fields.

Elliptic Curves (cont.)

On the real numbers and with parameters $a, b \in \mathbb{R}$, an elliptic curve looks like this:



Example: $y^2 = x^3 - 3x + 3$ over R

Elliptic Curves in Cryptography

In cryptography, we are interested in elliptic curves modulo a prime number $p > 3$:

Definition

An elliptic curve over $\mathbb{Z}_p$ is the set of all pairs $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ which fulfill

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

together with an imaginary point at infinity θ, where $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2 \neq 0 \bmod p$.

The last condition on a and b ensures that the elliptic curve is non-singular, that is, the plot has no self-intersections or vertices.

Computations on Elliptic Curves

Some special considerations are required to convert elliptic curves into a group of points:

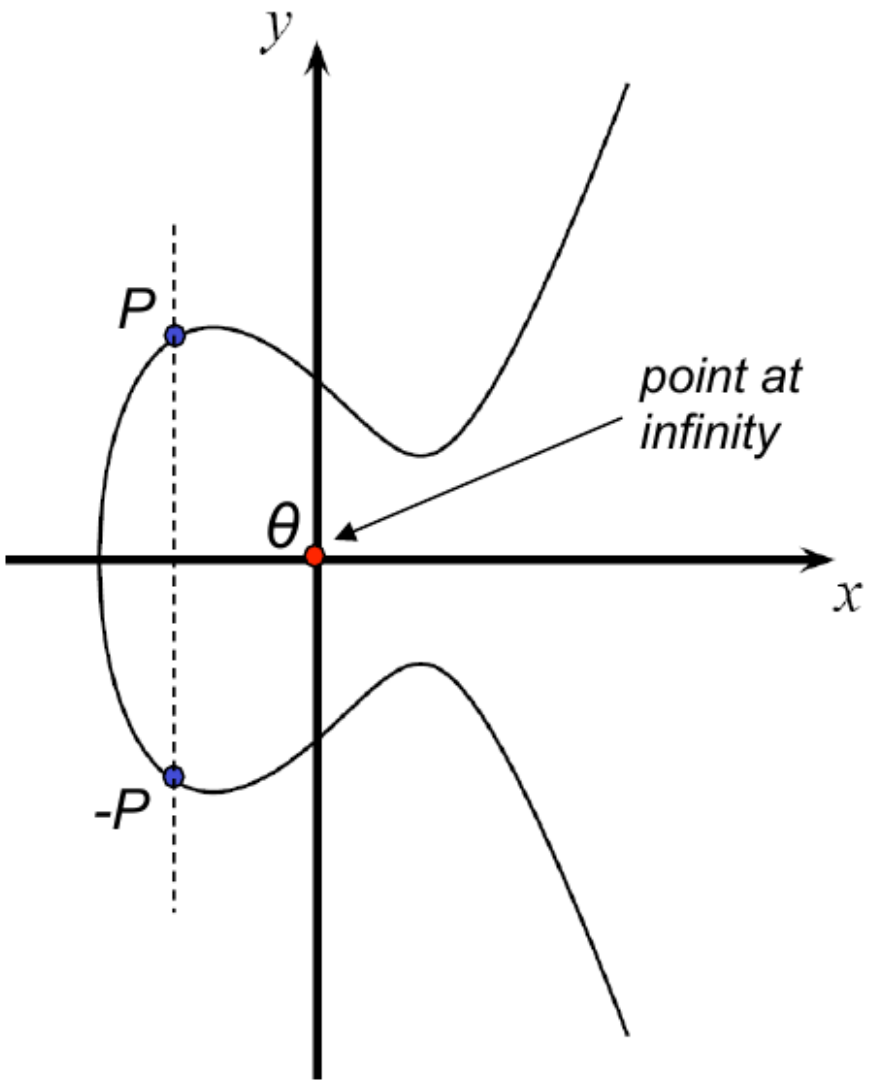
- In any group, a special element is required to allow for the identity operation, i.e., given $P \in E$:

$$P + \theta = P = \theta + P.$$

  This identity point (which is not on the curve) is additionally added to the group definition.
  This (infinite) identity point is denoted by θ.

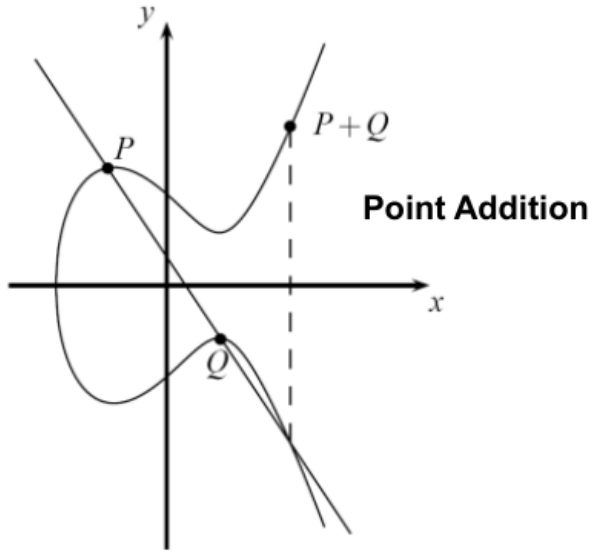
- Elliptic curves are symmetric along the x-axis.
  Up to two solutions y and $-y$ exist for each quadratic residue x of the elliptic curve.
  For each point $P = (x, y)$, the inverse or negative point is defined as $-P = (x, -y)$.

Computations on Elliptic Curves (cont.)



Computations on Elliptic Curves (cont.)

- Generating a *group of points* on elliptic curves based on point addition operation $P+Q = R$, *i.e.*, $(x_P,y_P)+(x_Q,y_Q) = (x_R,y_R)$
- Geometric Interpretation of point addition operation
  - *Draw straight line through P and Q; if P=Q use tangent line instead*
  - *Mirror third intersection point of drawn line with the elliptic curve along the x-axis*

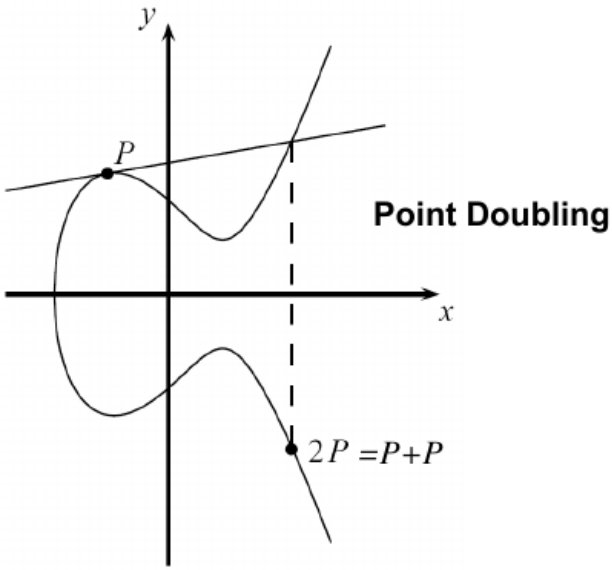


- Elliptic Curve Point Addition and Doubling Formulas

$x_3 = s_2 - x_1 - x_2 \bmod p$ and $y_3 = s(x_1 - x_3) - y_1 \bmod p$

where

$$s = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \bmod p & \text{; if P} \neq \text{Q (point addition)} \\ \dfrac{3x_1^2 + a}{2y_1} \bmod p & \text{; if P = Q (point doubling)} \end{cases}$$



Computations on Elliptic Curves - Example

Consider the elliptic curve

$$E : y^2 = x^3 + 2x + 2 \mod 17$$

and the point $P = (5, 1)$.
Let us compute $2P = P + P = (x_1, y_1) + (x_2, y_2)$. We have:

$$2P = P + P = (5, 1) + (5, 1) = (x_3, y_3).$$

Compute:

$$s = \frac{3x_1^2 + a}{2y_1} = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 = 9 \cdot 9 = 13 \mod 17.$$

$$x_3 = s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 = 6 \mod 17.$$

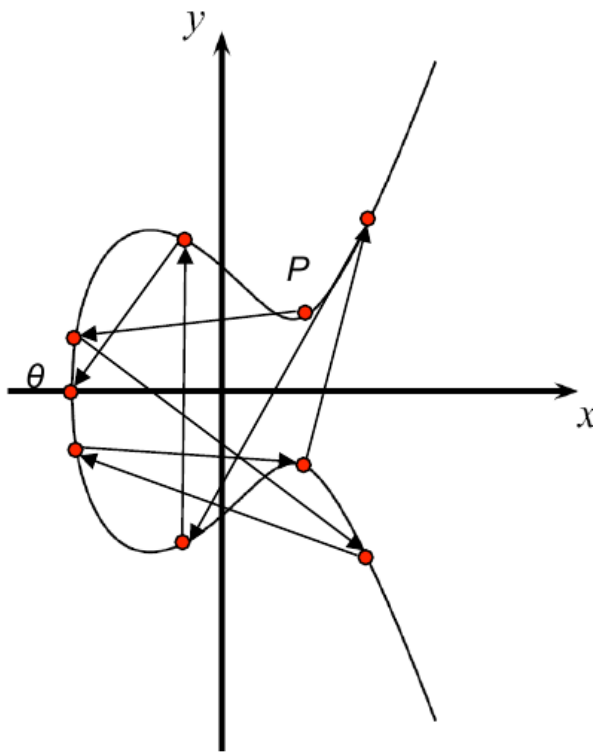
$$y_3 = s(x_1 - x_3) - y_1 = 13 \cdot (5 - 6) - 1 = -14 = 3 \mod 17.$$

Finally, we have $2P = (5, 1) + (5, 1) = (6, 3)$.

Computations on Elliptic Curves (cont.)

The points on an elliptic curve and the point at infinity θ form cyclic subgroups

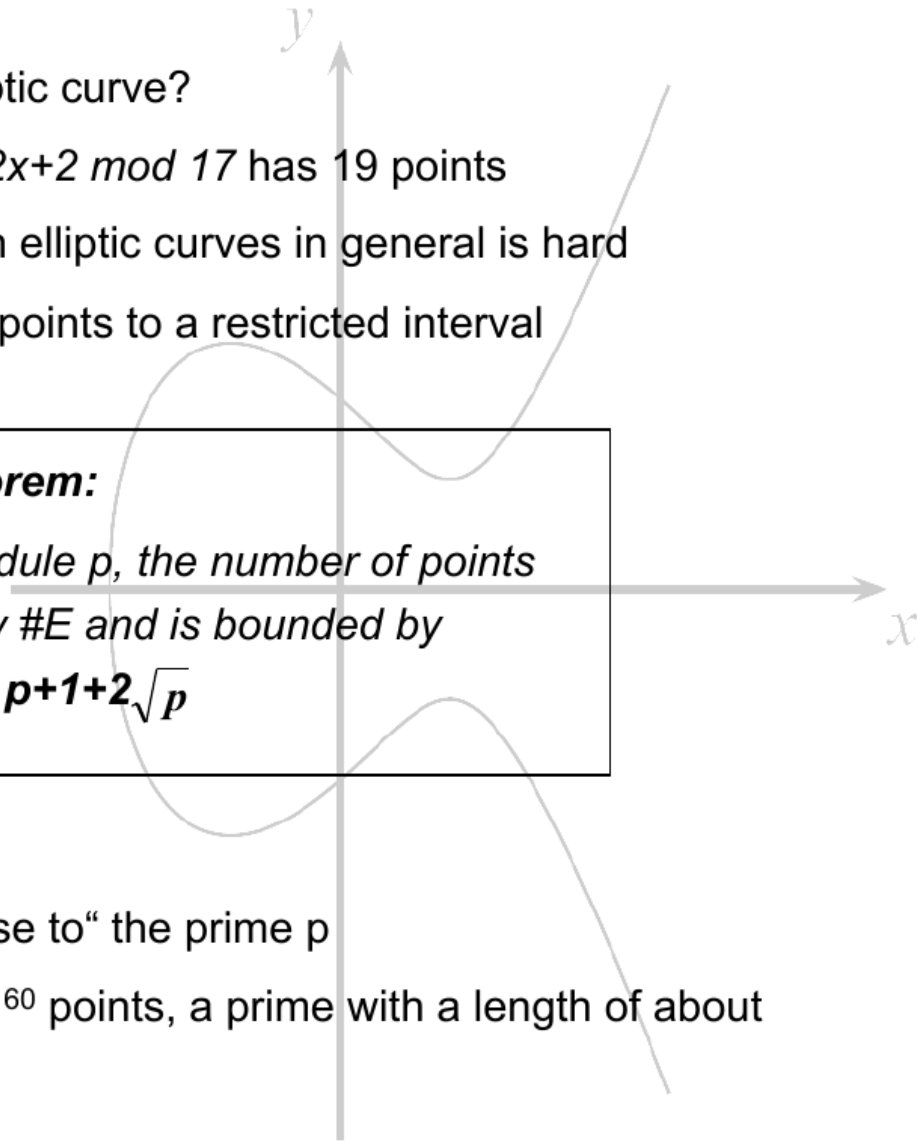
$2P = (5,1)+(5,1) = (6,3)$
$3P = 2P+P = (10,6)$
$4P = (3,1)$
$5P = (9,16)$
$6P = (16,13)$
$7P = (0,6)$
$8P = (13,7)$
$9P = (7,6)$
$10P = (7,11)$
$11P = (13,10)$
$12P = (0,11)$
$13P = (16,4)$
$14P = (9,1)$
$15P = (3,16)$
$16P = (10,11)$
$17P = (6,14)$
$18P = (5,16)$
$19P = \theta$



This elliptic curve has order #E = |E| = 19 since it contains 19 points in its cyclic group.

Number of Points on an Elliptic Curve

- How many points can be on an arbitrary elliptic curve?
  - Consider previous example: *E:* $y^2 = x^3+2x+2 \ mod \ 17$ has 19 points
  - However, determining the point count on elliptic curves in general is hard
- But Hasse's theorem bounds the number of points to a restricted interval



Definition: Hasse's Theorem:

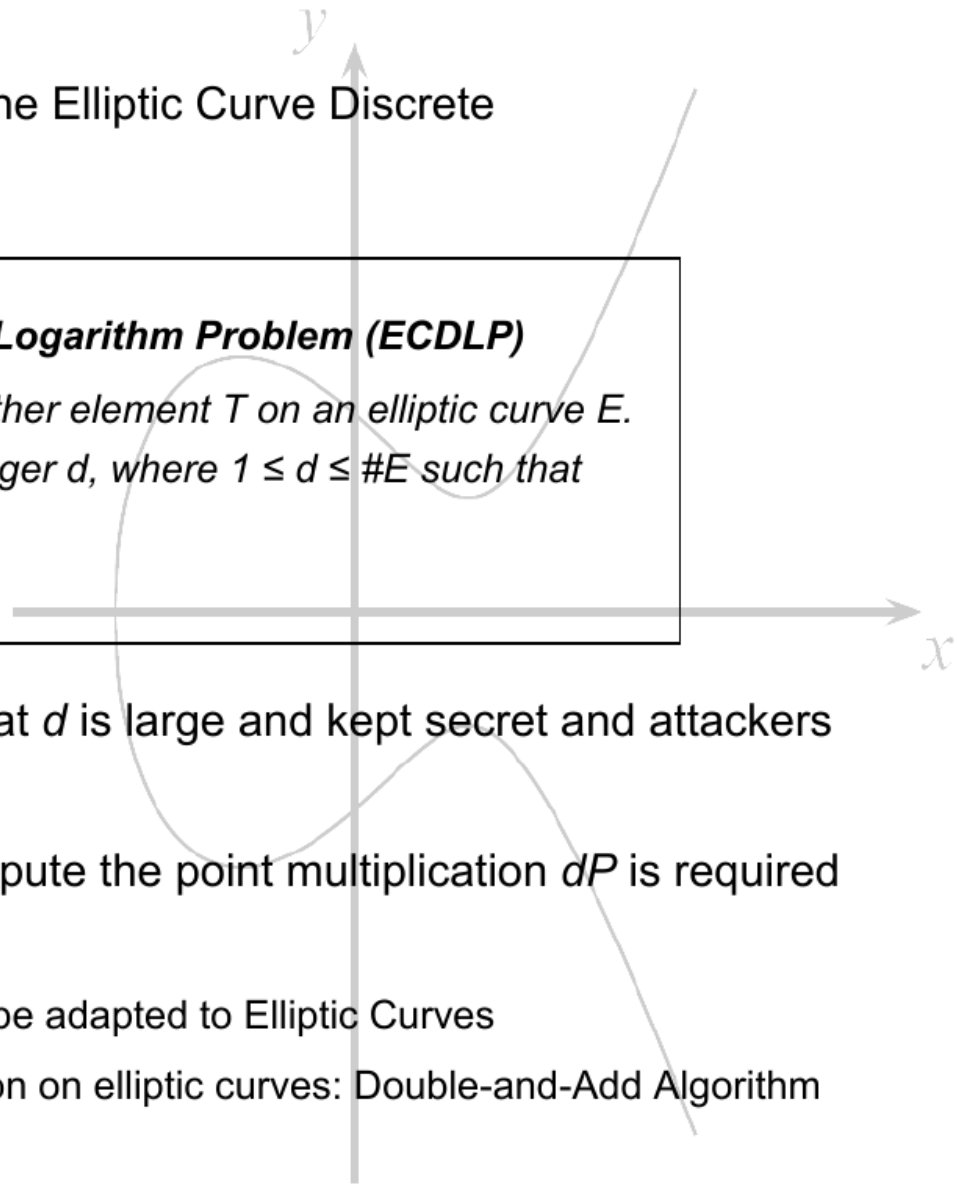
Given an elliptic curve module p, the number of points on the curve is denoted by #E and is bounded by

$$p+1-2\sqrt{p} \leq \#E \leq p+1+2\sqrt{p}$$

- **Interpretation:** The number of points is „close to" the prime p
- **Example:** To generate a curve with about 2^{160} points, a prime with a length of about 160 bits is required

Elliptic Curve Discrete Logarithm Problem

- Cryptosystems rely on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP)



Definition: Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given a primitive element P and another element T on an elliptic curve E. The ECDL problem is finding the integer d, where $1 \leq d \leq \#E$ such that

$$\underbrace{P + P + \ldots + P}_{d \text{ times}} = dP = T.$$

- Cryptosystems are based on the idea that d is large and kept secret and attackers cannot compute it easily
- If d is known, an efficient method to compute the point multiplication dP is required to create a reasonable cryptosystem
  - Known Square-and-Multiply Method can be adapted to Elliptic Curves
  - The method for efficient point multiplication on elliptic curves: Double-and-Add Algorithm

Double-and-Add Algorithm for Point Multiplication

- **Double-and-Add Algorithm**

Input: Elliptic curve E, an elliptic curve point P and a scalar d with bits d_i
Output: $T = d\,P$

Initialization:

$T = P$

Algorithm:

FOR $i = t - 1$ DOWNTO 0

  $T = T + T \bmod n$

  IF $d_i = 1$

    $T = T + P \bmod n$

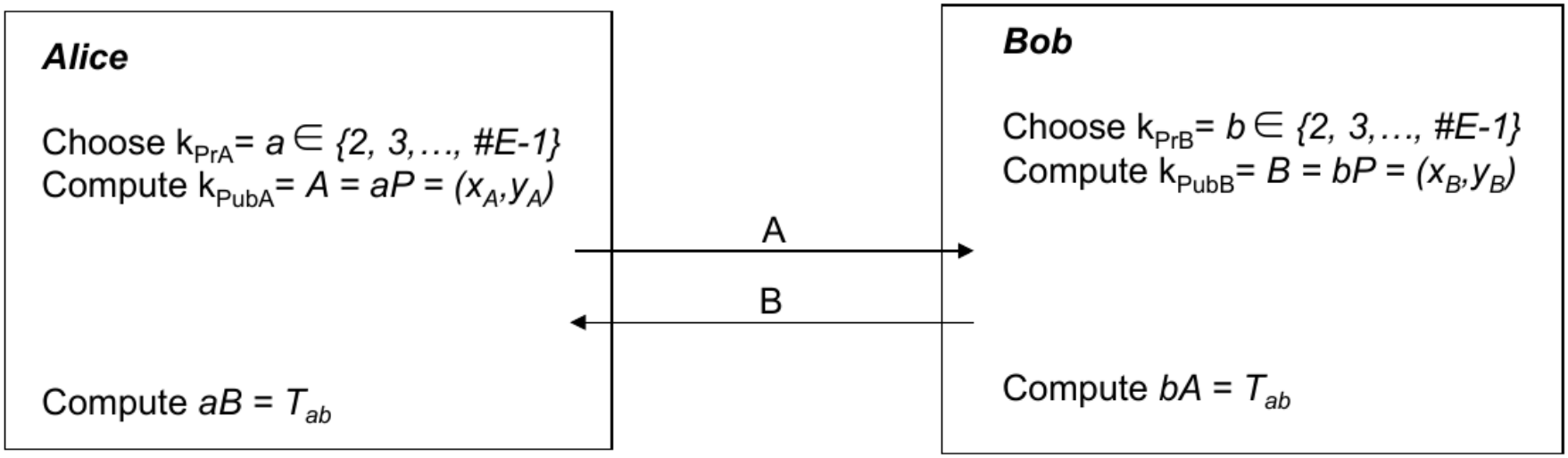
RETURN (T)

Example: $26P = (11010_2)P = (d_4 d_3 d_2 d_1 d_0)_2\, P$.

Step		
#0	$P = \mathbf{1}_2 P$	inital setting
#1*a*	$P+P = 2P = \mathbf{10}_2 P$	DOUBLE (bit d_3)
#1*b*	$2P+P = 3P = 10^2\, P+1_2 P = \mathbf{11}_2 P$	ADD (bit d_3=1)
#2*a*	$3P+3P = 6P = 2(11_2 P) = \mathbf{110}_2 P$	DOUBLE (bit d_2)
#2*b*		no ADD ($d_2 = 0$)
#3*a*	$6P+6P = 12P = 2(110_2 P) = \mathbf{1100}_2 P$	DOUBLE (bit d_1)
#3*b*	$12P+P = 13P = 1100_2 P+1_2\, P = \mathbf{1101}_2 P$	ADD (bit d_1=1)
#4*a*	$13P+13P = 26P = 2(1101_2 P) = \mathbf{11010}_2 P$	DOUBLE (bit d_0)
#4*b*		no ADD ($d_0 = 0$)

Elliptic Curve Diffie-Hellman Protocol (ECDH)

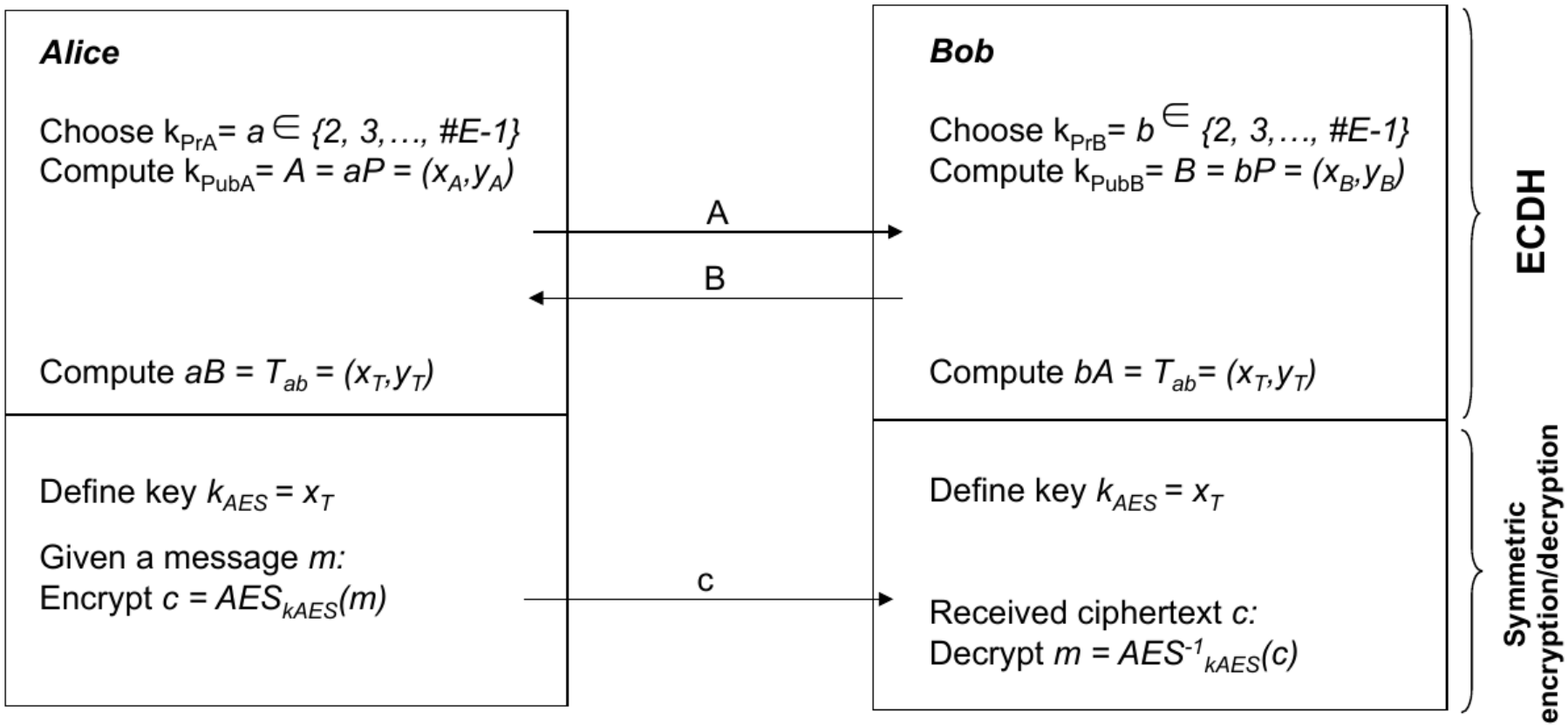
- Given a prime p, a suitable elliptic curve E and a point $P=(x_P,y_P)$
- The Elliptic Curve Diffie-Hellman Key Exchange is defined by the following protocol:



- Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$
- Proof for correctness:
  - Alice computes aB=a(bP)=abP
  - Bob computes bA=b(aP)=abP since group is associative
- One of the coordinates of the point T_{AB} (usually the x-coordinate) can be used as session key (often after applying a hash function)

Elliptic Curve Diffie-Hellman Protocol (ECDH) (cont.)

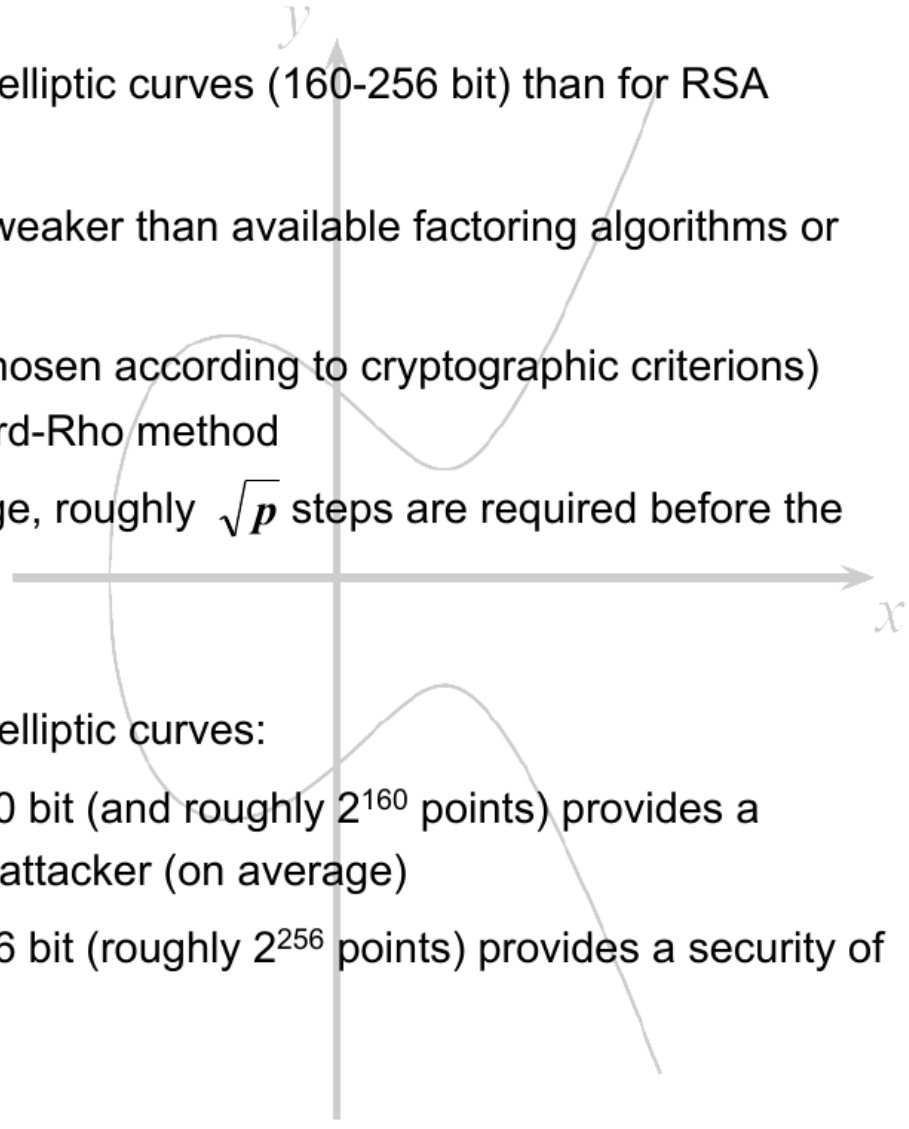
- The ECDH is often used to derive session keys for (symmetric) encryption
- One of the coordinates of the point T_{AB} (usually the x-coordinate) is taken as session key



- In some cases, a hash function (see next chapters) is used to derive the session key

ECDH - Security Aspects

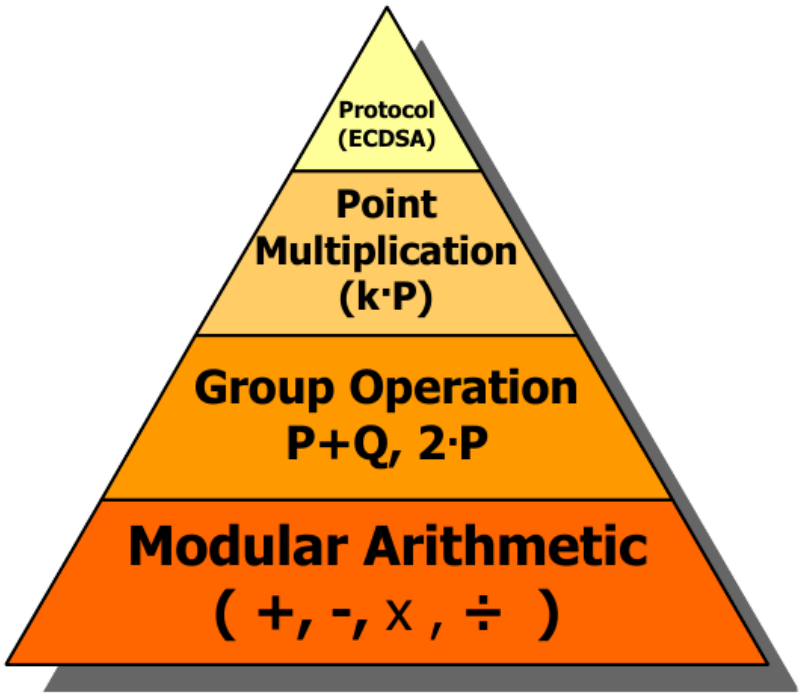
- Why are parameters signficantly smaller for elliptic curves (160-256 bit) than for RSA (1024-3076 bit)?
  - Attacks on groups of elliptic curves are weaker than available factoring algorithms or integer DL attacks
  - Best known attacks on elliptic curves (chosen according to cryptographic criterions) are the Baby-Step Giant-Step and Pollard-Rho method
  - Complexity of these methods: on average, roughly $\sqrt{p}$ steps are required before the ECDLP can be successfully solved



- Implications to practical parameter sizes for elliptic curves:
  - An elliptic curve using a prime p with 160 bit (and roughly 2^{160} points) provides a security of 2^{80} steps that required by an attacker (on average)
  - An elliptic curve using a prime p with 256 bit (roughly 2^{256} points) provides a security of 2^{128} steps on average

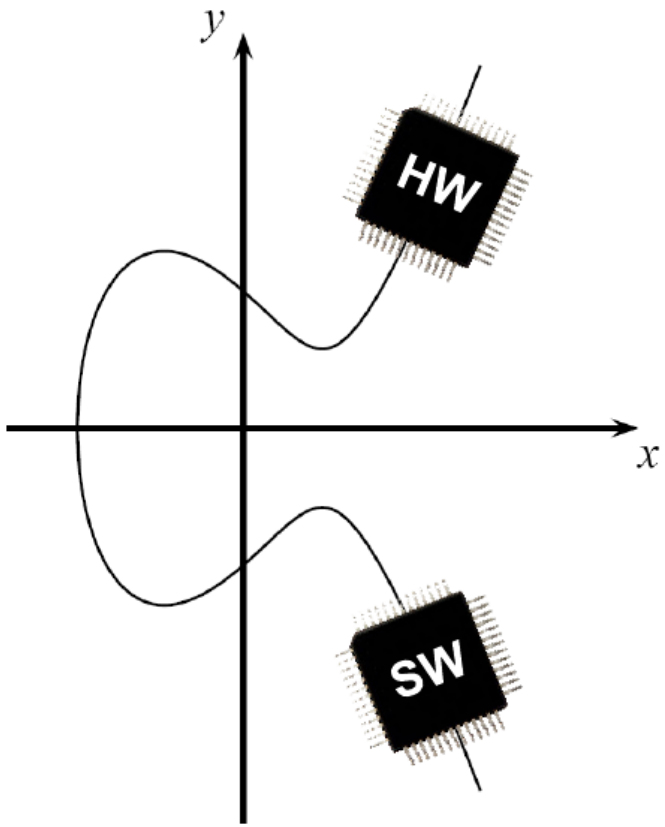
Implementations in Hardware and Software

- Elliptic curve computations usually regarded as consisting of four layers:
  - Basic modular arithmetic operations are computationally most expensive
  - Group operation implements point doubling and point addition
  - Point multiplication can be implemented using the Double-and-Add method
  - Upper layer protocols like ECDH and ECDSA
- Most efforts should go in optimizations of the modular arithmetic operations, such as
  - Modular addition and subtraction
  - Modular multiplication
  - Modular inversion



Implementations in Hardware and Software (cont.)

- Software implementations
  - Optimized 256-bit ECC implementation on 3GHz 64-bit CPU requires about *2 ms* per point multiplication
  - Less powerful microprocessors (e.g, on SmartCards or cell phones) even take significantly longer (*>10 ms*)
- Hardware implementations
  - High-performance implementations with 256-bit special primes can compute a point multiplication in a few hundred microseconds on reconfigurable hardware
  - Dedicated chips for ECC can compute a point multiplication even in a few ten microseconds



Elliptic Curve Digital Signature Algorithm (ECDSA)

The ECDSA standard is defined for elliptic curves over $\mathbb{Z}_p$ and Galois fields $GF(2^m)$, the former being often preferred in practice.

Key Generation for ECDSA

1. Use an elliptic curve E with
   - modulus p
   - coefficients a and b
   - a point A which generates a cyclic group of prime order q
2. Choose a random integer d with $0 < d < q$.
3. Compute $B = dA$.

The keys are now:
$k_{pub} = (p, a, b, q, A, B)$
$k_{pr} = (d)$

Similar to DSA, the cyclic group has an order q which should have a size of at least 160 bit or more for higher security levels.

ECDSA - Signature and Verification

ECDSA Signature Generation

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $R = k_E A$.
3. Let $r = x_R$.
4. Compute $s \equiv (h(x) + d \cdot r)\, {k_E}^{-1} \bmod q$.

ECDSA Signature Verification

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$.
2. Compute auxiliary value $u_1 \equiv w \cdot h(x) \bmod q$.
3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q$.
4. Compute $P = u_1 A + u_2 B$.
5. The verification $ver_{k_{pub}}(x,(r,s))$ follows from:

$$x_P \begin{cases} \equiv r \bmod q \implies \text{valid signature} \\ \not\equiv r \bmod q \implies \text{invalid signature} \end{cases}$$

ECDSA - Proof of Verification

Proof. Consider the hash value $h(x)$ of the message x. We show that (r, s) verifies the condition $r = x_P \mod q$. We have:

$$s = (h(x) + dr)k_E^{-1} \mod q,$$

which is equivalent to:

$$k_E = s^{-1}h(x) + ds^{-1}r \mod q.$$

and further to:

$$k_E = u_1 + du_2 \mod q.$$

Since the point A generates a cyclic group of order q, we can multiply both sides of the equation with A and we get:

$$k_E A = (u_1 + du_2)A = u_1 A + du_2 A = u_1 A + u_2 B.$$

But this is the condition that we check in the verification process by comparing the x-coordinates of $P = u_1 A + u_2 B$ and $R = k_E A$.

ECDSA - Example

Note that finding an elliptic curve with good cryptographic properties is a nontrivial task.
Bob wants to send a message to Alice, signed with ECDSA. He chooses the elliptic curve:

$$E : y^2 = x^3 + 2x + 2 \mod 17.$$

Because all points of the curve form a cyclic group of prime order 19, there are no subgroups and hence in this case $q = \#E = 19$.

Bob - key generation:

- Choose E with $p = 17$, $a = 2$, $b = 2$, and $A = (5, 1)$ with $q = 19$.
- Choose $d = 7$.
- Compute $B = dA = 7 \cdot (5, 1) = (0, 6)$.
- Send $(p, a, b, q, A, B) = (17, 2, 2, 19, (5, 1), (0, 6))$ to Alice.

ECDSA - Example (cont.)

Bob - signature:

- Compute the hash of a message x, say $h(x) = 26$.
- Choose ephemeral key $k_E = 10$
- Compute $R = 10 \cdot (5, 1) = (7, 11)$, so $r = x_R = 7$.
- Compute $s = (26 + 7 \cdot 7) \cdot 2 = 17 \mod 19$.
- Send $(x, (r, s)) = (x, (7, 17))$ to Alice.

Alice - verification:

- Compute $w = 17^{-1} = 9 \mod 19$.
- Compute $u_1 = 9 \cdot 26 = 6 \mod 19$.
- Compute $u_2 = 9 \cdot 7 = 6 \mod 19$.
- Compute $P = 6 \cdot (5, 1) + 6 \cdot (0, 6) = (7, 11)$.
- Check $x_P = r \mod 19$, so it is a valid signature.

Conclusions

- Elliptic Curve Cryptography (ECC) is based on the discrete logarithm problem. It requires, for instance, arithmetic modulo a prime.
- ECC can be used for key exchange, for digital signatures and for encryption.
- ECC provides the same level of security as RSA or discrete logarithm systems over $\mathbb{Z}_p$ with considerably shorter operands (approximately 160-256 bit vs. 1024-3072 bit), which results in shorter ciphertexts and signatures.
- In many cases ECC has performance advantages over other public-key algorithms.
- ECC is slowly gaining popularity in applications, compared to other public-key schemes, i.e., many new applications, especially on embedded platforms, make use of elliptic curve cryptography.

Selective Bibliography


T. Güneysu, C. Paar, J. Pelzl, *Elliptic Curve Cryptography slides*. www.crypto-textbook.com


C. Paar, J. Pelzl, *Understanding Cryptography*, Springer, 2009.