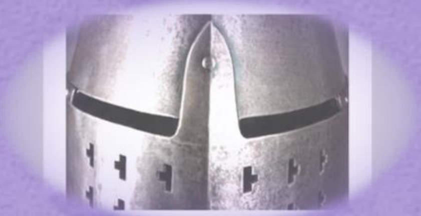


Cryptography and Network Security

Sixth Edition

by William Stallings

Chapter 10

Other Public-Key Cryptosystems

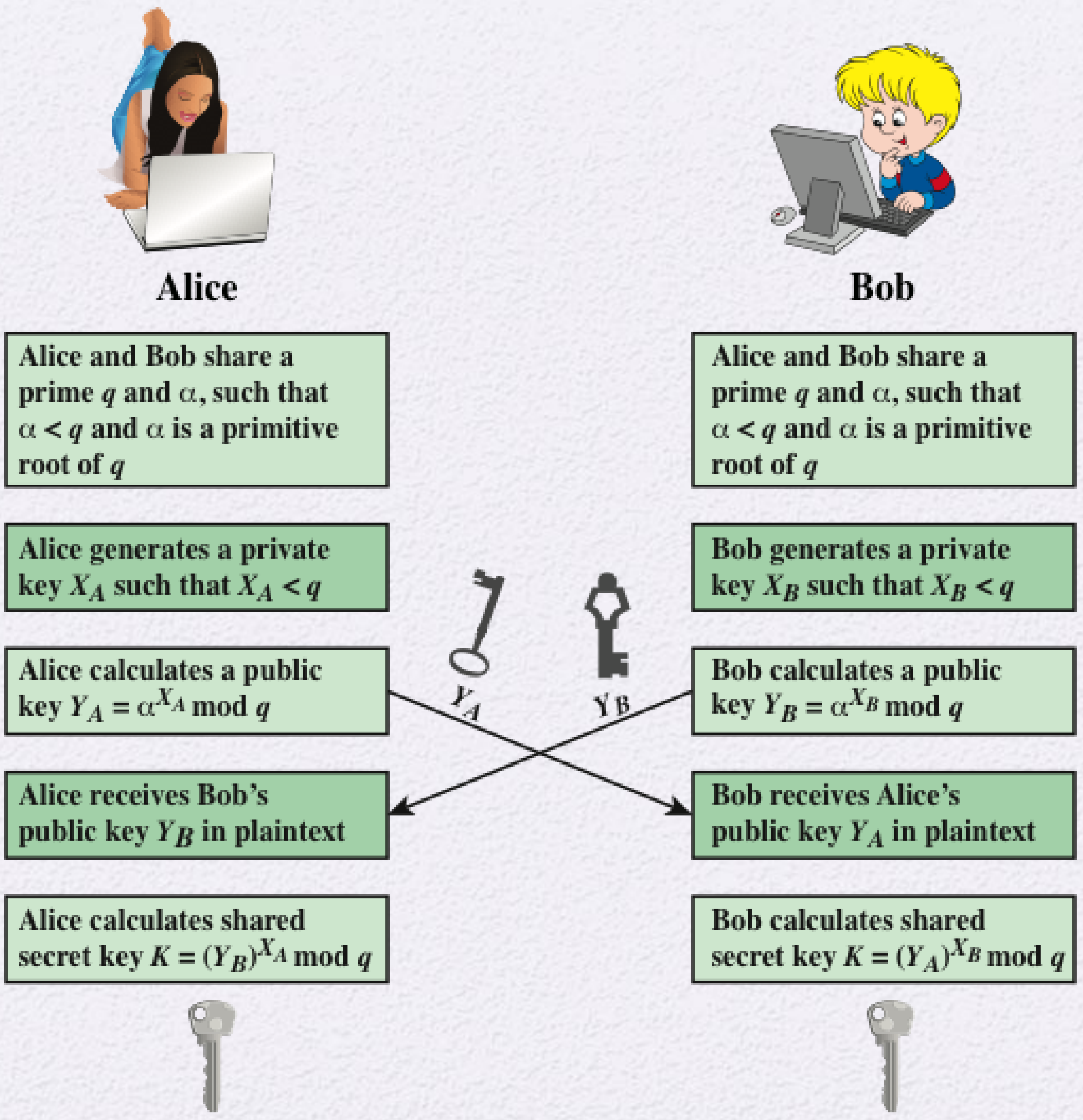
"Amongst the tribes of Central Australia every man, woman, and child has a secret or sacred name which is bestowed by the older men upon him or her soon after birth, and which is known to none but the fully initiated members of the group. This secret name is never mentioned except upon the most solemn occasions; to utter it in the hearing of men of another group would be a most serious breach of tribal custom. When mentioned at all, the name is spoken only in a whisper, and not until the most elaborate precautions have been taken that it shall be heard by no one but members of the group. The native thinks that a stranger knowing his secret name would have special power to work him ill by means of magic."

—The Golden Bough,

Sir James George Frazer

Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages
- The algorithm itself is limited to the exchange of secret values
- Its effectiveness depends on the difficulty of computing discrete logarithms



Alice and Bob share a prime q and α, such that $\alpha < q$ and α is a primitive root of q

Alice generates a private key X_A such that $X_A < q$

Alice calculates a public key $Y_A = \alpha^{X_A} \bmod q$

Alice receives Bob's public key Y_B in plaintext

Alice calculates shared secret key $K = (Y_B)^{X_A} \bmod q$

Alice and Bob share a prime q and α, such that $\alpha < q$ and α is a primitive root of q

Bob generates a private key X_B such that $X_B < q$

Bob calculates a public key $Y_B = \alpha^{X_B} \bmod q$

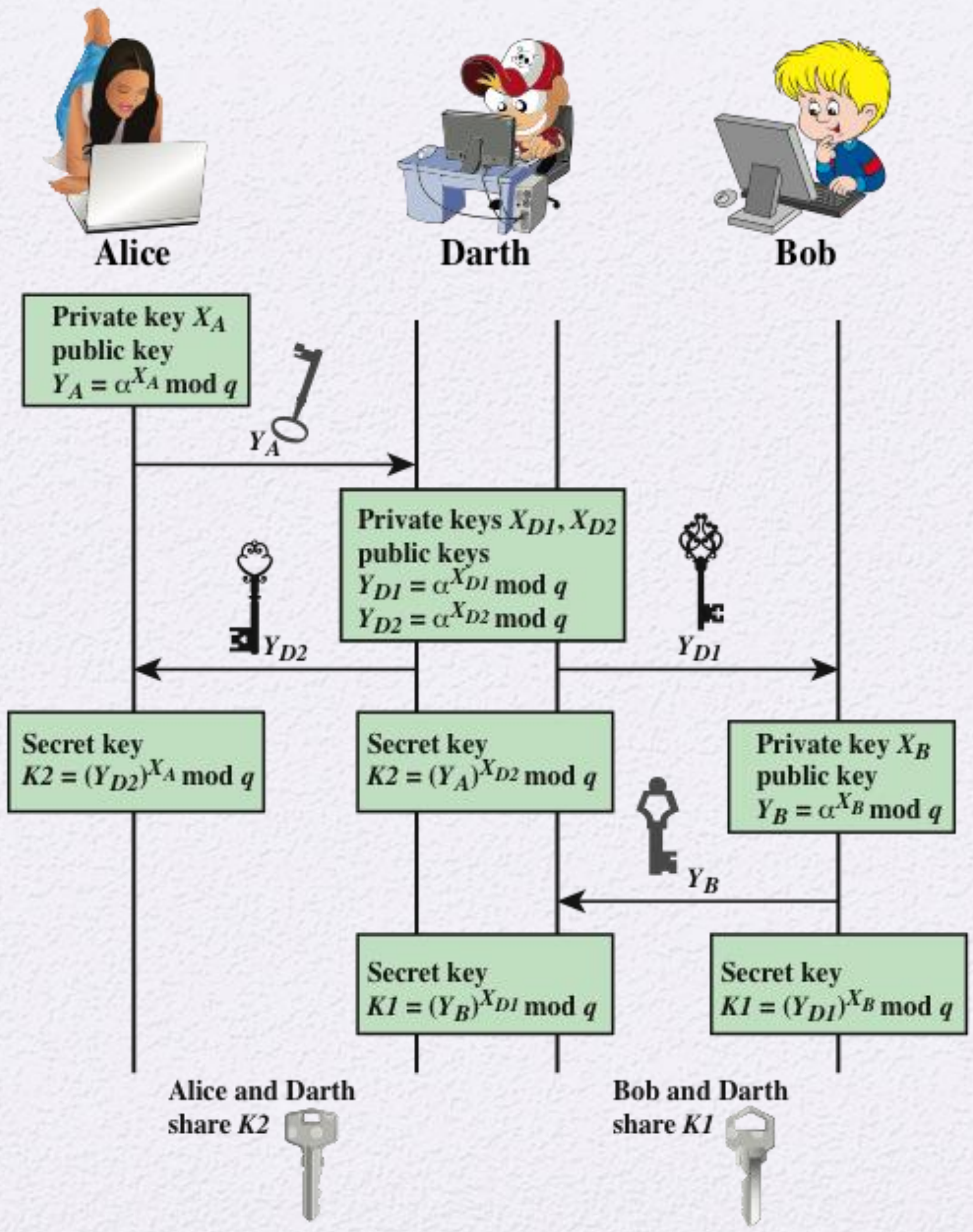
Bob receives Alice's public key Y_A in plaintext

Bob calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

Figure 10.1 Diffie-Hellman Key Exchange

Key Exchange Protocols

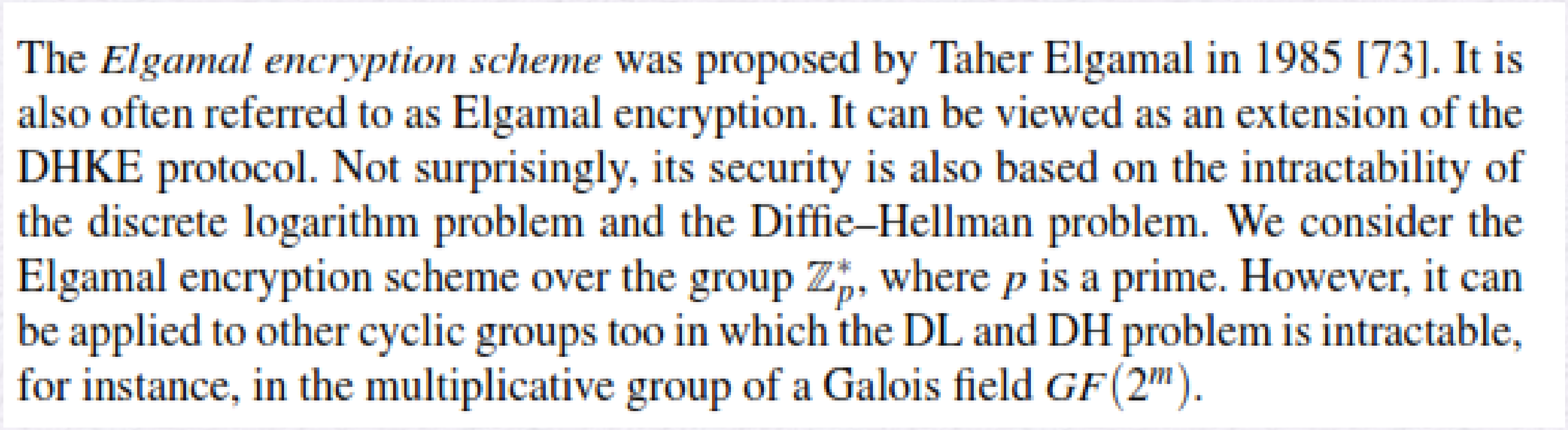
- Users could create random private/public Diffie-Hellman keys each time they communicate
- Users could create a known private/public Diffie-Hellman key and publish in a directory, then consulted and used to securely communicate with them
- Vulnerable to Man-in-the-Middle-Attack
- Authentication of the keys is needed



Alice
Darth
Bob
Private key X_A
public key
$Y_A = \alpha^{X_A} \bmod q$
Y_A
Private keys X_{D1}, X_{D2}
public keys
$Y_{D1} = \alpha^{X_{D1}} \bmod q$
$Y_{D2} = \alpha^{X_{D2}} \bmod q$
Y_{D2}
Y_{D1}
Secret key
$K2 = (Y_{D2})^{X_A} \bmod q$
Secret key
$K2 = (Y_A)^{X_{D2}} \bmod q$
Private key X_B
public key
$Y_B = \alpha^{X_B} \bmod q$
Y_B
Secret key
$K1 = (Y_B)^{X_{D1}} \bmod q$
Secret key
$K1 = (Y_{D1})^{X_B} \bmod q$
Alice and Darth share $K2$
Bob and Darth share $K1$

Figure 10.2 Man-in-the-Middle Attack

Elgamal Encryption



The *Elgamal encryption scheme* was proposed by Taher Elgamal in 1985 [73]. It is also often referred to as Elgamal encryption. It can be viewed as an extension of the DHKE protocol. Not surprisingly, its security is also based on the intractability of the discrete logarithm problem and the Diffie–Hellman problem. We consider the Elgamal encryption scheme over the group $\mathbb{Z}_p^*$, where p is a prime. However, it can be applied to other cyclic groups too in which the DL and DH problem is intractable, for instance, in the multiplicative group of a Galois field $GF(2^m)$.

- We consider two parties, Alice and Bob. If Alice wants to send an encrypted message x to Bob, both parties first perform a Diffie–Hellman key exchange to derive a shared key k_M
- For this we assume that a large prime p and a primitive element a have been generated. Now, the new idea is that Alice uses this key as a multiplicative mask to encrypt x as
- $y = x \cdot k_M \bmod p$.

ElGamal Cryptography

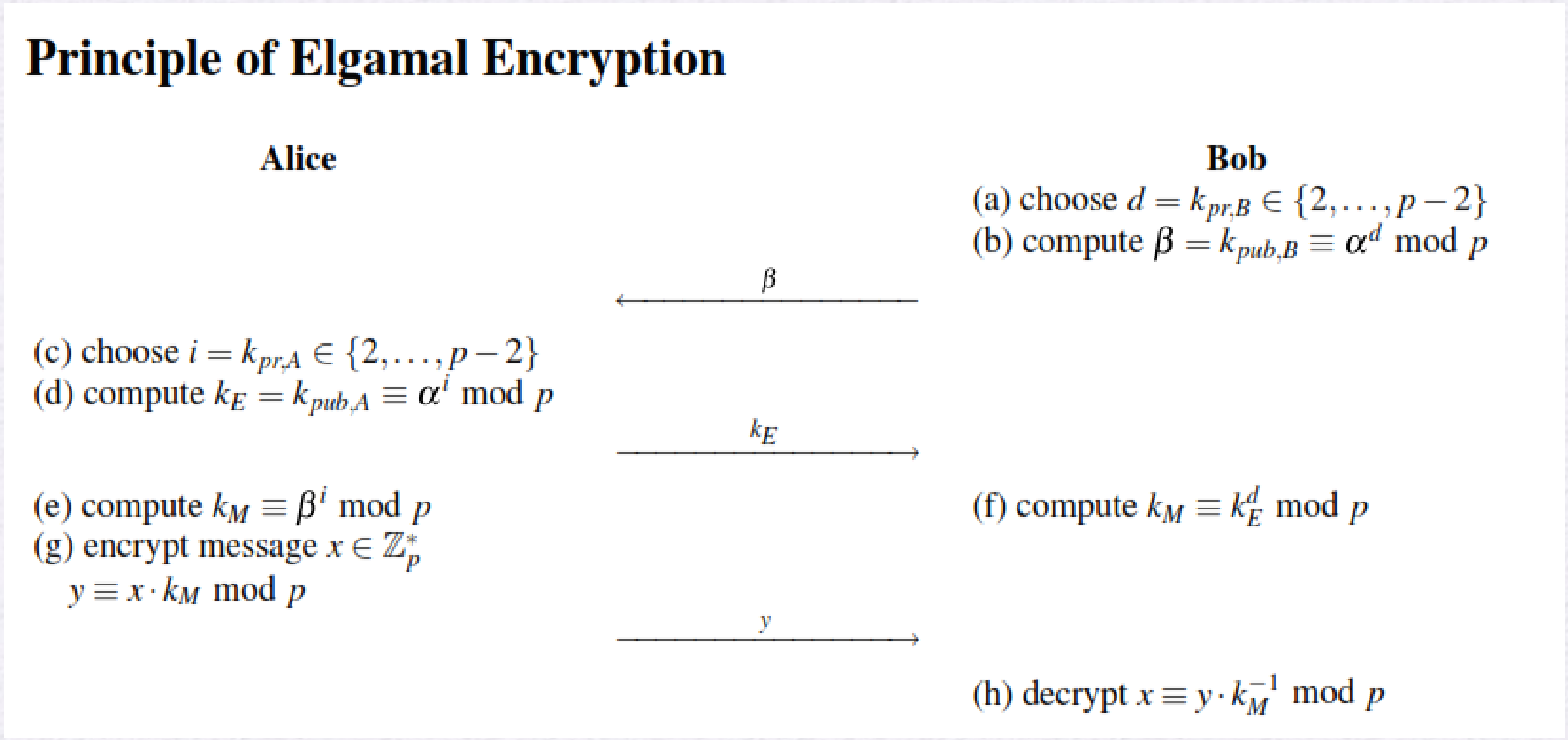
Announced in 1984 by T. Elgamal

Public-key scheme based on discrete logarithms closely related to the Diffie-Hellman technique

Used in the digital signature standard (DSS) and the S/MIME e-mail standard

Global elements are a prime number *q* and *a* which is a primitive root of *q*

Security is based on the difficulty of computing discrete logarithms

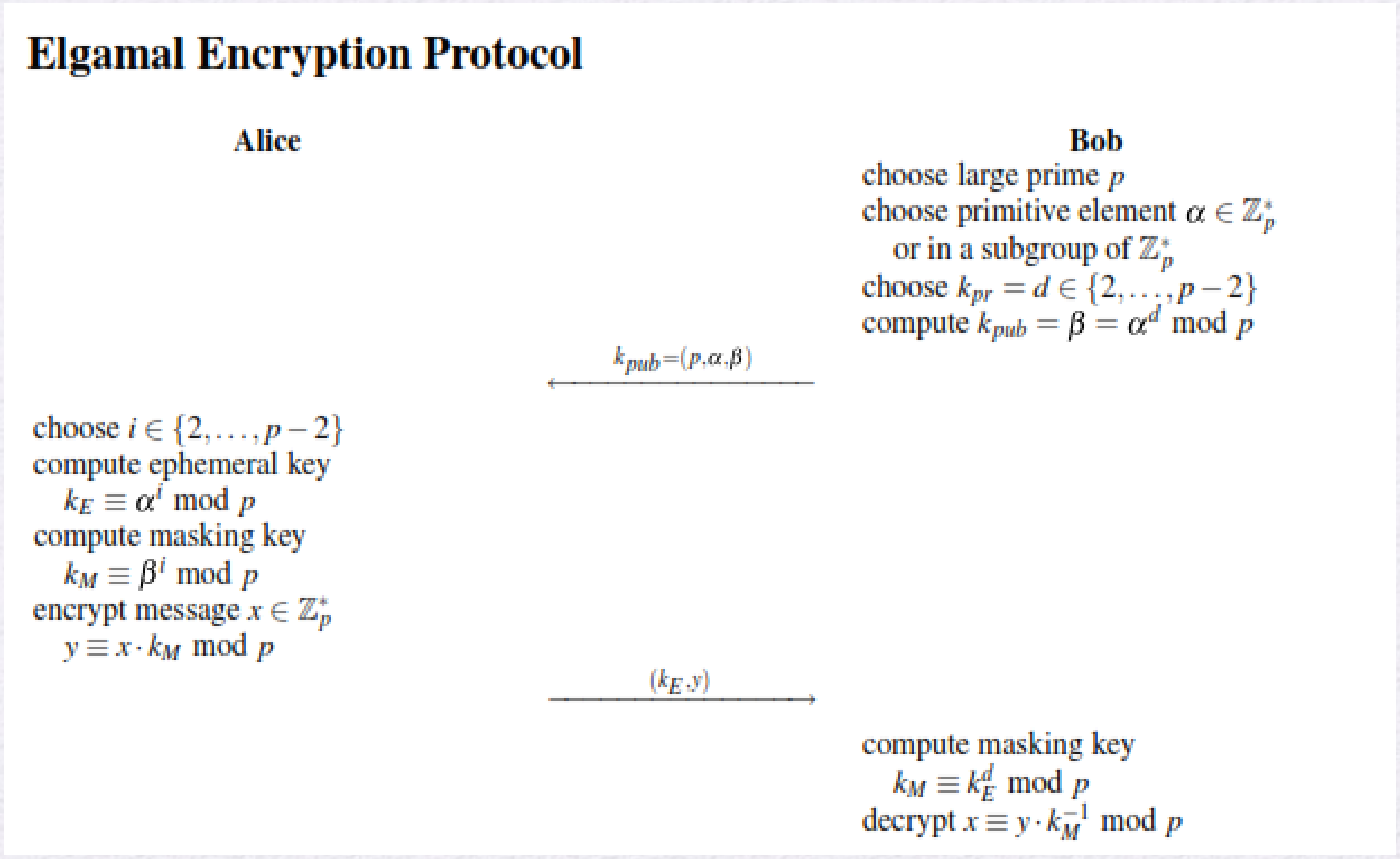


Principle of Elgamal Encryption

Alice		**Bob**
		(a) choose $d = k_{pr,B} \in \{2,\ldots,p-2\}$ (b) compute $\beta = k_{pub,B} \equiv \alpha^d \bmod p$
	$\xleftarrow{\quad\beta\quad}$	
(c) choose $i = k_{pr,A} \in \{2,\ldots,p-2\}$ (d) compute $k_E = k_{pub,A} \equiv \alpha^i \bmod p$		
	$\xrightarrow{\quad k_E\quad}$	
(e) compute $k_M \equiv \beta^i \bmod p$ (g) encrypt message $x \in \mathbb{Z}_p^*$ $y \equiv x \cdot k_M \bmod p$		(f) compute $k_M \equiv k_E^d \bmod p$
	$\xrightarrow{\quad y\quad}$	
		(h) decrypt $x \equiv y \cdot k_M^{-1} \bmod p$

The protocol consists of two phases, the classical DHKE (Steps a–f) which is followed by the message encryption and decryption (Steps g and h, respectively). Bob computes his private key d and public key β. This key pair does not change, i.e., it can be used for encrypting many messages. Alice, however, has to generate a new public–private key pair for the encryption of every message. Her private key is denoted by i and her public key by k_E. The latter is an ephemeral (existing only temporarily) key, hence the index “E”. The joint key is denoted by k_M because it is used for masking the plaintext.

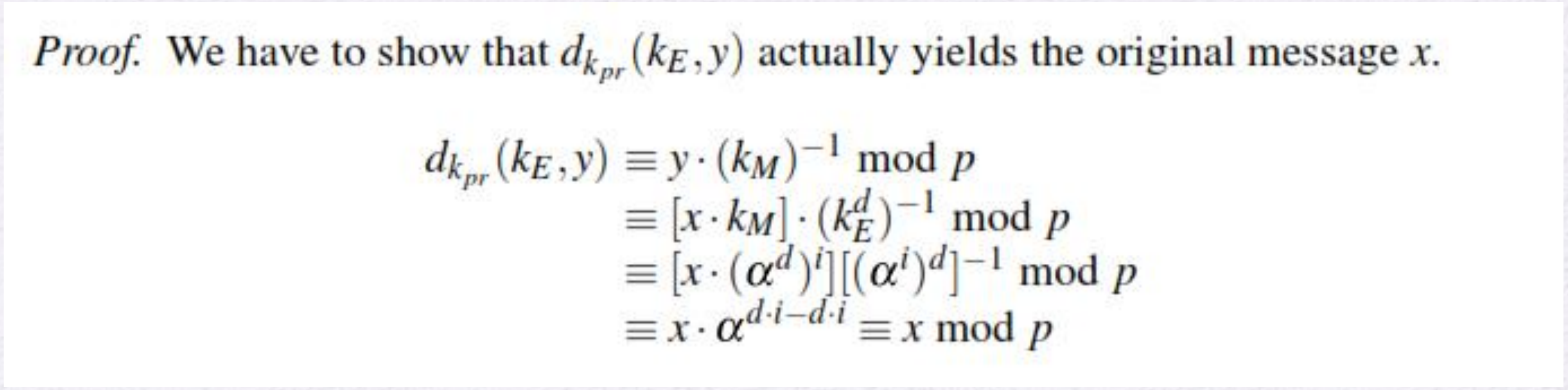
For the actual encryption, Alice simply multiplies the plaintext message x by the masking key k_M in $\mathbb{Z}_p^*$. On the receiving side, Bob reverses the encryption by multipliying with the inverse mask. Note that one property of cyclic groups is that, given any key $k_M \in \mathbb{Z}_p^*$, every messages x maps to another ciphertext if the two values are multiplied. Moreover, if the key k_M is randomly drawn from $\mathbb{Z}_p^*$, every ciphertext $y \in \{1, 2, \ldots, p-1\}$ is equally likely.



Elgamal Encryption Protocol

Alice		**Bob**
		choose large prime p choose primitive element $\alpha \in \mathbb{Z}_p^*$ or in a subgroup of $\mathbb{Z}_p^*$ choose $k_{pr} = d \in \{2, \ldots, p-2\}$ compute $k_{pub} = \beta = \alpha^d \bmod p$
	$\xleftarrow{k_{pub}=(p,\alpha,\beta)}$	
choose $i \in \{2, \ldots, p-2\}$ compute ephemeral key $k_E \equiv \alpha^i \bmod p$ compute masking key $k_M \equiv \beta^i \bmod p$ encrypt message $x \in \mathbb{Z}_p^*$ $y \equiv x \cdot k_M \bmod p$		
	$\xrightarrow{(k_E, y)}$	
		compute masking key $k_M \equiv k_E^d \bmod p$ decrypt $x \equiv y \cdot k_M^{-1} \bmod p$

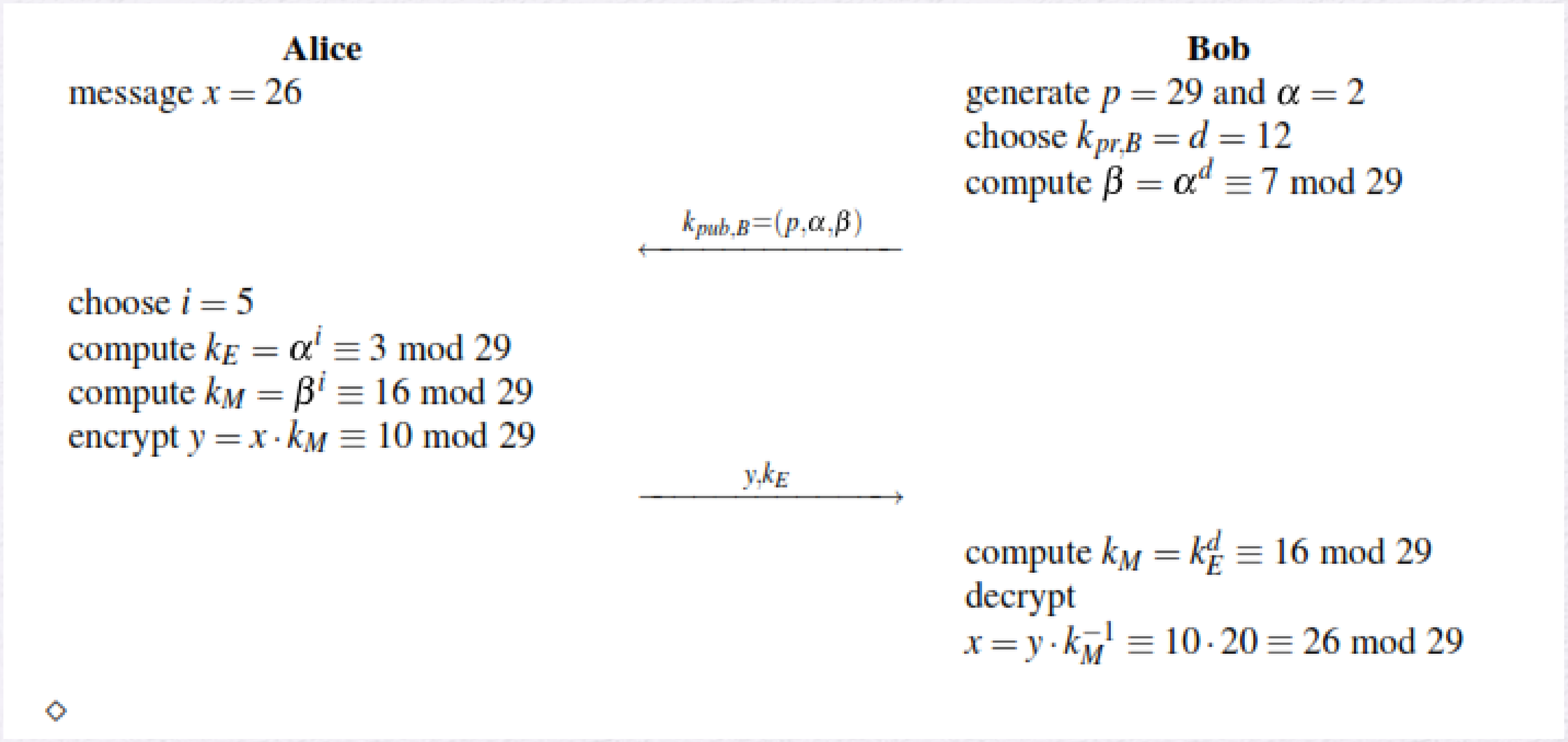
Proof



Proof. We have to show that $d_{k_{pr}}(k_E, y)$ actually yields the original message x.

$$\begin{aligned} d_{k_{pr}}(k_E, y) &\equiv y \cdot (k_M)^{-1} \bmod p \\ &\equiv [x \cdot k_M] \cdot (k_E^d)^{-1} \bmod p \\ &\equiv [x \cdot (\alpha^d)^i][(\alpha^i)^d]^{-1} \bmod p \\ &\equiv x \cdot \alpha^{d \cdot i - d \cdot i} \equiv x \bmod p \end{aligned}$$

- In this example, Bob generates the Elgamal keys and Alice encrypts the message x = 26.



Alice		**Bob**
message $x = 26$		generate $p = 29$ and $\alpha = 2$
		choose $k_{pr,B} = d = 12$
		compute $\beta = \alpha^d \equiv 7 \bmod 29$
	$\xleftarrow{k_{pub,B}=(p,\alpha,\beta)}$	
choose $i = 5$		
compute $k_E = \alpha^i \equiv 3 \bmod 29$		
compute $k_M = \beta^i \equiv 16 \bmod 29$		
encrypt $y = x \cdot k_M \equiv 10 \bmod 29$		
	$\xrightarrow{y, k_E}$	
		compute $k_M = k_E^d \equiv 16 \bmod 29$
		decrypt
		$x = y \cdot k_M^{-1} \equiv 10 \cdot 20 \equiv 26 \bmod 29$

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (\mathrm{C}_2 K^{-1}) \bmod q$

Figure 10.3 The ElGamal Cryptosystem

ECC

- ECC was independently invented in 1987 by Neal Koblitz and in 1986 by Victor Miller.

- During the 1990s there was much speculation about the security and practicality of ECC, especially if compared to RSA.

- An important step for building confidence in ECC was the issuing of two ANSI banking standards for elliptic curve digital signature and key establishment

- ECC schemes are allowed as asymmetric algorithms. Elliptic curves are also widely used in commercial standards such as IPsec or Transport Layer Security (TLS).

Elliptic Curve Arithmetic

- Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA
  - The key length for secure RSA use has increased over recent years and this has put a heavier processing load on applications using RSA
- Elliptic curve cryptography (ECC) is showing up in standardization efforts including the IEEE P1363 Standard for Public-Key Cryptography
- Principal attraction of ECC is that it appears to offer equal security for a far smaller key size
- Confidence level in ECC is not yet as high as that in RSA

ECC

- ECC is based on the generalized discrete logarithm problem.
- Cyclic group on which we can build our cryptosystem.

Contd…

- We start by considering certain polynomials (e.g., functions with sums of exponents
- of x and y), and we plot them over the real numbers.

Contd…

Example 9.1. Let's look at the polynomial equation $x^2 + y^2 = r^2$ over the real numbers $\mathbb{R}$. If we plot all the pairs (x, y) which fulfill this equation in a coordinate sys-

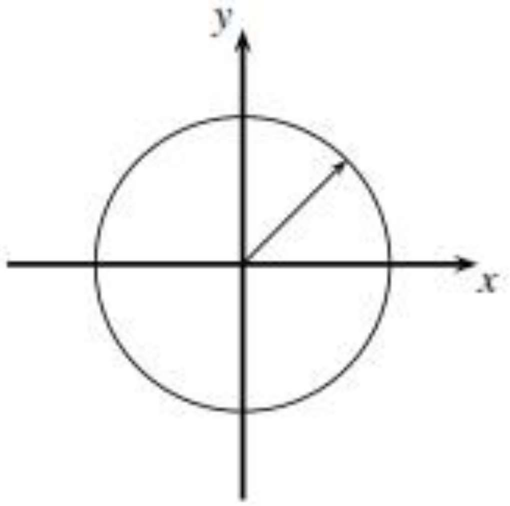


Fig. 9.1 Plot of all points (x, y) which fulfill the equation $x^2 + y^2 = r^2$ over $\mathbb{R}$

Contd…

Example 9.2. A slight generalization of the circle equation is to introduce coefficients to the two terms x^2 and y^2, i.e., we look at the set of solutions to the equation $a \cdot x^2 + b \cdot y^2 = c$ over the real numbers. It turns out that we obtain an ellipse, as

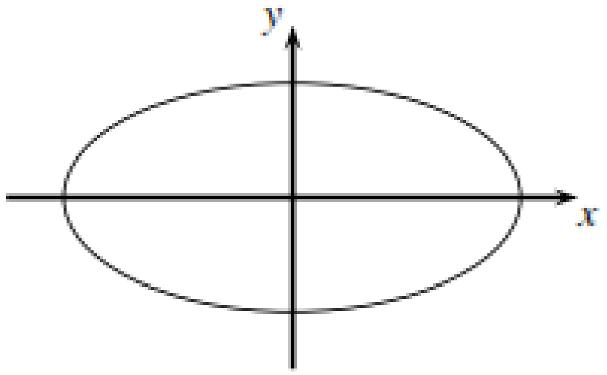
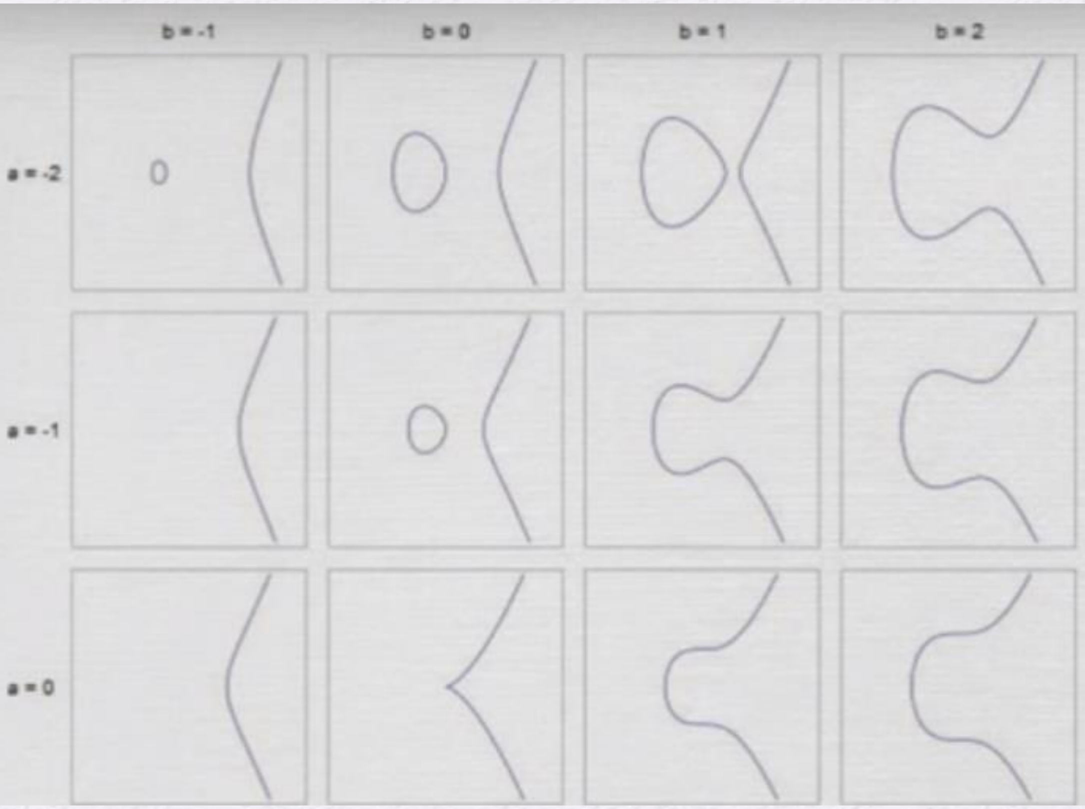


Fig. 9.2 Plot of all points (x, y) which fulfill the equation $a \cdot x^2 + b \cdot y^2 = c$ over $\mathbb{R}$

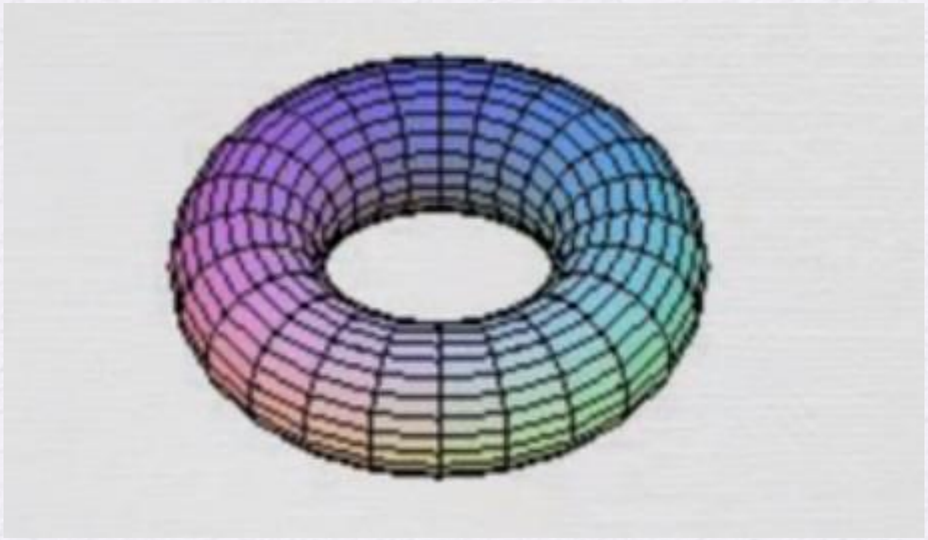
- An elliptic curve is a special type of polynomial equation. For cryptographic use, we need to consider the curve not over the real numbers but over a finite field. The most popular choice is prime fields GF(p) , where all arithmetic is performed modulo a prime p.

$\mathbb{R}$

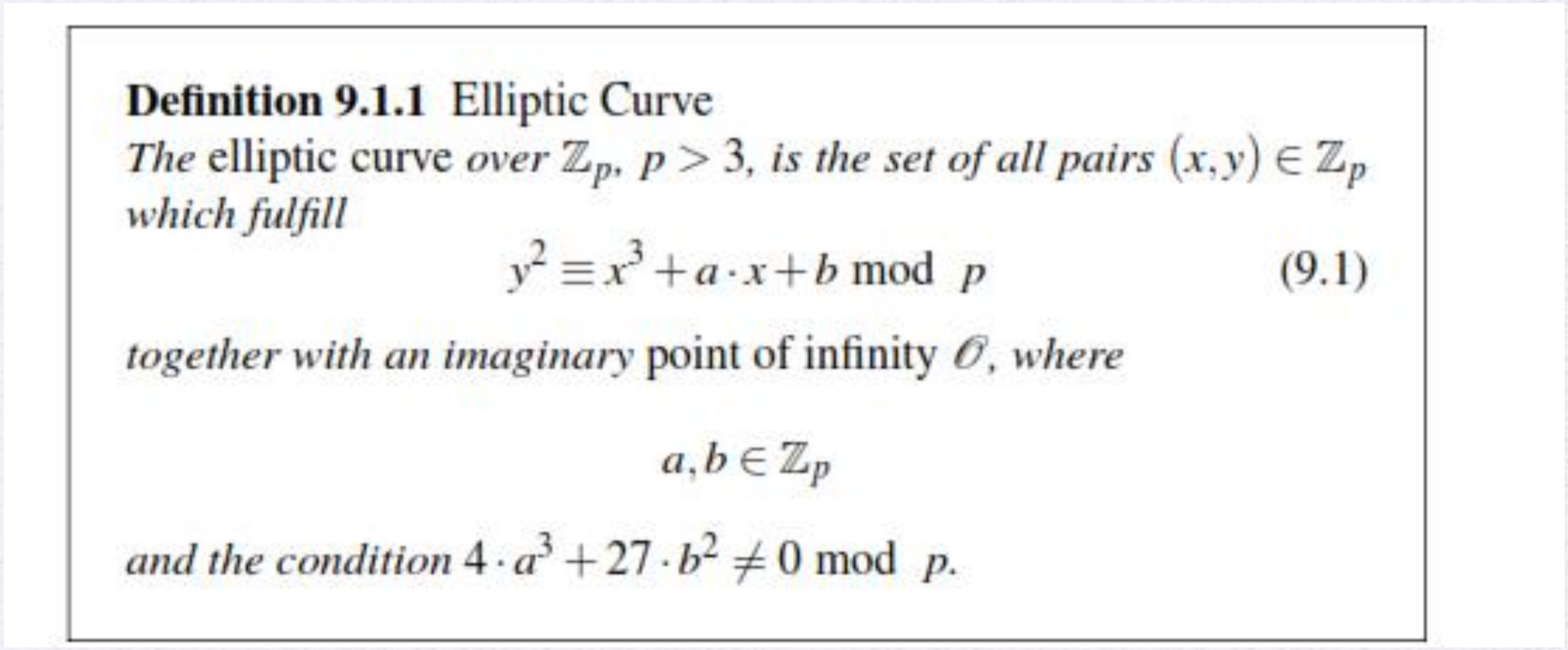


The curve with a=0,b=0 is singular

EC for Complex Numbers



Definition



Definition 9.1.1 Elliptic Curve
The elliptic curve *over* $\mathbb{Z}_p$, $p > 3$, *is the set of all pairs* $(x,y) \in \mathbb{Z}_p$ *which fulfill*

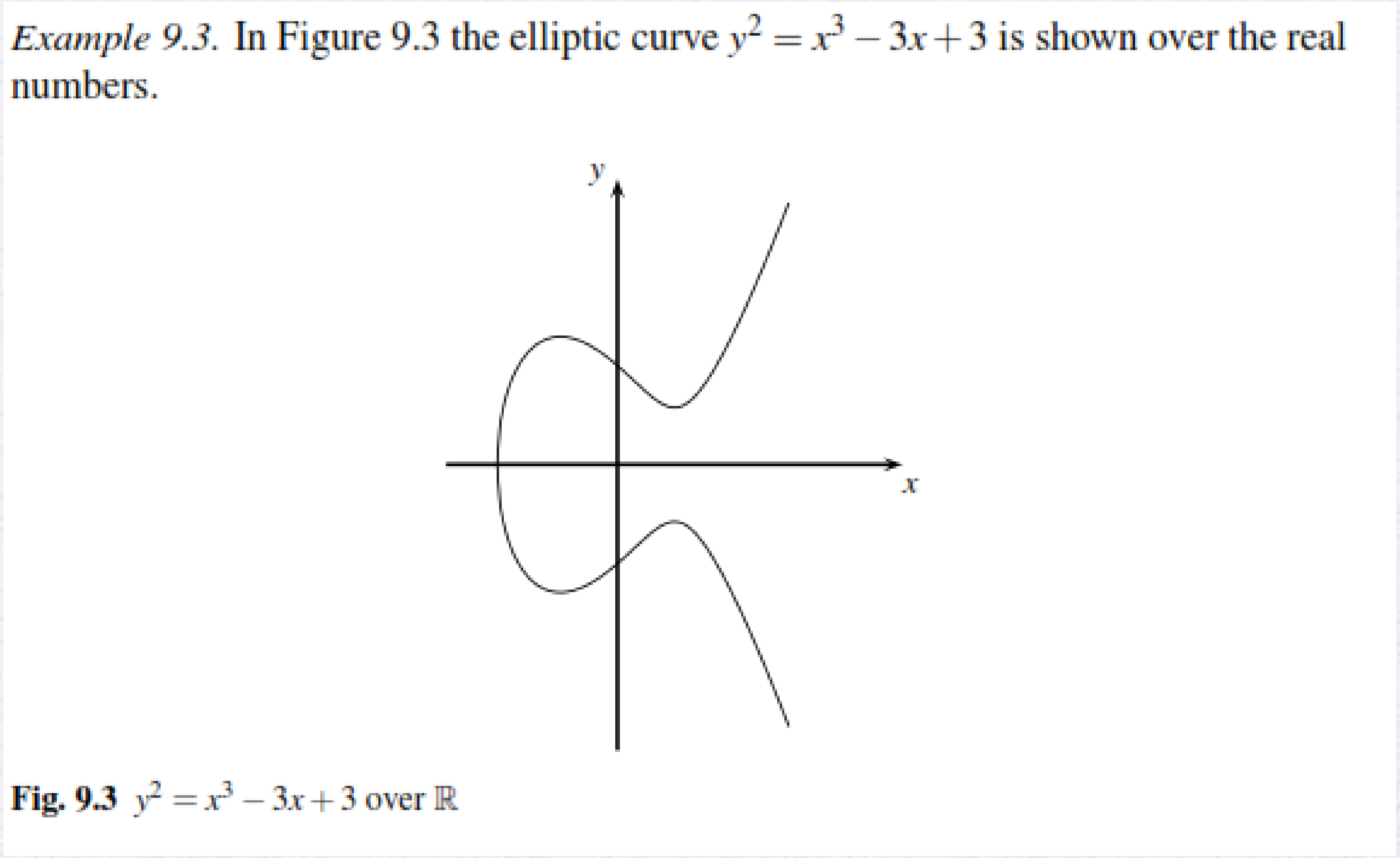
$$y^2 \equiv x^3 + a \cdot x + b \bmod p \tag{9.1}$$

together with an imaginary point of infinity $\mathcal{O}$, *where*

$$a, b \in \mathbb{Z}_p$$

and the condition $4 \cdot a^3 + 27 \cdot b^2 \neq 0 \bmod p$.

Example



Example 9.3. In Figure 9.3 the elliptic curve $y^2 = x^3 - 3x + 3$ is shown over the real numbers.

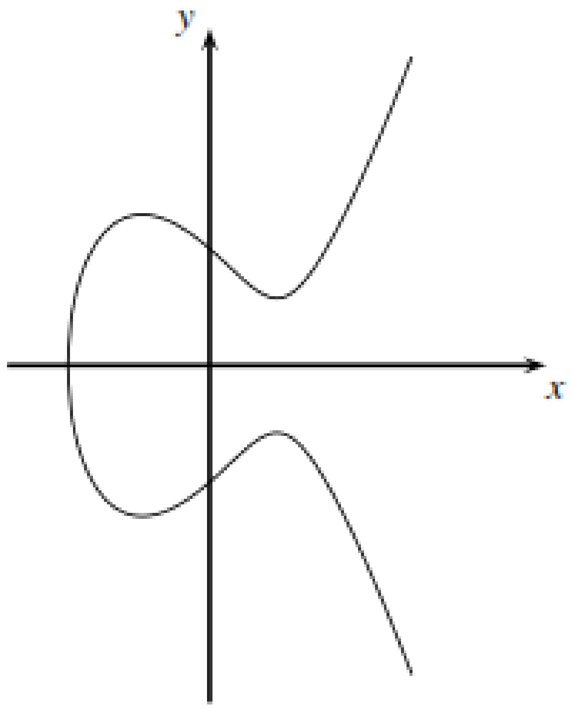
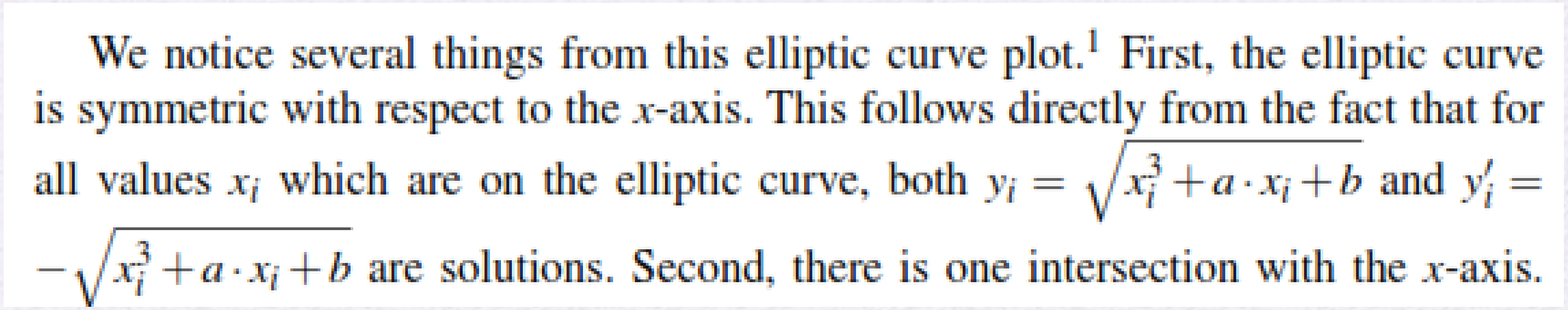
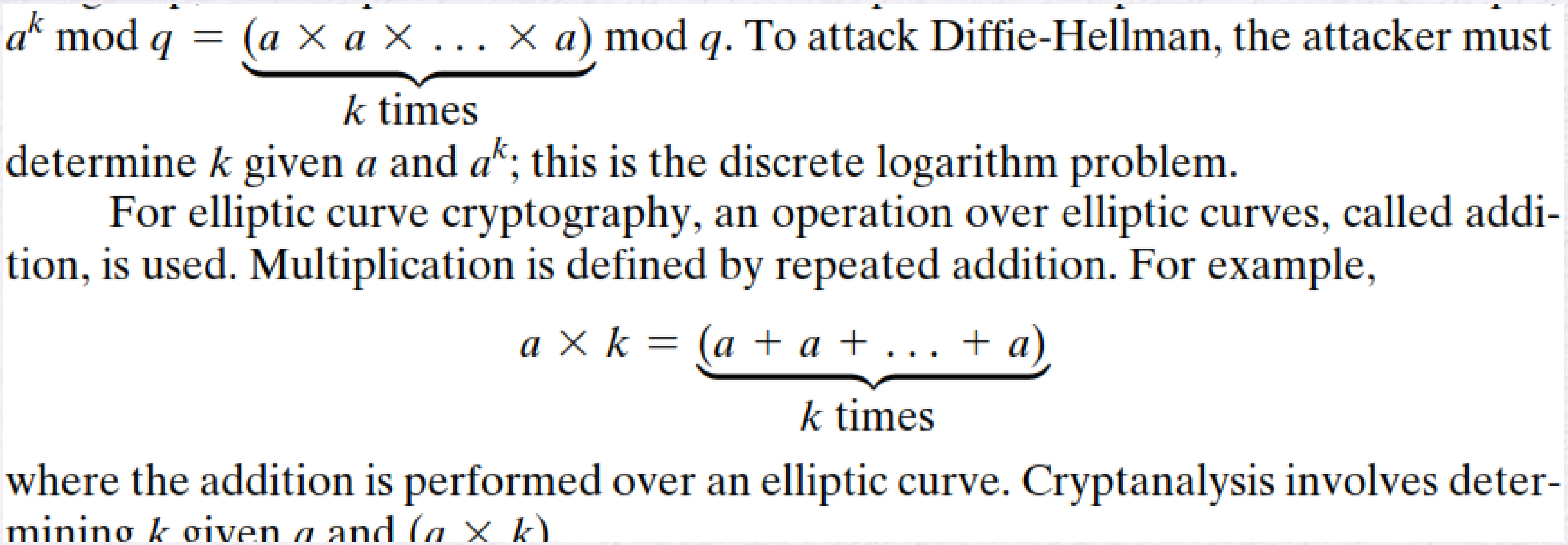


Fig. 9.3 $y^2 = x^3 - 3x + 3$ over $\mathbb{R}$

Contd…



We notice several things from this elliptic curve plot.[1] First, the elliptic curve is symmetric with respect to the x-axis. This follows directly from the fact that for all values x_i which are on the elliptic curve, both $y_i = \sqrt{x_i^3 + a \cdot x_i + b}$ and $y_i' = -\sqrt{x_i^3 + a \cdot x_i + b}$ are solutions. Second, there is one intersection with the x-axis.



$a^k \bmod q = \underbrace{(a \times a \times \ldots \times a)}_{k \text{ times}} \bmod q$. To attack Diffie-Hellman, the attacker must determine k given a and a^k; this is the discrete logarithm problem.

For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition. For example,

$$a \times k = \underbrace{(a + a + \ldots + a)}_{k \text{ times}}$$

where the addition is performed over an elliptic curve. Cryptanalysis involves determining k given a and $(a \times k)$

Abelian Group

- A set of elements with a binary operation, denoted by •, that associates to each ordered pair (a, b) of elements in G an element $(a \bullet b)$ in G, such that the following axioms are obeyed:

(A1) Closure:	If a and b belong to G, then $a \bullet b$ is also in G
(A2) Associative:	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all a, b, c in G
(A3) Identity element:	There is an element e in G such that $a \bullet e = e \bullet a = a$ for all a in G
(A4) Inverse element:	For each a in G there is an element a' in G such that $a \bullet a' = a' \bullet a = e$
(A5) Commutative:	$a \bullet b = b \bullet a$ for all a, b in G

- An **elliptic curve is defined by an equation in two variables with coefficients.**
- For cryptography, the variables and coefficients are restricted to elements in a finite field, which results in the definition of a finite abelian group.

Weierstrass equation:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, e are real numbers and x and y take on values in the real numbers.[4] For our purpose, it is sufficient to limit ourselves to equations of the form

$$y^2 = x^3 + ax + b \tag{10.1}$$

- elliptic curve is a single element denoted O *and called the point at infinity or the zero point,*

$$4a^3 + 27b^2 \neq 0$$

K - Field

$$E = \{(x, y) \mid y^2 = x^3 + ax + b\}$$

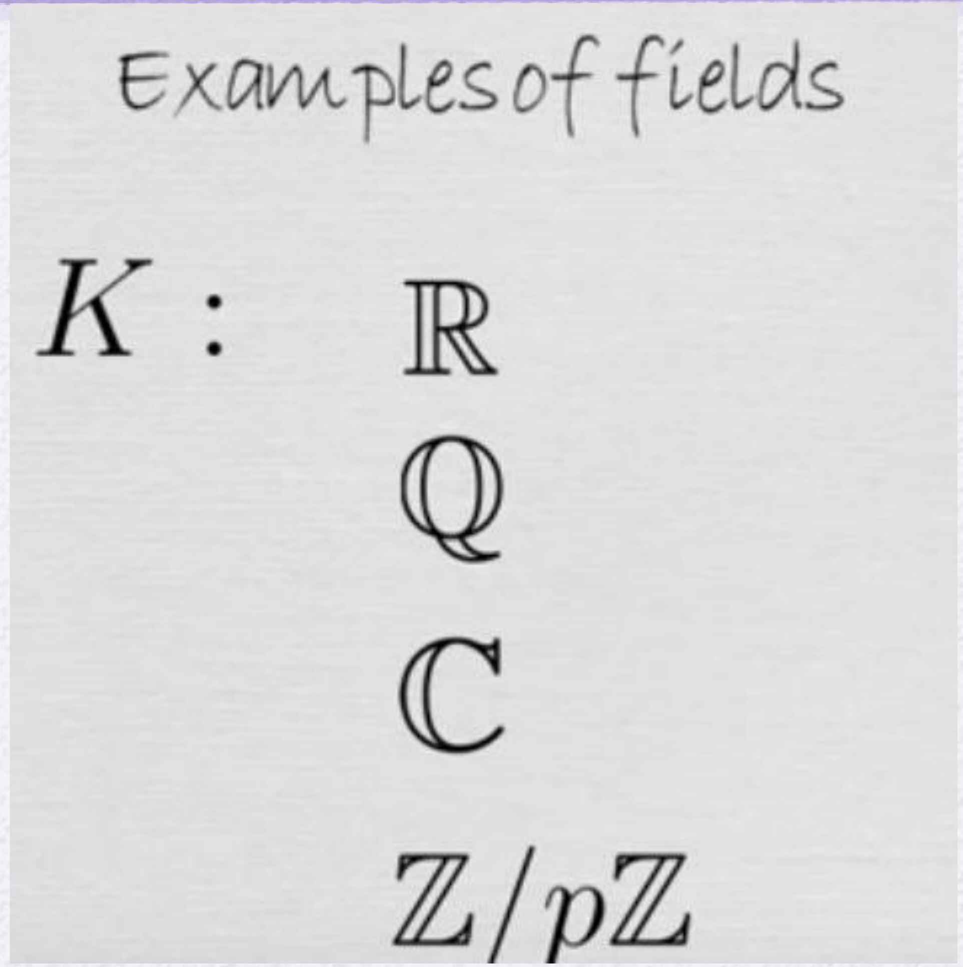
$a, b \in K$

point at infinity: $\mathcal{O}$

$4a^3 + 27b^2 \neq 0$

Examples of Field K

- Real Numbers
- Rational Numbers
- Complex Numbers
- Integer Modulo Prime


Examples of fields

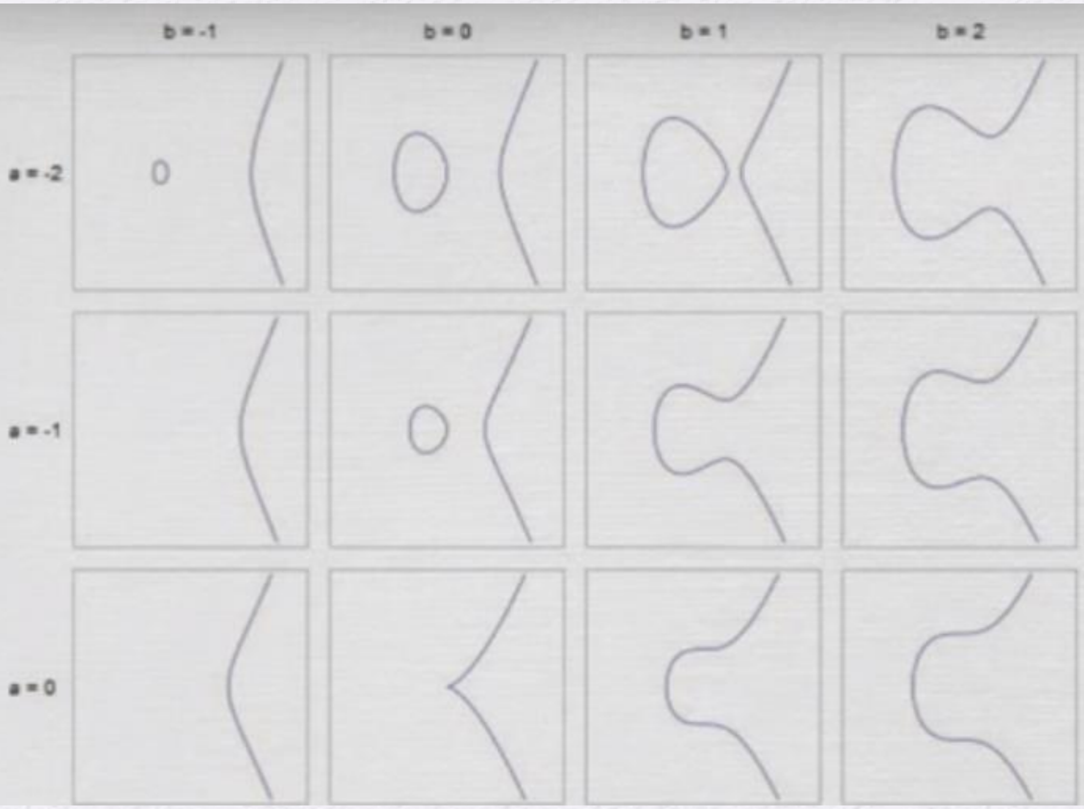
$K : \quad \mathbb{R}$

$\mathbb{Q}$

$\mathbb{C}$

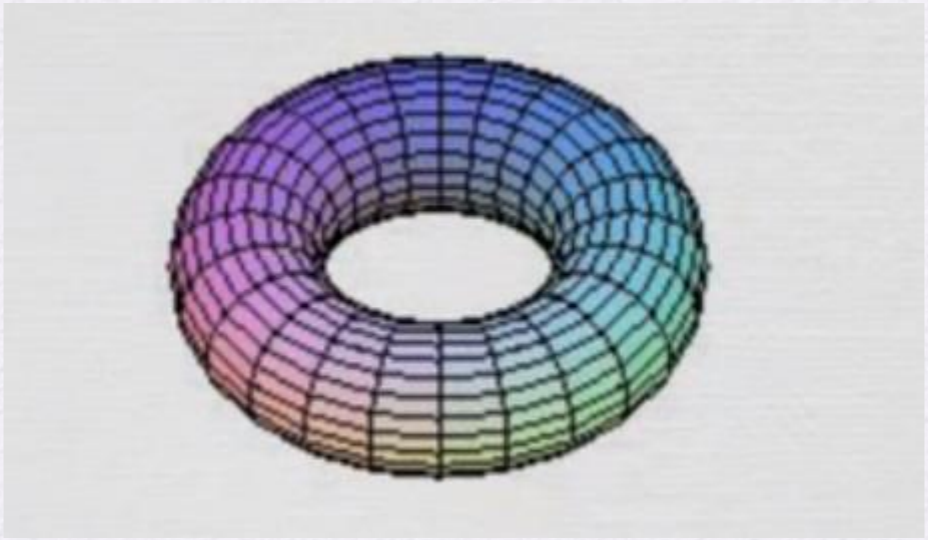
$\mathbb{Z}/p\mathbb{Z}$

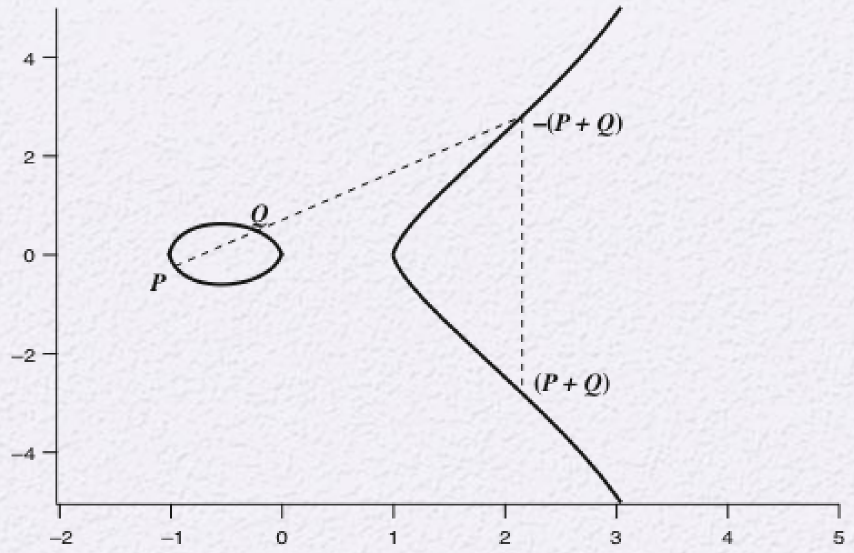
$\mathbb{R}$



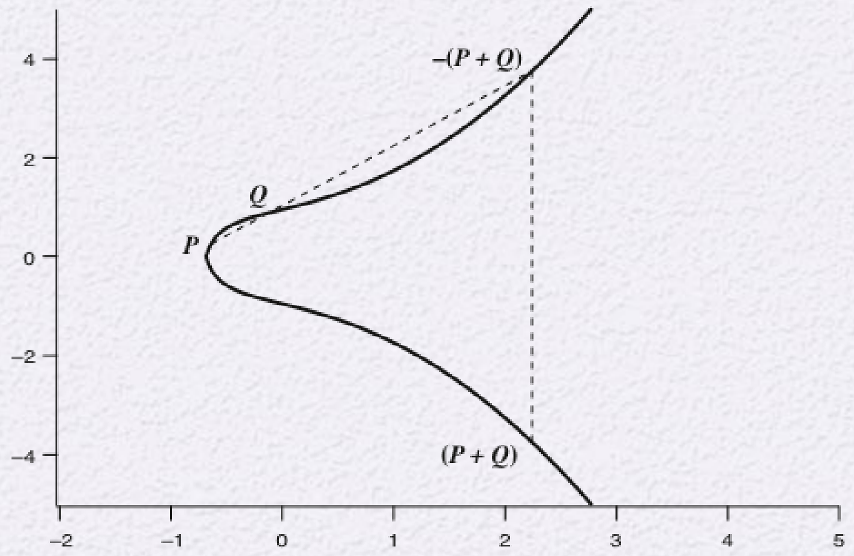
The curve with a=0,b=0 is singular

EC for Complex Numbers



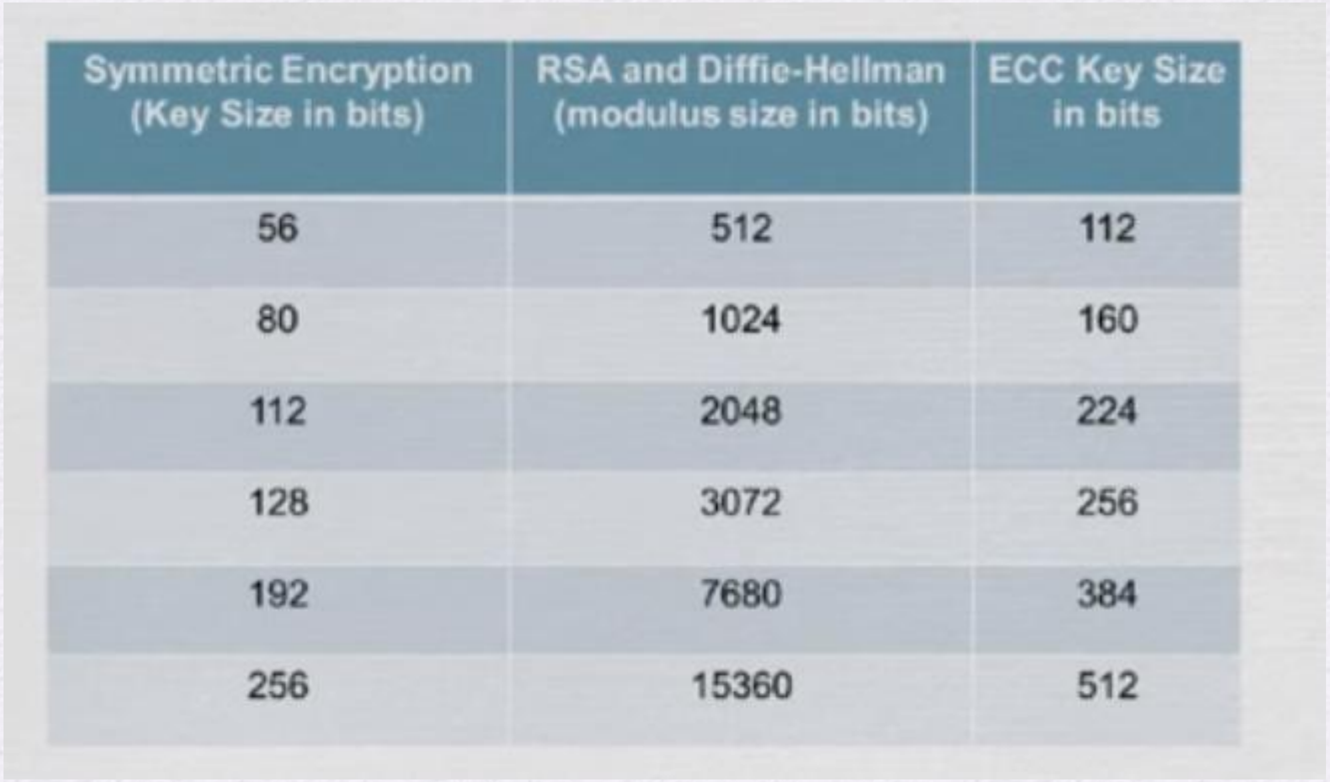


(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

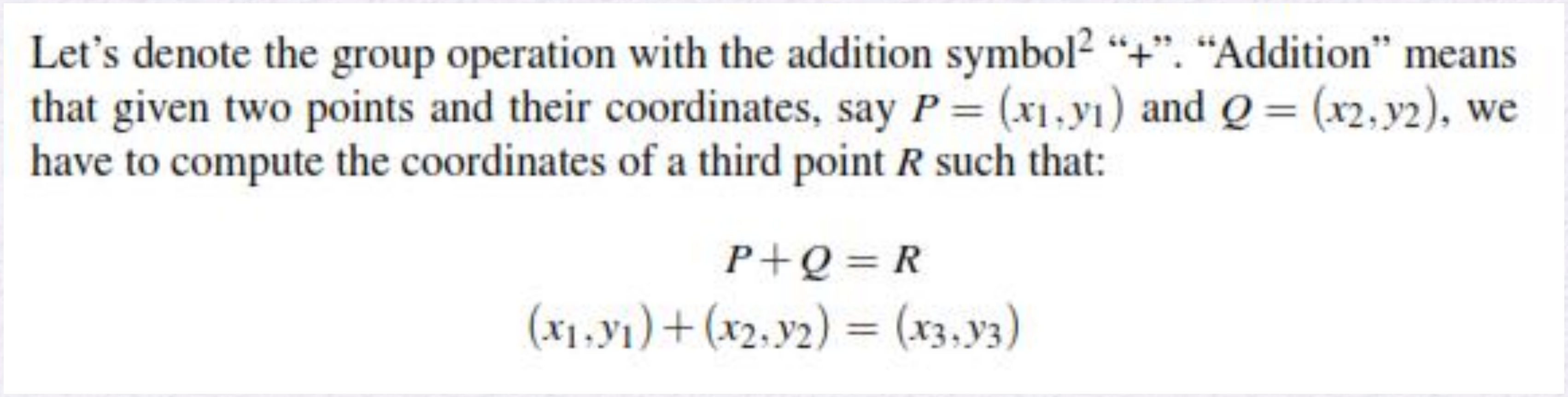
Figure 10.4 Example of Elliptic Curves



Symmetric Encryption (Key Size in bits)	RSA and Diffie-Hellman (modulus size in bits)	ECC Key Size in bits
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

- If three points on an elliptic curve lie on a straight line, their sum is *O. From this definition, we can define the rules of addition* over an elliptic curve.

Group Operations



Let's denote the group operation with the addition symbol[2] "+". "Addition" means that given two points and their coordinates, say $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, we have to compute the coordinates of a third point R such that:

$$P + Q = R$$
$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

Point addition

- Point Addition P+Q This is the case where we compute R = P +Q and P not equal to Q.
- Draw a line through P and Q and obtain a third point of intersection between the elliptic curve and the line. Mirror this third intersection point along the x-axis. This mirrored point is, by definition, the point R.

Point addition

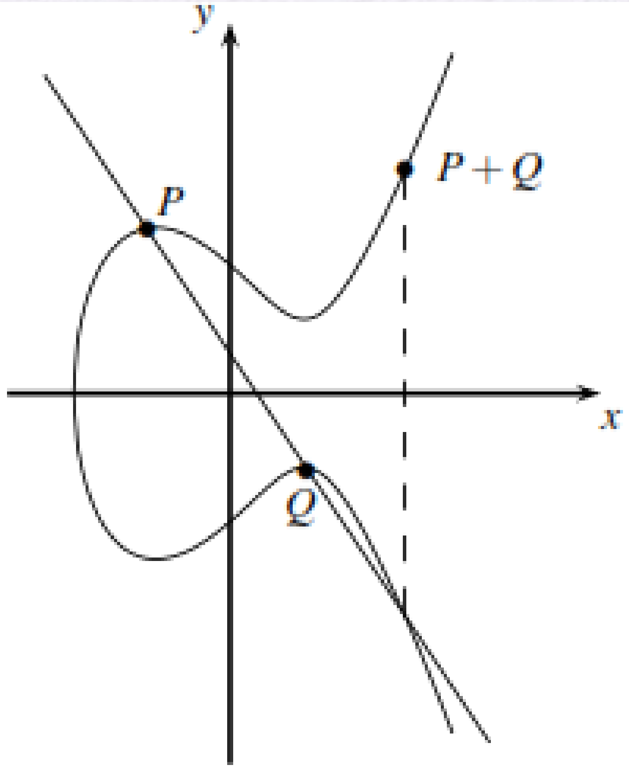
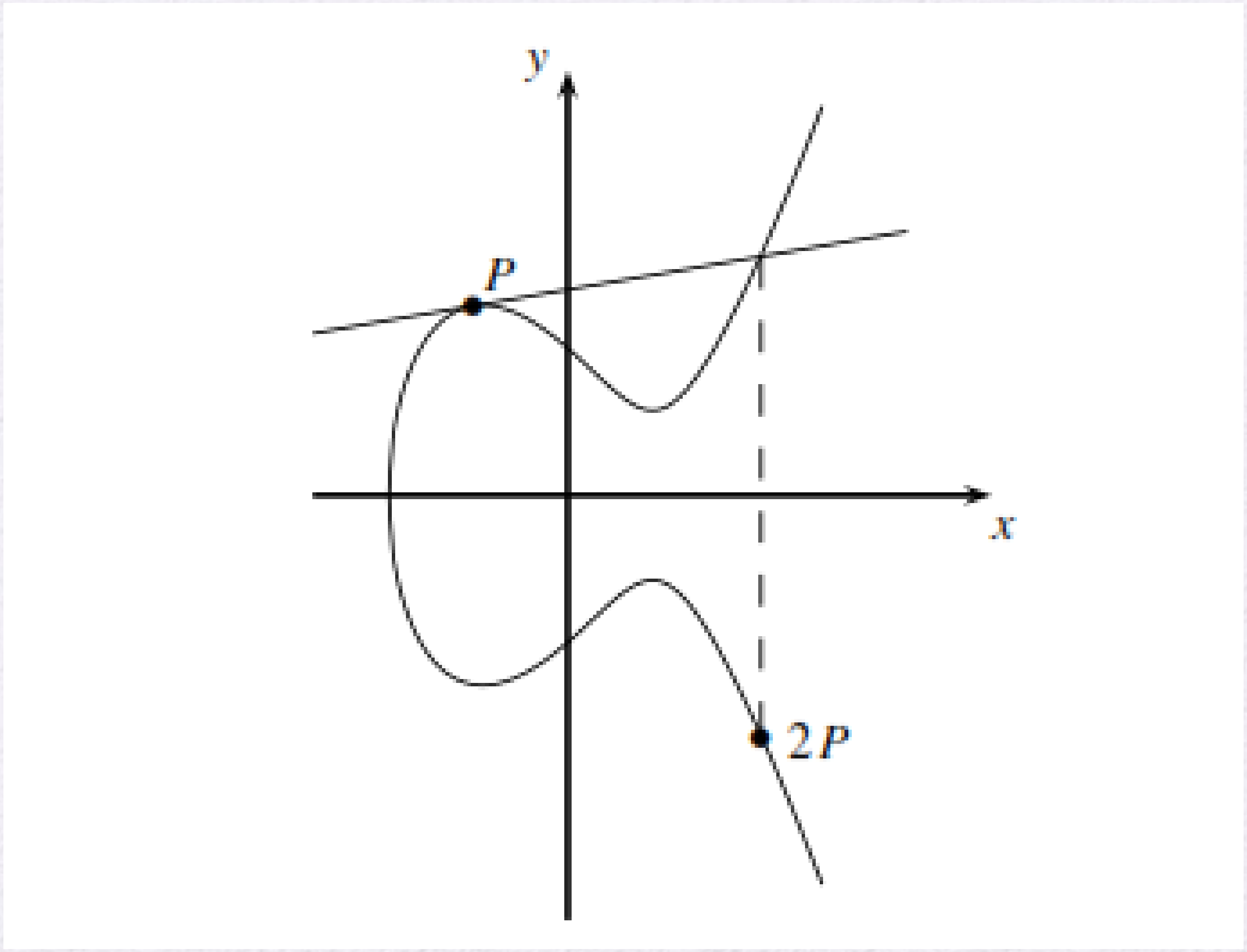


Fig. 9.4 Point addition on an elliptic curve over the real numbers

Point Doubling

- Point Doubling P+P This is the case where we compute P+Q but P =Q. Hence, we can write R = P +P = 2P. We need a slightly different construction here. We draw the tangent line through P and obtain a second point of intersection between this line and the elliptic curve. We mirror the point of the second intersection along the x-axis

Point Doubling



- this tangent-and-chord method was used to construct a third point if two points were already known, while only using the four standard algebraic operations add, subtract, multiply and divide.
- It turns out that if points on the elliptic curve are added in this very way, the set of points also fulfill most conditions necessary for a group, that is, closure, associativity, existence of an identity element and existence of an inverse.

- In particular, we can take the curve equation from above, but we now consider it over prime fields GF(p) rather than over the real numbers.

Elliptic Curve Point Addition and Point Doubling

$$x_3 = s^2 - x_1 - x_2 \bmod p$$
$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where

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

Identity

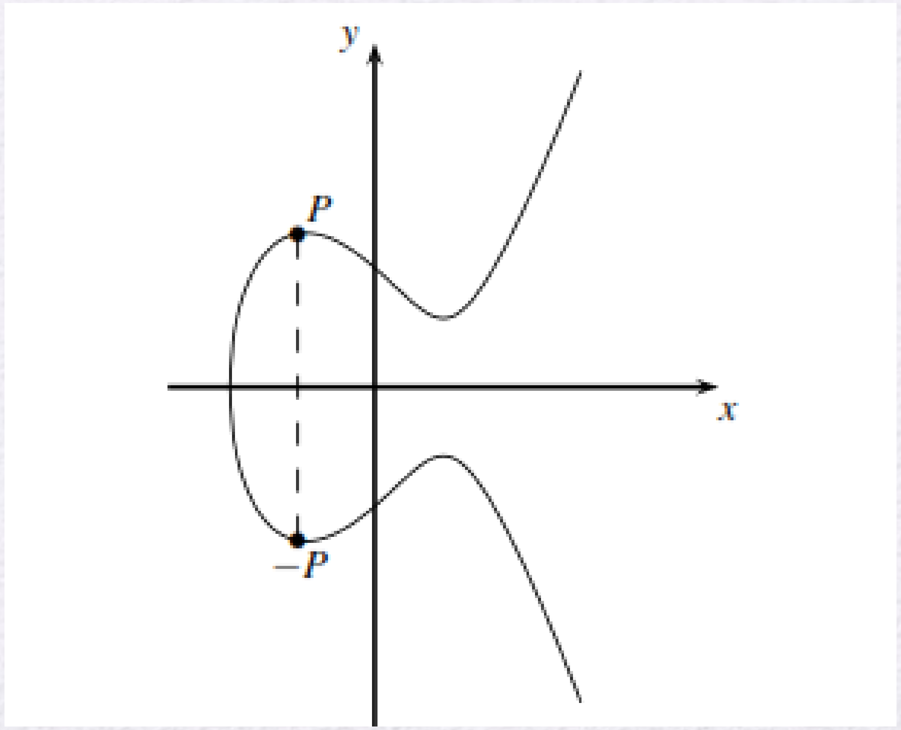
- One thing that is still missing is an identity (or neutral) element O such that: P+O = P for all points P on the elliptic curve.
- point at infinity as the neutral element O. This point at infinity can be visualized as a point that is located towards “plus” infinity along the y-axis or towards “minus” infinity along the y-axis.

Inverse

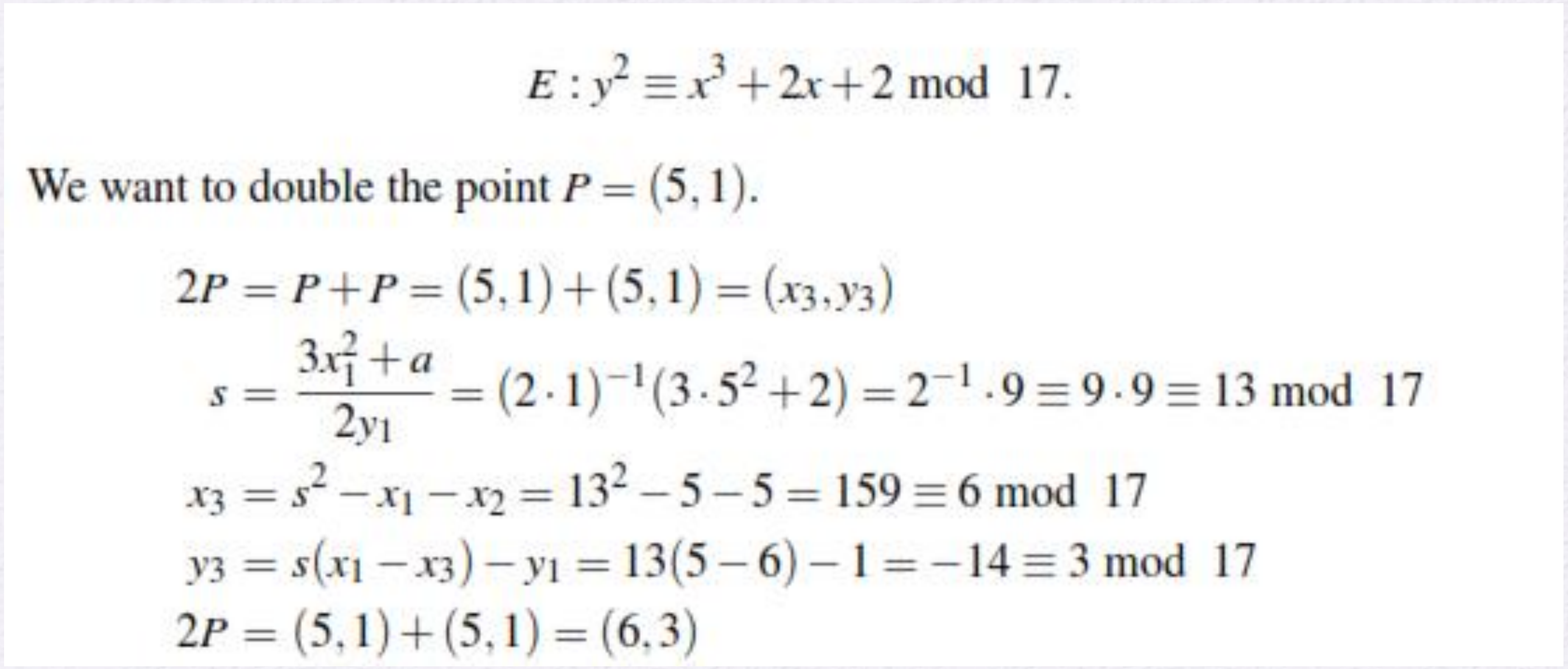
- According the group definition, we can now also define the inverse -P of any group element P as: P+(-P)=O.

Inverse

The question is how do we find $-P$? If we apply the tangent-and-chord method from above, it turns out that the inverse of the point $P = (x_p, y_p)$ is the point $-P = (x_p, -y_p)$, i.e., the point that is reflected along the x-axis. Figure 9.6 shows the point



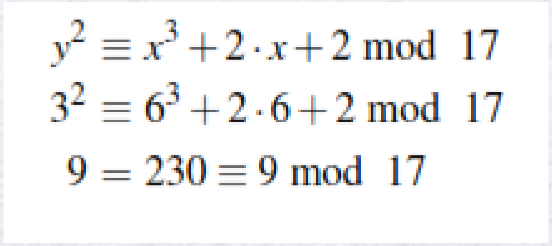
Example



$$E: y^2 \equiv x^3 + 2x + 2 \bmod 17.$$

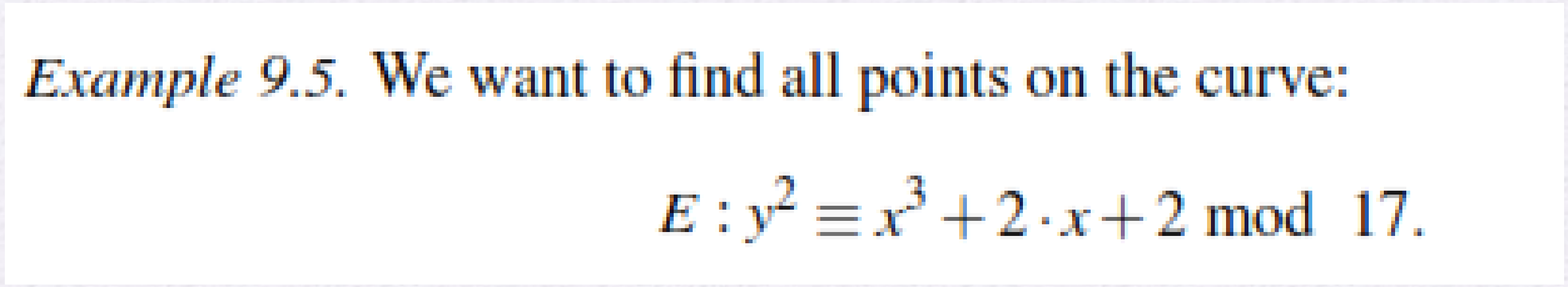
We want to double the point $P = (5,1)$.

$$\begin{aligned}
2P &= P + P = (5,1) + (5,1) = (x_3, y_3) \\
s &= \frac{3x_1^2 + a}{2y_1} = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 \equiv 9 \cdot 9 \equiv 13 \bmod 17 \\
x_3 &= s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \bmod 17 \\
y_3 &= s(x_1 - x_3) - y_1 = 13(5-6) - 1 = -14 \equiv 3 \bmod 17 \\
2P &= (5,1) + (5,1) = (6,3)
\end{aligned}$$



$$\begin{aligned}
y^2 &\equiv x^3 + 2 \cdot x + 2 \bmod 17 \\
3^2 &\equiv 6^3 + 2 \cdot 6 + 2 \bmod 17 \\
9 &= 230 \equiv 9 \bmod 17
\end{aligned}$$

Example



Example 9.5. We want to find all points on the curve:

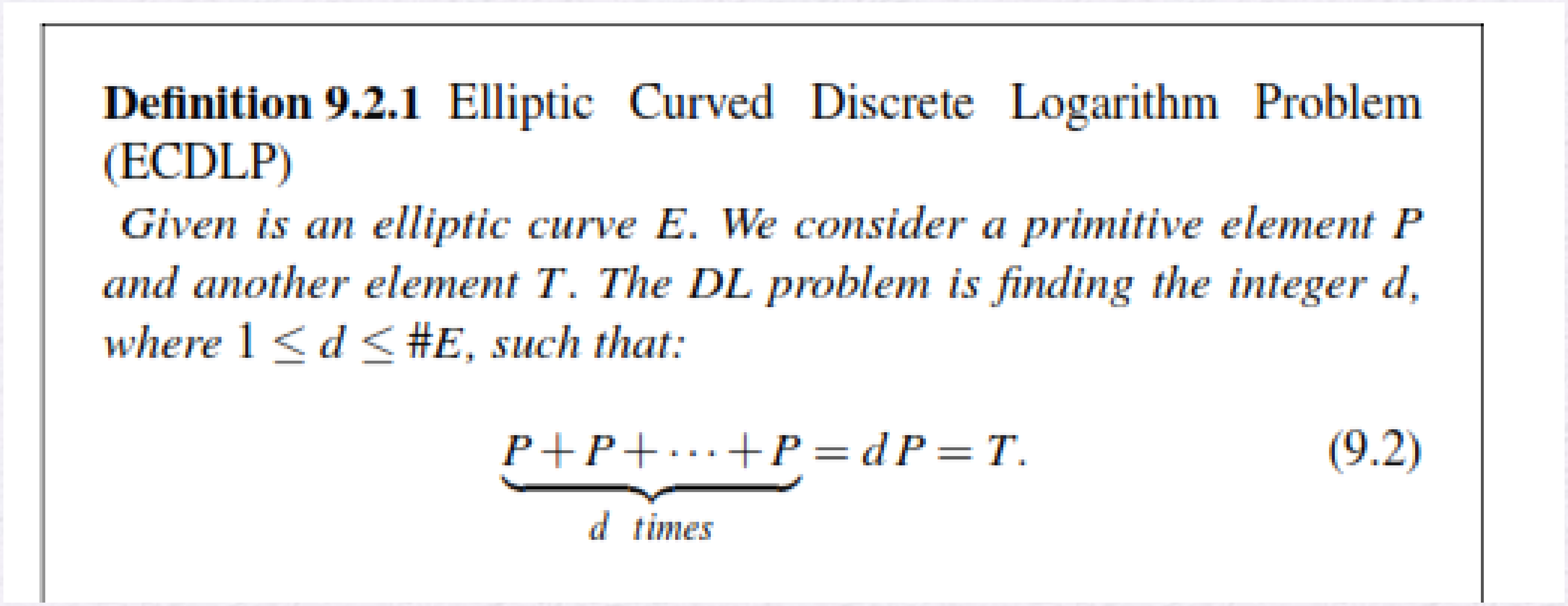
$$E : y^2 \equiv x^3 + 2 \cdot x + 2 \bmod 17.$$

- we start with the primitive element P =(5, 1)

$$\begin{aligned}
2P &= (5,1)+(5,1)=(6,3) & 11P &= (13,10)\\
3P &= 2P+P=(10,6) & 12P &= (0,11)\\
4P &= (3,1) & 13P &= (16,4)\\
5P &= (9,16) & 14P &= (9,1)\\
6P &= (16,13) & 15P &= (3,16)\\
7P &= (0,6) & 16P &= (10,11)\\
8P &= (13,7) & 17P &= (6,14)\\
9P &= (7,6) & 18P &= (5,16)\\
10P &= (7,11) & 19P &= \mathcal{O}
\end{aligned}$$

$$\begin{aligned}
20P &= 19P+P=\mathcal{O}+P=P\\
21P &= 2P
\end{aligned}$$

ECDLP



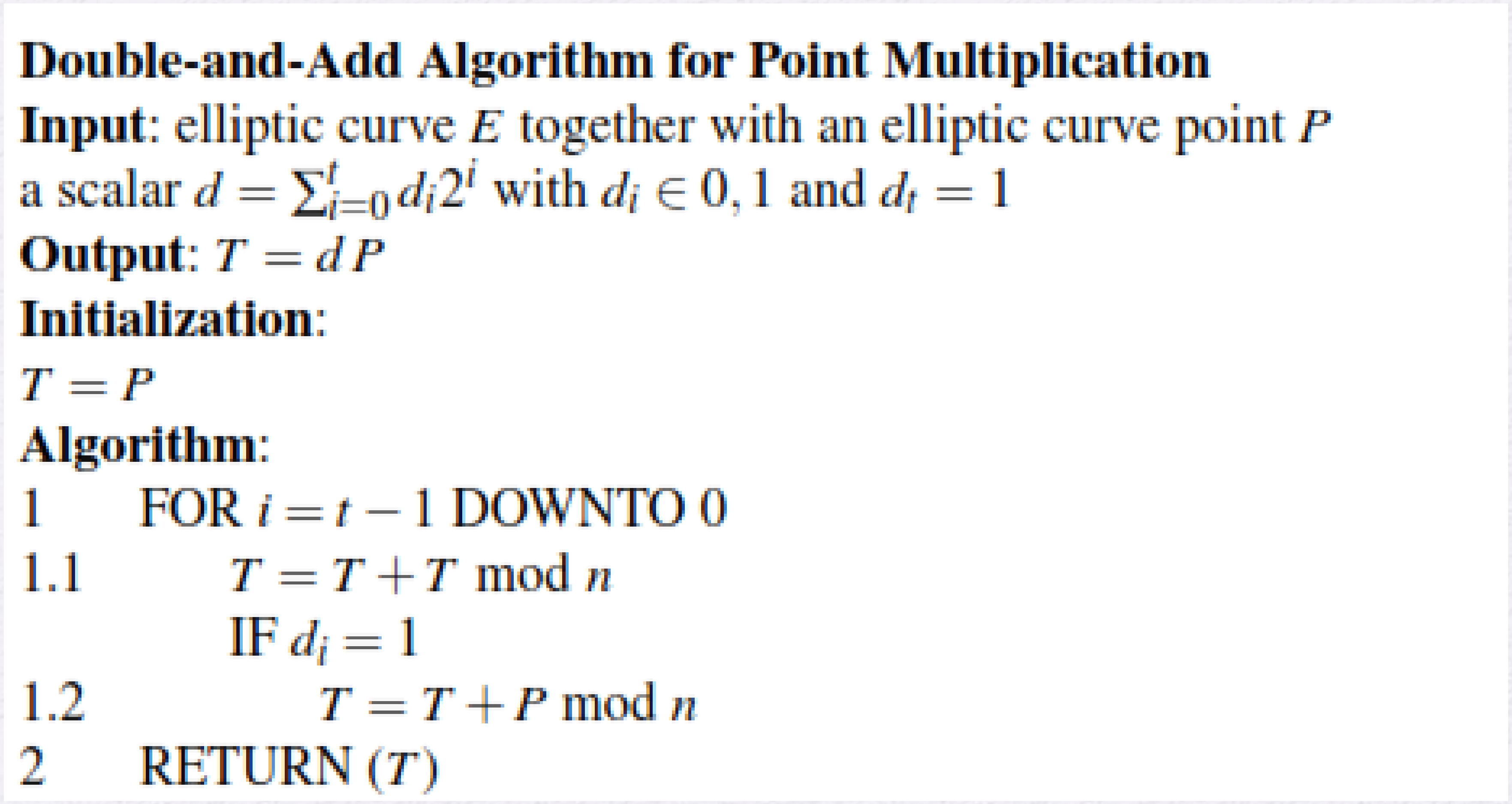
Definition 9.2.1 Elliptic Curved Discrete Logarithm Problem (ECDLP)

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

$$\underbrace{P + P + \cdots + P}_{d \;\; times} = d\,P = T. \qquad (9.2)$$

Efficient

- Point multiplication is analog to exponentiation in multiplicative groups.
- In order to do it efficiently, we can directly adopt the square-and-multiply algorithm.



Double-and-Add Algorithm for Point Multiplication

Input: elliptic curve E together with an elliptic curve point P
a scalar $d = \sum_{i=0}^{t} d_i 2^i$ with $d_i \in 0, 1$ and $d_t = 1$

Output: $T = d\,P$

Initialization:

$T = P$

Algorithm:

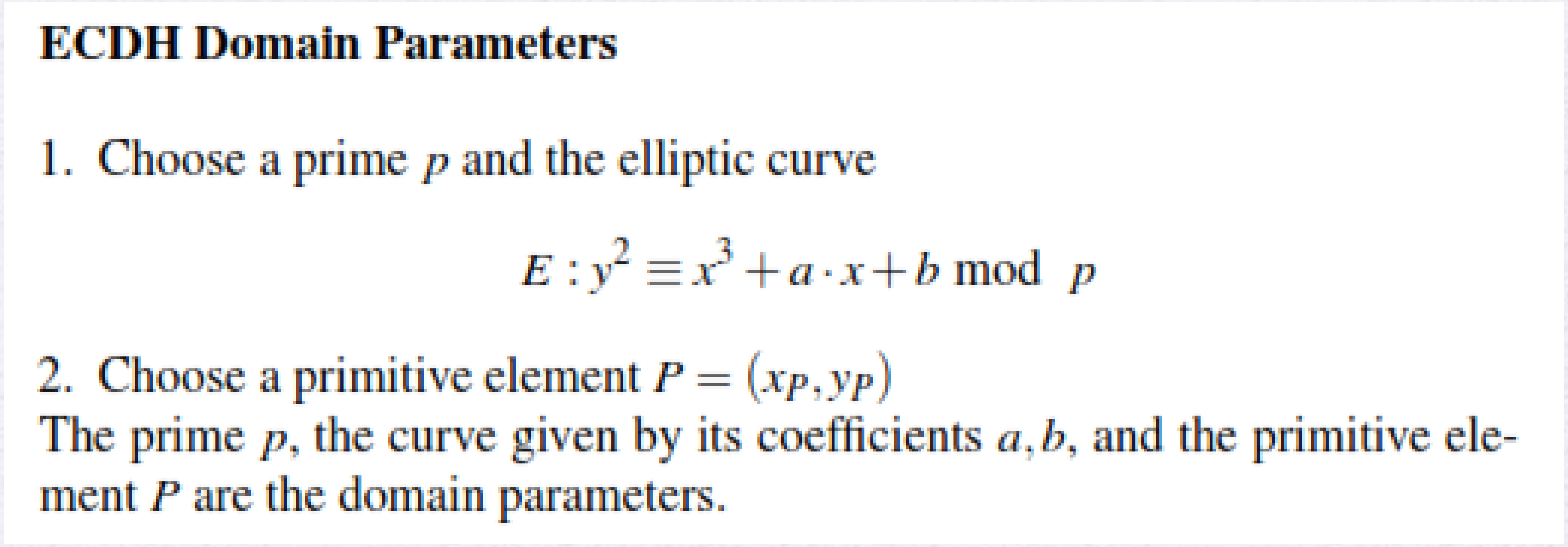
```
1      FOR $i = t - 1$ DOWNTO 0
1.1        $T = T + T \bmod n$
           IF $d_i = 1$
1.2            $T = T + P \bmod n$
2      RETURN ($T$)
```

Example

$$26\,P = (11010_2)\,P = (d_4d_3d_2d_1d_0)_2\,P.$$

Step		
#0	$P = \mathbf{1}_2\,P$	inital setting, bit processed: $d_4 = 1$
#1a	$P + P = 2P = \mathbf{10}_2\,P$	DOUBLE, bit processed: d_3
#1b	$2P + P = 3P = 10_2\,P + 1_2\,P = \mathbf{11}_2\,P$	ADD, since $d_3 = 1$
#2a	$3P + 3P = 6P = 2(11_2\,P) = \mathbf{110_2}\,P$	DOUBLE, bit processed: d_2
#2b		no ADD, since $d_2 = 0$
#3a	$6P + 6P = 12P = 2(110_2\,P) = \mathbf{1100_2}\,P$	DOUBLE, bit processed: d_1
#3b	$12P + P = 13P = 1100_2\,P + 1_2\,P = \mathbf{1101}_2\,P$	ADD, since $d_1 = 1$
#4a	$13P + 13P = 26P = 2(1101_2\,P) = \mathbf{11010_2}\,P$	DOUBLE, bit processed: d_0
#4b		no ADD, since $d_0 = 0$

ECDHP



ECDH Domain Parameters

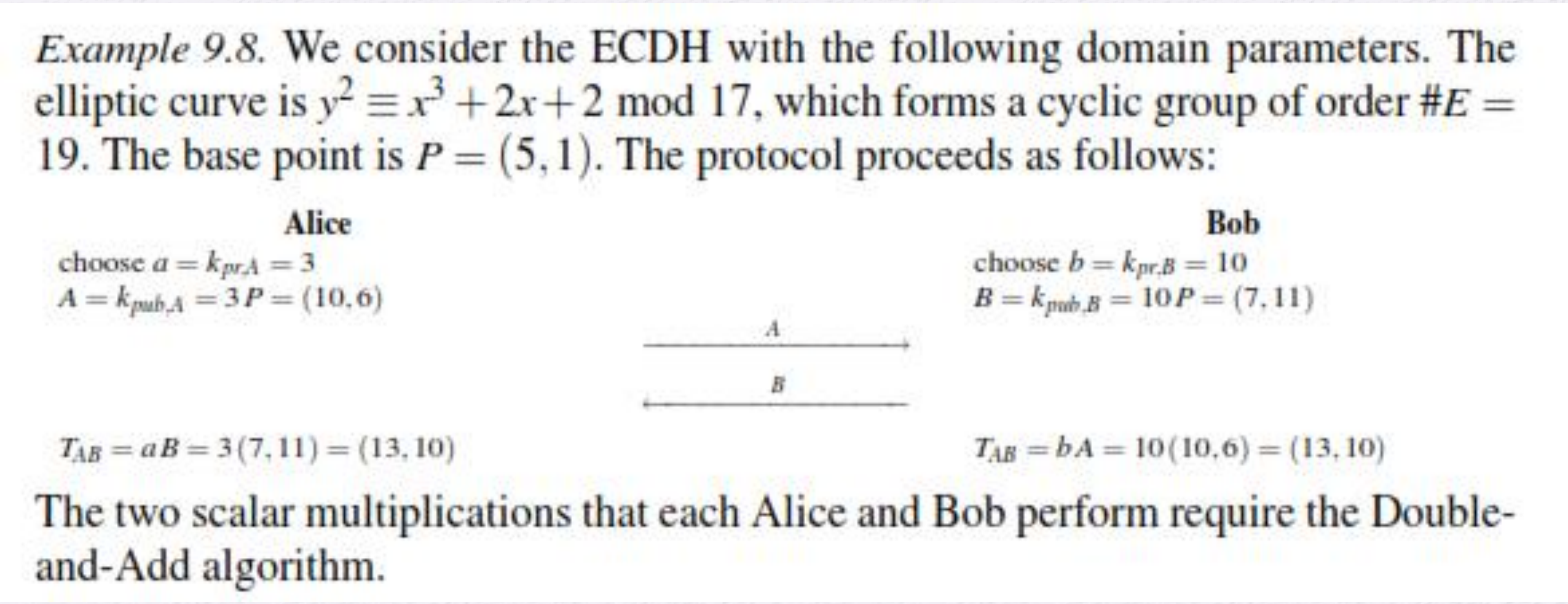
1. Choose a prime p and the elliptic curve

$$E: y^2 \equiv x^3 + a \cdot x + b \bmod p$$

2. Choose a primitive element $P = (x_P, y_P)$

The prime p, the curve given by its coefficients a, b, and the primitive element P are the domain parameters.

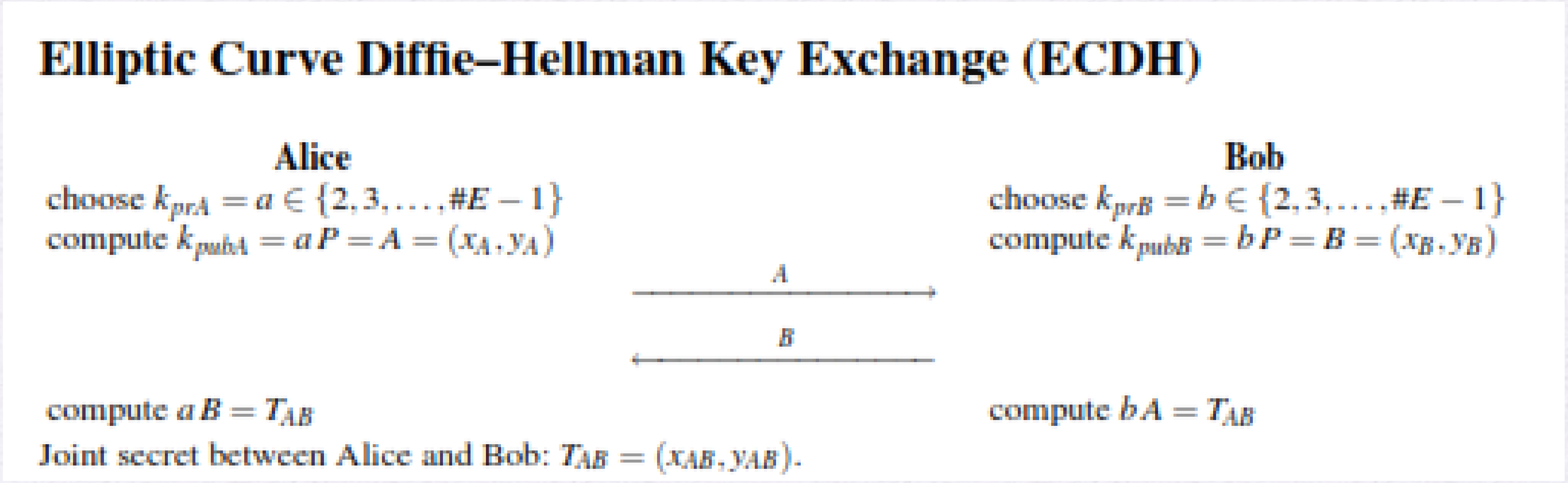
Example



Example 9.8. We consider the ECDH with the following domain parameters. The elliptic curve is $y^2 \equiv x^3 + 2x + 2 \bmod 17$, which forms a cyclic group of order $\#E = 19$. The base point is $P = (5,1)$. The protocol proceeds as follows:

Alice		**Bob**
choose $a = k_{pr,A} = 3$ $A = k_{pub,A} = 3P = (10,6)$		choose $b = k_{pr,B} = 10$ $B = k_{pub,B} = 10P = (7,11)$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$T_{AB} = aB = 3(7,11) = (13,10)$		$T_{AB} = bA = 10(10,6) = (13,10)$

The two scalar multiplications that each Alice and Bob perform require the Double-and-Add algorithm.



Elliptic Curve Diffie–Hellman Key Exchange (ECDH)

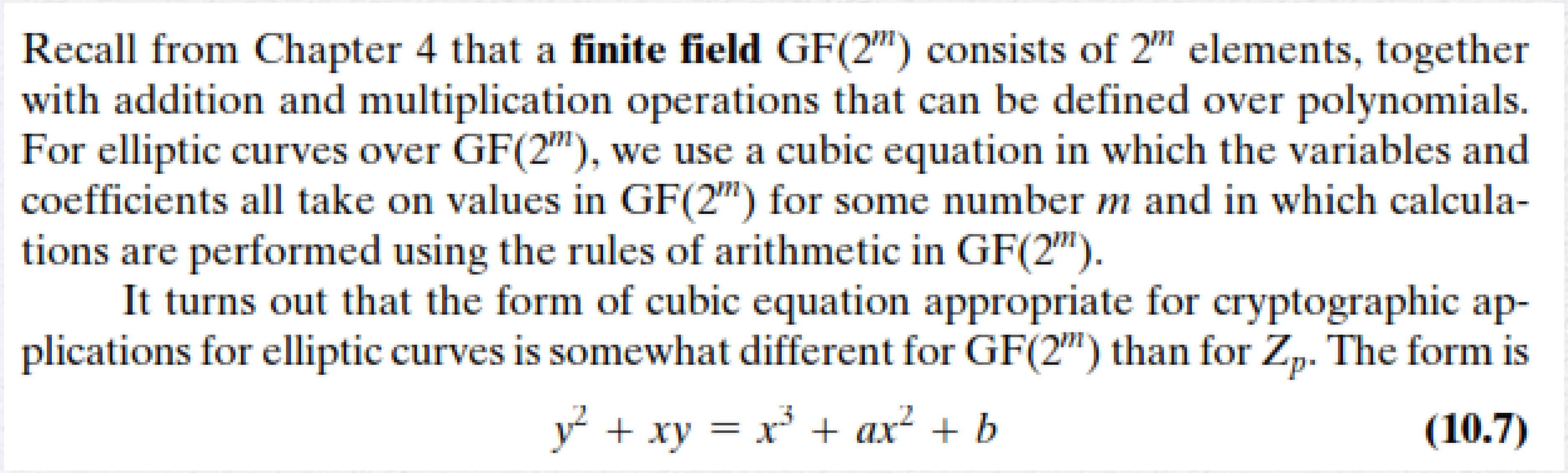
Alice		**Bob**
choose $k_{prA} = a \in \{2, 3, \ldots, \#E-1\}$		choose $k_{prB} = b \in \{2, 3, \ldots, \#E-1\}$
compute $k_{pubA} = aP = A = (x_A, y_A)$		compute $k_{pubB} = bP = B = (x_B, y_B)$
	$\xrightarrow{\quad A \quad}$	
	$\xleftarrow{\quad B \quad}$	
compute $aB = T_{AB}$		compute $bA = T_{AB}$

Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$.

ECDHP

- If an attacker Oscar wants to break the ECDH, he has the following information: E, p, P, A, and B. He wants to compute the joint secret between Alice and Bob
- $T = a \cdot b \cdot P$. This is called the elliptic curve Diffie–Hellman problem (ECDHP).

Example

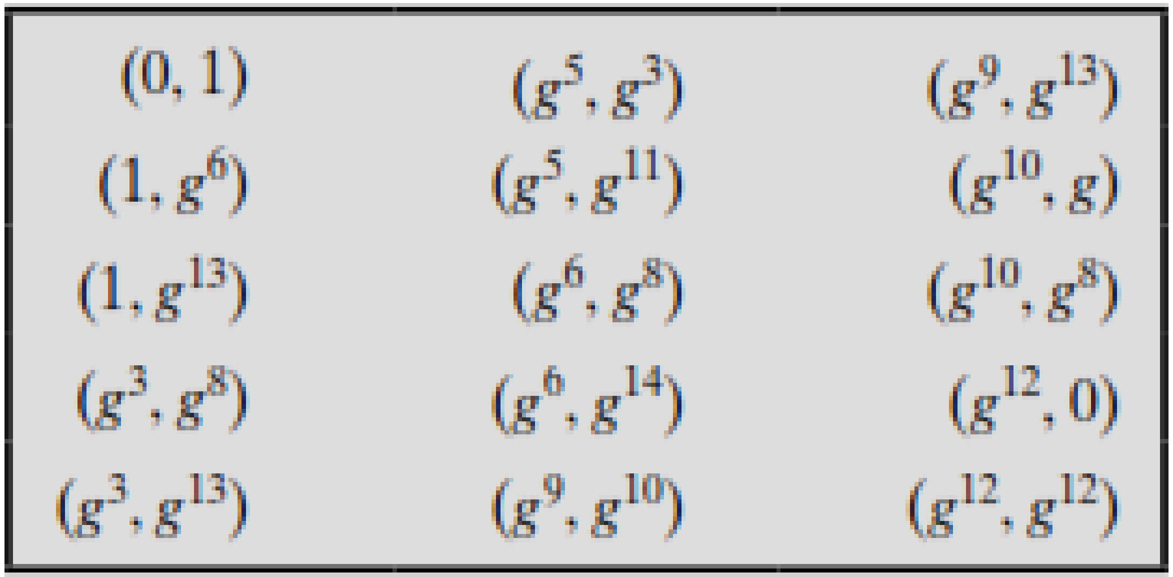


Recall from Chapter 4 that a **finite field** GF(2^m) consists of 2^m elements, together with addition and multiplication operations that can be defined over polynomials. For elliptic curves over GF(2^m), we use a cubic equation in which the variables and coefficients all take on values in GF(2^m) for some number *m* and in which calculations are performed using the rules of arithmetic in GF(2^m).

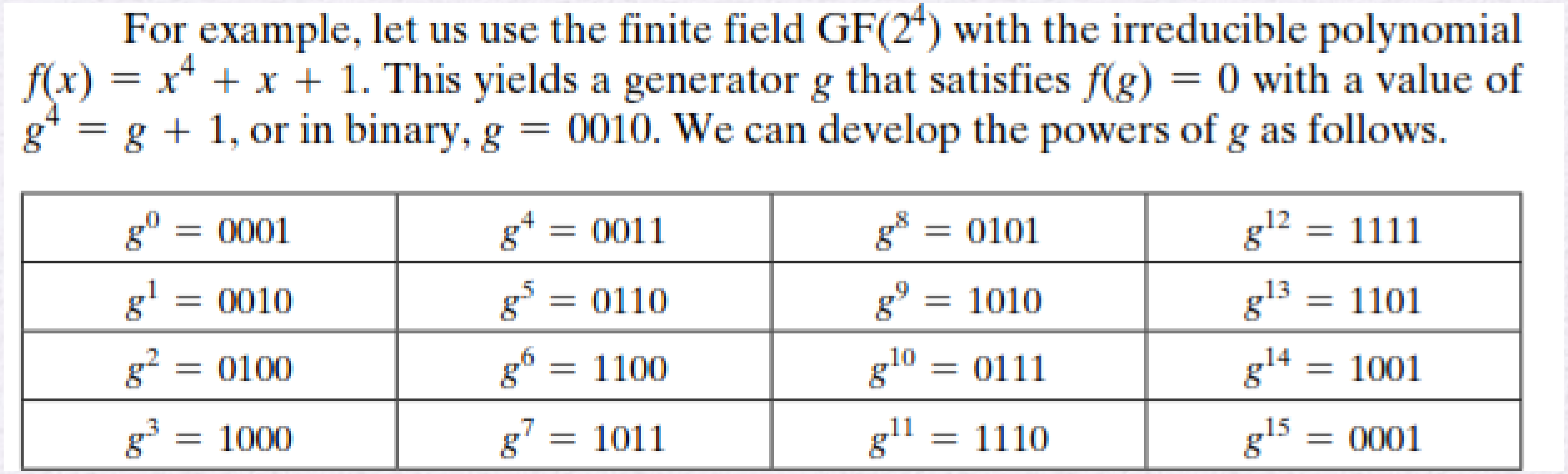
It turns out that the form of cubic equation appropriate for cryptographic applications for elliptic curves is somewhat different for GF(2^m) than for Z_p. The form is

$$y^2 + xy = x^3 + ax^2 + b \quad \textbf{(10.7)}$$

Table 10.2 Points (other than O) on the Elliptic Curve $E_{2^4}(g^4, 1)$

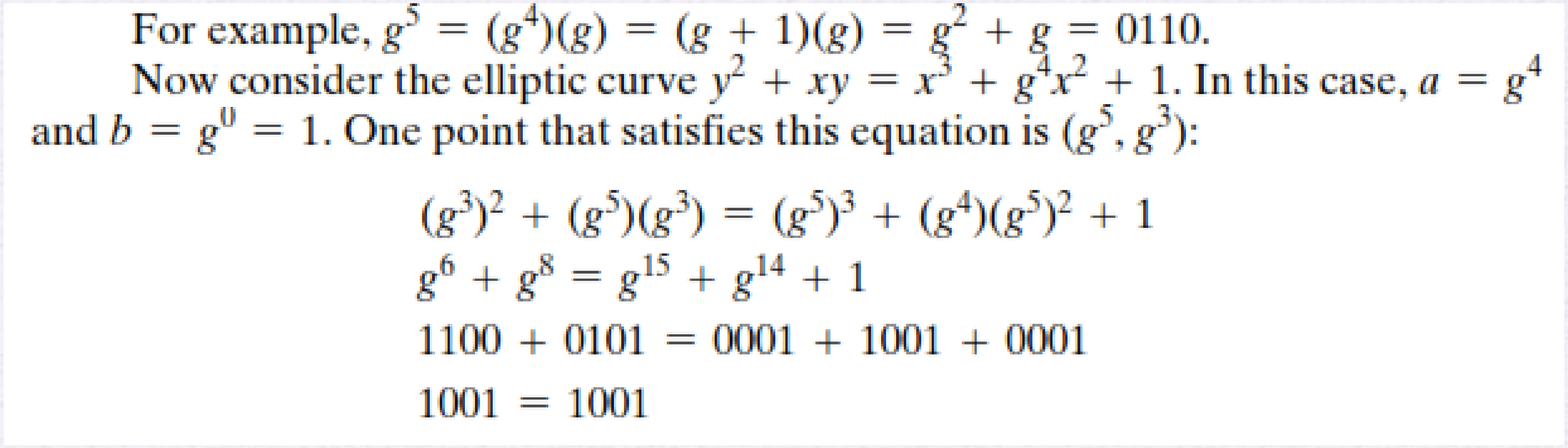


$(0, 1)$	(g^5, g^3)	(g^9, g^{13})
$(1, g^6)$	(g^5, g^{11})	(g^{10}, g)
$(1, g^{13})$	(g^6, g^8)	(g^{10}, g^8)
(g^3, g^8)	(g^6, g^{14})	$(g^{12}, 0)$
(g^3, g^{13})	(g^9, g^{10})	(g^{12}, g^{12})



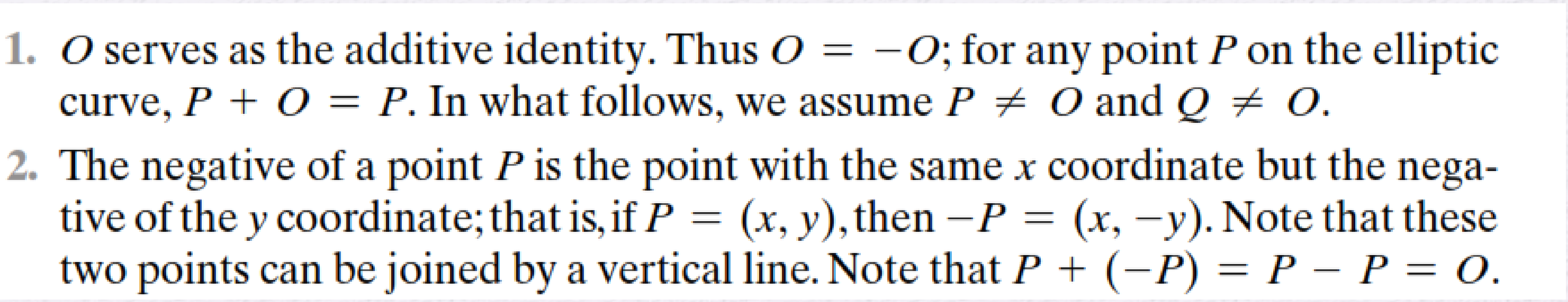
For example, let us use the finite field GF(2^4) with the irreducible polynomial $f(x) = x^4 + x + 1$. This yields a generator g that satisfies $f(g) = 0$ with a value of $g^4 = g + 1$, or in binary, $g = 0010$. We can develop the powers of g as follows.

$g^0 = 0001$	$g^4 = 0011$	$g^8 = 0101$	$g^{12} = 1111$
$g^1 = 0010$	$g^5 = 0110$	$g^9 = 1010$	$g^{13} = 1101$
$g^2 = 0100$	$g^6 = 1100$	$g^{10} = 0111$	$g^{14} = 1001$
$g^3 = 1000$	$g^7 = 1011$	$g^{11} = 1110$	$g^{15} = 0001$

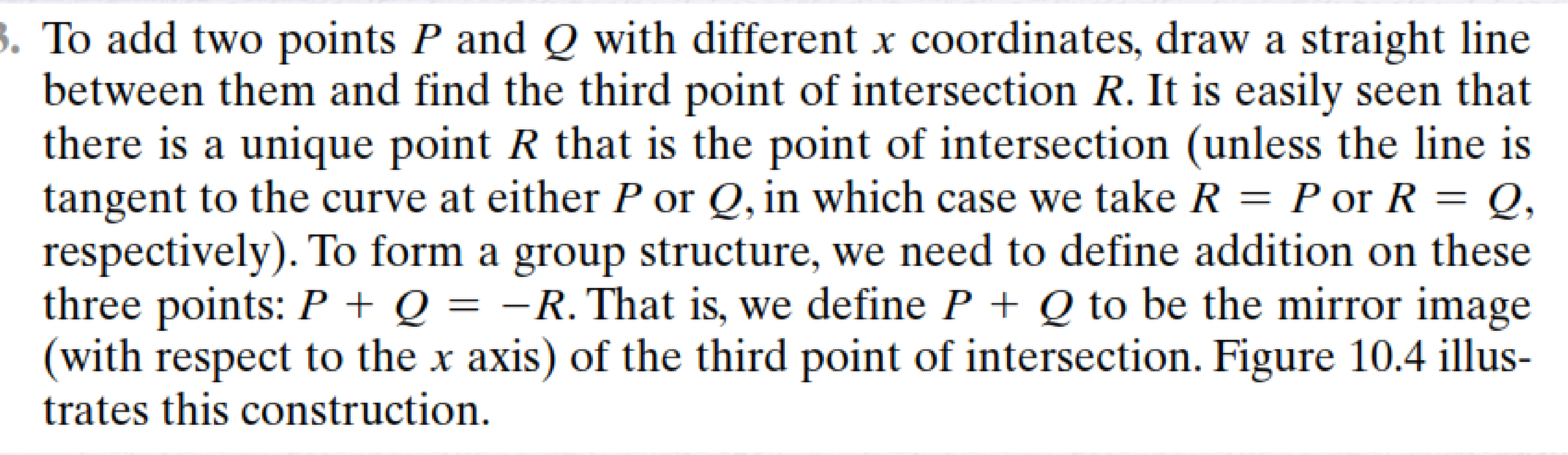

For example, $g^5 = (g^4)(g) = (g + 1)(g) = g^2 + g = 0110$.

Now consider the elliptic curve $y^2 + xy = x^3 + g^4x^2 + 1$. In this case, $a = g^4$ and $b = g^0 = 1$. One point that satisfies this equation is (g^5, g^3):

$$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$$
$$g^6 + g^8 = g^{15} + g^{14} + 1$$
$$1100 + 0101 = 0001 + 1001 + 0001$$
$$1001 = 1001$$



1. O serves as the additive identity. Thus $O = -O$; for any point P on the elliptic curve, $P + O = P$. In what follows, we assume $P \neq O$ and $Q \neq O$.
2. The negative of a point P is the point with the same x coordinate but the negative of the y coordinate; that is, if $P = (x, y)$, then $-P = (x, -y)$. Note that these two points can be joined by a vertical line. Note that $P + (-P) = P - P = O$.



3. To add two points P and Q with different x coordinates, draw a straight line between them and find the third point of intersection R. It is easily seen that there is a unique point R that is the point of intersection (unless the line is tangent to the curve at either P or Q, in which case we take $R = P$ or $R = Q$, respectively). To form a group structure, we need to define addition on these three points: $P + Q = -R$. That is, we define $P + Q$ to be the mirror image (with respect to the x axis) of the third point of intersection. Figure 10.4 illustrates this construction.

Elliptic Curve Addition

- Adding distinct points P and Q
  - $P=(x_P, y_P)$, $Q=(x_Q, y_Q)$ are not negative of each other
  - P + Q = R where

    $s= (y_P-y_Q)/ (x_P-x_Q)$

    $x_R = s^2-x_P-x_Q$

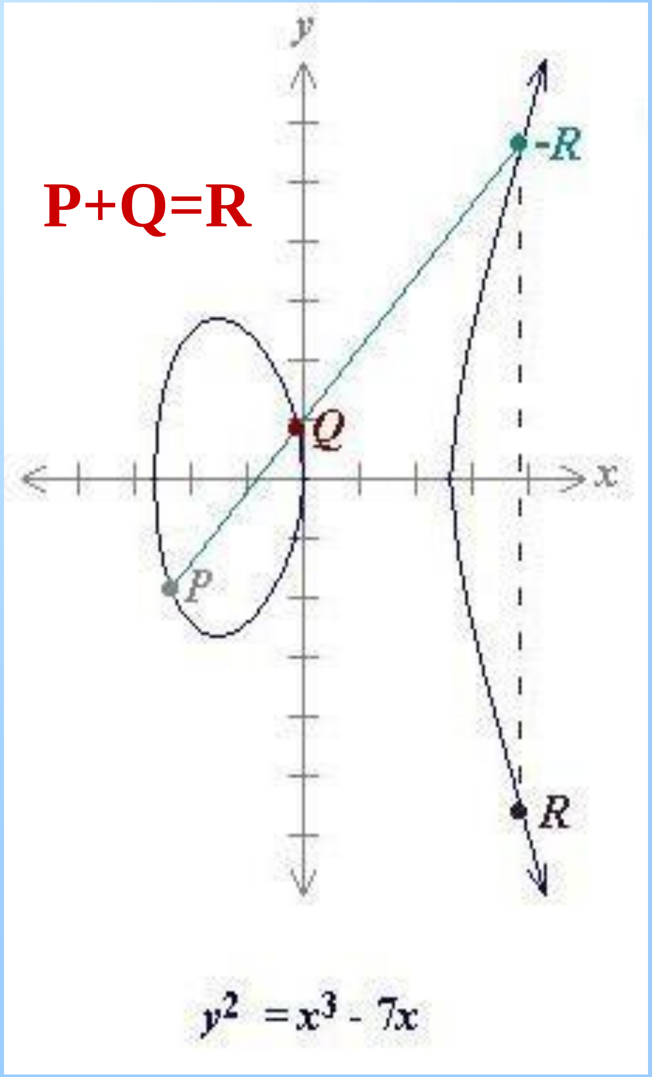
    $y_R = -y_P+s (x_P-x_R)$
  - s is the slope of the line through P and Q
  - Example:

    P(-2.35, -1.86), Q(-0.1, 0.836)

    –R(3.89, 5.62), R(3.89, -5.62)

    P+Q=R=(3.89, -5.62)

P+Q=R

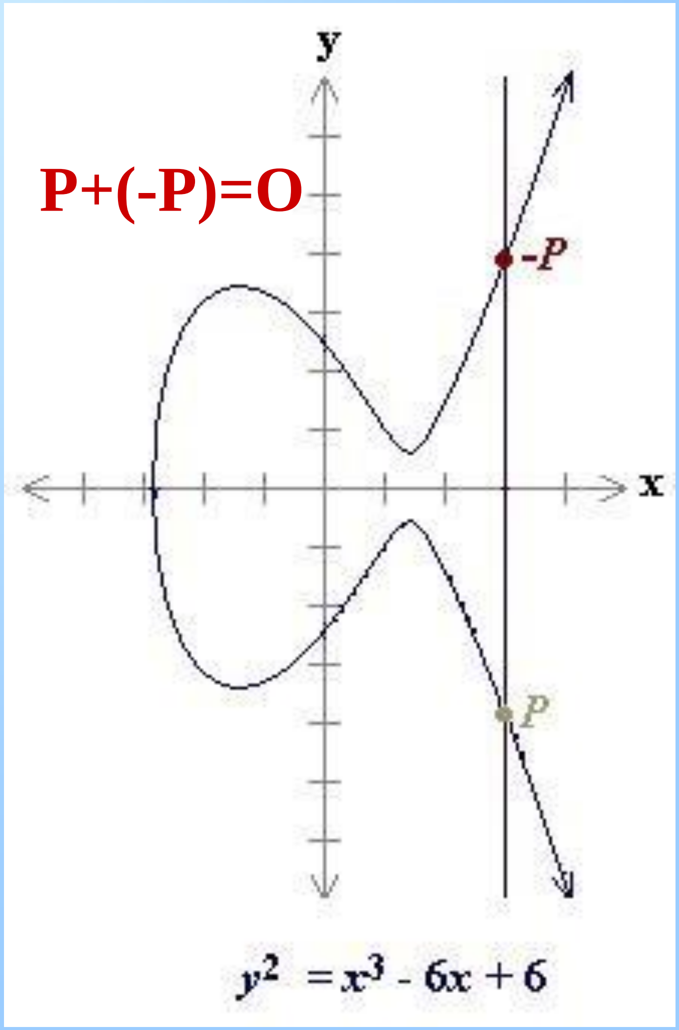


Elliptic Curve Addition

- Adding Points P and -P
  - ♣ Elliptic curve group includes the point at infinity O.

    P+(-P)=O

    $\Rightarrow$ P+O=P
  - ♣ All elliptic curves have the point at infinity O



Elliptic Curve Addition

- Doubling the Point P if $y_P \neq 0$
  - ♣ $P=(x_P, y_P), y_P \neq 0$
  - ♣ $P+P=2P=R$

    $s= (3x_P^2 +a)/ (2y_P)$

    $x_R = s^2-2x_P$

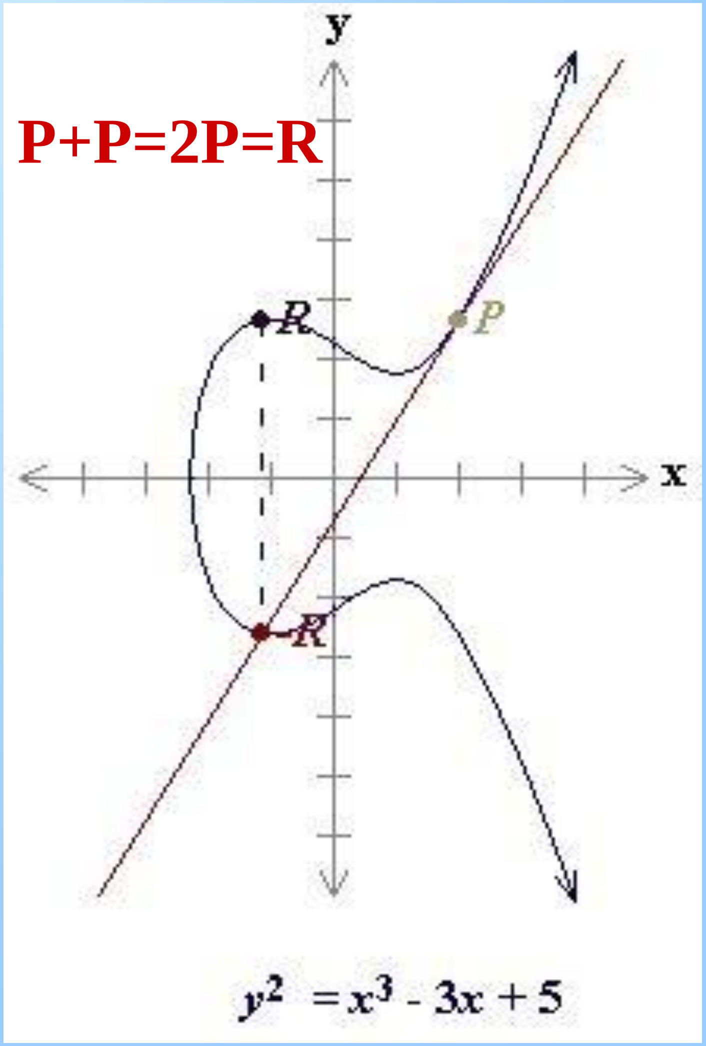
    $y_R = -y_P+s (x_P-x_R)$
  - ♣ Example:

    P(2, 2.65),

    –R(-1.11, -2.64),

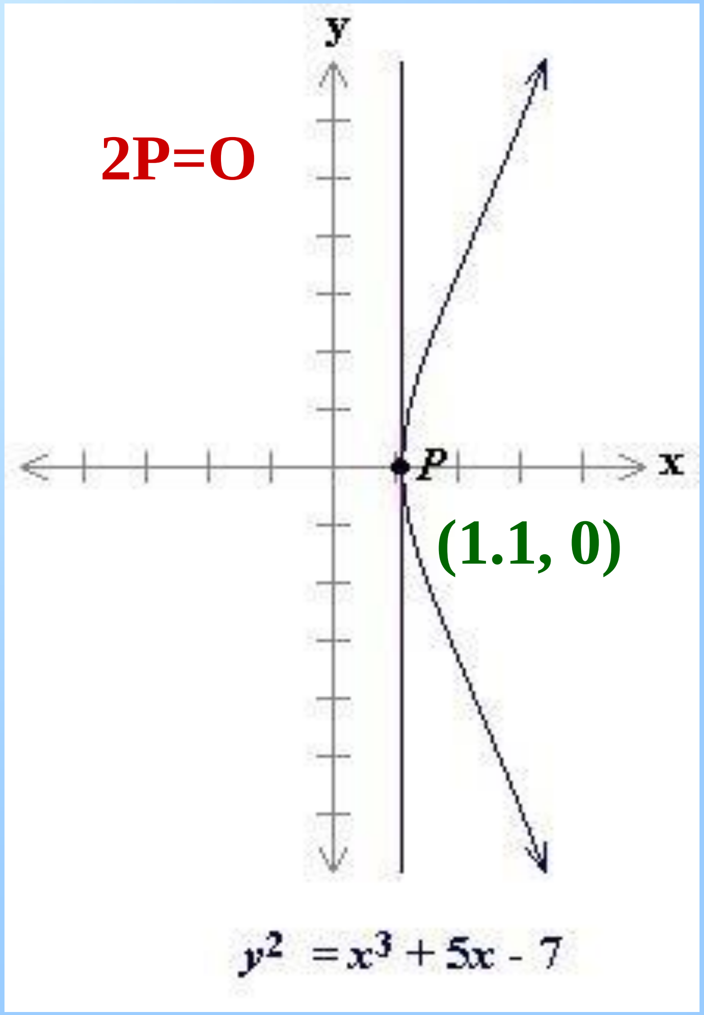
    R(-1.11, 2.64)

    2P=R=(-1.11, -2.64)



Elliptic Curve Addition

- Doubling the Point P if $y_P=0$
  - ♣ P+P=2P=O
  - ♣ 3P= 2P+P=O+P=P
  - ♣ 4P= 3P+P=P+P=O
  - ♣ 5P=P, 6P=O, 7P=P, etc



Elliptic Curve Over Finite Field F_z

- Major Difference between Elliptic Curve Over F_z and Over Real Numbers

♣ Elliptic curve over F_z has a finite number of points

♣ Unlike elliptic curve over real numbers, computations over F_z involve no round off error

♣ Computations are performed by modulo z

Elliptic Curve Over Finite Field F_z

- Set of points (x, y) satisfy
  $y^2 = x^3 + ax + b \bmod z$,
  where z is a prime number>3, a, b, x, y $\in F_z$
- Adding Distinct Points P and Q
  - $P=(x_p, y_p)$, $-P=(x_p, -y_p \bmod z)$.
  - P+Q=R where

    $s= (y_P-y_Q)/ (x_P-x_Q) \bmod z$

    $x_R = s^2-x_P-x_Q \bmod z$

    $y_R = -y_P+s (x_P-x_R) \bmod z$
- Doubling the Point P if $y_p \neq 0$
  - 2P=R where

    $s= (3x_P^2 +a)/ (2y_P) \bmod z$

    $x_R = s^2-2x_P \bmod z$

    $y_R = -y_P+s (x_P-x_R) \bmod z$

Conclude Elliptic Curve Theory

- Crucial Property of an Elliptic Curve

♣ Define a rule for “adding” two points which are on the elliptic curve, to obtain a 3rd point which is also on the elliptic curve

♣ Include a special point O, which does not satisfy the elliptic curve equation

- Order of a Point

♣ Order of a point P on the elliptic curve is the smallest integer r such that r*P=O

Elliptic Curve Discrete Logarithm Problem (ECDLP)

- Public-key cryptography systems use hard-to-solve problems as the basis of the algorithm
  - Prime factorization is a hard problem used by RSA
- ECDLP is a "hard" problem used by ECC
  - Given two points Q & G on elliptic curve, such that Q = d*G
  - Can we easily find integer d?
- Q is public key, d is private key
- Relatively easy to perform, but extremely difficult to reverse

Elliptic Curve Diffie-Helman Protocol

Q: public key
d: private key
G: a fixed point on elliptic curve

Generates d_{Alice}

Computes $Q_{Alice}=d_{Alice}*G$

shared secret key

Computes $P_1=d_{Alice}*Q_{Bob}$
$=d_{Alice}*(d_{Bob}*G)$

Publish Curve Point Q_{Alice} →

← Publish Q_{Bob}

Use this computed point P_1 or P_2 as the shared secret key ↔

$P_1=P_2=d_{Alice}*d_{Bob}*G$

Generates d_{Bob}

Computes $Q_{Bob}=d_{Bob}*G$

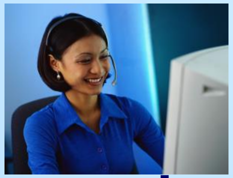
shared secret key

Computes $P_2=d_{Bob}*Q_{Alice}$
$=d_{Bob}*(d_{Alice}*G)$

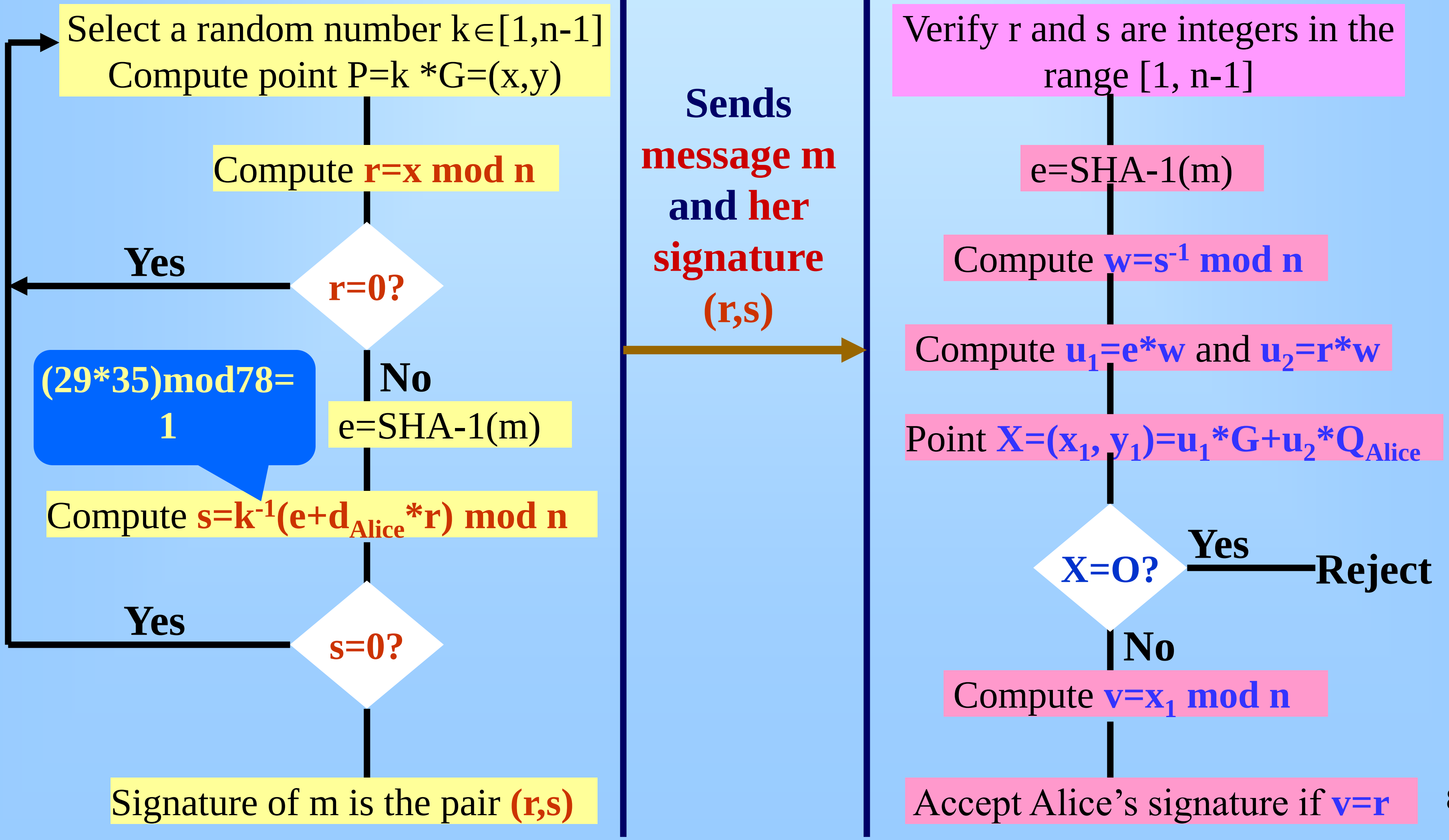
Given a curve point G and the result of d*G, it is difficult to compute d.

Elliptic Curve Digital Signature Authentication

Q$_{Alice}$: public key
d$_{Alice}$: private key



G: a point on elliptic curve
n: order of point G, n*G=O



Select a random number $k \in [1,n-1]$
Compute point $P=k*G=(x,y)$
Compute $r=x \bmod n$
Yes
$r=0?$
No
$(29*35) \bmod 78 = 1$
$e=SHA\text{-}1(m)$
Compute $s=k^{-1}(e+d_{Alice}*r) \bmod n$
Yes
$s=0?$
Signature of m is the pair (r,s)
Sends message m and her signature (r,s)
Verify r and s are integers in the range $[1, n-1]$
$e=SHA\text{-}1(m)$
Compute $w=s^{-1} \bmod n$
Compute $u_1=e*w$ and $u_2=r*w$
Point $X=(x_1, y_1)=u_1*G+u_2*Q_{Alice}$
$X=O?$
Yes
Reject
No
Compute $v=x_1 \bmod n$
Accept Alice's signature if $v=r$

Security Analysis

- ECC can offer same levels of security with small size keys comparable to RSA and other public key cryptography methods

RSA Key Size	Time to Break Key (MIPS Years)	**ECC** Key Size for Equivalent Security	**RSA:ECC** Key Size Ratio
512	10^4	106	5:1
768	10^8	132	6:1
1,024	10^{11}	160	7:1
2,048	10^{20}	210	10:1
21,000	10^{78}	600	35:1

- Designed for devices with limited memory, bandwidth, computational power, e.g. smartcards and PDAs

Group Operation

+ **ADDITION**

point at infinity

Given two points in the set $E = \{(x,y) \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$

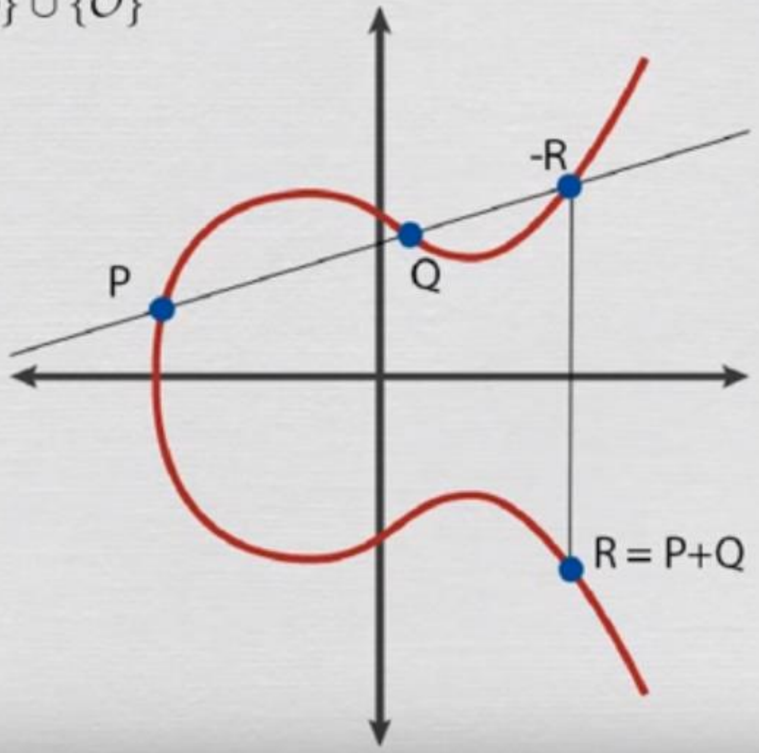
$P+Q=?$

Algebraically

$$s = \frac{y_P - y_Q}{x_P - x_Q}$$

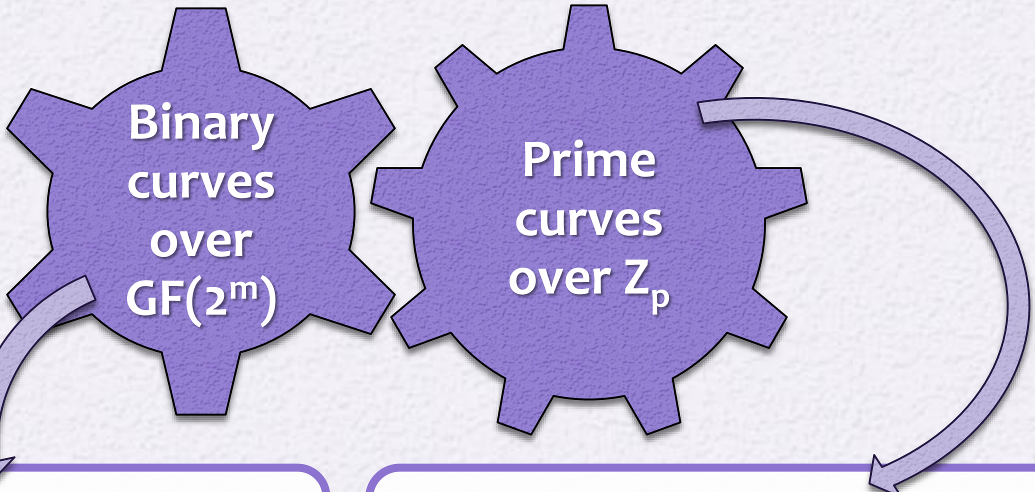
$$x_R = s^2 - (x_P + x_Q)$$

$$y_R = s(x_P - x_R) - y_P$$



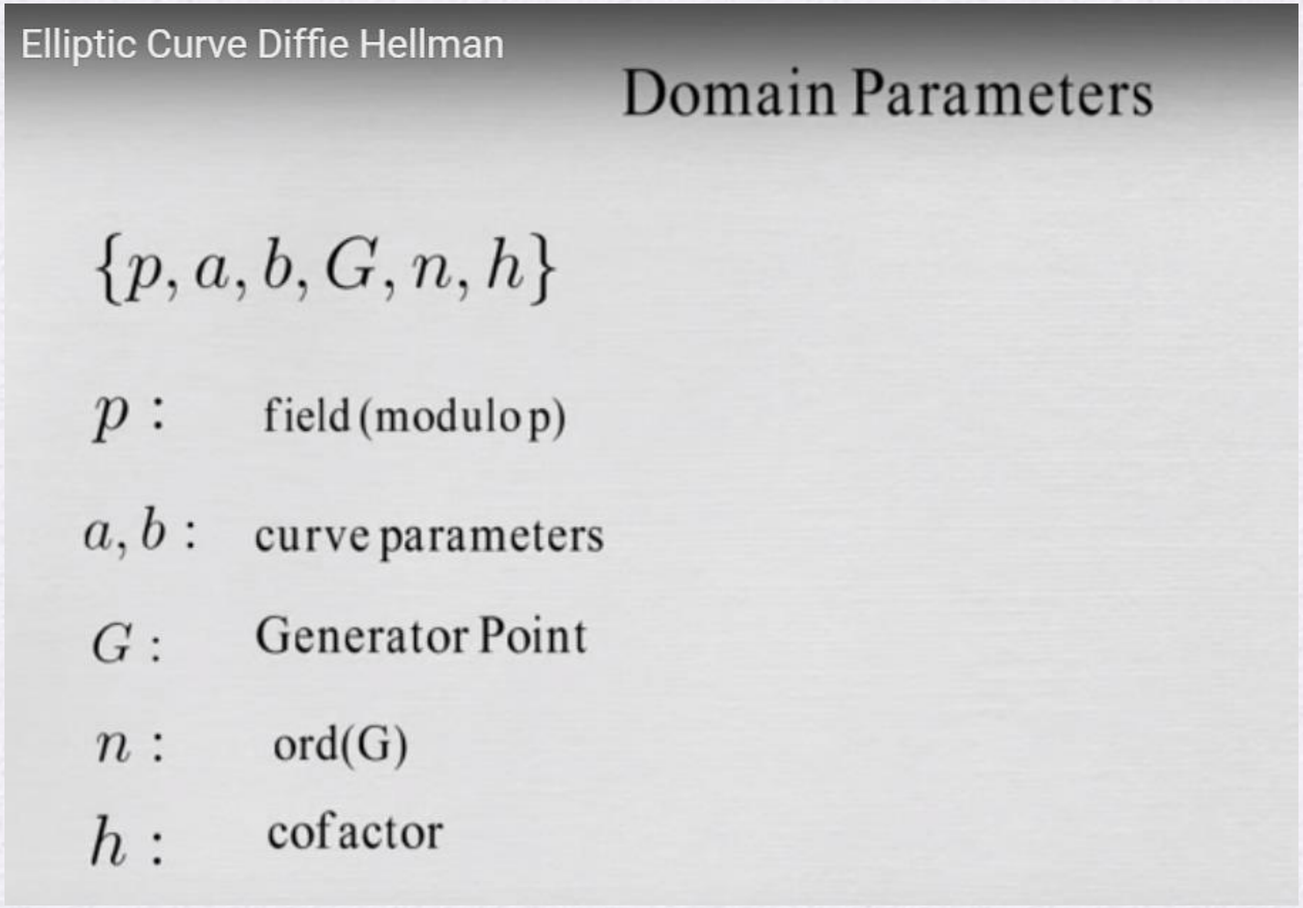
Elliptic Curves Over Z_p

- Elliptic curve cryptography uses curves whose variables and coefficients are finite
- Two families of elliptic curves are used in cryptographic applications:



- **Variables and coefficients all take on values in $GF(2^m)$ and in calculations are performed over $GF(2^m)$**
- **Best for hardware applications**

- **Use a cubic equation in which the variables and coefficients all take on values in the set of integers from 0 through p-1 and in which calculations are performed modulo p**
- **Best for software applications**

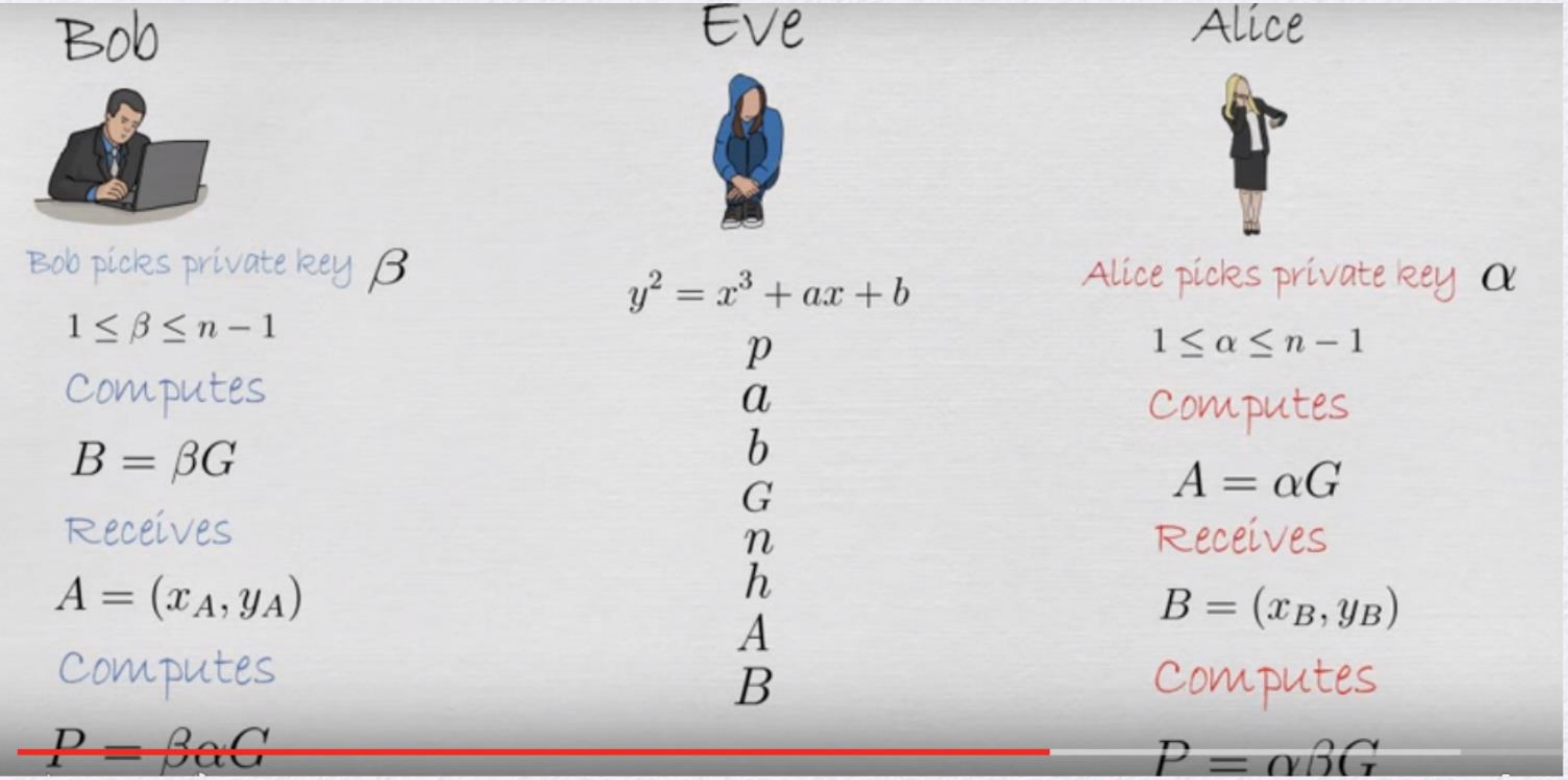

Elliptic Curve Diffie Hellman

Domain Parameters

$$\{p, a, b, G, n, h\}$$

p :	field (modulo p)
a, b :	curve parameters
G :	Generator Point
n :	ord(G)
h :	cofactor

EC Diffie Hellman key exchange



Bob

Bob picks private key β

$1 \leq \beta \leq n-1$

Computes

$B = \beta G$

Receives

$A = (x_A, y_A)$

Computes

$P = \beta\alpha G$

Eve

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

p

a

b

G

n

h

A

B

Alice

Alice picks private key α

$1 \leq \alpha \leq n-1$

Computes

$A = \alpha G$

Receives

$B = (x_B, y_B)$

Computes

$P = \alpha\beta G$

Example

$$E: \quad y^2 \equiv x^3 + 2x + 2 \pmod{17}$$

$$G = (5, 1)$$

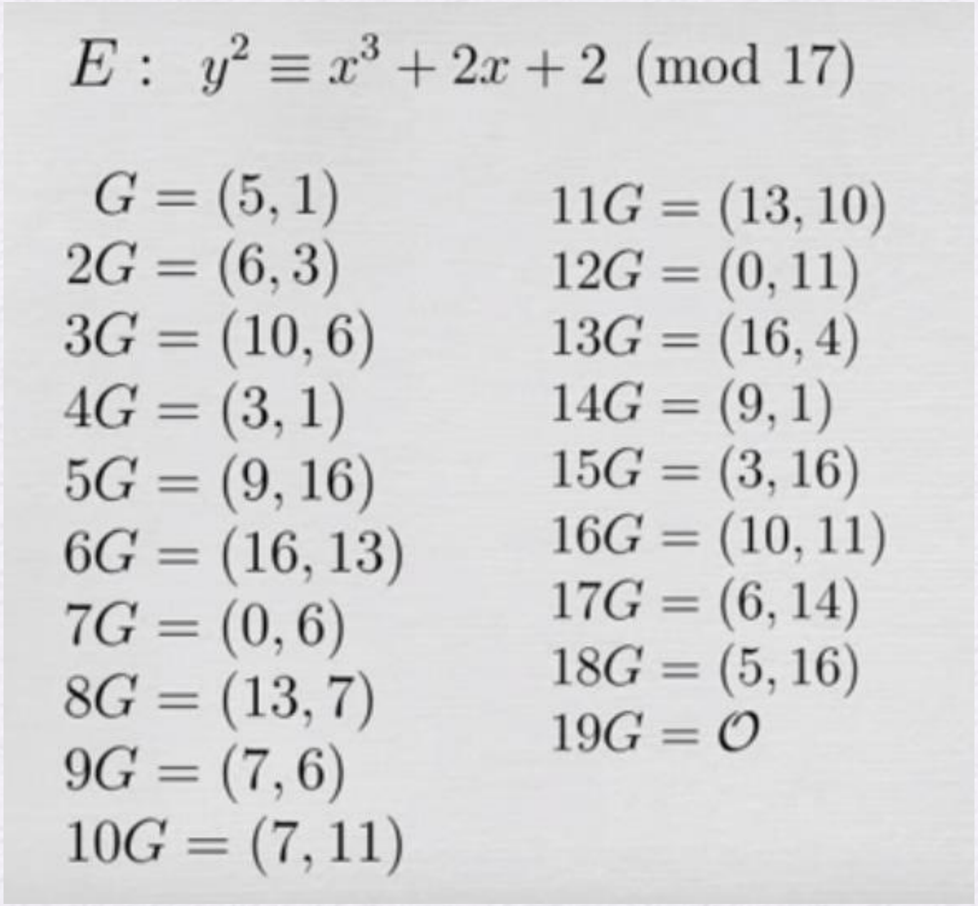
Cyclic Group

COMPUTE $2G = G + G$

$$s = \frac{3x_G^2 + a}{2y_G} \qquad s \equiv \frac{3(5^2) + 2}{2(1)} \equiv 77 \cdot 2^{-1} \equiv 9 \cdot 9 \equiv 13 \pmod{17}$$

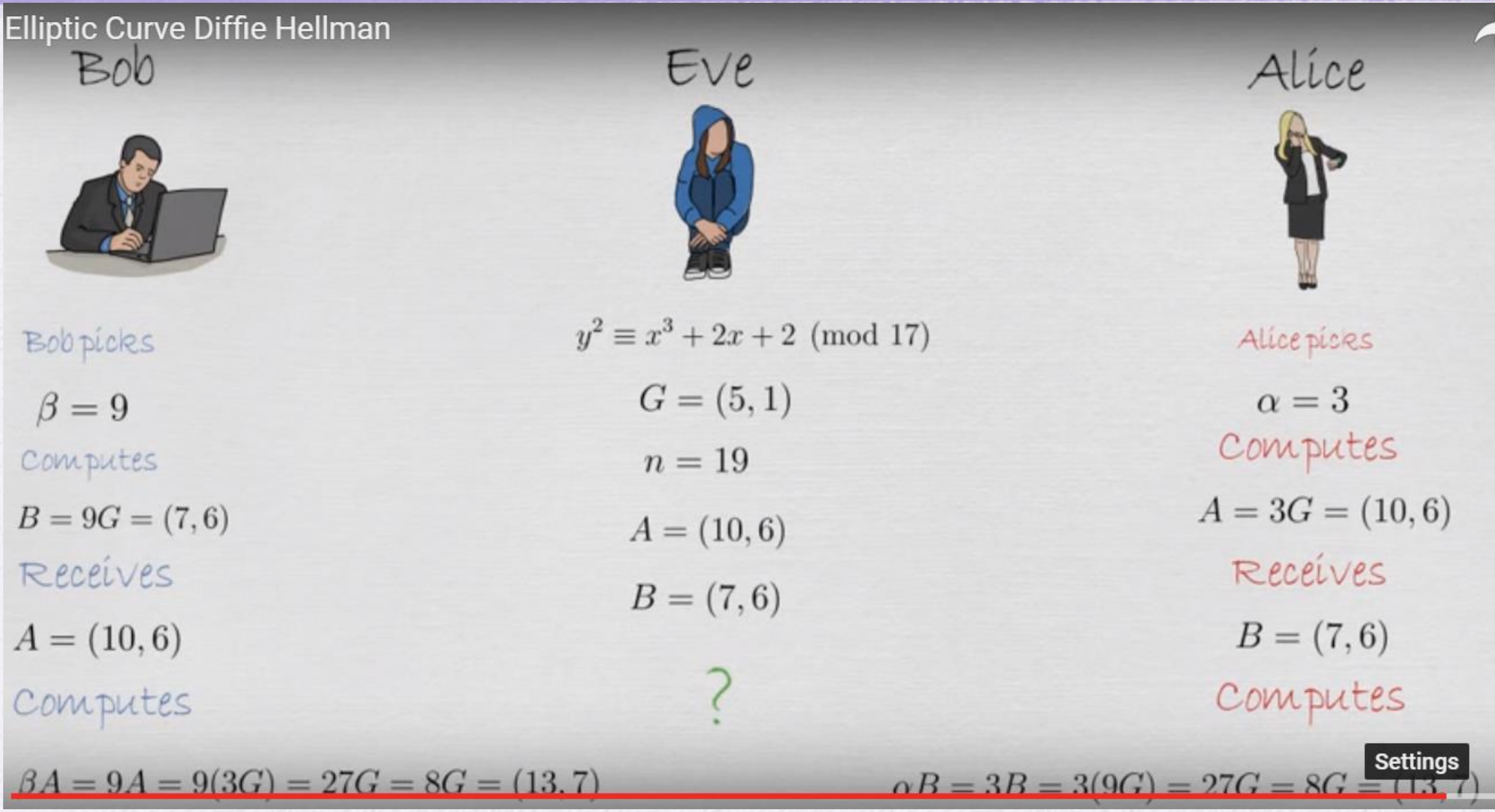
$$x_{2G} = s^2 - 2x_G \qquad x_{2G} \equiv 13^2 - 2(5) \equiv 16 - 10 \equiv 6 \pmod{17}$$

$$y_{2G} = s(x_G - x_{2G}) - y_G \qquad y_{2G} \equiv 13(5 - 6) - 1 \equiv -13 - 1 \equiv -14 \equiv 3 \pmod{17}$$



$$E: \; y^2 \equiv x^3 + 2x + 2 \pmod{17}$$

$G = (5, 1)$	$11G = (13, 10)$
$2G = (6, 3)$	$12G = (0, 11)$
$3G = (10, 6)$	$13G = (16, 4)$
$4G = (3, 1)$	$14G = (9, 1)$
$5G = (9, 16)$	$15G = (3, 16)$
$6G = (16, 13)$	$16G = (10, 11)$
$7G = (0, 6)$	$17G = (6, 14)$
$8G = (13, 7)$	$18G = (5, 16)$
$9G = (7, 6)$	$19G = \mathcal{O}$
$10G = (7, 11)$	



Elliptic Curve Diffie Hellman

Bob

Bob picks

$\beta = 9$

Computes

$B = 9G = (7, 6)$

Receives

$A = (10, 6)$

Computes

$\beta A = 9A = 9(3G) = 27G = 8G = (13, 7)$

Eve

$y^2 \equiv x^3 + 2x + 2 \pmod{17}$

$G = (5, 1)$

$n = 19$

$A = (10, 6)$

$B = (7, 6)$

?

Alice

Alice picks

$\alpha = 3$

Computes

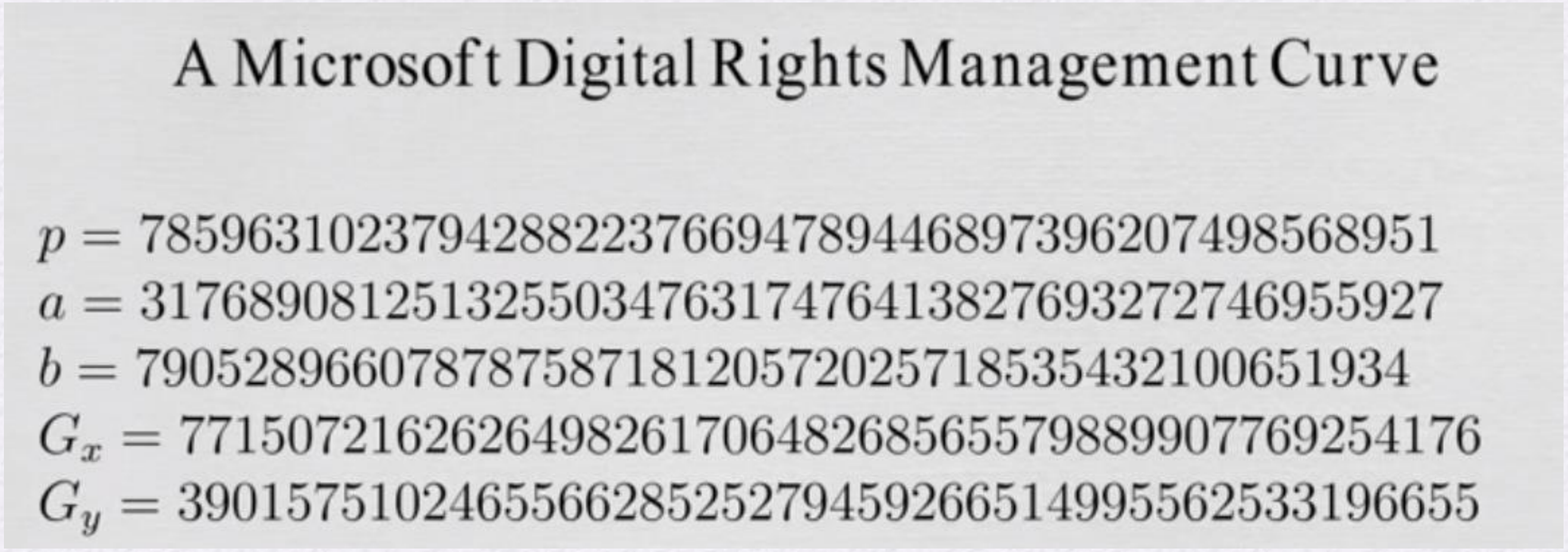
$A = 3G = (10, 6)$

Receives

$B = (7, 6)$

Computes

$\alpha B = 3B = 3(9G) = 27G = 8G = (13, 7)$

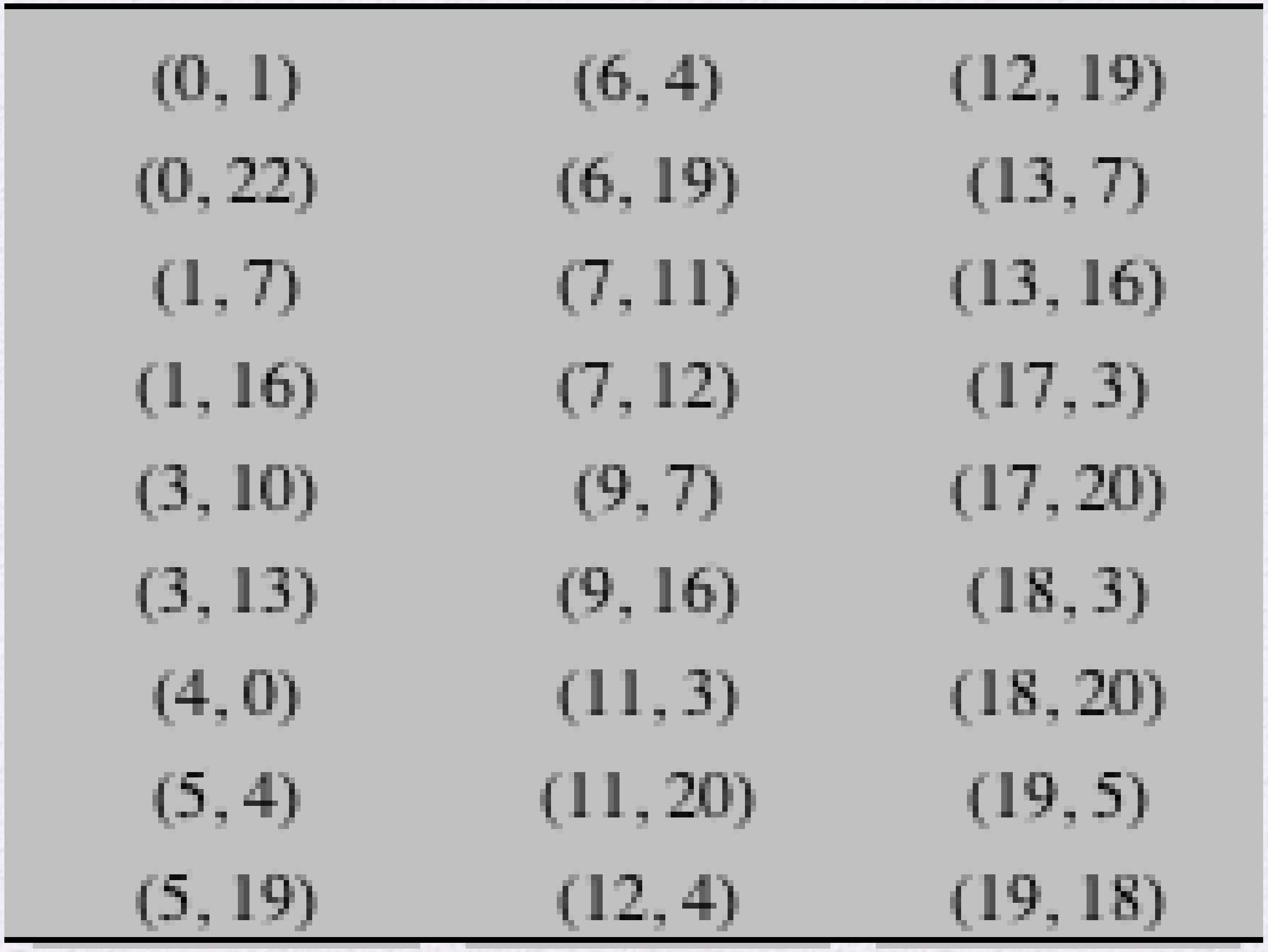


A Microsoft Digital Rights Management Curve

$$p = 785963102379428822376694789446897396207498568951$$
$$a = 317689081251325503476317476413827693272746955927$$
$$b = 79052896607878758718120572025718535432100651934$$
$$G_x = 771507216262649826170648268565579889907769254176$$
$$G_y = 390157510246556628525279459266514995562533196655$$

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)

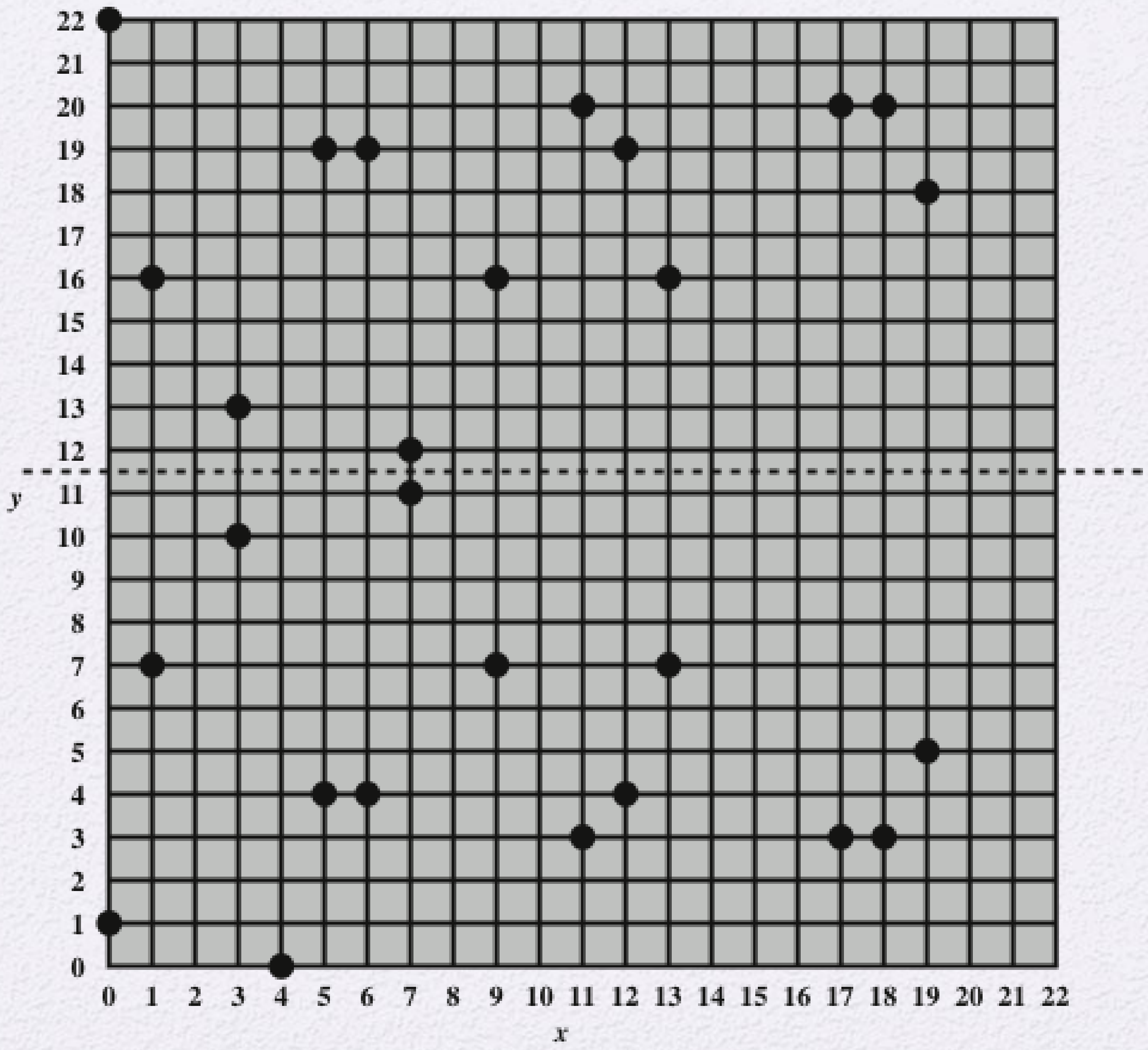


Figure 10.5 The Elliptic Curve $E_{23}(1,1)$

Elliptic Curves Over GF(2^m)

- Use a cubic equation in which the variables and coefficients all take on values in GF(2^m) for some number *m*
- Calculations are performed using the rules of arithmetic in GF(2^m)
- The form of cubic equation appropriate for cryptographic applications for elliptic curves is somewhat different for GF(2^m) than for Z_p
    - It is understood that the variables *x* and *y* and the coefficients *a* and *b* are elements of GF(2^m) and that calculations are performed in GF(2^m)

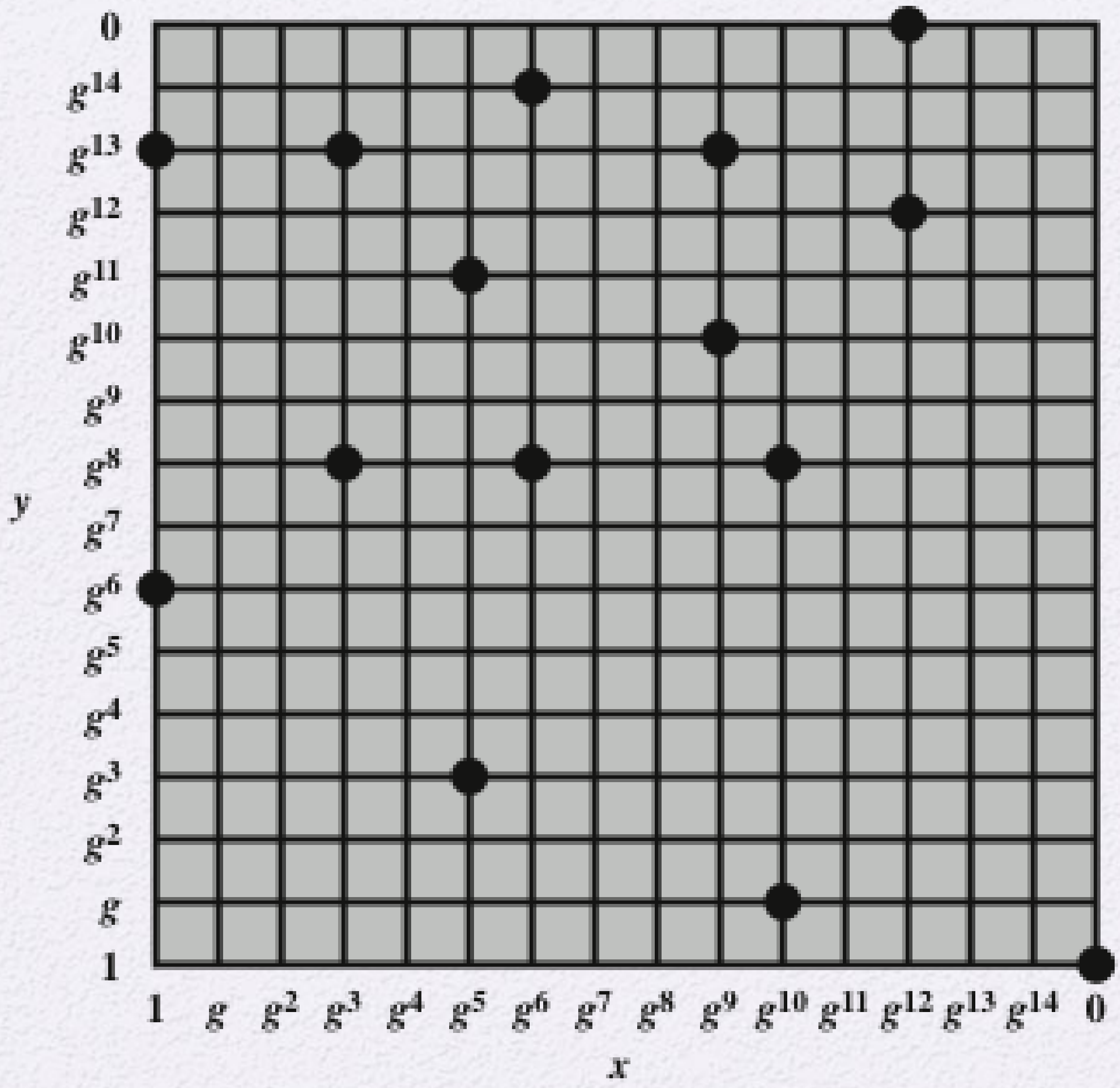


Figure 10.6 The Elliptic Curve $E_{2^4}(g^4, 1)$

Elliptic Curve Cryptography (ECC)

- Addition operation in ECC is the counterpart of modular multiplication in RSA
- Multiple addition is the counterpart of modular exponentiation

To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm

- $Q=kP$, where Q, P belong to a prime curve
- Is "easy" to compute Q given k and P
- But "hard" to find k given Q, and P
- Known as the elliptic curve logarithm problem

- Certicom example: $E_{23}(9,17)$

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

Figure 10.7 ECC Diffie-Hellman Key Exchange

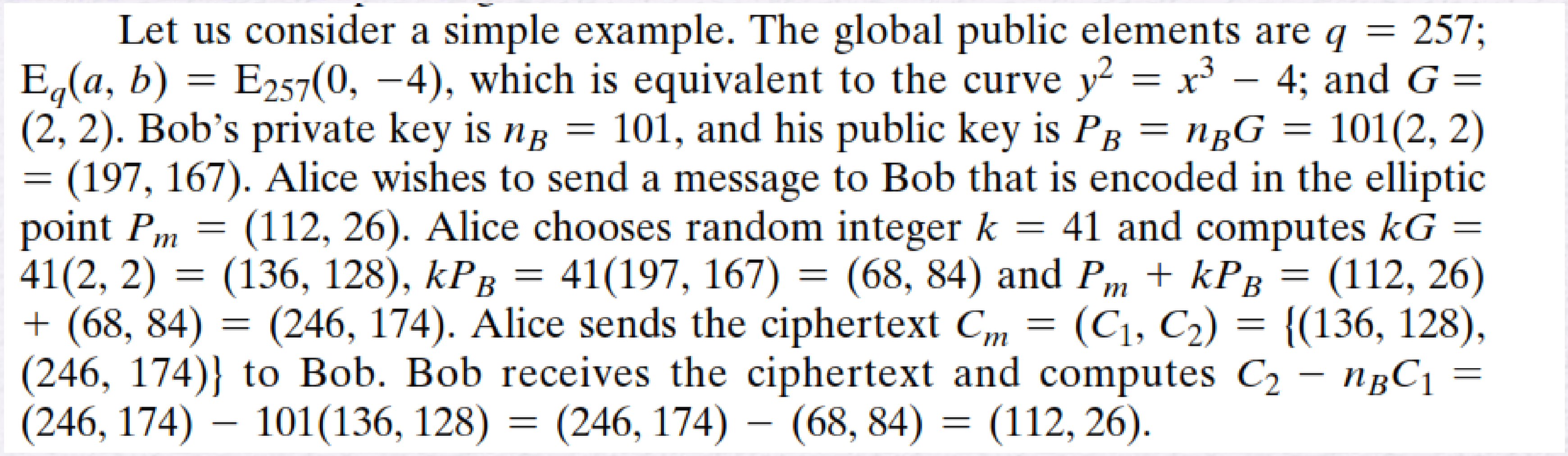
ECC Encryption/Decryption

- Several approaches using elliptic curves have been analyzed
- Must first encode any message *m* as a point on the elliptic curve P_m
- Select suitable curve and point G as in Diffie-Hellman
- Each user chooses a private key n_A and generates a public key $P_A = n_A * G$
- To encrypt and send message P_m to B, A chooses a random positive integer *k* and produces the ciphertext C_m *consisting of the pair of points:*

$$C_m = \{kG, P_m + kP_B\}$$

- To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

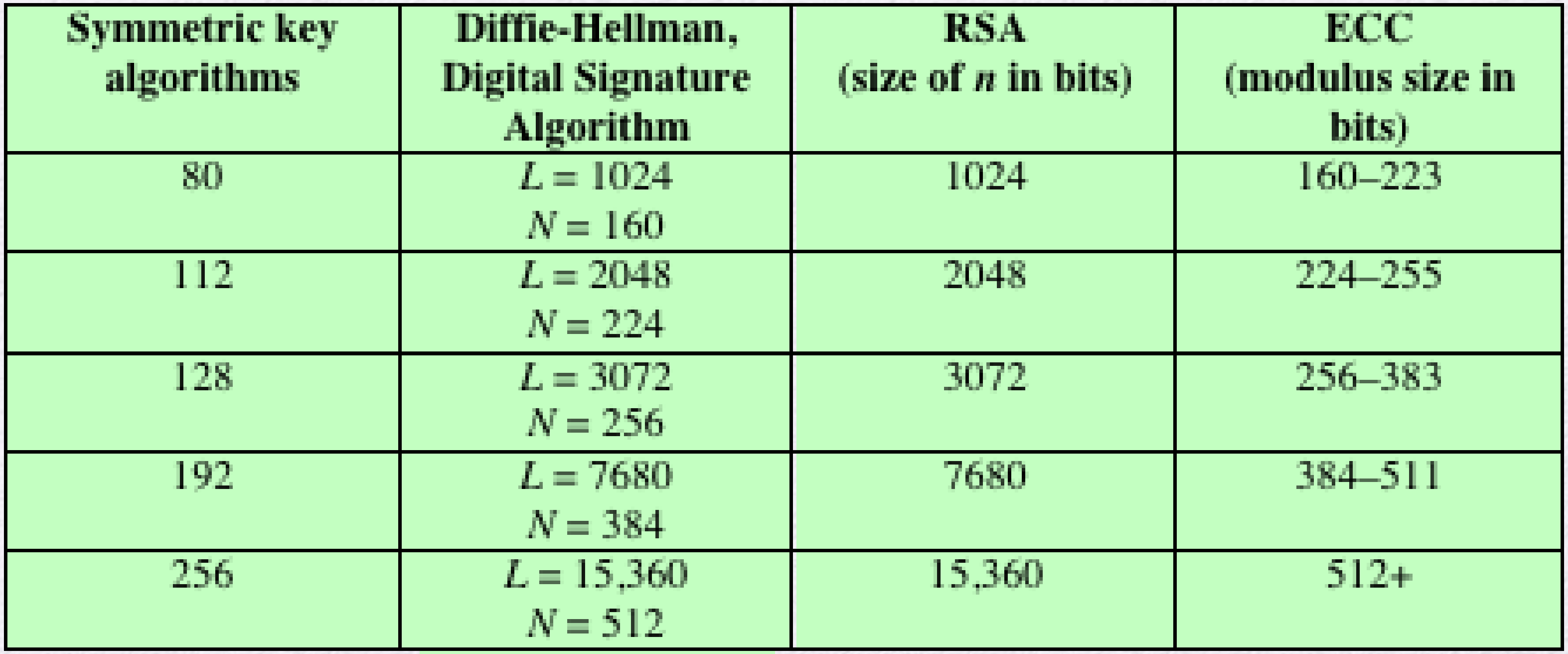


Let us consider a simple example. The global public elements are $q = 257$; $E_q(a, b) = E_{257}(0, -4)$, which is equivalent to the curve $y^2 = x^3 - 4$; and $G = (2, 2)$. Bob's private key is $n_B = 101$, and his public key is $P_B = n_B G = 101(2, 2) = (197, 167)$. Alice wishes to send a message to Bob that is encoded in the elliptic point $P_m = (112, 26)$. Alice chooses random integer $k = 41$ and computes $kG = 41(2, 2) = (136, 128)$, $kP_B = 41(197, 167) = (68, 84)$ and $P_m + kP_B = (112, 26) + (68, 84) = (246, 174)$. Alice sends the ciphertext $C_m = (C_1, C_2) = \{(136, 128), (246, 174)\}$ to Bob. Bob receives the ciphertext and computes $C_2 - n_B C_1 = (246, 174) - 101(136, 128) = (246, 174) - (68, 84) = (112, 26)$.

Security of Elliptic Curve Cryptography

- Depends on the difficulty of the elliptic curve logarithm problem
- Fastest known technique is “Pollard rho method”
- Compared to factoring, can use much smaller key sizes than with RSA
- For equivalent key lengths computations are roughly equivalent
- Hence, for similar security ECC offers significant computational advantages

Table 10.3
Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)



Symmetric key algorithms	**Diffie-Hellman, Digital Signature Algorithm**	**RSA (size of *n* in bits)**	**ECC (modulus size in bits)**
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

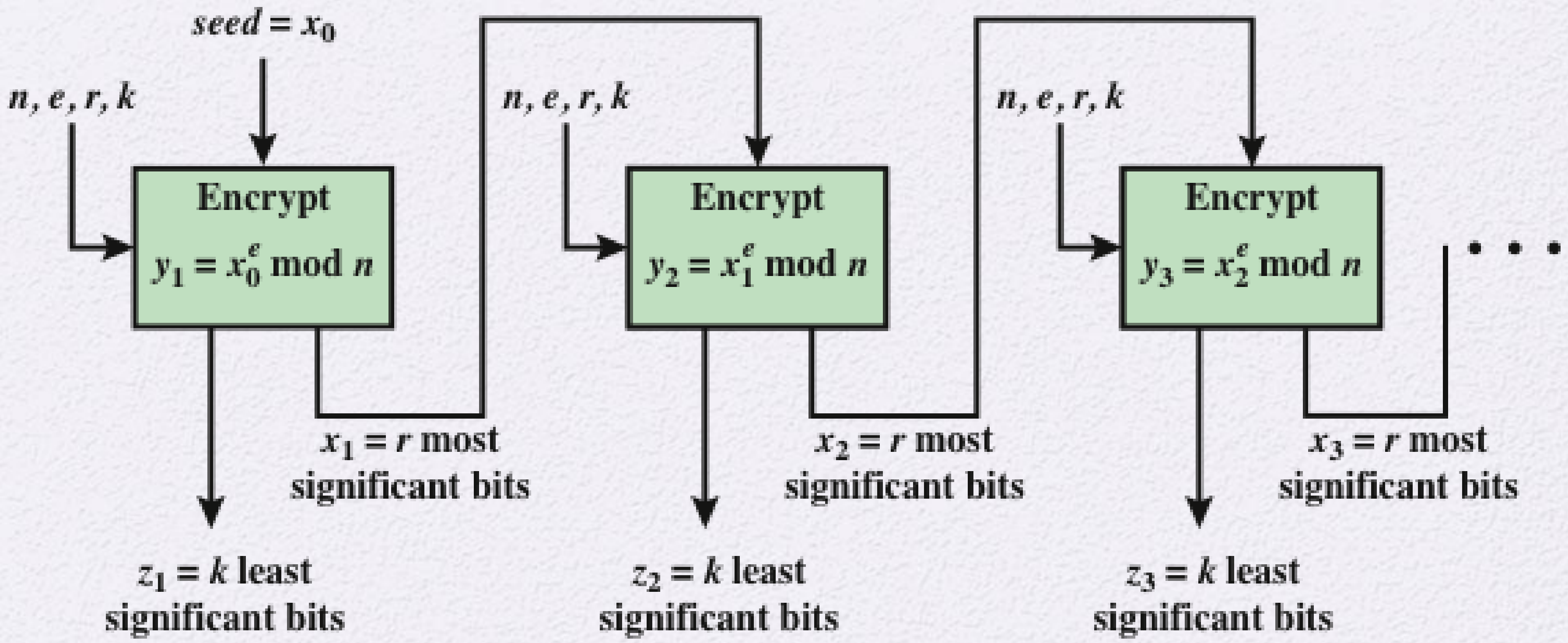
Note: *L* = size of public key, *N* = size of private key

Pseudorandom Number Generation (PRNG) Based on Asymmetric Cipher

- An asymmetric encryption algorithm produces apparently ransom output and can be used to build a PRNG

- Much slower than symmetric algorithms so they're not used to generate open-ended PRNG bit streams

- Useful for creating a pseudorandom function (PRF) for generating a short pseudorandom bit sequence



$seed = x_0$
n, e, r, k
Encrypt
$y_1 = x_0^e \bmod n$
$x_1 = r$ most significant bits
$z_1 = k$ least significant bits
n, e, r, k
Encrypt
$y_2 = x_1^e \bmod n$
$x_2 = r$ most significant bits
$z_2 = k$ least significant bits
n, e, r, k
Encrypt
$y_3 = x_2^e \bmod n$
$x_3 = r$ most significant bits
$z_3 = k$ least significant bits

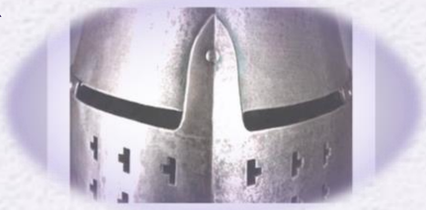
Figure 10.8 Micali-Schnorr Pseudorandom Bit Generator

PRNG Based on Elliptic Curve Cryptography

- Developed by the U.S. National Security Agency (NSA)
- Known as dual elliptic curve PRNG (DEC PRNG)
- Recommended in NIST SP 800-90, the ANSI standard X9.82, and the ISO standard 18031
- Has been some controversy regarding both the security and efficiency of this algorithm compared to other alternatives
    - The only motivation for its use would be that it is used in a system that already implements ECC but does not implement any other symmetric, asymmetric, or hash cryptographic algorithm that could be used to build a PRNG

Summary

- Diffie-Hellman Key Exchange
  - The algorithm
  - Key exchange protocols
  - Man-in-the-middle attack
- Elgamal cryptographic system
- Elliptic curve cryptography
  - Analog of Diffie-Hellman key exchange
  - Elliptic curve encryption/decryption
  - Security of elliptic curve cryptography



- Elliptic curve arithmetic
  - Abelian groups
  - Elliptic curves over real numbers
  - Elliptic curves over Z_p
  - Elliptic curves over $GF(2^m)$
- Pseudorandom number generation based on an asymmetric cipher
  - PRNG based on RSA
  - PRNG based on elliptic curve cryptography