Cryptography and Network Security

Behrouz Forouzan

Chư ơ ng 10

Khóa bất đối xứng mật mã

Chư ơ ng 10

Mục tiêu

Để phân biệt hai hệ mật mã: khóa đối xứng và khóa bất đối xứng

Giới thiệu các hàm một chiều cửa bẫy và cách sử dụng chúng trong các hệ mật mã khóa bất đối xứng

Giới thiệu hệ thống mật mã ba lô như một trong những ý tư ởng đầu tiên trong mật mã khóa bất đối xứng

Thảo luận về hệ thống mật mã RSA

Thảo luận về hệ thống mật mã Rabin

Thảo luận về ElGamal hệ thống mật mã

Thảo luận về hệ thống mật mã đư ờng cong elip

10-1 GIỚI THIỆU

Mật mã khóa đối xứng và bất đối xứng sẽ tồn tại song song và tiếp tục phục vụ cộng đồng. Chúng tôi thực sự tin rằng chúng bổ sung cho nhau; ư u điểm của cái này có thể bù đắp đư ợc như ợc điểm của cái kia.

Các chủ đề đư ợc thảo luận trong phần này:

10.1.1 Chìa khóa
10.1.2 Ý tư ởng chung
10.1.3 Cần cả hai
10.1.4 Chức năng một chiều của Trapdoor
10.1.5 Hệ thống mật mã ba lô

10-1 GIỚI THIỆU

Mật mã khóa đối xứng và bất đối xứng sẽ tồn tại song song và tiếp tục phục vụ cộng đồng. Chúng tôi thực sự tin rằng chúng bổ sung cho nhau; ư u điểm của cái này có thể bù đắp đư ợc như ợc điểm của cái kia.

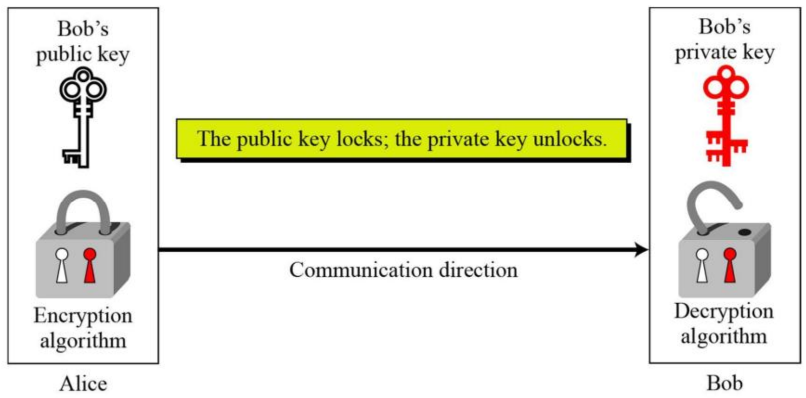
Ghi chú

Mật mã khóa đối xứng dựa trên việc chia sẻ bí mật; Mật mã khóa bất đối xứng dựa trên bí mật cá nhân.

10.1.1 Chìa khóa

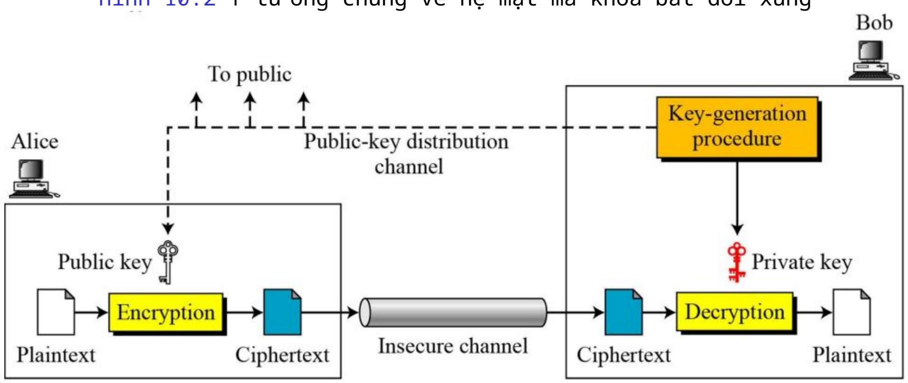
Mật mã khóa bất đối xứng sử dụng hai khóa riêng biệt: một khóa riêng và một khóa chung.

Hình 10.1 Khóa và mở khóa trong hệ mật mã khóa bất đối xứng



10.1.2 Ý tư ởng chung

Hình 10.2 Ý tư ởng chung về hệ mật mã khóa bất đối xứng



10.1.2 Tiếp theo

Bản rõ/Bản mã • Không giống như trong mật mã khóa đối xứng, bản rõ và bản mã đư ợc coi là số nguyên trong mật mã khóa bất đối xứng.

• Thông điệp phải đư ợc mã hóa dư ới dạng số nguyên (hoặc tập hợp số nguyên) trư ớc khi mã hóa; số nguyên (hoặc tập hợp số nguyên) phải đư ợc giải mã thành tin nhắn sau khi giải mã. • Mật mã khóa bất đối xứng thư ờng đư ợc sử dụng để mã hóa hoặc giải mã một phần thông tin nhỏ, chẳng hạn như khóa mật mã cho mật mã khóa đối xứng

10.1.2 Tiếp theo

Mã hóa/Giải mã

$$\boxed{C = f(K_{public}, P) \; ; \; P = g(K_{private}, C)}$$

• Mã hóa và giải mã bằng khóa bất đối xứng mật mã là các hàm toán học đư ợc áp dụng trên các số đại diện cho bản rõ và bản mã. • Chức năng mã hóa f chỉ đư ợc sử dụng để mã hóa; • Hàm giải mã g chỉ đư ợc sử dụng để giải mã

10.1.3 Cần cả hai

- Có một thực tế rất quan trọng đôi khi bị hiểu lầm: Sự ra đời của mật mã khóa bất đối xứng không loại bỏ nhu cầu về mật mã khóa đối xứng. Lý do là mật mã khóa bất đối xứng, sử dụng các hàm toán học để mã hóa và giải mã, chậm hơn nhiều so với mật mã khóa đối xứng. • Mật mã khóa đối xứng vẫn cần thiết để mã hóa các thông điệp lớn. • Mật mã khóa bất đối xứng là

vẫn cần xác thực,

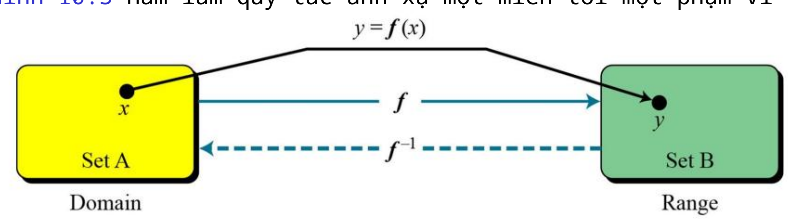
chữ ký số và trao đổi khóa bí mật.

10.1.4 Chức năng một chiều của cửa sập

Ý tưởng chính đằng sau mật mã khóa bất đối xứng là khái niệm hàm một chiều cửa bẫy.

Chức năng

Hình 10.3 Hàm làm quy tắc ánh xạ một miền tới một phạm vi



10.1.4 Tiếp theo

Hàm một chiều (OWF) là hàm thỏa mãn hai tính chất sau:

1. f dễ tính toán. Cho x, y=f(x) dễ tính 1 thì khó tính. Cho y, không khả thi 2. f tính x=f-1(x)

Hàm một chiều cửa bẫy (TOWF) là hàm một chiều với thuộc tính thứ ba:

3. Cho y và một cửa sập, x có thể tính được dễ dàng.

10.1.4 Tiếp theo

Ví dụ 10. 1

Khi n lớn thì n = p × q là hàm một chiều. Cho p và q , việc tính n luôn dễ dàng ; với n, rất khó tính được p và q. Đây là vấn đề nhân tố hóa.

Ví dụ 10. 2 k mod

và n, có thể dễ dànglàinhửađsậpcKhi Chlớn, thìvàhàm, y = x hàm một chiều. Cho x, k rất khó tính được x. Đây là bài toán logarit rời rạc. Tuy nhiên, nếu chúng ta biết cửa sập, k′ sao cho k × k k′ mod n tìm được x. y

'

= 1 mod (n), ta có thể dùng x =

10.1.5 Hệ thống mật mã ba lô

Ý tư ởng tuyệt vời đầu tiên về mật mã khóa công khai đến từ Merkle và Hellman, trong hệ thống mật mã ba lô của họ.

Mặc dù hệ thống này đư ợc cho là không an toàn với các tiêu chuẩn ngày nay

Định nghĩa, cho hai bộ k a = [a1 , a2 , ., ak] và x = [x1 , x2 , ., xk].

$$\boxed{s = knapsackSum\,(a, x) \;=\; x_1a_1 + x_2a_2 + \cdots + x_ka_k}$$

Cho a và x, dễ dàng tính đư ợc s. Tuy nhiên, với s và a rất khó tìm đư ợc x.

Bộ siêu tăng

$$\boxed{a_i \geq a_1 + a_2 + \cdots + a_{i-1}}$$

10.1.5 Tiếp theo

Algorithm 10.1 *knapsacksum and inv_knapsackSum for a superincreasing k-tuple*

```
knapsackSum (x [1 … k], a [1 … k])          inv_knapsackSum (s, a [1 … k])
{                                           {
   s ← 0                                       for (i = k down to 1)
   for (i = 1 to k)                            {
   {                                              if s ≥ a_i
      s ← s + a_i × x_i                           {
   }                                                 x_i ← 1
   return s                                          s ← s − a_i
}                                                 }
                                                  else x_i ← 0
                                               }
                                               return x [1 … k]
                                            }
```


10.1.5 Tiếp theo

Ví dụ 10. 3

Một ví dụ rất đơ n giản, giả sử rằng a = [17, 25, 46, 94, 201,400] và s = 272 đã cho. Bảng 10.1 cho thấy cách tìm thấy bộ dữ liệu x bằng cách sử dụng thủ tục inv_knapsackSum trong Thuật toán 10.1. Trong trư ờng hợp này x = [0, 1, 1, 0, 1, 0], có nghĩa là 25, 46 và 201 nằm trong ba lô.

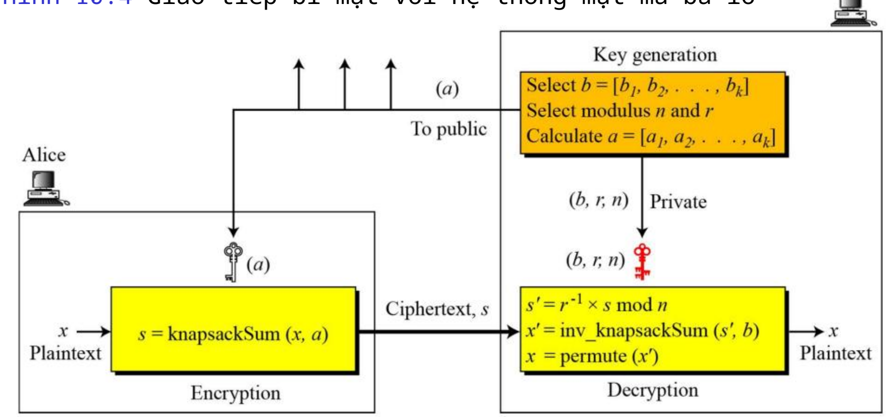
Table 10.1 *Values of i, a_i, s, and x_i in Example 10.3*

i	a_i	s	$s \geq a_i$	x_i	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

10.1.5 Tiếp theo

Giao tiếp bí mật với ba lô.

Hình 10.4 Giao tiếp bí mật với hệ thống mật mã ba lô



10.1.5 Tiếp theo

Giao tiếp bí mật với ba lô.

Tạo khóa: a. Tạo một k-

tuple siêu tăng b=[b1 ,b2 , ., bk] b. Chọn mô đun n, sao cho n>

b1+b2+ .+ bk c.Chọn số nguyên ngẫu nhiên r nguyên tố cùng nhau với

n và 1<=r<=n-1

d. Tạo một k-tupe t=[t tạm thời trong đó t t 1 , 2 , ., t k]

i=r ×bi mod n e. Chọn một hoán

vị của k đối tư ợng và tìm một bộ mới a=permute(t) f. Khóa công khai là k-

tuple a. khóa

riêng là n, r và k-tuple b

10.1.5 Tiếp theo

Giao tiếp bí mật với ba lô.

Mã hóa: a. Alice

chuyển tin nhắn của mình thành k-tuple x=(x1 , x2 , ., xk) trong đó xi là 0 hoặc

1. Tuple x là bản rõ. b. Alice sử dụng thủ tục knapsackSum để tính

s. Sau đó

cô ấy gửi giá trị của s làm bản mã.

Giải mã: Bob nhận đư ợc bản mã s: a.Bob tính s'=r -1 × s

mod nbBob sử dụng inv_knapsackSum để tạo x'.

c.Bob hoán vị x' để tìm x. Các bộ x là bản rõ đư ợc phục

hồi.

10.1.5 Tiếp theo

Ví dụ 10. 4

Đây là một ví dụ tầm thư ờng (rất không an toàn) chỉ để hiển thị quy trình.

1. Key generation:
   a. Bob creates the superincreasing tuple $b = [7, 11, 19, 39, 79, 157, 313]$.
   b. Bob chooses the modulus $n = 900$ and $r = 37$, and [4 2 5 3 1 7 6] as permutation table.
   c. Bob now calculates the tuple $t = [259, 407, 703, 543, 223, 409, 781]$.
   d. Bob calculates the tuple a = permute $(t) = [543, 407, 223, 703, 259, 781, 409]$.
   e. Bob publicly announces a; he keeps n, r, and b secret.

2. Suppose Alice wants to send a single character "g" to Bob.
   a. She uses the 7-bit ASCII representation of "g", $(1100111)_2$, and creates the tuple $x = [1, 1, 0, 0, 1, 1, 1]$. This is the plaintext.
   b. Alice calculates $s = knapsackSum\ (a, x) = 2165$. This is the ciphertext sent to Bob.

3. Bob can decrypt the ciphertext, $s = 2165$.
   a. Bob calculates $s' = s \times r^{-1} \bmod n = 2165 \times 37^{-1} \bmod 900 = 527$.
   b. Bob calculates $x' = Inv\_knapsackSum\ (s', b) = [1, 1, 0, 1, 0, 1, 1]$.
   c. Bob calculates $x = permute\ (x') = [1, 1, 0, 0, 1, 1, 1]$. He interprets the string $(1100111)_2$ as the character "g".

10-2 HỆ THỐNG MẬT MÃ RSA

Thuật toán khóa công khai phổ biến nhất là hệ thống mật mã RSA, đư ợc đặt theo tên của các nhà phát minh ra nó (Rivest, Shamir và Adleman).

Các chủ đề đư ợc thảo luận trong phần này:

10.2.1 Giới thiệu 10.2.2

Quy trình 10.2.3 Một

số ví dụ tầm thư ờng 10.2.4 Tấn công

vào RSA

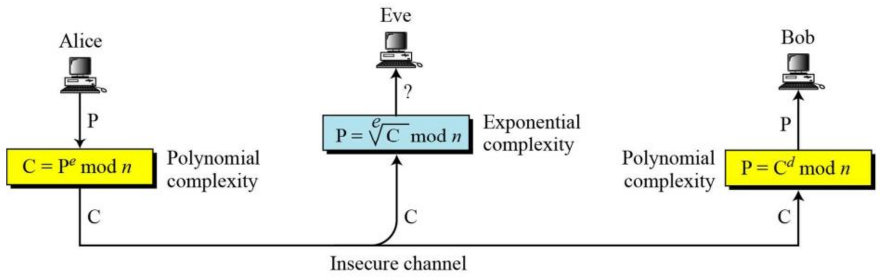
10.2.5 Khuyến nghị

10.2.6 Đệm mã hóa bất đối xứng tối ư u (OAEP)

10.2.7 Ứng dụng

10.2.1 Giới thiệu

Hình 10.5 Độ phức tạp của các thao tác trong RSA



RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.

10.2.2 Tiếp theo

Hai cấu trúc đại số

Vòng mã hóa/giải mã: R = <Zn , +, × >

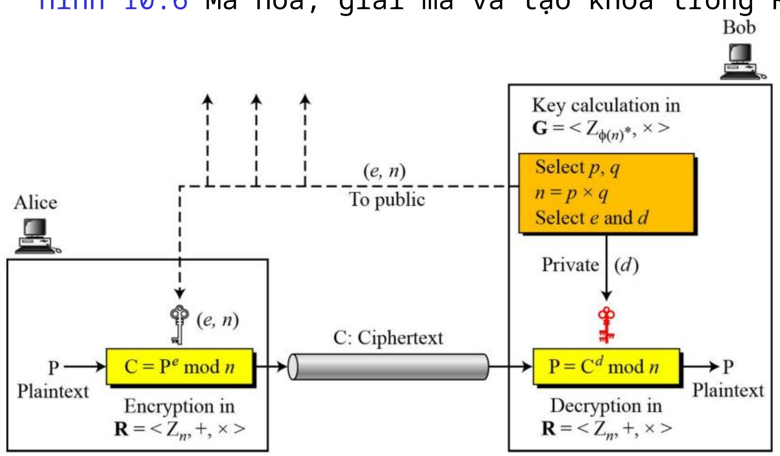
Nhóm tạo khóa: G = <Z $_{(N)}$*, × >

RSA uses two algebraic structures: a public ring $R = <Z_n, +, \times>$ and a private group $G = <Z_{\phi(n)}*, \times>$.

In RSA, the tuple (e, n) is the public key; the integer d is the private key.

10.2.2 Quy trình

Hình 10.6 Mã hóa, giải mã và tạo khóa trong RSA



10.2.2 Tiếp tục

Algorithm 10.2 *RSA Key Generation*

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p-1) \times (q-1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$

Public_key $\leftarrow$ (e, n) // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

Mã hóa

Algorithm 10.3 *RSA encryption*

```
RSA_Encryption (P, e, n)                 // P is the plaintext in Zn and P < n
{
   C ← Fast_Exponentiation (P, e, n)     // Calculation of (P^e mod n)
   return C
}
```

In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

giải mã

Algorithm 10.4 *RSA decryption*

```
RSA_Decryption (C, d, n)                 //C is the ciphertext in Zn
{
   P ← Fast_Exponentiation (C, d, n)     // Calculation of (C^d mod n)
   return P
}
```


Bằng chứng về RSA

Chúng ta có thể chứng minh rằng sự mã hóa là nghịch đảo của nhau bằng cách sử dụng phiên bản thứ hai của định lý Euler

If $n = p \times q$, $a < n$, and k is an integer, then $a^{k\times\phi(n)+1} \equiv a \pmod{n}$.

Giả sử rằng bản rõ mà Bob lấy được là P1 và chứng minh rằng nó bằng P

$P_1 = C^d \bmod n = (P^e \bmod n)^d \bmod n = P^{ed} \bmod n$

$ed = k\phi(n) + 1$ // d and e are inverses modulo $\phi(n)$

$P_1 = P^{ed} \bmod n \rightarrow P_1 = P^{k\phi(n)+1} \bmod n$

$P_1 = P^{k\phi(n)+1} \bmod n = \quad P \bmod n$ // Euler's theorem (second version)

10.5

Bob chọn 7 và 11 là p và q và tính n = 77. Giá trị của (n) = (7 1)(11 1) = 60. Bây giờ Bob chọn hai số mũ e và d từ Z60 . Nếu anh ta chọn e là 13 thì d là 37. Lư u ý rằng e × d mod 60 = 1 (chúng nghịch đảo của mỗi số. Bây giờ hãy tư ởng tư ợng Alice muốn gửi bản rõ 5 cho Bob.

Cô ấy sử dụng số mũ công khai 13 để mã hóa 5.

Plaintext: 5	$C = 5^{13} = 26 \bmod 77$	Ciphertext: 26

Bob nhận đư ợc bản mã 26 và sử dụng khóa riêng 37 để giải mã bản mã:

Ciphertext: 26	$P = 26^{37} = 5 \bmod 77$	Plaintext: 5

10.2.3 Một số ví dụ tầm thư ờng

10.6

Bây giờ giả sử rằng một ngư ời khác, John, muốn gửi tin nhắn cho Bob. John có thể sử dụng cùng một khóa chung do Bob công bố (có thể là trên trang web của anh ấy), 13; Bản rõ của John là 63. John tính như sau:

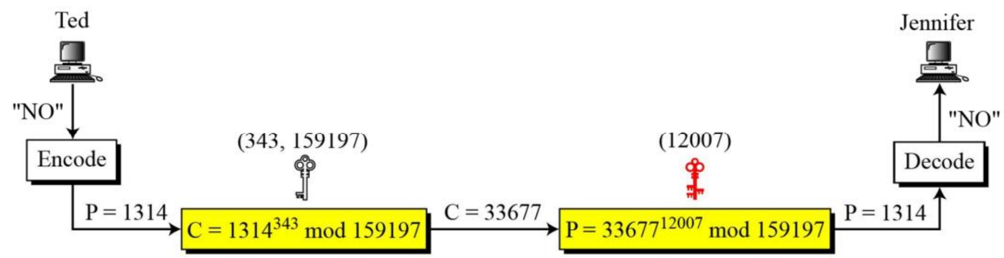
Plaintext: 63 $C = 63^{13} = 28 \bmod 77$ Ciphertext: 28

Bob nhận đư ợc bản mã 28 và sử dụng khóa riêng 37 của mình để giải mã bản mã:

Ciphertext: 28 $P = 28^{37} = 63 \bmod 77$ Plaintext: 63

10.2.3 Tiếp theo

Hình 10.7 Mã hóa và giải mã trong Ví dụ 10.7



10.2.3 Một số ví dụ tầm thư ờng

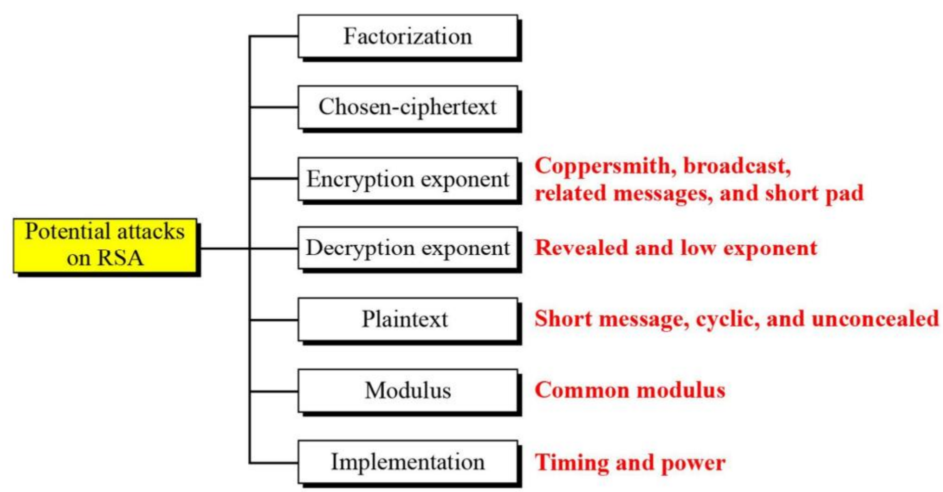
10.7

Jennifer tạo ra một cặp chìa khóa cho chính mình. Cô ấy chọn p = 397 và q = 401. Cô ấy tính n = 159197. Sau đó cô ấy tính (n) = 158400. Sau đó cô ấy chọn e = 343 và d = 12007. Hãy chỉ ra cách Ted có thể gửi tin nhắn cho Jennifer nếu anh ấy biết e và N.

Giả sử Ted muốn gửi tin nhắn “KHÔNG” cho Jennifer. Anh ta thay đổi mỗi ký tự thành một số (từ 00 đến 25), với mỗi ký tự đư ợc mã hóa thành hai chữ số. Sau đó, anh ta ghép hai ký tự đư ợc mã hóa và nhận đư ợc một số có bốn chữ số. Bản rõ là 1314. Hình 10.7 thể hiện quá trình này.

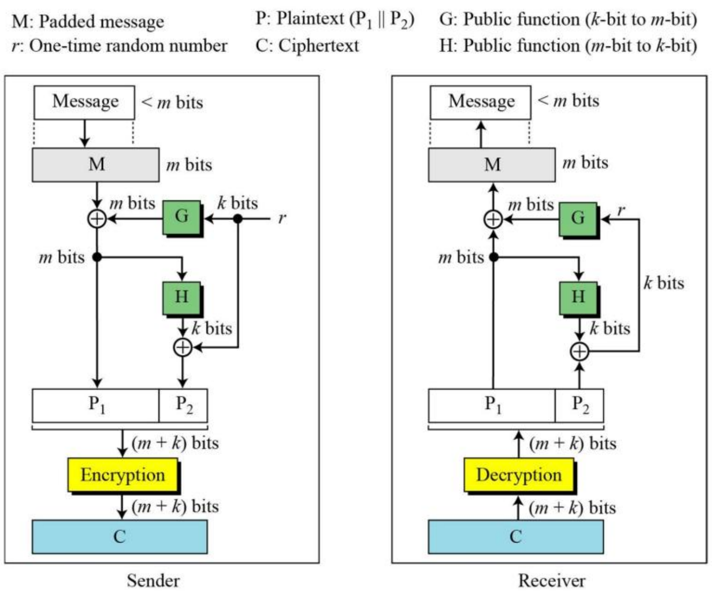
10.2.4 Tấn công vào RSA

Hình 10.8 Phân loại các cuộc tấn công tiềm ẩn vào RSA



10.2.6 OAEP

Hình 10.9 Vùng đệm mã hóa bất đối xứng tối ư u (OAEP)



10.2.6 Tiếp theo

Ví dụ 10. 8

Đây là một ví dụ thực tế hơ n. Chúng tôi chọn p và q 512 bit , tính n và (n), sau đó chọn e và kiểm tra độ nguyên tố tư ơ ng đối với (n). Sau đó chúng tôi tính toán d.

Cuối cùng, chúng tôi hiển thị kết quả mã hóa và giải mã. Số nguyên p là số có 159 chữ số.

p =	96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986660649242019180878066742109606335421992661209

q =	12060191957231446918276794204450896001555925054637033936061798321731482148483764659215389453209175225273226830107120695604602513887145524969000359660045617

10.2.6 Tiếp theo

Ví dụ 10. 8

Tiếp theo

Mô đun n = p × q. Nó có 309 chữ số.

n =	115935041739676149688925098646158875237714573754541447754855261376147885408326350817276878815968325168468849300625485764111250162414552339182927162507656772727460097082714127730434960500556347274566628060099924037102991424472292215772798531727033839381334692684137327622000966676671831831088373420823444370953

(n) = (p 1)(q 1) có 309 chữ số.

$\phi(n)$ =	115935041739676149688925098646158875237714573754541447754855261376147885408326350817276878815968325168468849300625485764111250162414552339182927162507656751054233608492916752034482627988117554787657013923444405716989581728196098226361075467211864612171359107358640614008885170265377277264467341066243857664128

10.2.6 Tiếp theo

Ví dụ 10. 8

Tiếp theo

Bob chọn e = 35535 (lý tư ởng là 65537) và kiểm tra nó để đảm bảo nó nguyên tố cùng nhau với (n). Sau đó anh ta tìm nghịch đảo của e modulo (n) và gọi nó là d.

e =	35535
d =	580083028600377639360936612896779175946690620896509621804228661113805938528223587317062869100300217108590443384021707298690876006115306202524959884448047568240966247081485817130463240644077704833134010850947385295645071936774061197326557424237217617674620776371642076003370853332885321447088595513667029483 1

10.2.6 Tiếp theo

Ví dụ 10. 8 Tiếp theo

Alice muốn gửi tin nhắn "ĐÂY LÀ KIỂM TRA", tin nhắn này có thể được thay đổi thành giá trị số bằng cách sử dụng sơ đồ mã hóa 00 26 (26 là ký tự khoảng trắng).

P =	1907081826081826002619041819

Bản mã do Alice tính được là C = Pe ,đó là

C =	4753091236462268272063655506105451809423717960704917165232392430544529606131993285666178434183591141511974112520056829797945717360361012782188478927415660904800235071907152771859149751884658886321011483541033616578984679683867637337657774656250792805211481418440481418443081277305900469287424855916646210865 6

10.2.6 Tiếp theo

Ví dụ 10. 8 Tiếp theo

Bob có thể khôi phục bản rõ từ bản mã bằng cách sử dụng P = Cd,đó là

P =	1907081826081826002619041819

Bản rõ được khôi phục là "ĐÂY LÀ KIỂM TRA" sau khi giải mã.

10-3 HỆ THỐNG MẬT MÃ RABIN

Hệ thống mật mã Rabin có thể được coi là hệ thống mật mã RSA trong đó giá trị của e và d là cố định.

Mã hóa là C ≡ P2 (mod n) và giải mã là P ≡ C1/2 (mod n).

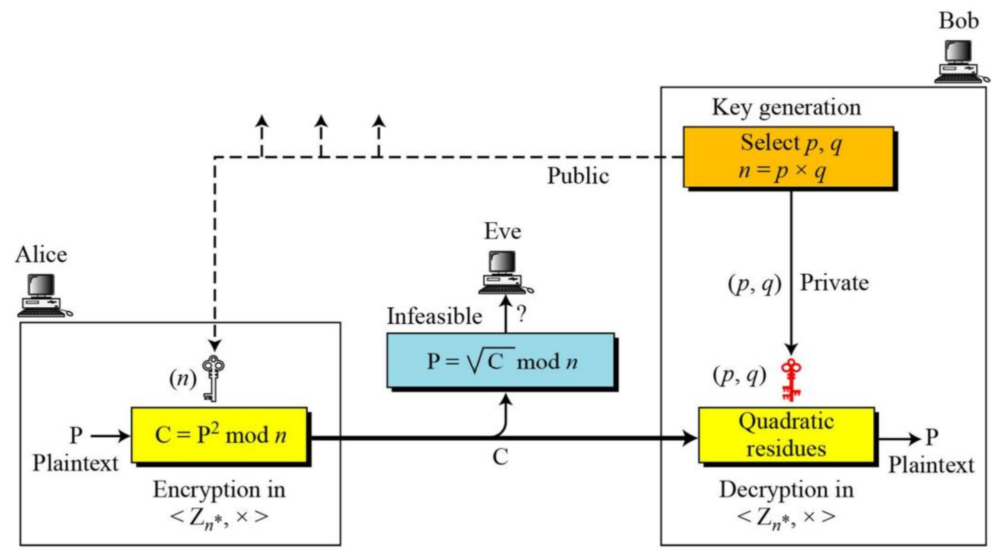
Các chủ đề được thảo luận trong phần này:

10.3.1 Quy trình

10.3.2 Bảo mật của Hệ thống Rabin

10-3 Tiếp theo

Hình 10.10 Hệ thống mật mã Rabin



10.3.1 Quy trình

Tạo khóa

Algorithm 10.6 *Key generation for Rabin cryptosystem*

```
Rabin_Key_Generation
{
    Choose two large primes p and q in the form 4k + 3 and p ≠ q.
    n ← p × q
    Public_key ← n                  // To be announced publicly
    Private_key ← (q, n)            // To be kept secret
    return Public_key and Private_key
}
```


10.3.1 Tiếp theo

Mã hóa

Algorithm 10.7 *Encryption in Rabin cryptosystem*

```
Rabin_Encryption (n, P)          // n is the public key; P is the ciphertext from Zn*
{
    C ← P^2 mod n                // C is the ciphertext
    return C
}
```


10.3.1 Tiếp theo

giải mã

Algorithm 10.8 *Decryption in Rabin cryptosystem*

```
Rabin_Decryption (p, q, C)          // C is the ciphertext; p and q are private keys
{
    a1 ← +(C^((p+1)/4)) mod p
    a2 ← −(C^((p+1)/4)) mod p
    b1 ← +(C^((q+1)/4)) mod q
    b2 ← −(C^((q+1)/4)) mod q
    // The algorithm for the Chinese remainder algorithm is called four times.
    P1 ← Chinese_Remainder (a1, b1, p, q)
    P2 ← Chinese_Remainder (a1, b2, p, q)
    P3 ← Chinese_Remainder (a2, b1, p, q)
    P4 ← Chinese_Remainder (a2, b2, p, q)
    return P1, P2, P3, and P4
}
```

Ghi chú

Hệ thống mật mã Rabin không mang tính quyết định: Việc giải mã tạo ra bốn bản rõ.

10.3.1 Tiếp theo

Ví dụ 10. 9

Đây là một ví dụ rất tầm thường để thể hiện ý tưởng.

1. Bob chọn p = 23 và q = 7. Lưu ý rằng cả hai đều tương ứng với 3 mod 4.

2. Bob tính n = p × q = 161.

3. Bob thông báo n một cách công khai; anh ấy giữ p và q ở chế độ riêng tư .

4. Alice muốn gửi bản rõ P = 24. Lưu ý rằng 161 và 24 tương đối nguyên tố; 24 nằm trong Z161*. Cô tính C = 242 = 93 mod 161 và gửi bản mã 93 cho Bob.

10.3.1 Tiếp theo

Ví dụ 10. 9

5. Bob nhận 93 và tính bốn giá trị:

a1 = +(93 (23+1)/4) mod 23 = 1 mod 23

a2 = (93 (23+1)/4) mod 23 = 22 mod 23

b1 = +(93 (7+1)/4) mod 7 = 4 mod

7 b2 = (93 (7+1)/4) mod 7 = 3 mod 7

6. Bob lấy bốn câu trả lời có thể có, (a1 , b1), (a1 , b2), (a2 , b1), và (a2 , b2), và sử dụng định lý số dư Trung Quốc để tìm bốn bản rõ có thể có: 116, 24, 137 và 45. Lư u ý rằng chỉ có câu trả lời thứ hai là bản rõ của Alice.

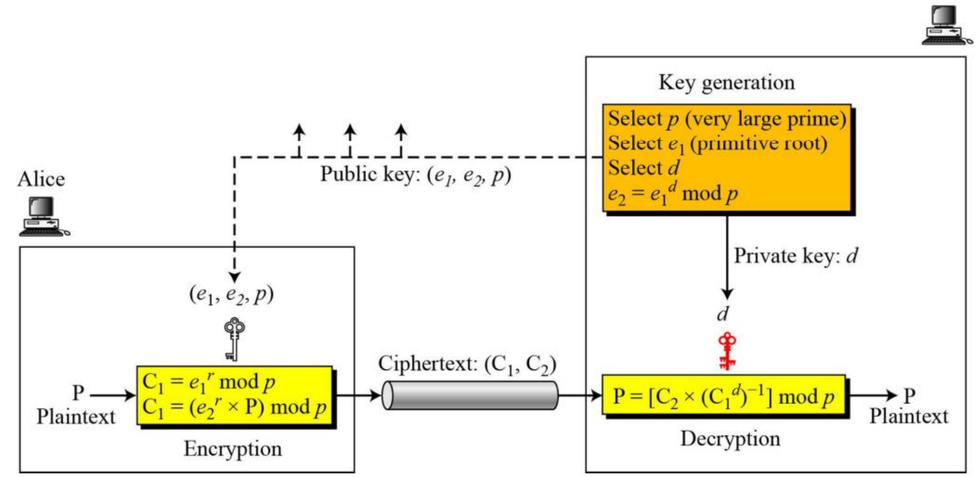
10-4 HỆ THỐNG MẬT MÃ ELGAMAL

Ngoài RSA và Rabin, một hệ thống mật mã khóa công khai khác là ElGamal. ElGamal dựa trên bài toán logarit rời rạc đã thảo luận ở Chư ơ ng 9.

Các chủ đề đư ợc thảo luận trong phần này:

10.4.2 Quy trình

Hình 10.11 Tạo khóa, mã hóa và giải mã trong ElGamal



10.4.2 Tiếp tục

Tạo khóa

Algorithm 10.9 *ElGamal key generation*

ElGamal_Key_Generation
{
Select a large prime p
Select d to be a member of the group $\mathbf{G} = <\mathbf{Z}_p^*, \times>$ such that $1 \le d \le p-2$
Select e_1 to be a primitive root in the group $\mathbf{G} = <\mathbf{Z}_p^*, \times>$
$e_2 \leftarrow e_1^{d} \bmod p$
Public_key $\leftarrow (e_1, e_2, p)$ // To be announced publicly
Private_key $\leftarrow d$ // To be kept secret
return Public_key and Private_key
}

10.4.2 Tiếp tục

Algorithm 10.10 *ElGamal encryption*

```
ElGamal_Encryption (e1, e2, p, P)                    // P is the plaintext
{
    Select a random integer r in the group G = <Zp*, ×>
    C1 ← e1^r mod p
    C2 ← (P × e2^r) mod p                            // C1 and C2 are the ciphertexts
    return C1 and C2
}
```


10.4.2 Tiếp tục

Algorithm 10.11 *ElGamal decryption*

```
ElGamal_Decryption (d, p, C1, C2)                    // C1 and C2 are the ciphertexts
{
    P ← [C2 (C1^d)^-1] mod p                         // P is the plaintext
    return P
}
```

Ghi chú

Độ phức tạp hoạt động bit của mã hóa hoặc giải mã trong hệ thống mật mã ElGamal là đa thức.

10.4.3 Tiếp theo

Ví dụ 10. 10

Đây là một ví dụ tầm thường. Bob chọn p = 11 và e1 = 2. và d = 3 e2 = e1 ^{d} = 8. Vậy khóa chung là (2, 8, 11) và khóa riêng là 3. Alice chọn r = 4 và tính C1 và C2 cho bản rõ 7.

Plaintext: 7

$C_1 = e_1^{\,r} \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (\mathrm{P} \times e_2^{\,r}) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

Bob nhận được bản mã (5 và 6) và tính toán bản rõ.

$[C_2 \times (C_1^{\,d})^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$

Plaintext: 7

10.4.3 Tiếp theo

Ví dụ 10. 11

Thay vì sử dụng P = [C2 × (C1 ^{d}) 1] mod p để giải mã, chúng ta có thể tránh tính toán nghịch đảo của phép nhân và sử dụng P = [C2 × C1 p 1 d] mod p (xem Định lý nhỏ Fermat trong Chương 9). Trong Ví dụ 10.10, chúng ta có thể tính P = [6 × 5 11 1 3] mod 11 = 7 mod 11.

Ghi chú

Đối với hệ thống mật mã ElGamal, p phải có ít nhất 300 chữ số và r phải là mới đối với mỗi lần mã hóa.

10.4.3 Tiếp tục

Ví dụ 10. 12

Bob sử dụng số nguyên ngẫu nhiên 512 bit. Số nguyên p là một số có 155 chữ số (lý tư ởng là 300 chữ số). Sau đó Bob chọn e1 , d và tính e2 , như hiển thị bên dư ới:

$p =$	115348992725616762449253137170143317404900945326098349598143469219 056898698622645932129754737871895144368891765264730936159299937280 61165964347353440008577
$e_1 =$	2

$d =$	1007
$e_2 =$	978864130430091895087668569380977390438800628873376876100220622332 554507074156189212318317704610141673360150884132940857248537703158 2066010072558707455

10.4.3 Tiếp tục

Ví dụ 10. 10

Alice có bản rõ P=3200 để gửi cho Bob. Cô ấy chọn r = 545131, tính C1 và C2 rồi gửi chúng cho Bob.

P =	3200
$r =$	545131
$\mathbf{C_1} =$	887297069383528471022570471492275663120260067256562125018188351429 417223599712681114105363661705173051581533189165400973736355080295 736788569060619152881
$\mathbf{C_2} =$	708454333048929944577016012380794999567436021836192446961774506921 244696155165800779455593080345889614402408599525919579209721628879 6813505827795664302950

Bob tính bản rõ P = C2 × ((C1) d 1 mod p = 3200 mod p.)

P =	3200

10-5 HỆ THỐNG MẬT MÃ ĐƯ ỜNG ELliptic

Mặc dù RSA và ElGamal là các hệ thống mật mã khóa bất đối xứng an toàn, như ng tính bảo mật của chúng đi kèm với cái giá phải trả là khóa lớn. Các nhà nghiên cứu đã tìm kiếm các lựa chọn thay thế có cùng mức độ bảo mật với kích thư ớc khóa nhỏ hơ n. Một trong những lựa chọn thay thế đầy hứa hẹn này là hệ thống mật mã đư ờng cong elip (ECC).

Các chủ đề đư ợc thảo luận trong phần này:
10.5.1 Đư ờng cong Elliptic trên số thực
10.5.2 Đư ờng cong Elliptic trên GF(p)
10.5.3 Đư ờng cong Elliptic trên GF(2n)
10.5.4 Mật mã đư ờng cong elip mô phỏng ElGamal

10.5.1 Đư ờng cong Elliptic trên số thực

Phư ơ ng trình tổng quát của đư ờng cong elip là

$$y^2 + b_1xy + b_2y = x^3 + a_1x^2 + a_2x + a_3$$

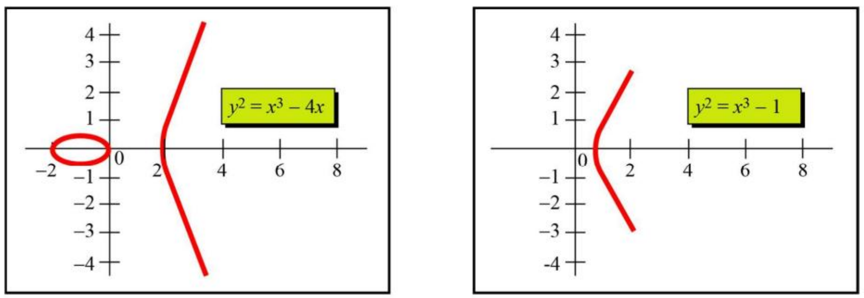
Đư ờng cong elip trên số thực sử dụng một lớp đư ờng cong elip đặc biệt có dạng

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

Ví dụ 10. 13

Hình 10.12 cho thấy hai đư ờng cong elip có phư ơ ng trình y2 = x3 4x và y2 = x3 1. Cả hai đều không suy biến. Tuy nhiên, nghiệm thứ nhất có ba nghiệm thực (x = 2, x = 0, và x = 2), như ng nghiệm thứ hai chỉ có một nghiệm thực (x = 1) và hai nghiệm ảo.

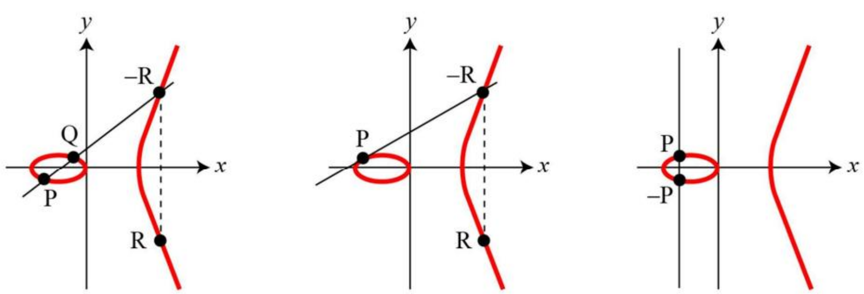
Hình 10.12 Hai đư ờng cong elip trên một trư ờng thực



a. Three real roots b. One real and two imaginary roots

10.5.1 Tiếp theo

Hình 10.13 Ba trư ờng hợp cộng trong đư ờng cong elip



a. (R = P + Q) b. (R = P + P) c. (O = P + (– P))

10.5.1 Tiếp theo

1.
$$\lambda = (y_2 - y_1) / (x_2 - x_1)$$
$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda\,(x_1 - x_3) - y_1$$

2.
$$\lambda = (3x_1{}^2 + a)/(2y_1)$$
$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda\,(x_1 - x_3) - y_1$$

3. Điểm chặn ở vô cực; điểm O là điểm ở vô cực hoặc điểm 0, là đẳng thức cộng của nhóm.

10.5.2 Đư ờng cong Elliptic trên GF(p)

Tìm một nghịch đảo

Nghịch đảo của một điểm (x, y) là (x, y), trong đó y là nghịch đảo cộng của y. Ví dụ: nếu p = 13 thì nghịch đảo của (4, 2) là (4, 11).

Tìm điểm trên đư ờng cong

Thuật toán 10.12 trình bày mã giả để tìm các điểm trên đư ờng cong Ep(a, b).

10.5.2 Tiếp theo

Algorithm 10.12 *Pseudocode for finding points on an elliptic curve*

```
ellipticCurve_points (p, a, b)                         // p is the modulus
{
    x ← 0
    while (x < p)
    {
        w ← (x^3 + ax + b) mod p                       // w is y^2
        if (w is a perfect square in Z_p) output (x, √w) (x, −√w)
        x ← x + 1
    {
}
```


Ví dụ 10. 14

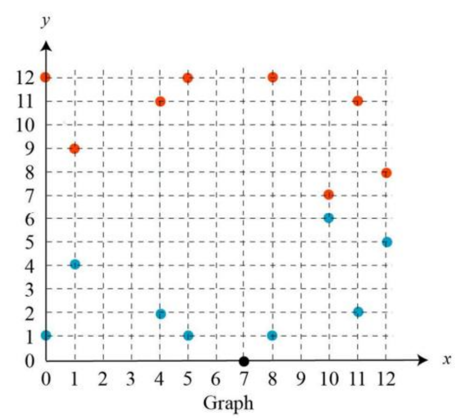
Phư ơ ng trình là y2 = x3 + x + 1 và việc tính toán đư ợc thực hiện theo modulo 13.

Hình 10.14 Các điểm trên đư ờng cong elip trên GF(p)

P = –P ⟶ (7, 0)

P	–P
(0, 1)	(0, 12)
(1, 4)	(1, 9)
(4, 2)	(4, 11)
(5, 1)	(5, 12)
(7, 0)	(7, 0)
(8, 1)	(8, 12)
(10, 6)	(10, 7)
(11, 2)	(11, 11)
(12, 5)	(12, 18)

Points



Graph

10.5.2 Tiếp theo

Ví dụ 10. 15 Ta

cộng hai điểm trong Ví dụ 10.14, R = P + Q, trong đó P = (4, 2) và Q = (10, 6). Một. λ = (6 2) × (10 4) 1 mod 13 = 4 × 6 1 mod 13 = 5 mod 13. b. x = (52 4 10) mod 13 = 11 mod 13. c. y = [5 (4 11) 2] mod 13 = 2 mod 13. d. R = (11, 2), là một điểm trên đư ờng cong trong Ví dụ 10.14.

10.5.3 Đư ờng cong Elliptic trên GF(2n)

Để xác định đư ờng cong elip trên GF(2n), ngư ời ta cần thay đổi phư ơ ng trình bậc ba. Phư ơ ng trình chung là

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

Tìm nghịch đảo Nếu P = (x, y), thì P = (x, x + y).

Tìm điểm trên đư ờng cong Chúng ta có thể viết một thuật toán để tìm các điểm trên đư ờng cong bằng cách sử dụng các bộ tạo đa thức đã thảo luận trong Chư ơ ng 7..

10.5.3 Tiếp theo

Tìm nghịch đảo Nếu P

= (x, y), thì P = (x, x + y).

Tìm điểm trên đường cong Chúng ta

có thể viết một thuật toán để tìm các điểm trên đường cong bằng cách

sử dụng các bộ tạo đa thức đã thảo luận trong Chương 7.

Thuật toán này được để lại như một bài tập. Sau đây là một ví dụ

rất tầm thường.

10.5.3 Tiếp theo

Ví dụ 10. 16

Chúng ta chọn GF(23) với các phần tử {0, 1, g, g2 , g3 , g4 , g5 , g6 } bằng cách sử dụng đa thức tối giản của f(x) = x3 + x + 1, nghĩa là 3 + g + 1 = 0 hoặc g3 = g + 1. Các lũy thừa khác của g có thể được tính g tương ứng. Sau đây cho thấy các giá trị của g.

0	000	$g^3 = g + 1$	011
1	001	$g^4 = g^2 + g$	110
g	010	$g^5 = g^2 + g + 1$	111
g^2	100	$g^6 = g^2 + 1$	101

10.5.3 Tiếp theo

Ví dụ 10. 16 Tiếp theo

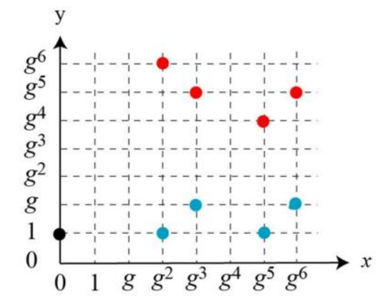
Sử dụng đường cong elip y2 + xy = x3 + g3x 2 + 1, với a = g3 và b = 1, chúng ta có thể tìm được các điểm trên đường cong này, như trong Hình 10.15.

Hình 10.15 Các điểm trên đường cong elip trên GF(2n)

P = –P →

P	–P
$(0, 1)$	$(0, 1)$
$(g^2, 1)$	(g^2, g^6)
(g^3, g^2)	(g^3, g^5)
$(g^5, 1)$	(g^5, g^4)
(g^6, g)	(g^6, g^5)

Points



Graph

10.5.3 Tiếp theo

Cộng hai điểm 1. Nếu P =

(x1 , y1), Q = (x2 , y2), Q P, và Q P, thì R = (x3 , y3)

= P + Q có thể được tìm thấy dưới dạng

$$\lambda = (y_2 + y_1) / (x_2 + x_1)$$
$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \qquad y_3 = \lambda\,(x_1 + x_3) + x_3 + y_1$$

Nếu Q = P thì R = P + P (hoặc R = 2P) có thể tìm được dưới dạng

$$\lambda = x_1 + y_1 / x_1$$
$$x_3 = \lambda^2 + \lambda + a \qquad y_3 = x_1^2 + (\lambda + 1)\,x_3$$

10.5.3 Tiếp theo

Ví dụ 10. 17

Chúng ta hãy tìm R = P + Q, trong đó P = (0, 1) và Q = (g^2 , 1).

Ta có λ = 0 và R = (g^5 , g^4).

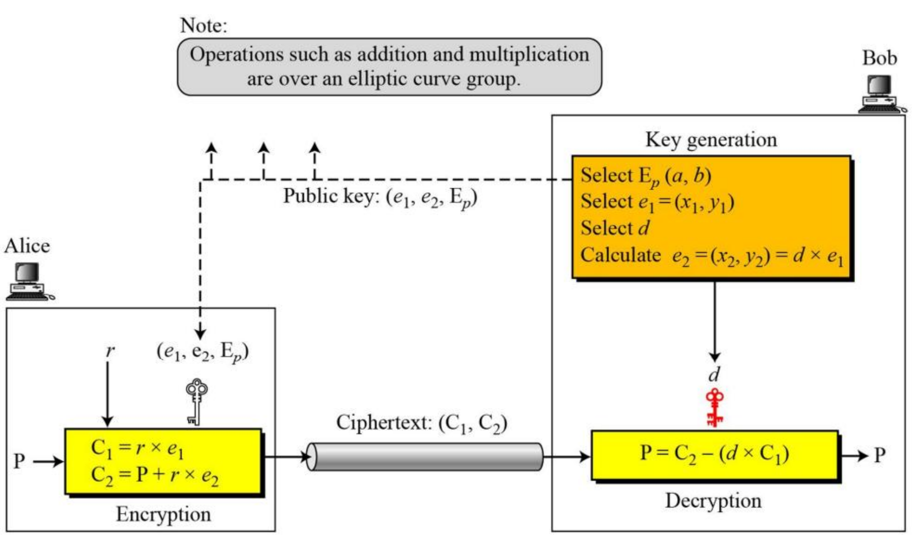
Ví dụ 10. 18 Hãy

tìm R = 2P, trong đó P = (g^2 , 1). Ta có λ = g^2 + 1/g^2 = g^2 + g^5 = g + 1

và R = (g^6 , g^5).

10.5.4 ECC Mô phỏng ElGamal

Hình 10.16 Hệ mật mã ElGamal sử dụng đường cong elip



10.5.4 Tiếp theo

Tạo khóa công khai và khóa riêng

E(a, b) e_1 (x_1 , y_1) d e_2 (x_2 , y_2) = d × e_1 (x_1 , y_1)

Mã hóa

$$C_1 = r \times e_1 \qquad C_2 = P + r \times e_2$$

giải mã

$$P = C_2 - (d \times C_1)$$ The minus sign here means adding with the inverse.

Ghi chú

Tính bảo mật của ECC phụ thuộc vào độ khó của việc giải bài toán logarit đường cong elip.

10.5.4 Tiếp theo

Ví dụ 10. 19

Đây là một ví dụ rất đơn giản về mã hóa sử dụng đường cong elip trên GF(p).

1. Bob chọn E67(2, 3) làm đường cong elip trên GF(p).
2. Bob chọn e_1 = (2, 22) và d = 4.
3. Bob tính e_2 = (13, 45), trong đó e_2 = d × e_1 .
4. Bob công bố bộ dữ liệu (E, e_1 , e_2).
5. Alice muốn gửi bản rõ P = (24, 26) cho Bob. Cô ấy chọn r = 2.