11464 : INFORMATION SYSTEMS SECURITY

Chapter 6: Public-Key Cryptography

Public Key Cryptography

By

Mustafa Al-Fayoumi

Outline

- Overview of Public-key cryptosystems
- The RSA algorithm
  - Description of the algorithm
  - Computational aspects
  - Exponentiation Algorithms
- Diffie-Hellman Key Exchange
  - Primitive Roots

Basic Protocol for Public-Key Encryption

Alice		Bob
	$\xleftarrow{K_{pubB}}$	$(K_{pubB}, K_{prB}) = K$
x		
$y = e_{K_{pubB}}(x)$	$\xrightarrow{y}$	
		$x = d_{K_{prB}}(y)$

Security Mechanisms of Public-Key Cryptography

- Here are main mechanisms that can be realized with asymmetric cryptography:
  - **Key Distribution** (e.g., Diffie-Hellman key exchange, RSA) without a preshared secret (key)
  - **Nonrepudiation and Digital Signatures** (e.g., RSA, DSA or ECDSA) to provide message integrity
  - **Identification**, using challenge-response protocols with digital signatures
  - **Encryption** (e.g., RSA / Elgamal)
- Disadvantage: Computationally very intensive (1000 times slower than symmetric Algorithms!)

Basic Key Transport Protocol

- In practice: **Hybrid systems**, incorporating asymmetric and symmetric algorithms

  1. Key exchange (for symmetric schemes) and **digital signatures** are performed with (slow) **asymmetric** algorithms

  2. Encryption of data is done using (fast) symmetric ciphers, e.g., **block ciphers or stream ciphers**

How to build Public-Key Algorithms

- Asymmetric schemes are based on a **„one-way function“ *f()***:
  - Computing $y = f(x)$ is computationally easy
  - Computing $x = f\text{-}1(y)$ is computationally infeasible
- One way functions are based on **mathematically hard problems.**
  - Three main families:
    - **Factoring integers** (RSA, ...):
      - Given a composite integer *n*, find its prime factors (Multiply two primes: easy)
    - **Discrete Logarithm** (Diffie-Hellman, Elgamal, DSA, …):
      - Given *a*, *y* and *m*, find *x* such that $ax = y$ mod *m* (Exponentiation *ax* : easy)
    - **Elliptic Curves (EC)** (ECDH, ECDSA):
      - Generalization of discrete logarithm
- Note: The problems are considered mathematically hard, but no proof exists (so far).

Key Lengths and Security Levels

Symmetric	*ECC*	*RSA, DL*	*Remark*
64 Bit	128 Bit	≈ 700 Bit	Only short term security (a few hours or days)
80 Bit	160 Bit	≈ 1024 Bit	Medium security (except attacks from big governmental institutions etc.)
128 Bit	256 Bit	≈ 3072 Bit	Long term security (without quantum computers)

Rivest-Shamir-Adleman (RSA) Scheme

- Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman based on the factoring problem.
- Most widely used general-purpose approach to public-key encryption
- Is a cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n
  - A typical size for n is 1024 bits, or 309 decimal digits

RSA Scheme

- **Algorithm Key Generation Algorithm for RSA Public-Key Encryption by Alice, Alice should do the following:**
  - Choose large primes: p, q
  - Compute n = pq,
  - Compute ϕ(n) = (p-1)(q-1)
  - Select a random integer e, such that:
    - $1 < e < \emptyset$
    - $GCD(e, \emptyset) = 1$
  - Use the extended Euclidean algorithm to compute the unique integer d, such that $1 < d < \emptyset(n)$ as follows:
    - $ed \equiv 1 \, mod \, \emptyset(n)$ (i.e., $d = e^{-1} \, mod \, \emptyset(n)$)

Recall: φ(n) Function:

1. If n is a prime, then ϕ(n) =n - 1.

2. If n is a product of two primes, NOT equal, then ϕ (n) = (p -1) (q-1).

3. If n is a product of two primes, equal, then ϕ (n) = (p -1) q.

- **Keys: public, *(e, n)*; private, *(d, φ)*;**

Remember : In mathematical background lectures, we learned and applied some algorithms to find the inverse (d) like: Exhaustive search, Fraction Method and Multiply Theta. More reference:

RSA Scheme

)

- **Algorithm RSA Public-Key Encryption and Decryption**
- **Encryption: Bob should do the following:**
  - Obtain Alice's authentic public key (n, e)
  - Represent the message as an integer m in the interval [0, n-1].
  - Compute $c = m^e \ mod \ n$ (e.g. using one of Exponentiation Algorithms)
  - Send the Cipertect (c) to Alice.
- **Decryption: Alice should do the following:**
  - Get the Ciphertext (c) from Bob
  - Recover the plaintext (m) as follows:
    - $m = \ c^d \ mod \ n$

Exponentiation Algorithms:

1. **Fast Exponentiation Algorithm for Encryption and Decryption**
2. **Repeated Square-and-Multiply Algorithm for Exponentiation in Zn**

self-assessment

- Explain why the public key and private key in the RSA scheme are inverses in group $mod\ \emptyset(n)$ and not inverses in group $mod\ n$, where n is the product of two distinct large prime numbers?

Example: RSA with small numbers

ALICE		BOB
Message $x = 4$		1. Choose $p = 3$ and $q = 11$
		2. Compute $n = p * q = 33$
		3. $\Phi(n) = (3-1) * (11-1) = 20$
		4. Choose $e = 3$
	$K_{pub} = (33,3)$ ←	5. $d \equiv e^{-1} \equiv 7 \bmod 20$
$y = x^e \equiv 4^3 \equiv 31 \bmod 33$		
	$y = 31$ →	
		$y^d = 31^7 \equiv 4 = x \bmod 33$

Example of RSA Scheme

Alice

Key Generation

1) Select $p=5$, $q=7$
2) Compute $n = p * q = 35$
3) $\Phi(n) = (5-1) * (7-1) = 24$
4) Choose $e = 5$
5) $d \equiv e^{-1} \equiv 5 \bmod 20$

$K_{pub} = (35,5)$ →

Bob

Message (m) = 4

Encryption

$y = m^e \equiv 4^5 \equiv 9 \bmod 35$

← $c = 9$

Decryption

$c^d = 9^5 \equiv 4 = m \bmod 35$

Example of RSA Scheme

Alice

Key Generation

1) Select p=3, q=11
2) Compute $n = p * q = 33$
3) $\Phi(n) = (5-1) * (7-1) = 20$
4) Choose $e = 3$
5) $d \equiv e^{-1} \equiv 7 \bmod 20$

$$Kpub = (33,3) \longrightarrow$$

Bob

Message (m) = 4

Encryption

$c = m^e \equiv 4^3 \equiv 31 \bmod 33$

$$\longleftarrow c = 31$$

Decryption

$c^d = 31^7 \equiv 4 = m \bmod 33$

Example of RSA Scheme

- p = 5, q=7; n = 5*7 = 35; ϕ(n) = (5-1)(7-1)=24
- Exponent e = 5;
  - $1 < 5 < 24$
  - GCD (e, ϕ(n)) = (5, 24) = 1;
- $ed \equiv 1 \; mod \; \emptyset(n)$ (i.e., $d = e^{-1} \; mod \; \emptyset(n)$)

Public key: (e=5, n=35)

Private key: (d=5, φ (n)=24

GCD(5, 24):
24 mod 5 = 4
5 mod 4 = 1
4 mod 1 = 0 – Stop. GCD(5, 24) = 1

By using fraction method find the inverse ($e^{-1} \; mod \; \emptyset(n)$) as follows:
Def= φ/e = 24/5=4.8
D= 1/e = 1/5 = 0.2 → ***d = 5***
Repeat
d= d+def **5*5 ≡ *1 (mod 24)***
Until d= integer

Encryption M=4

$C = 4^5 \equiv 9 \; (mod \; 35)$

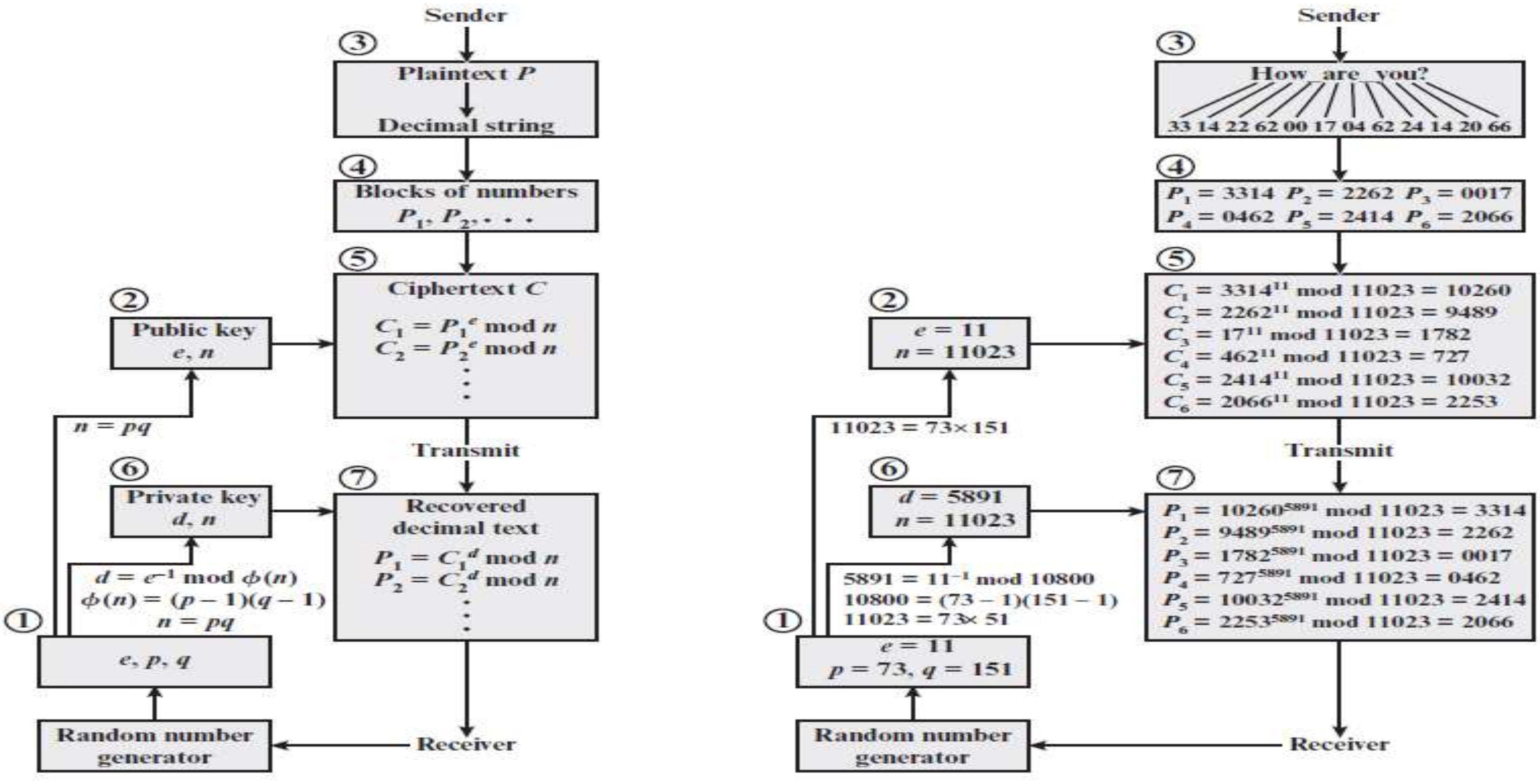
$$E(M) \equiv M^e \; (mod \; n)$$

Decryption C=9

$M = 9^5 \equiv 4 \; (mod \; 35)$

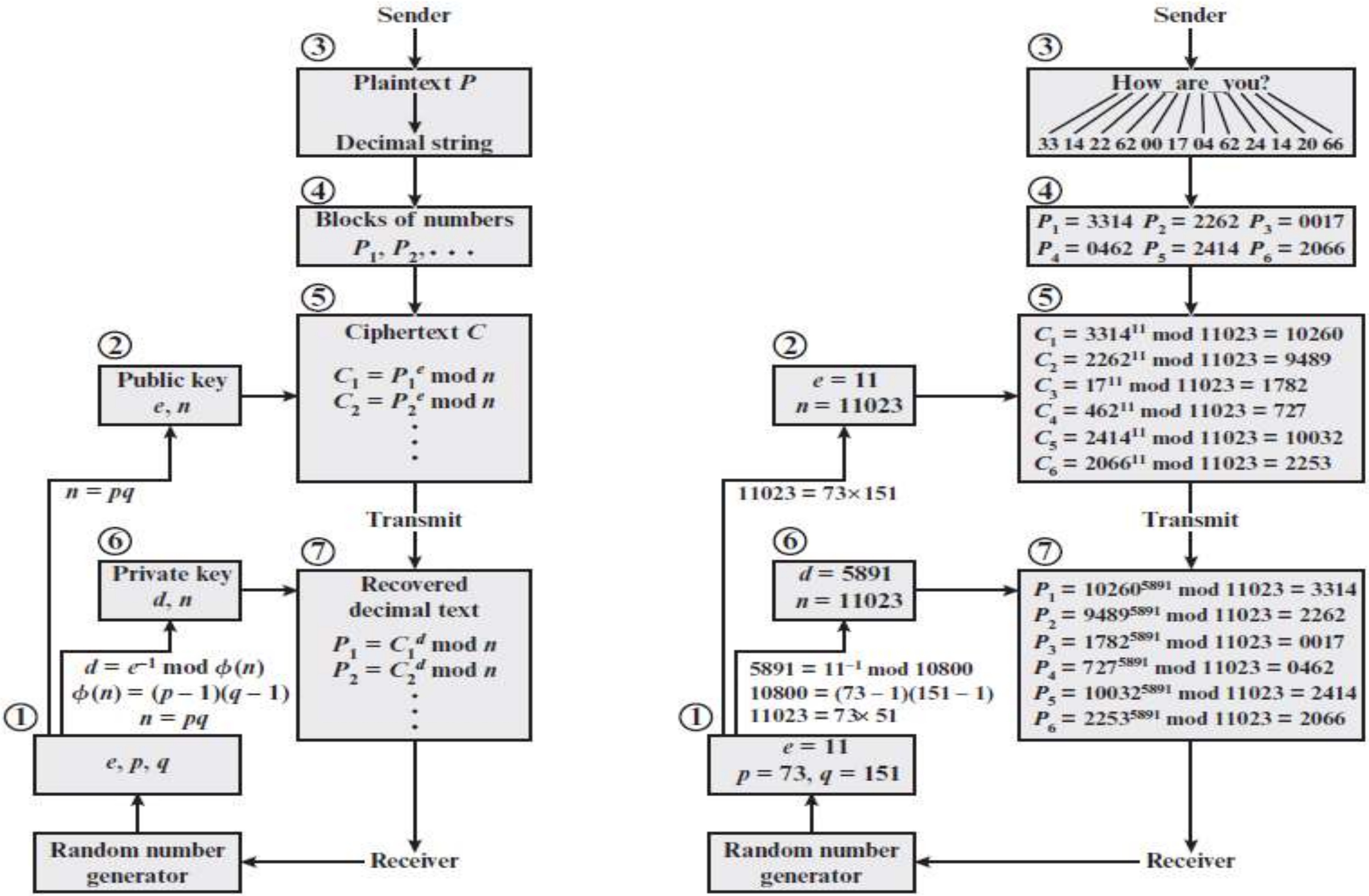
$$M \equiv C^d \; (mod \; n)$$

Computational Aspects (RSA Processing of Multiple Blocks)



(a) General approach

(b) Example



(a) General approach

(b) Example

Plaintext: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

RSA Processing of Multiple Blocks

- ***p = 43, q=59; n = 43*59 = 2357; φ(n) = 42*58 =2436***
- ***Exponent e = 13; (e, φ(n)) = (13, 42*58) = 1;*** → $d = 937$

  $937 * 13 \equiv 1 \pmod{2436}$
- **Block length is 4**

PUBLIC KEY CRYPTOGRAPHY

$C_1 = 1520^{13} \equiv 95 \pmod{2537}$

Public key: (13, 2357)

Private key: (937, 2357)

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}
1520	0111	0802	1004	2402	1724	1519	1406	1700	1507	2423

$E(M_i) \equiv M_i^e \pmod{n}$

0095	1648	1410	1299	0811	2333	2132	0370	1185	1457	1084
c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}

$0095^{937} \equiv 1520 \pmod{2537}$

$M_i \equiv C^d \pmod{n}$

Practical RSA parameters

- Practical RSA parameters are much, much larger. The RSA modulus n should be at least 1024 bit long, which results in a bit length for p and q of 512. Here is an example of RSA parameters for this bit length:
- p=
  E0DFD2C2A288ACEBC705EFAB30E4447541A8C5A47A37185C5A9B98389CE4DE19199AA3069B404F
  D98C801568CB9170EB712BF 10B4955CE9C9DC8CE6855C6123h
- q=
  EBE0FCF21866FD9A9F0D72F7994875A8D92E67AEE4B515136B2778A8048B149828AEA30BD0BA34B
  977982A3D42168F594CA99F3981DDABFAB2369F229640115h
- n=
  CF33188211FDF6052BDBB1A37235E0ABB5978A45C71FD381A91D12FC76DA0544C47568AC83D85
  5D47CA8D8A779579AB72E635D0B0AAAC22D28341E998E90F82122A2C06090F43A37E0203C2B72
  E401FD06890EC8EAD4F07E686E906F01B2468AE7B30CBD670255C1FEDE1A2762CF4392C0759499C
  C0ABECFF008728D9A11ADFh
- e=
  40B028E1E4CCF07537643101FF72444A0BE1D7682F1EDB553E3AB4F6DD8293CA1945DB12D796AE9
  244D60565C2EB692A89B8881D58D278562ED60066DD8211E67315CF89857167206120405B08B54
  D10D4EC4ED4253C75FA74098FE3F7FB751FF5121353C554391E114C85B56A9725E9BD5685D6C9C7
  EED8EE442366353DC39h
- d=
  C21A93EE751A8D4FBFD77285D79D6768C58EBF283743D2889A395F266C78F4A28E86F545960C2C
  E01EB8AD5246905163B28D0B8BAABB959CC03F4EC499186168AE9ED6D88058898907E61C7CCCC5
  84D65D801CFE32DFC983707F87F5AA6AE4B9E77B9CE630E2C0DF05841B5E4984D059A35D7270D5
  00514891F7B77B804BED81h

Preparation for next Lecture

- Continue reading about RSA Scheme:
  - **Security of the RSA:**
    - Trial Division
    - Pollard's rho and Pollard's p-1 algorithms
    - Oblivious Transfer

Brute Force Attack

Problem: Confidential Massage

A	B
$PU_A = (e = 7, n = 187)$	$PU_B = (e = 5, n = 299)$
$PR_A = (d = 23, n = 187)$	$PR_B = (d = 53, n = 299)$
$PU_B = (e = 5, n = 299)$	$PU_A = (e = 7, n = 187)$

Assume **A** send encrypted message to **B** – **A ----> B**

Brute Force Attack

Problem: Confidential Massage

A	B
$PU_A = (e = 7, n = 187)$	$PU_B = (e = 5, n = 299)$
$PR_A = (d = 23, n = 187)$	$PR_B = (d = 53, n = 299)$
$PU_B = (e = 5, n = 299)$	$PU_A = (e = 7, n = 187)$

Assume **A** send encrypted message to **B** – **A ----> B**

$$
\begin{aligned}
C &= E(PU_B, M) \\
&= M^e \bmod n \\
&= 15^5 \bmod 299 \\
&= 214
\end{aligned}
$$

$$C = 214$$

$$
\begin{aligned}
M &= D(PP_B, C) \\
&= C^d \bmod n \\
&= 214^{53} \bmod 299 \\
&= 15
\end{aligned}
$$

Brute Force Attack

- Just try all possibilities for **M**:

  M=1→ $214 \equiv 1^5 \mod 299$→ $214 \neq 1$ ✗

  M=2→ $214 \equiv 2^5 \mod 299$→ $214 \neq 32$ ✗

  M=3→ $214 \equiv 3^5 \mod 299$→ $214 \neq 243$ ✗

  .

  .

  .

  M=15→ $214 \equiv 15^5 \mod 299$→ $214 = 214$ ✓

- So, a brute force attack will try all values of M
- How stop the brute force attack?
- Make **M** large, and **M** to be large, ***n*** must be large because **M** must be less than ***n***. So, the RSA algorithm need to choose an ***n*** which is very enough large that is one of security condition.

Integer Factoring Problem

- **Factorisation:**
- With the exception of the number 1, all numbers can be decomposed into two or more numbers that multiply together to make the number.
- For example, the number 6 can be factorized as follows:
  - $6 = 3 \times 2 \times 1$.
    - 3, 2 and 1 are referred to as factors of 6.
  - 6 can also be factorized as: $6 = 6 \times 1$.
    - So 6 and 1 are also factors of 6.
- The process of decomposing a number in this way is called **factorisation.**

Security of RSA: **Trial division**

- Once it is established that an integer n is composite, before expending vast amounts of time with more powerful techniques, the first thing that should be attempted is trial division by all "small" primes. Here, "small" is determined as a function of the size of n.
- As an extreme case, trial division can be attempted by all primes up to $\sqrt{n}$. If this is done, trial division will completely factor n but the procedure will take roughly $\sqrt{n}$ divisions in the worst case when n is a product of two primes of the same size.
- In general, if the factors found at each stage are tested for primality, then trial division to factor n completely takes $o(p$

Trial division

- **Fact** Let n be chosen uniformly at random from the interval $[1\,, x]$.
- (i) If $\frac{1}{2} \leq \alpha \leq 1$, then the probability that the largest prime factor of n is $\leq x^{\propto}$ is approximately $1 + \ln \propto$. Thus, for example, the probability that n has a prime factor $> \sqrt{x}$ is $\ln 2$ ≈ 0.69.
- (ii) The probability that the second-largest prime factor of n is $\leq x^{0.2117}$ is about $\frac{1}{2}$.
- (iii) The expected total number of prime factors of n is ln $ln\, n + O(1)$. (If $n = \prod {p_i}^{e_i}$, the *total* number of prime factors of n is $\sum e_i$)

Trial division

Algorithm:

1. Choose an odd integer number that is not prime number (n).
2. Compute (S) as follows $\lfloor Sqrt\ (n) \rfloor$.
3. If (S is a prime) and (n mod S = 0) then return S.
4. Repeat

   4.1 S = S – 1

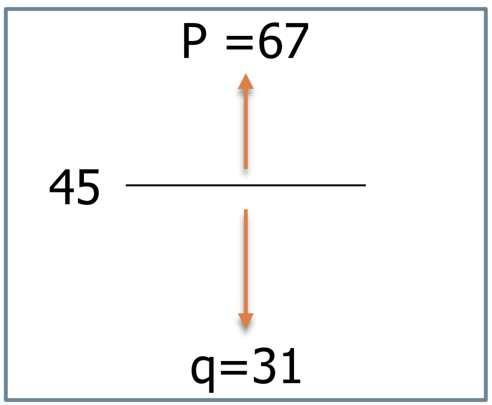
   4.2 Check if S is a prime

   4.3 Compute n mod S.

   Until (S is a prime) and (n mod S = 0) then return S.
5. End

Example

- Example suppose n = 2077
- Then the value of $\sqrt{2077}$ = 45.574115460..
- Find $\lfloor 45.574115460.. \rfloor$ = 45



- P=31 and q= 2077/31= 67

OR

- P=67 and q=2077/67=31

45	Not prime
44	Not prime
43	2077 mode 43 = 13
42	Not prime
41	2077 mode 41 = 27
40	Not prime
39	Not prime
38	Not prime
37	2077 mode 37 = 5
36	Not prime
35	Not prime
34	Not prime
33	Not prime
32	Not prime
31	2077 mode 31 = 0

45	Not prime
46	Not prime
47	2077 mode 47 = 9
48	Not prime
49	Not prime
50	Not prime
51	Not prime
52	Not prime
53	2077 mode 53 = 10
54	Not prime
55	Not prime
56	Not prime
57	Not prime
58	Not prime
59	2077 mode 59 = 12
60	Not prime
61	2077 mode 61 = 3
62	Not prime
63	Not prime
64	Not prime
65	Not prime
66	Not prime
67	2077 mode 67 = 0

Preparation for next Lecture

- Continue reading about RSA Scheme:
  - **Exponentiation Algorithms:**
    - Fast Exponentiation Algorithm for Encryption and Decryption
    - Repeated Square-and-Multiply Algorithm for Exponentiation in Zn

Finding power - Exponentiation

- Example: $c = m^e = 21^{11} \bmod 29$
- Raising **21** to the **power 11**, multiplying 11 copies of 21 together, looks like a lengthy and error prone task which will result in calculations involving numbers with many digits **(in fact 21^{11} = 350 277 500 542 221)** ??
- However, to work out the result we can take advantage of **two things**:
  1. We only need to work with numbers up to 29
  2. We can break down the operation of raising 21 to the power of 11 into a number of stages

Finding power

- 21^{11} means multiplying eleven copies of twenty-one together. A clue as to how this calculation might be broken down is given by writing the exponent of 11 as a sum of, for instance, three components 8 + 2 + 1 then:
  - $21^{11} \equiv 21^{8+2+1}$
- This shows that multiplying 11 copies of 21 together is the same as first multiplying eight copies of 21 together and then multiplying the result by the product of a further two copies of 21, giving a total of 10 copies.
- Next, to make the total number of copies 11 the result would need to be multiplied by another copy of 21. $\mathbf{21^{11}}$ can therefore be written as $\mathbf{21^{8} \times 21^{2} \times 21^{1}}$

Finding power

- **A second observation** can also be valuable. It is that, for instance, $21^8 = 21^{4+4} = 21^4 \times 21^4$
- That is, 21^8 is the same as multiplying two copies of 21^4 together. This can be summarized in the notation of exponentiation as $(21^4)^2$
- Note also that 21^4 can be found by multiplying two copies of 21^2 together so that $21^4 = 21^2 \times 21^2 = (21^2)^2$ and $21^8 = (21^4)^2$

Finding power

- **Now, exploiting the advantage of working modulo 29:**

$$21^2 \equiv 441 \bmod 29$$
$$\equiv 441 - 15 \times 29 \equiv 6 \bmod 29$$

- Using the result for 21^2 and taking a further step gives 21^4 as:

$$21^4 \equiv \left(21^2\right)^2 \equiv 6^2 \equiv 36 \bmod 29$$
$$\equiv 36 - 1 \times 29 \equiv 7 \bmod 29$$

- And then utilizing the result for 21^4 to obtain 21^8 gives:

$$21^8 \equiv \left(21^4\right)^2 \equiv 7^2 \equiv 49 \bmod 29$$
$$\equiv 1 \times 29 + 20 \equiv 20 \bmod 29$$

Finding power

- With these results the encryption calculation can be completed **without the need to perform arithmetic on very large numbers:**

$$21^{11} \bmod 29 \equiv 21^{8+2+1} \equiv 21^{8} \times 21^{2} \times 21 \equiv 20 \times 6 \times 21 \bmod 29$$

$$\equiv 120 \times 21 \equiv (4 \times 29 + 4) \times 21 \bmod 29$$

$$\equiv 4 \times 21 \equiv 84 \equiv 2 \times 29 + 26 \bmod 29$$

$$\equiv 26 \bmod 29$$

- **So the result of encryption the letter (21) by using the encryption key 11 is letter (26)**

Finding power

$$19^{13} \equiv 19^{8+4+1}\ mod\ 77$$

- Calculating the powers of 19 modulo 77 gives:

$$19^2 \equiv 361 \equiv (361 - 4 \times 77) \equiv 53\ mod\ 77$$

$$19^4 \equiv (19^2)^2 \equiv 53^2 \equiv 2809 \equiv (2809 - 36 \times 77) \equiv 37\ mod\ 77$$

$$19^8 \equiv (19^4)^2 \equiv 37^2 \equiv 1369 \equiv (1369 - 17 \times 77) \equiv 60\ mod\ 77$$

So

$$19^{13} \equiv 19^{8+4+1}\ mod\ 77 \equiv 60 \times 37 \times 19 \equiv 2220 \times 19$$
$$\equiv (2220 - 28 \times 77) \times 19\ mod\ 77 \equiv 64 \times 19 \equiv 1216$$
$$\equiv (1216 - 15 \times 77)\ mod\ 77 \equiv 61\ mod\ 77$$

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange: Overview

- Proposed in 1976 by **Whitfield Diffie and Martin Hellman**
- **Widely used**, e.g. in Secure Shell (SSH), Transport Layer Security (TLS), and Internet Protocol Security (IPSec)
- The Diffie–Hellman Key Exchange (DHKE) is a key exchange protocol and **not** used for encryption
- (For the purpose of encryption based on the DHKE, ElGamal can be used.)

Diffie-Hellman Key Exchange: Overview

- The purpose of key distribution (or key exchange) protocols is to allow a shared key to be securely transmitted between the principals
- **Diffie-Hellman key exchange protocol**: (**Important**)
  - Is a classic protocol which enables Bob and Alice to agree on a key for encrypting subsequent messages, which **does not require them to explicitly send the key**
- Its effectiveness depends on the difficulty of computing discrete logarithms

Diffie–Hellman Key Exchange: Set-up

1. Choose a large prime p.
2. Choose an integer $\alpha \in \{2,3,\ldots,p-2\}$.
3. Publish p *and* α.

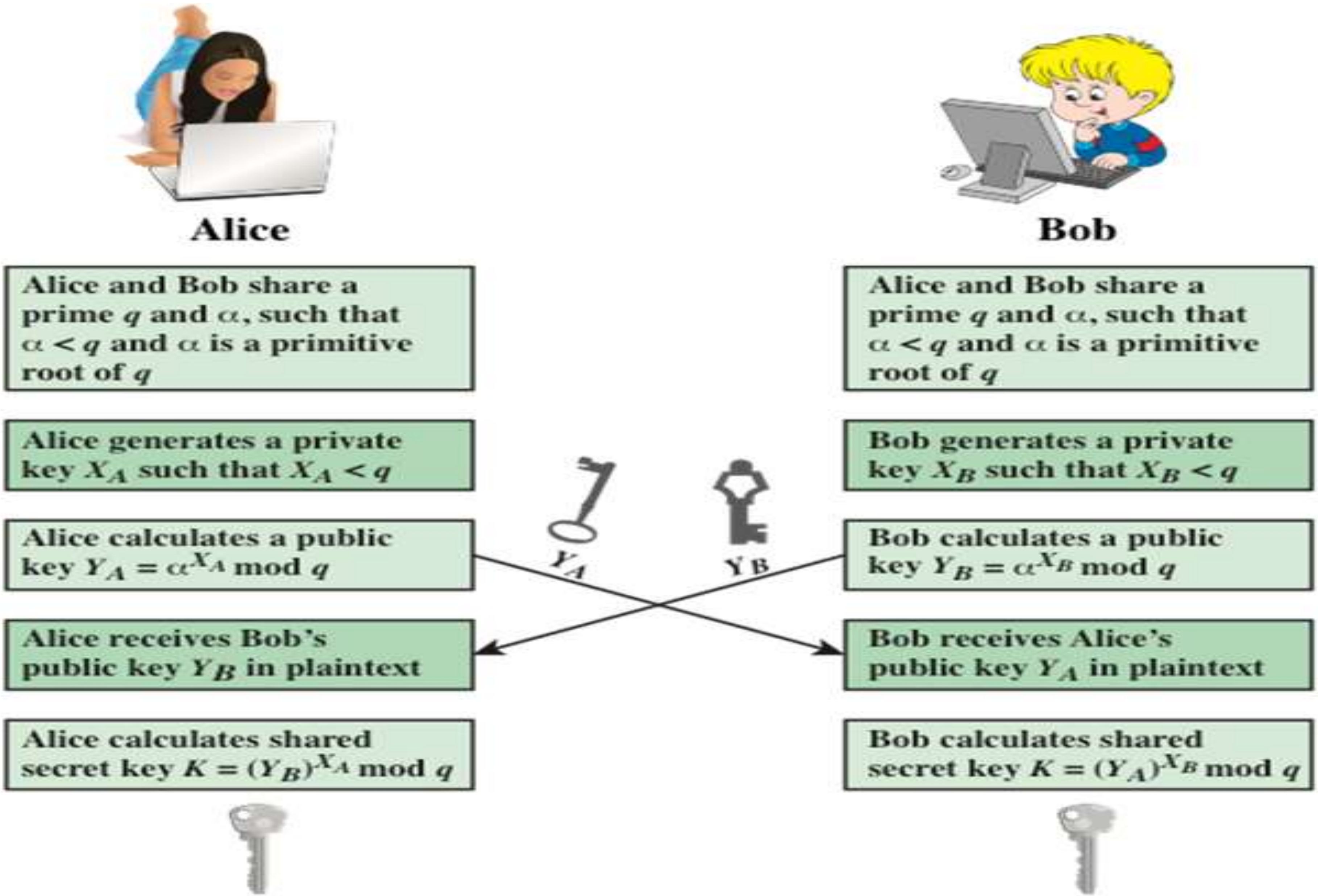


Figure Diffie-Hellman Key Exchange

One Immediate Application: The Diffie-Hellman Algorithm

Problem: Establish *common* keys (for symmetric cryptography) to be used by two individuals so that **intruders** cannot discover them in a feasible amount of computer time.

These are known to all!

Let

- $\boldsymbol{q}$ be a large prime (primitive root)
- α be an integer relatively prime to $\boldsymbol{p}$

Pick X_A relatively prime to q-*1*

Pick X_B relatively prime to *q-1*

$$Y_A \equiv \alpha^{X_A} (\bmod q), \quad 0 < Y_A < q$$

$$Y_B \equiv \alpha^{X_B} (\bmod q), \quad 0 < Y_B < q$$

$$K = (X_B)^{X_A} \bmod q \equiv \alpha^{X_B X_A} (\bmod q), 0 < K < q \; = \; K = (Y_A)^{X_B} (\bmod q) \equiv \alpha^{X_A X_B} (\bmod q), 0 < K < q$$

We can now use the joint key K for encryption, e.g., with AES

$$Y = AES_{K_{AB}}(X) \qquad X = AES^{-1}_{K_{AB}}(Y)$$

A Simple Example of a DH Exchange

Domain parameters

$\boldsymbol{q} = 17$

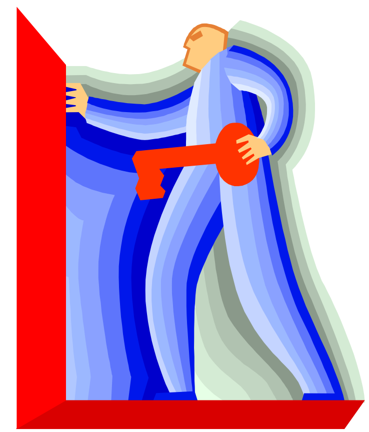
$\alpha = 2$

$X_A = 3$

$$Y_A \equiv \alpha^{X_A} (\bmod q) = 8 (\bmod 17) = 8$$

$$Y_B \equiv \alpha^{X_B} (\bmod q) = 32 (\bmod 17) = 15$$

$X_B = 5$



$$K = Y_B^{X_A} (\bmod q) = 3375 (\bmod 17) = 9 \quad = \quad K = Y_A^{X_B} (\bmod q) = 32768 \ (\bmod 17) = 9$$

Diffie–Hellman Key Exchange: Set-up

Example:

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that α = 7 and q = 23. The steps are as follows:

1. Alice chooses X_A = **3** and calculates $Y_A = 7^3 \bmod 23 =$ **21**.
2. Bob chooses X_B = **6** and calculates $Y_B = 7^6 \bmod 23 =$ **4**.
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key **$K = 4^3 \bmod 23 = 18$**.
6. Bob calculates the symmetric key **$K = 21^6 \bmod 23 = 18$**.
7. The value of K is the same for both Alice and Bob;

$$(\alpha^{X_A})^{X_B} \; mod \; q = \; 7^{18} \; mod \; 23 = 18$$

Primitive Roots

- Primitive Roots In the group $G = \langle Z_n{*}, \times \rangle$, when the order of an element is the same as f(n), that element is called the primitive root of the group.

Table 8.3 Powers of Integers, Modulo 19

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

Primitive Roots

Example

- Table 9.5 shows the result of $a^i \equiv x \pmod 7$ for the group $G = <Z_7^*, \times>$. In this group, $\phi(7) = 6$.

Table 9.5 *Example 9.50*

		$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
	$a = 1$	$\boxed{x: 1}$	x: 1	x: 1	x: 1	x: 1	x: 1
	$a = 2$	x: 2	x: 4	$\boxed{x: 1}$	x: 2	x: 4	x: 1
Primitive root →	$a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	$\boxed{x: 1}$
	$a = 4$	x: 4	x: 2	$\boxed{x: 1}$	x: 4	x: 2	x: 1
Primitive root →	$a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	$\boxed{x: 1}$
	$a = 6$	x: 6	$\boxed{x: 1}$	x: 6	x: 1	x: 6	x: 1

Primitive Roots

- If the group $G = \langle Z_n^*, \times\rangle$ has any primitive root, the number of primitive roots is $f(f(n))$.
- Cyclic Group If g is a primitive root in the group, we can generate the set Z_n^* as $Z_n^* = \{g^1, g^2, g^3, \ldots, g^{f(n)}\}$

Example:

- The group $G = \langle Z_{10}^*, \times\rangle$ has two primitive roots because $f(10) = 4$ and $f(f(10)) = 2$. It can be found that the primitive roots are 3 and 7. The following shows how we can create the whole set Z_{10}^* using each primitive root.

$g = 3 \rightarrow$	$g^1 \bmod 10 = 3$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 7$	$g^4 \bmod 10 = 1$
$g = 7 \rightarrow$	$g^1 \bmod 10 = 7$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 3$	$g^4 \bmod 10 = 1$

The group $G = \langle Z_n^*, \times\rangle$ is a cyclic group if it has primitive roots.
The group $G = \langle Z_p^*, \times\rangle$ is always cyclic.