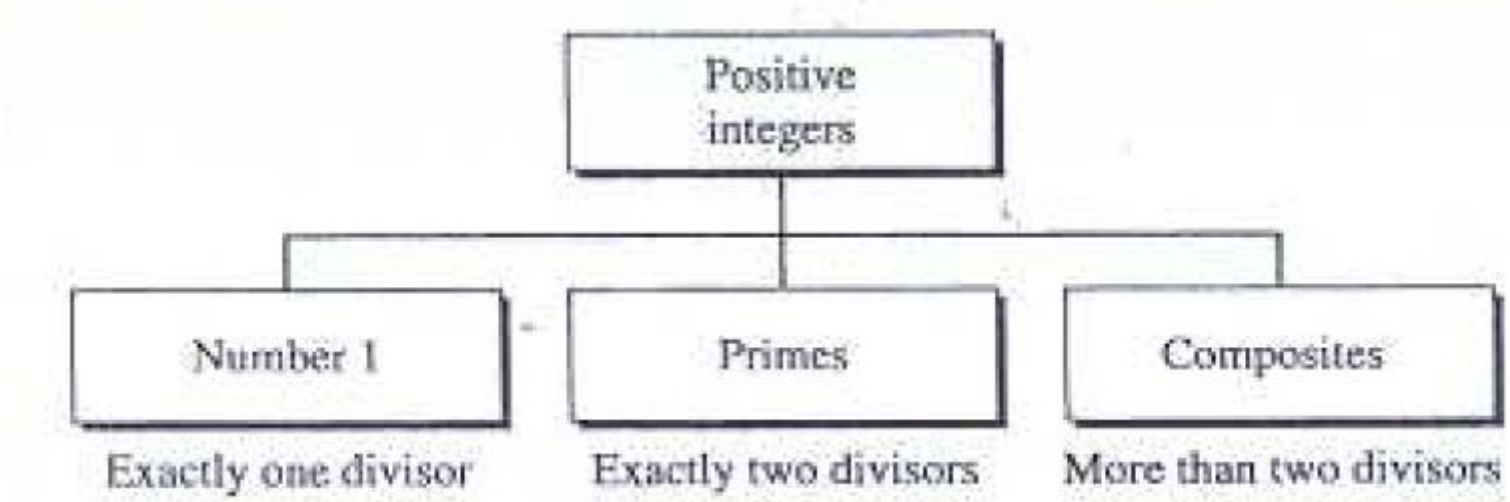
Module 3 : **Asymmetric Key Cryptography**

3.1 Mathematics of Asymmetric Key Cryptography: Primes, Primality Testing, Factorization, Quadratic Congruence, Exponentiation and Logarithm

3.2 Public key cryptography: Principles of public key, cryptosystems, The RSA algorithm, attacks on RSA

3.3 Key management: Diffie Hellman Key exchange, Man-in Middle attack

- Symmetric vs Asymmetric key cryptography
- Stream ciphers vs block ciphers
- Public key cryptography
- RSA
- Elliptic curve cryptography
- Public Key infrastructure (PKI)


Positive integers
Number 1
Primes
Composites
Exactly one divisor
Exactly two divisors
More than two divisors

Coprimes

- Two positive integers, a and b, are relatively prime, or coprime, if gcd (a, b) = 1.
- Note that the number 1 is relatively prime with any integer.
- If p is a prime, then all integers 1 to p-1 are relatively prime to p.
- Cardinality of Primes
- Given a number n, how many primes are smaller than or equal to n?

- Infinite number of primes can be there.
- Assume that the only primes are in the set (2, 3, 5, 7, 11, 13, 17). Here P = 510510 and P + 1 = 510511 However, 510511 = 19 * 97 * 277 ; none of these primes were in the original list. Therefore, there are three primes greater than 17.

Checking for Primeness

- If the number is divisible by any primes less than sqrt(n) then it is not prime.
- Is 97 a prime?
- Solution
- The floor of sqrt(97) = 9 The primes less than 9 are 2, 3, 5, and 7. is 97 divisible by any of these numbers?
- It is not, so 97 is a prime.

- Is 301 a prime?
- Solution
- The floor of sqrt(301) = 17
- We need to check 2, 3, 5, 7, 11, 13, and 17.
- The numbers 2, 3, and 5 do not divide 301, but 7 does.
- Therefore 301 is not a prime.

Sieve of Eratosthenes

- The Greek mathematician Eratosthenes devised a method to find all primes less than n.
- The method is called the sieve of Eratosthenes.
- Suppose we want to find all prime less than 100.
- Write down all the numbers between 2 and 100. Because sqrt(100) = 10,
- we need to see if any number less than 100 is divisible by 2, 3, 5, and 7.

Table 9.1 *Sieve of Eratosthenes*

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

The following shows the process:

1. Cross out all numbers divisible by 2 (except 2 itself).
2. Cross out all numbers divisible by 3 (except 3 itself).
3. Cross out all numbers divisible by 5 (except 5 itself).
4. Cross out all numbers divisible by 7 (except 7 itself).
5. The numbers left over are primes.

Euler Totient Function [ϕ](n)

- If consider arithmetic modulo n, then a **reduced set of residues** is a subset of the complete set of residues modulo n which are relatively prime to n
- eg for n=10,
- the complete set of residues is {0,1,2,3,4,5,6,7,8,9}
- the reduced set of residues is {1,3,7,9}
- The number of elements in the reduced set of residues is called the **Euler Totient function**
- **[ϕ](n)**

- p (p prime) [ϕ](p) =p-1
- pr (p prime) [ϕ](p) =pr-1(p-1)
- p.q (p,q prime) [ϕ](p) =(p-1)(q-1)

1. $\phi(1) = 0$.
2. $\phi(p) = p - 1$ if p is a prime.
3. $\phi(m \times n) = \phi(m) \times \phi(n)$ if m and n are relatively prime.
4. $\phi(p^e) = p^e - p^{e-1}$ if p is a prime.

- What is the value of Phi * (13)?
- Because 13 is a prime, phi(13) = (13 - 1) = 12

- What is the value of Phi(10) ?
- We can use the third rule: phi(10) = phi(2) * phi(5) = 1 * 4 = 4 because 2 and 5 are primes.

- What is the value of phi * (240) ?
- We can write 240 = 2 ^ 4 * 3 ^ 1 * 5 ^ 1
- Then phi(240) = (2 ^ 4 - 2 ^ 3)(3 ^ 1 - 3 ^ 0)(5 ^ 1 - 5 ^ 0) = 64

- Can we say that phi(49) = phi(7) * phi(7) = 6 * 6 =36?
- No. The third rule applies when m and n are relatively prime.
- Here 49 = 7 ^ 2 We need to use the fourth rule: phi(49) = 7 ^ 2 - 7 ^ 1 = 42

- What is the number of elements in Z_{14} *?
- The answer is phi(14) = phi(7) * phi(2) = 6 * 1 = 6 The members are 1, 3, 5, 9, 11, and 13.

Fermat's Little Theorem

- Fermat's little theorem plays a very important role in number theory and cryptography.
- First Version
- The first version says that if p is a prime and a is an integer such that p does not divide a, then a ^ (p - 1) = 1 mod p.
- Second Version
- The second version removes the condition on a. It says that if p is a prime and a is an integer,
- then a ^ p = a mod p.

- Find the result of 6 ^ 10 mod 11.
- Solution
- We have 6 ^ 10 mod 11 = 1.
- This is the first version of Fermat's little theorem where p = 11 .

- Find the result of 3 ^ 12 mod 11.
- Solution Here the exponent (12) and the modulus (11) are not the same.
- With substitution this can be solved using Fermat's little theorem.
- 3 ^ 12 mod 11 = (3 ^ 11 * 3) mod 11 = (3 ^ 11) mod 11) (3 mod 11) = (3 * 3) mod 11 = 9

Multiplicative Inverses

- A very interesting application of Fermat's theorem is in finding some multiplicative inverses quickly if the modulus is a prime.
- If p is a prime and a is an integer such that p does not divide a,
- then
- a ^ (p - 1) = 1 mod p.
- a^{-1} mod p = a ^ (p - 2) mod p

a. $8^{-1} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17 = 15 \bmod 17$

b. $5^{-1} \bmod 23 = 5^{23-2} \bmod 23 = 5^{21} \bmod 23 = 14 \bmod 23$

c. $60^{-1} \bmod 101 = 60^{101-2} \bmod 101 = 60^{99} \bmod 101 = 32 \bmod 101$

d. $22^{-1} \bmod 211 = 22^{211-2} \bmod 211 = 22^{209} \bmod 211 = 48 \bmod 211$

Euler's Theorem

- Euler's theorem can be thought of as a generalization of Fermat's little theorem.
- The modulus in the Fermat theorem is a prime, the modulus in Euler's theorem is an integer.
- First Version

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

- Second Version :
- it removes the condition that a and n should be coprime.
- If n = p x q, a <n, and k an integer,
- Then

$$a^{k \times \phi(n) + 1} \equiv a \pmod{n}$$

Example 9.15

Find the result of $6^{24} \bmod 35$.

Solution

We have $6^{24} \bmod 35 = 6^{\Phi(35)} \bmod 35 = 1$.

Example 9.16

Find the result of $20^{62} \bmod 77$.

Solution

If we let $k = 1$ on the second version, we have $20^{62} \bmod 77 = (20 \bmod 77)\ (20^{\Phi(77)+1} \bmod 77) \bmod 77 = (20)(20) \bmod 77 = 15$.

Application of Eulers

- Multiplicative Inverse

$$a^{-1} \bmod n = a^{\phi(n)-1} \bmod n.$$

a. $8^{-1} \bmod 77 = 8^{\phi(77)-1} \bmod 77 = 8^{59} \bmod 77 = 29 \bmod 77$

b. $7^{-1} \bmod 15 = 7^{\phi(15)-1} \bmod 15 = 7^{7} \bmod 15 = 13 \bmod 15$

c. $60^{-1} \bmod 187 = 60^{\phi(187)-1} \bmod 187 = 60^{159} \bmod 187 = 53 \bmod 187$

d. $71^{-1} \bmod 100 = 71^{\phi(100)-1} \bmod 100 = 71^{39} \bmod 100 = 31 \bmod 100$

8^59 = 8^5^11*8^4
= 8^5 mod 77 = 43
43^11*8^4 = 43^2^5*43*8^4

Generating Prime Numbers

- Mersenne Prime

$$M_p = 2^p - 1$$

- Failed at M11
- Fermat Prime

$$F_n = 2^{2^n} + 1$$

- Failed at F5

Chinese Remainder Theorem

If integers $n_1, \ldots, n_k$ **are pairwise coprime,**

Then the system of congruences

$$\begin{cases} x \equiv a_1 \bmod n_1 \\ x \equiv a_2 \bmod n_2 \\ \quad\vdots \\ x \equiv a_k \bmod n_k \end{cases}$$

has a unique solution modulo $N = n_1 n_2 \ldots n_k$ **:**

$$x \equiv \sum_{i=1}^{k} a_i N_i y_i \bmod N$$

Where $N_i = N/n_i$ **and** $y_i = N_i^{-1} \bmod n_i$

- **Example: Solve the following simultaneous linear congruences?**
  - $x \equiv 2 \pmod 3$
  - $x \equiv 3 \pmod 5$
  - $x \equiv 2 \pmod 7$
  -
  - **Solution:**

(1) Calculate $N = 3 \times 5 \times 7 = 105$

(2) Calculate

- $N1 = 105 / 3 = 35$,
- $N2 = 105 / 5 = 21$,
- $N3 = 105 / 7 = 15$

(3) Calculate

- $y1 = 35^{-1} \bmod 3 = 2$ since $35 \times 2 \equiv 1 \pmod 3$
- $y2 = 21^{-1} \bmod 5 = 1$ since $21 \times 1 \equiv 1 \pmod 5$
- $y3 = 15^{-1} \bmod 7 = 1$ since $15 \times 1 \equiv 1 \pmod 7$

(4) The solution to the simultaneous equations is

$$x = 2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1$$
$$= 233 \pmod{105} \equiv 23 \pmod{105}$$

- $x \equiv 1 \pmod 2$
  $x \equiv 2 \pmod 4$
  $x \equiv 3 \pmod 5$

- $X = 1 \bmod 3$
- $X = 1 \bmod 4$
- $X = 1 \bmod 5$
- $X = 0 \bmod 7$

Symmetric Key cryptography

- Uses same key in encryption and decryption process
- The encryption and decryption processes are inverse operations of each other
- Every user pair has to maintain the separate symmetric encryption key
- Both users have to maintain secrecy of the key
- Can be used to authenticate only each other, cannot be used to authenticate themselves to everybody else
- Classical symmetric key ciphers: Caesar Cipher, Playfair cipher, columnar cipher, railfence cipher etc
- Classical block ciphers :ECB, OFB, CFB, CBC
- Modern block ciphers : DES, AES

Symmetric Key cryptography

- A group of 100 users needs how many symmetric keys in total?

Symmetric Key cryptography

- A group of 100 users needs how many symmetric keys in total?

$$No\ of\ keys\ needed = \frac{n(n-1)}{2}$$

$$= 100 * \frac{100-1}{2}$$

$$= 100 * \frac{99}{2}$$

$$No\ of\ shared\ secret\ keys = 4950$$

Asymmetric Key cryptography

- Uses different keys in encryption and decryption process
- The encryption and decryption processes might be inverse operations of each other
- Every user has to maintain a pair of keys: public key and private key
- Only the private keys are kept secret, public keys can be announced to world
- Can be used to authenticate themselves to everybody else
- Typically follows stream ciphers
- Examples: RSA, ECC

Asymmetric Key cryptography

- A group of 100 users needs how many asymmetric keys in total?

Symmetric Key cryptography

- A group of 100 users needs how many asymmetric keys in total?

$$Total\ No\ of\ keys\ needed = 2 * N$$
$$= 2 * 100$$
$$= 200$$

No of keys each user has to maintain: 99 public keys of other users, 1 public key of own, 1 private key of own

$$= 99 + 1 + 1 = 101\ keys$$

Symmetric vs Asymmetric key differences

- **Speed:** Symmetric encryption is generally faster than asymmetric
- **Data Size:** Symmetric encryption is block cipher while Asymmetric is stream cipher
- **Key distribution:** In symmetric encryption, secure key distribution is crucial, as the same key is used for both encryption and decryption. Asymmetric encryption simplifies key distribution, as only the public key needs to be shared.
- **Key usage:** Symmetric encryption uses a single shared key for both encryption and decryption, while asymmetric encryption employs a pair of keys: a public key for encryption and a private key for decryption.
- **Use cases:** Symmetric encryption is ideal for bulk data encryption and secure communication within closed systems, whereas asymmetric encryption is often used for secure key exchanges, digital signatures, and authentication in open systems.

Ref: https://preyproject.com/blog/types-of-encryption-symmetric-or-asymmetric-rsa-or-aes#:~:text=Asymmetric%20and%20symmetric%20encryption%20are,a%20private%20key%20for%20decryption.

Symmetric vs Asymmetric key differences

- **Security:** Asymmetric encryption is considered more secure due to the use of two separate keys, making it harder for attackers to compromise the system. However, symmetric encryption can still provide strong security when implemented correctly with strong key management practices.
- **Scalability**: With a pair of keys, it is not difficult to communicate with multiple parties using Asymmetric key and that's how it is more scalable in large networks. With symmetric key, key management is a major issue
- **Resource Utilization:** Symmetric key encryption works on low usage of resources. Asymmetric encryption requires high consumption of resources.
- **Key Lengths:**128 or 256-bit key size for symmetric key cryptography while RSA uses 2048-bit or higher key size.

Ref: https://preyproject.com/blog/types-of-encryption-symmetric-or-asymmetric-rsa-or-aes#:~:text=Asymmetric%20and%20symmetric%20encryption%20are,a%20private%20key%20for%20decryption.

Public Key Cryptosystems

- Public-key/two-key/asymmetric cryptography involves the use of two keys:–
  - a public key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
  - A related private-key, known only to the recipient, used to decrypt messages, and sign(create) signatures
- Is asymmetric because–key used for encryption cannot be used for decryption and vice a versa
- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption

Some terms

- A message X in plaintext can be considered as X ==[X1, X2,… XM]
- The M elements of X are letters in some finite alphabet
- The message is sent by user A and is intended for destination user B
- B generates a related pair of keys a public key, PU_b and a private key, PR_b
- PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A
- With the message X and the encryption key PU_b as input, A forms the ciphertext

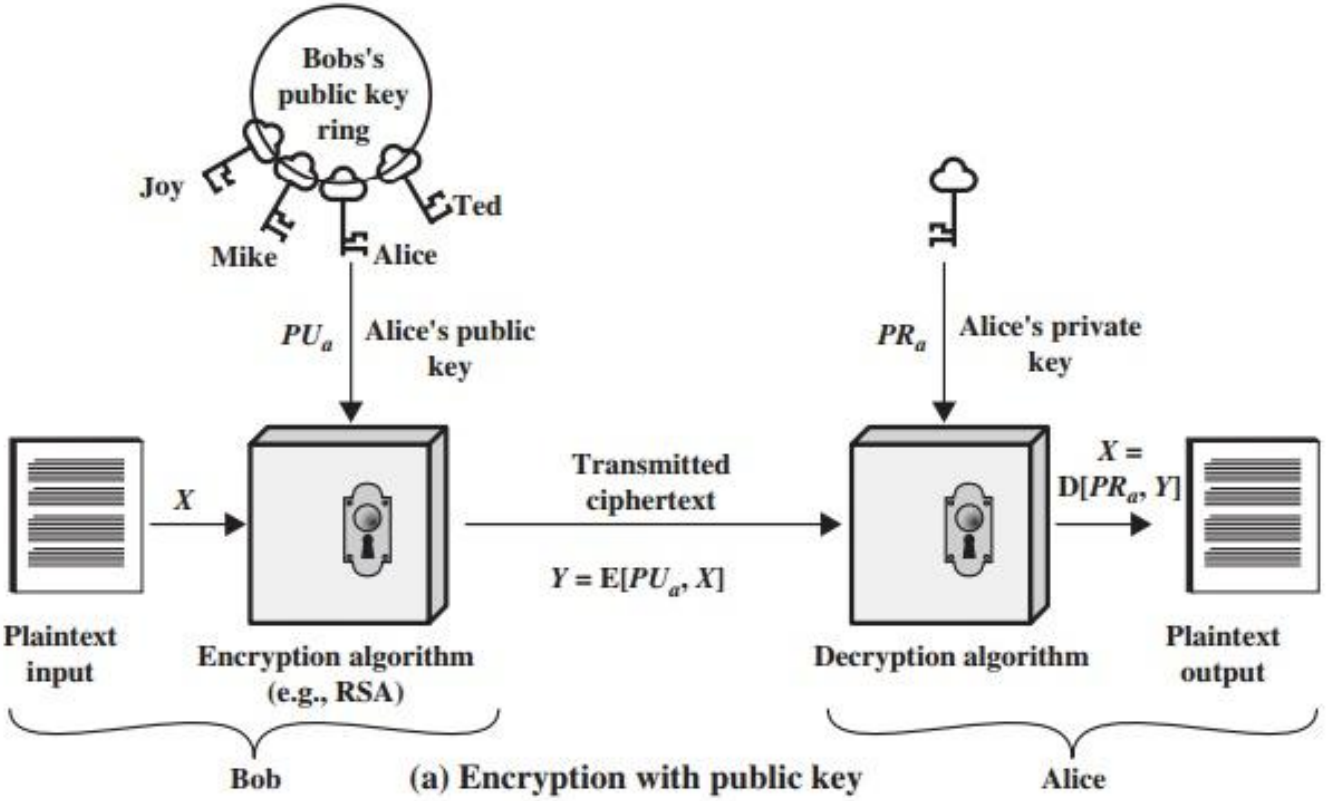
Y= [Y1, Y2, ..YN]

$$Y=E(Pu_{b,}X)$$

- The intended receiver, in possession of the matching private key, is able to invert the transformation

$$X = D(PR_b ,Y)$$

Encryption with Public key



(a) Encryption with public key

Encryption with Private key

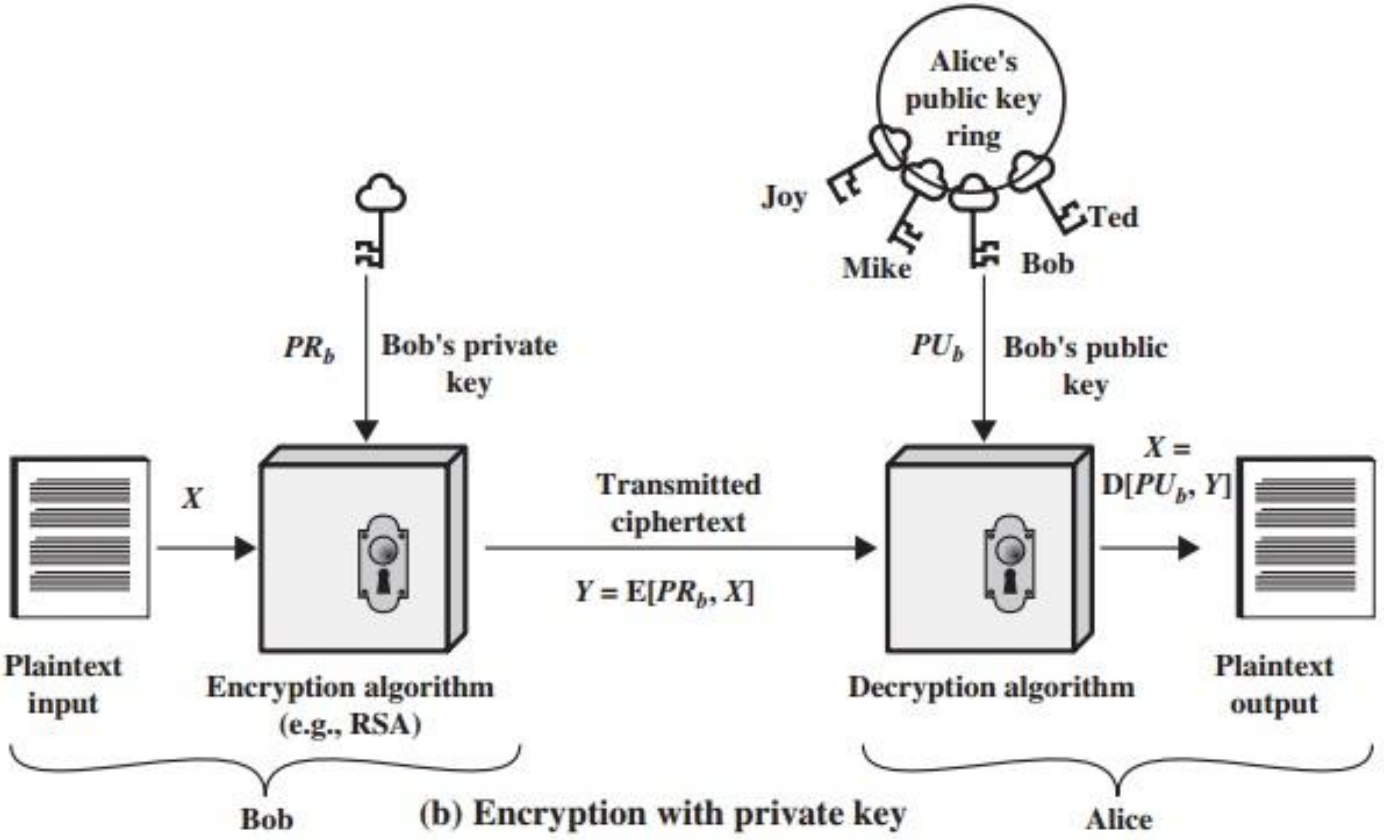


Figure 9.1 Public-Key Cryptography

Confidentiality with Public key cryptography

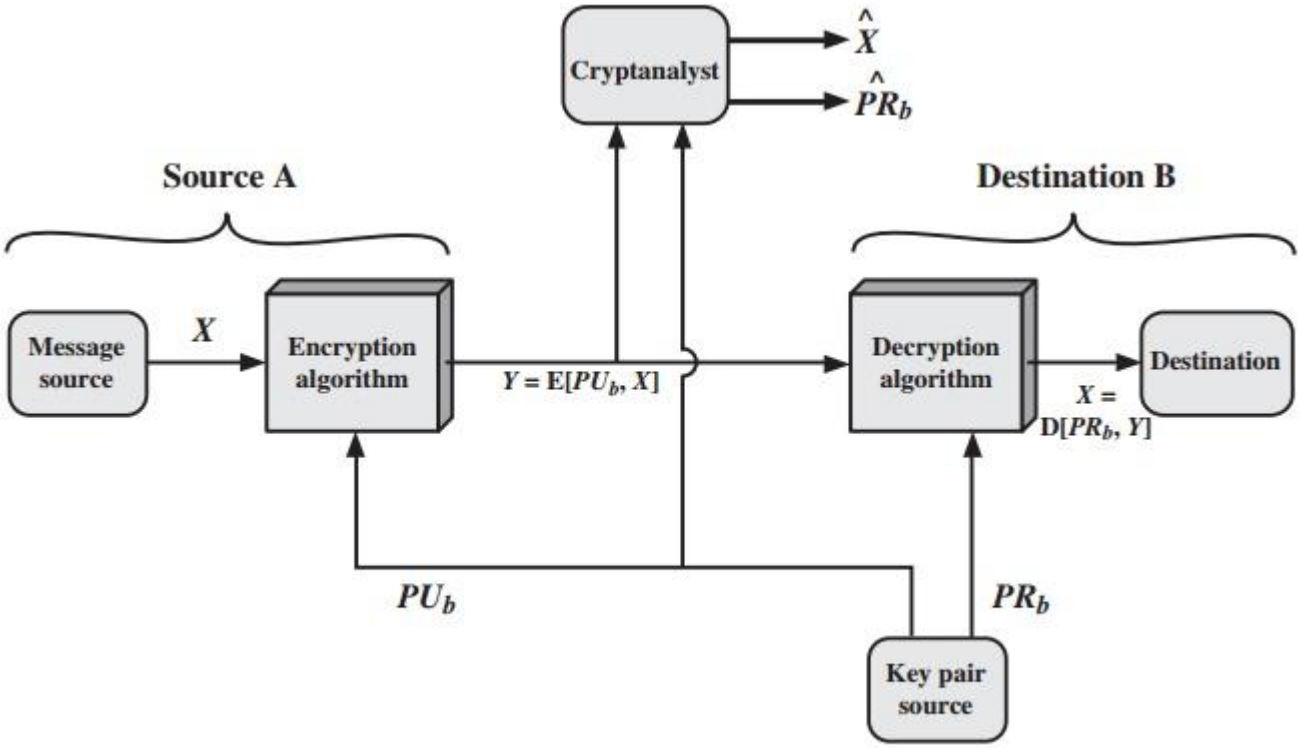


Figure 9.2 Public-Key Cryptosystem: Secrecy

Authentication with Public key cryptography

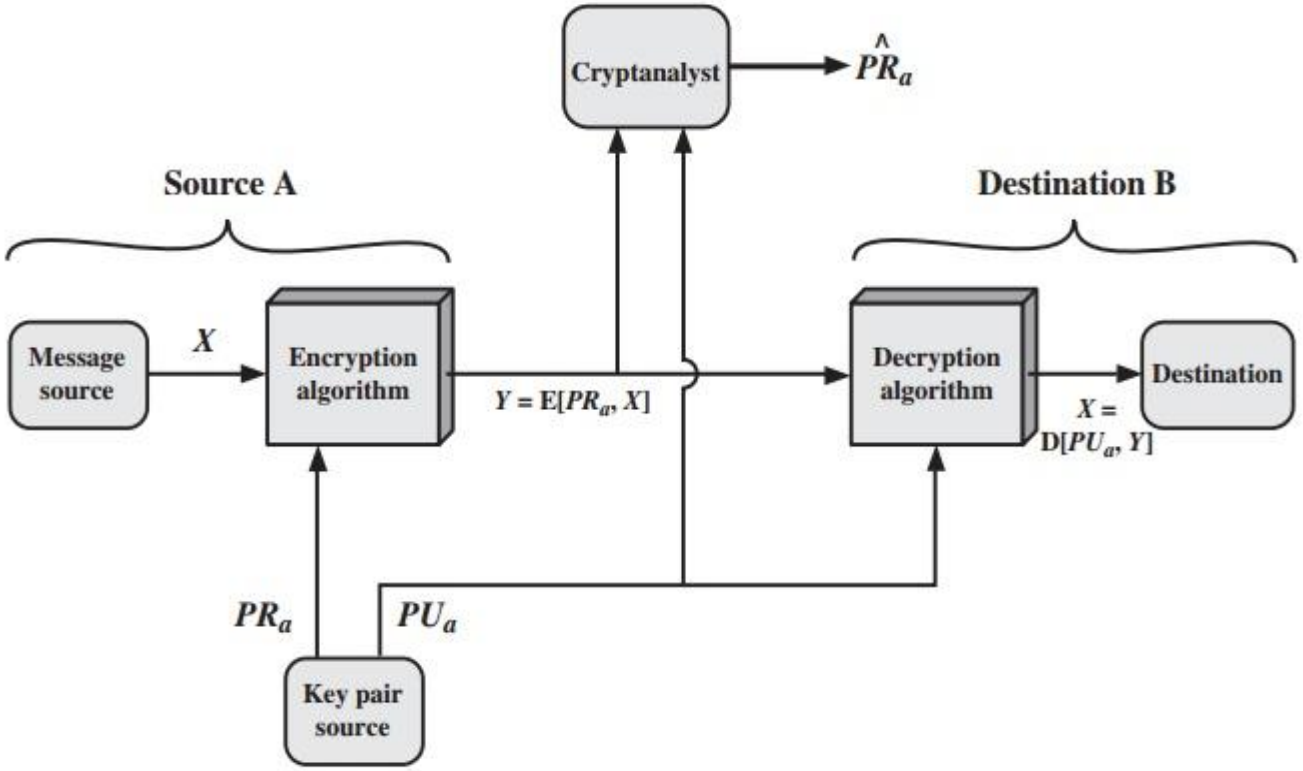


Figure 9.3 Public-Key Cryptosystem: Authentication

Combining Authentication and confidentiality

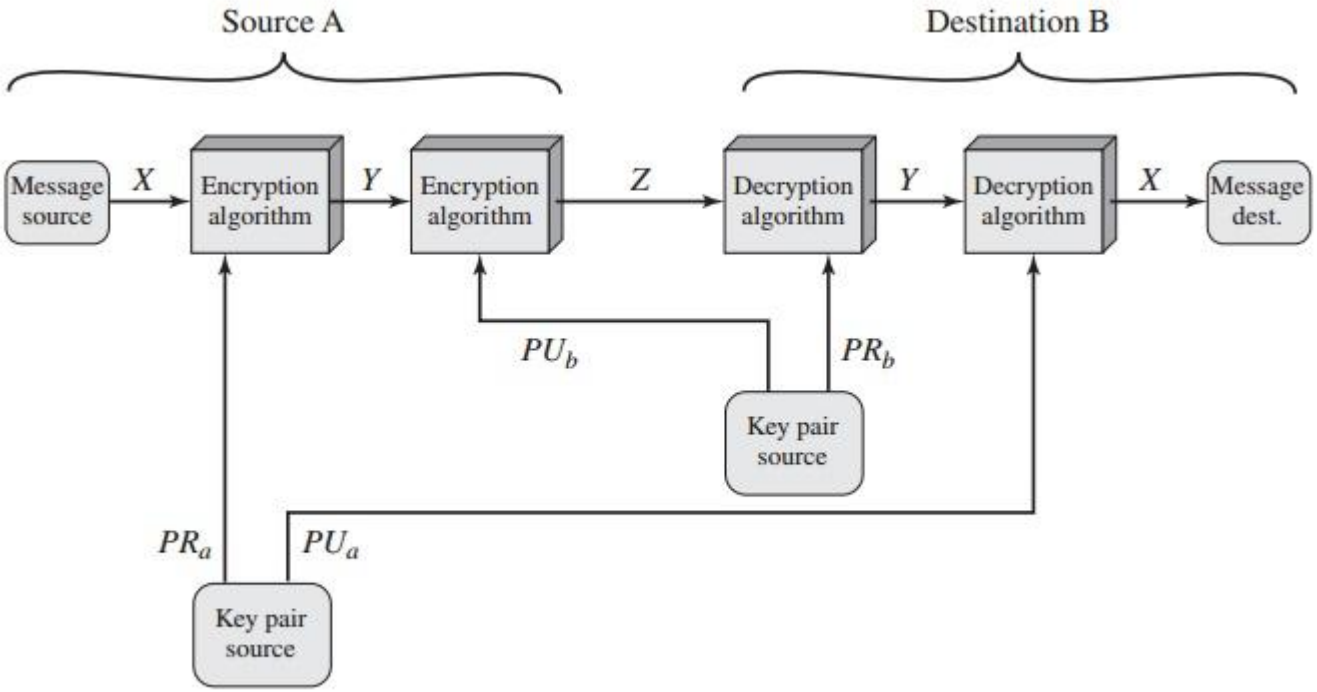


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Requirements of Public Key Cryptography

- It is computationally easy for a party B to generate a key pair (public key PU_b private key PR_b)
- It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext

- $C = E(Pu_b, M)$

- It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message

- $M= D(PR_b,C)= D[PR_{b,}E(PU_b,M)]$

Requirements of Public Key Cryptography

- Algorithm must fulfill:
- It is computationally infeasible for an adversary, knowing the public key, PU_b to determine the private key PR_b
- It is computationally infeasible for an adversary, knowing the public key, PU_b and a ciphertext C, to recover the original message M.

Public key cryptography

- The two keys can be applied in either order
- $M = D[Pu_b, E(PR_b, M)] = D[PR_b, E(Pu_b, M)]$

RSA

- "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman,
- RSAs publicly described the algorithm in 1977 at MIT, first published in 1978[RIVE78]
- Stream cipher system

RSA

- The RSA algorithm involves four steps:
  - key generation,
  - key distribution,
  - encryption,
  - and decryption.
- RSA uses modular exponentiation for encryption/decryption
- To attack it, Eve needs to calculate $\sqrt[e]{c} \bmod n$.

RSA Algorithm

Key Generation

Select p,q	p and q both prime; $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n),e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e,n\}$
Private key	$KR = \{d,n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption

Plaintext:	C
Ciphertext:	$M = C^d \pmod n$

Example 1

P=3, q=5 , thus n = p*q =15

Euler's Totient function to compute no of numbers co-prime with n:

$\Phi(n) = (p-1)*(q-1) = 2 * 4 = 8$

Choose e, such that Gcd (e, Φ (n)) = 1 and , $1< e< \Phi(n)$

Thus e could be any value from the set: ={3,5,7} as

GCD(e, Φ (n)) is 1 and $1<e<8$

Let e = 3

Compute d as: d*e mod Φ (n) = 1

d * 3 mod 8 = 1 (Use extended Euclidean's method to compute multiplicative inverse)

So, d = 3

Public key = (e,n) = (3,15)

Private key = (d,n) = (3,15)

Let M= 4

$C= M^e \bmod n = 4^3 \bmod 15= 4$

$M=C^d \bmod n = 4^3 \bmod 15= 4$

Example 2

- P=11, q=3, thus n = p*q =33
- $\Phi(n) = (p-1)*(q-1) = 10*2 = 20$
- Choose e such that $Gcd(e, \Phi(n)) = 1$ and $1<e<\Phi(n)$
- So e = {3,7,9,11,13,17,19}
- Let e= 3
- Compute d such that: d* e mod $\Phi(n) = 1$
- d * 3 mod 20 = 1
  - So, d = 7
- Public key = (e,n) = (3,33)
- Private key = (d,n) = (7,33)
- Let M= 7
- $C=M^e \bmod n = 7^3 \bmod 33 = 13$
- $M=C^d \bmod n = 13^7 \bmod 33 = \mathbf{7}$

Applications of RSA

- RSA is a versatile encryption and authentication algorithm with a wide range of applications in securing :
  - Digital communication,
  - Data security , and
  - Digital transactions.
- Its strength lies in its ability to provide secure encryption and digital signatures while allowing for public key distribution, making it a cornerstone of modern cryptography.
- Sign: Encrypt with private key
- Verify: Decrypt with associated public key

Applications of RSA

- **Secure Data Transmission**: RSA is commonly used for securing data transmission over insecure networks, such as the internet. It's employed in protocols like SSL/TLS for securing web traffic, SSH for secure remote access, and S/MIME for secure email communication.
- **Digital Signatures**: RSA is used to create digital signatures, which verify the authenticity and integrity of digital documents or messages. Digital signatures are vital for ensuring that the sender of a message is who they claim to be and that the message has not been tampered with.
- **Secure Key Exchange**: RSA can be used to securely exchange encryption keys between two parties. For example, it's used in the Diffie-Hellman key exchange protocol to establish secure communication channels between parties.

Applications of RSA

- **Secure Email**: RSA encryption is used in email encryption schemes like PGP (Pretty Good Privacy) and S/MIME to secure the content of emails, ensuring that only the intended recipient can decrypt and read the message.
- **Secure File Transfer**: RSA can be used to secure file transfers, ensuring that files are encrypted during transmission and can only be decrypted by authorized recipients.
- **Digital Certificates**: RSA is a fundamental component of digital certificates, which are used to verify the identity of websites, organizations, or individuals on the internet. Certificate Authorities (CAs) use RSA to create and sign digital certificates.

Applications of RSA

- **Secure Login and Authentication**: RSA is used in secure login mechanisms, such as two-factor authentication (2FA) and Secure Shell (SSH) logins. It helps ensure that only authorized users can access systems or accounts.
- **Secure E-commerce Transactions**: RSA plays a crucial role in securing online transactions, including credit card transactions and online banking. It protects sensitive financial information during transmission.
- **VPN Encryption**: Virtual Private Networks (VPNs) use RSA for encryption to establish secure connections over the internet. RSA encryption ensures that data transmitted between the user and the VPN server remains confidential.

Applications of RSA

- **Data Integrity Checking**: RSA signatures can be used to verify the integrity of software updates and downloads. Users can be confident that the software has not been tampered with during the download process.
- **Secure Chat and Messaging Apps**: Many secure chat and messaging apps use RSA encryption to secure conversations and messages, ensuring that only the intended recipients can read the messages.
- **Secure IoT Communication**: RSA can be used to secure communication in the Internet of Things (IoT) by encrypting data transmitted between IoT devices and cloud servers, protecting sensitive information and device control commands.
- **Secure Cloud Storage**: RSA encryption can be employed to protect data stored in the cloud, ensuring that only authorized users can access and decrypt it.

Advantages of RSA

- Security: highly secure when used with sufficiently long key lengths. Breaking RSA is a computationally intensive and becomes increasingly difficult as the key length grows.
- Public/Private Key Pair: Dual-key system in RSA eliminates the need for both parties to share a secret key beforehand
- Digital Signatures: used in secure communication and document verification.
- Key Exchange: To securely exchange secret keys for symmetric encryption algorithms such as SSL/TLS.
- Mathematical Soundness: RSA is based on the mathematical difficulty of factoring large semiprime numbers, which is believed to be a hard problem. Its security is rooted in well-established mathematical principles.
- Standardization: RSA has been widely standardized and is supported by many cryptographic libraries and software implementations, making it easy to integrate into various applications.

Disadvantages of RSA

- **Key Length:** key lengths can impact performance and increase computational overhead. Longer key lengths also require more storage for keys and certificates.
- **Computational Intensity:** encryption and decryption operations are computationally intensive, especially with longer key lengths. This slows down processing, making it less suitable for resource-constrained devices.
- **Vulnerability to Quantum Computing:** The security of RSA relies on the difficulty of factoring large numbers, which could be broken by quantum computers As quantum computing technology advances, RSA's security may be compromised.
- **Key Management:** RSA key management can be challenging, especially in large-scale systems. Safeguarding private keys and ensuring their integrity is critical. Loss or compromise of a private key can have severe security consequences.

Disadvantages of RSA

- **Performance Overhead:** The computational overhead of RSA encryption and decryption can impact the performance of systems, particularly in high-throughput applications like web servers.
  - Solution : Use hybrid encryption schemes (using RSA for key exchange and symmetric cryptography for data encryption) are often employed.
- **Lack of Forward Secrecy:** RSA does not provide forward secrecy, which means that if an attacker obtains a private key, they can decrypt all past and future communications encrypted with that key. This is a limitation in situations where forward secrecy is desired.
- **Key Length vs. Security:** As computing power advances, key lengths required to maintain security may need to be increased. This can be a challenge because longer key lengths increase computational demands and may not be supported by older hardware and software.

Summary: Attacks on RSA

- Brute Force Attack:
  - attacker attempts every possible private key until they find the one that successfully decrypts a message.
  - not practical against RSA when long key lengths
- Factorization Attack:
  - based on the difficulty of factoring the product of two large prime numbers.
  - Attack aims to find the prime factors of the modulus (n) to compute the private key.
  - Advanced algorithms like Pollard's rho algorithm and the General Number Field Sieve (GNFS) have been developed to factor large numbers efficiently.
  - The security of RSA relies on the difficulty of this factorization problem.
- Timing Attacks:
  - Timing attacks involve measuring the time it takes for an RSA operation to execute.
  - By analyzing the timing variations, an attacker can gain information about the private key, particularly in scenarios where RSA operations take different amounts of time based on the input data.

Summary:Attacks on RSA

- Chosen Plaintext Attack:
  - attacker can choose specific plaintexts and observe the corresponding ciphertexts produced by an RSA
  - With enough pairs of plaintext/ciphertext, an attacker may be able to deduce information about the private key.
- Ciphertext-Only Attack:
  - attacker has access only to encrypted messages without knowledge of the corresponding plaintexts.
  - This is a challenging scenario, but in some cases, it may be possible to gain partial information about the plaintext or the private key.

Summary:Attacks on RSA

- Common Modulus Attack:
  - attacker has access to two different ciphertexts encrypted with the same RSA modulus but different public exponents.
  - By exploiting mathematical relationships between these ciphertexts, an attacker may recover the plaintext.
- Quantum Computing Attacks:
  - Quantum computers, if developed sufficiently, could potentially factor large numbers efficiently using algorithms e.g. Shor's algorithm.
  - This would render RSA encryption vulnerable, highlighting the need for post-quantum cryptographic solutions.

Details- RSA attacks with examples

Common Modulus Attack

- If multiple entities share the same modulus n=pq with different pairs of (e_i, d_i), it is not secure. Do not share the same modulus!

- Cryptanalysis: If the same message M was encrypted to different users

  User u_1 : $C_1 = M^{e1} \bmod n$

  User u_2 : $C_2 = M^{e2} \bmod n$

If gcd(e_1,e_2)=1, there are a and b s.t. $ae_1 + be_2 = 1 \bmod n$

Then,

$(C_1)^a(C_2)^b \bmod n = (M^{e1})^a(M^{e2})^b \bmod n = M^{ae1+be2} \bmod n = M \bmod n$

Short Message Attack

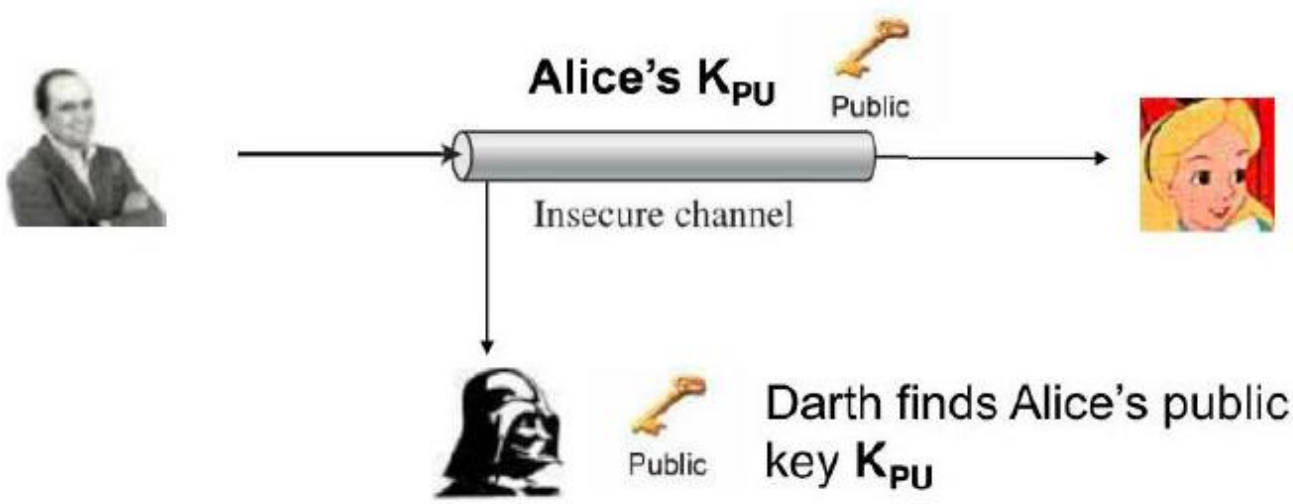
- Typical use of public key algorithm:
  Generating short messages
  - Symmetric keys (used then to send rest of message)
  - Social security numbers, etc.

- Idea:
  - Adversary acquires public key ***E***, ***n***
  - Uses them to encrypt all possible messages that may be sent (plausible if messages are short enough!) and stores in table
  - Intercepts encrypted message ***C*** and searches for match in the table

  Adversary can recover plaintext without decryption key!

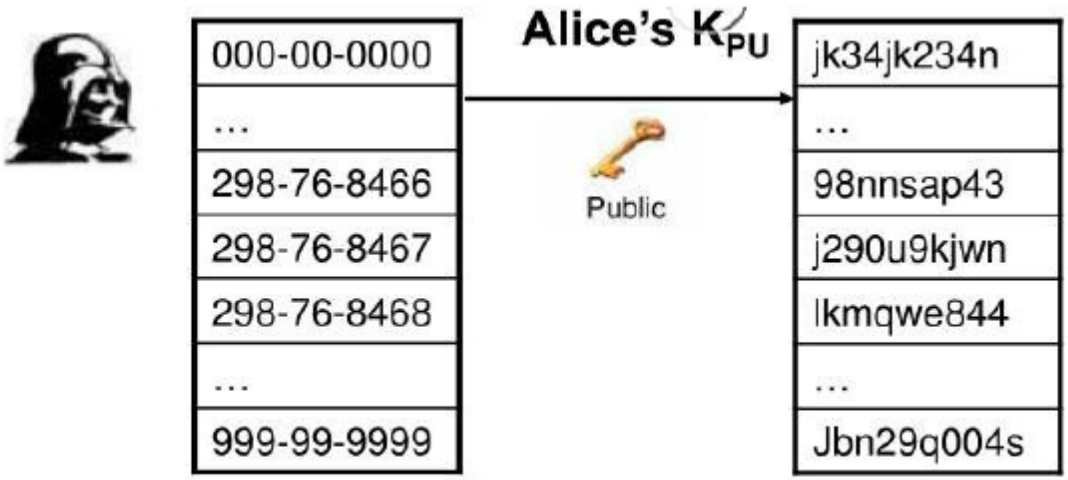
Short Message Attack (cont.)

- Example:
  Darth knows that Bob will use Alice's public key to send her a Social Security Number (9 digits)



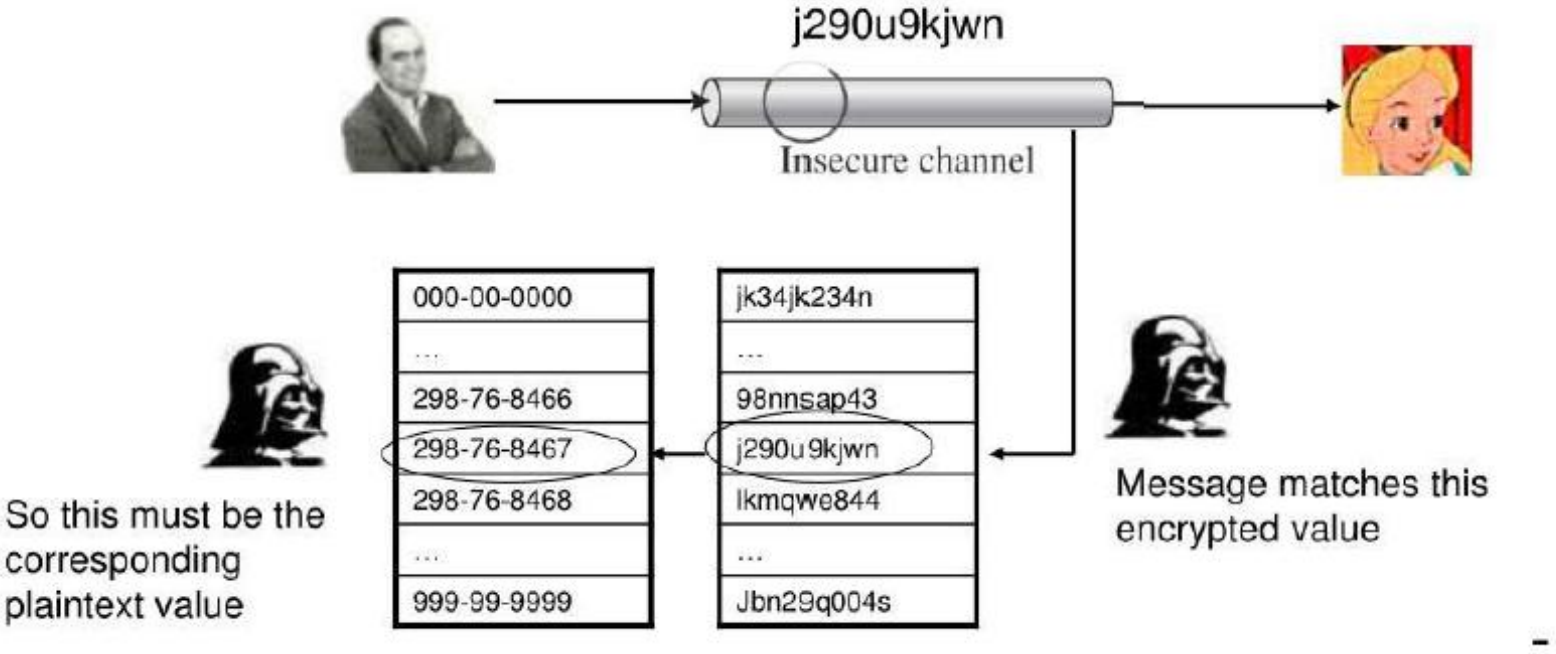
Short Message Attack (cont.)

- Darth uses Alice's public key K_{PU} to encrypt <u>all possible Social Security Numbers</u>
  (only a billion)



Short Message Attack (cont.)

- Darth intercepts Bob's SSN encrypted with Alice's public key
- Searches for match in table of encrypted values



RSA Cryptanalysis: Timing Attack (Theoretical)

- Timing attacks measure the time taken to perform cryptographic operations.
- Let's assume you're performing RSA encryption with an insecure implementation that leaks information through timing variations.
- You collect timing data for multiple encryptions of the same plaintext.
  - Encryption 1: 5 ms
  - Encryption 2: 4 ms
  - Encryption 3: 6 ms
- From the timing data, one might observe that the encryption operation took longer for Encryption 3.
- This could indicate that the plaintext had a particular property or structure.
- Over time, one might gather enough information to make educated guesses about the private key.
- In practice, mitigating timing attacks requires implementing secure cryptographic libraries and protecting against side-channel attacks.

Timing Attack

- If adversary knows the following:
  - Ciphertext ***C***
    - Can be intercepted
    - Can compute how long it takes to multiply ciphertext and compute mods
  - Total time decryption takes
    - Can be observed

  They could compute number of 1's in private ***D***

- Given enough known plaintexts, can reliably guess ***D*** completely

Timing Attack (cont.)

- Fast exponentiation algorithm used for decryption to compute C^D mod n:

```
result = 1
for (i = 0 to number of bits in D - 1) {
  if (ith bit of D = 1)
     result = (result * C) mod n
  C = C^2 mod n
  }
```

- Speed of decryption depends on number of **1**'s in ***D***
  - Each **1** requires additional multiplication operation
  - Each **0** skips that step

Timing Attack (cont.)

Solutions:

- "Pad" algorithm so all decryptions take same time

```
for (i = 0 to number of bits in D - 1) {
  if (ith bit of D = 1) result = (result * C) mod n
  else garbageVariable = (result * C) mod n
  C = C^2 mod n
  }
```

Cycling attack on RSA

- Since $C = M^e \bmod n$
- Encryption maps message m to one of the elements of the message space Z = (0,1,...,n -1).
- If the encryption is applied repeatedly on c, eventually a stage will arrive when C will get mapped to M.
- The adversary uses this fact to his advantage.
- He intercepts a ciphertext c, he carries out repeat encryptions of e till he gets back the intercepted ciphertext c.
- He goes back by one step because the message encrypted last must be the original plaintext m.

- Example : Bob sends ciphertext 37 to Alice after using her RSA public key [7, 77).
- The adversary intercepts the ciphertext and launches cycling attack using the public key of Al Write the computation carried out by the adversary to decrypt the ciphertext. Solution
- The adversary encrypts 37 repeatedly as given
- below:
- c_{1} = 37 ^ 7 mod 77 = 16
- c_{2} = 16 ^ 7 mod 77 = 58
- c_{3} = 58 ^ 7 mod 77 = 9
- c_{4} = 9 ^ 7 mod 77 = 37
- In the fourth step he gets c_{4} = 37
- Therefore, he concludes Bob's message is 9.

RSA Cryptanalysis: Factoring Small Modulus (Insecure Key Lengths):

Problem: Suppose you have an RSA public key : {3,187}. You want to find the private key (d) and factor the modulus.

- First, you need to find the prime factors of 187.
  - Attempt to factor 187:
  - 11 x 17 = 187
- Calculate $\phi(n) = (11-1)(17-1) = 160$
- Compute the private exponent (d) as the modular multiplicative inverse of 3 modulo 160. In this case, d = 107.
- With this small modulus, RSA is insecure, and it's possible to factor the modulus and compute the private key efficiently. Real-world RSA implementations use much larger moduli (e.g., 2048 bits or more) to resist factorization attacks.

RSA Cryptanalysis: Factoring Small Modulus (Insecure Key Lengths)

- **Key = {3,33}**
- **Factorize N:**
  N=33 can be expressed as 33=3×11
  So, p=3 and q=11
- **Calculate ϕ(N) = 2 * 10 =20**
- **Public key e:** Assume e=3 ‘e’ must satisfy 1<e<ϕ(N) and gcd(e,ϕ(N))=1
- Here, gcd(3,20)=1 so e is valid.
- **Private key d :** d is the modular multiplicative inverse of e modulo ϕ(N)

Solve e*d≡1 mod ϕ(N)
3*d≡1 mod 20

Using the extended Euclidean algorithm: d=7

- **Break RSA**
  Using the factorization of N, the private key d can now be computed. With d, you can decrypt any ciphertext encrypted with the public key (N,e)
- This is a toy example; real RSA uses large N (hundreds of digits) to make factorization infeasible.

RSA Cryptanalysis: Chosen Plaintext Attack (Theoretical)

- An attacker can choose plaintexts to be encrypted and observe the resulting ciphertexts.
- This scenario is **highly simplified** for illustration purposes:
- Attacker chooses plaintexts P1 = 10 and P2 = 20.
- Encrypted ciphertexts are : C1 = 37 and C2 = 77.
- If the attacker knows value of public key e = 3 (a common choice), they can calculate potential private keys:
- Private key d1: $C1^{d1} \equiv P1 \pmod{n} \Rightarrow 37^{d1} \equiv 10 \pmod{n}$
- Private key d2: $C2^{d2} \equiv P2 \pmod{n} \Rightarrow 77^{d2} \equiv 20 \pmod{n}$
- In reality, RSA encryption uses padding schemes that complicate this type of attack.
- Cryptanalysis of real-world RSA encryption typically involves far more complex and resource-intensive techniques.

How to make RSA strong?

- Use strong key lengths
- Employ secure padding schemes
- Regularly update cryptographic libraries and algorithms
- Follow best practices in key management and protection
- Actively explore post-quantum cryptographic alternatives.
- Use padding in message text

Key Management

- With public keys, we still have to worry about how the keys are distributed.
- There are a number of approaches:
  - Public announcement
    - Subject to forgery
  - Publicly available directory of public keys
    - Responsibility of some trusted entity or organization (Key Distribution Center)
    - Subject to tampering
  - Public Key Authority
    - Directory with public/private key for PKA
    - Could be a bottleneck
  - Public Key Certificates
    - Removes need to always go through a PKA to get a key
    - The PKA becomes a Certification Authority (CA)
    - CA issues certificate which contains public key and other information for a person or organization.

Diffie-Hellman Key Exchange

- Diffie and Hellman published the first public key algorithm
  - Referred to as ***Diffie-Hellman Key Exchange***
  - Used in a number of commercial products.
  - Oldest public key system still in use
  - Less general than RSA
    - It does neither encryption nor signatures

- Diffie-Hellman allows two individuals to agree on a shared private key, by exchanging public messages.

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Key Exchange

- a public-key distribution scheme
- cannot be used to exchange an arbitrary message
- rather it can establish a common key
- known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Key Exchange

- Diffie-Hellman key exchange depends for its effectiveness on the difficulty of computing ***discrete logarithms:***
  - We define a ***generator*** or ***primitive*** or ***primitive root*** of a prime number *p* as one whose powers generate all the integers from 1 to *p* - 1. So, if *a* is a primitive root of *p*, then the numbers:

    $a \bmod p,\ a^2 \bmod p,\ a^3 \bmod p,\ \ldots\ a^{p-1} \bmod p$

    are distinct and consist of the integers from 1 through *p* - 1 (in some permutation)
  - For any integer *b* and a generator *a* of *p*, we can find a unique exponent *i* such that:

    $$b \equiv a^i \bmod p \quad \text{where } 0 \le i \le (p-1)$$
  - The exponent *i* is referred to as the ***discrete logarithm*** (or ***index***) of *b* for the base *a*, mod *p*. It is the inverse of modular exponentiation, and finding the discrete logarithm *i*, given *b*, is known to be a *hard problem*.

Diffie-Hellman Key Exchange Algorithm

Global Public Elements

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

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

Diffie-Hellman Setup

- all users agree on global parameters:
- large prime integer or polynomial q
- a being a primitive root mod q
- each user (eg. A) generates their key
- chooses a secret key (number): $xA < q$
- compute their public key: $yA = a^{xA} \bmod q$
- each user makes public that key yA

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB}:

  $K_{AB} = a^{x_A \cdot x_B} \bmod q$

  $= y_A^{x_B} \bmod q$ (which **B** can compute)

  $= y_B^{x_A} \bmod q$ (which **A** can compute)

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

Diffie-Hellman Key Exchange Example:

Key exchange is based on the use of the prime number **q=353** and a primitive root of 353, in this case **α=3**. A and B select secret keys **X_A=97**, **X_B=233** respectively. Each computes its public key:

$$\text{A computes } Y_A = 3^{97} \bmod 353 = 40.$$
$$\text{B computes } Y_B = 3^{233} \bmod 353 = 248.$$

After they exchange public keys, each can compute the common secret key:

$$\text{A computes } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160.$$
$$\text{B computes } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160.$$

Attacker Example:

We assume an attacker would have available the following information:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

In this simple example, it would be possible by brute force to determine the secret key **160**. In particular, an attacker **E** can determine the common key by discovering a solution to the equation **$3^a \bmod 353 = 40$** or the equation **$x^b \bmod 353 = 248$**. The brute-force approach is to calculate **powers of 3 modulo 353**, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provides **$3^{97} \bmod 353 = 40$**. With larger numbers, the problem becomes impractical.

Key Exchange Protocol

- users could create random private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a Man-in-the-Middle Attack
- authentication of the keys is needed

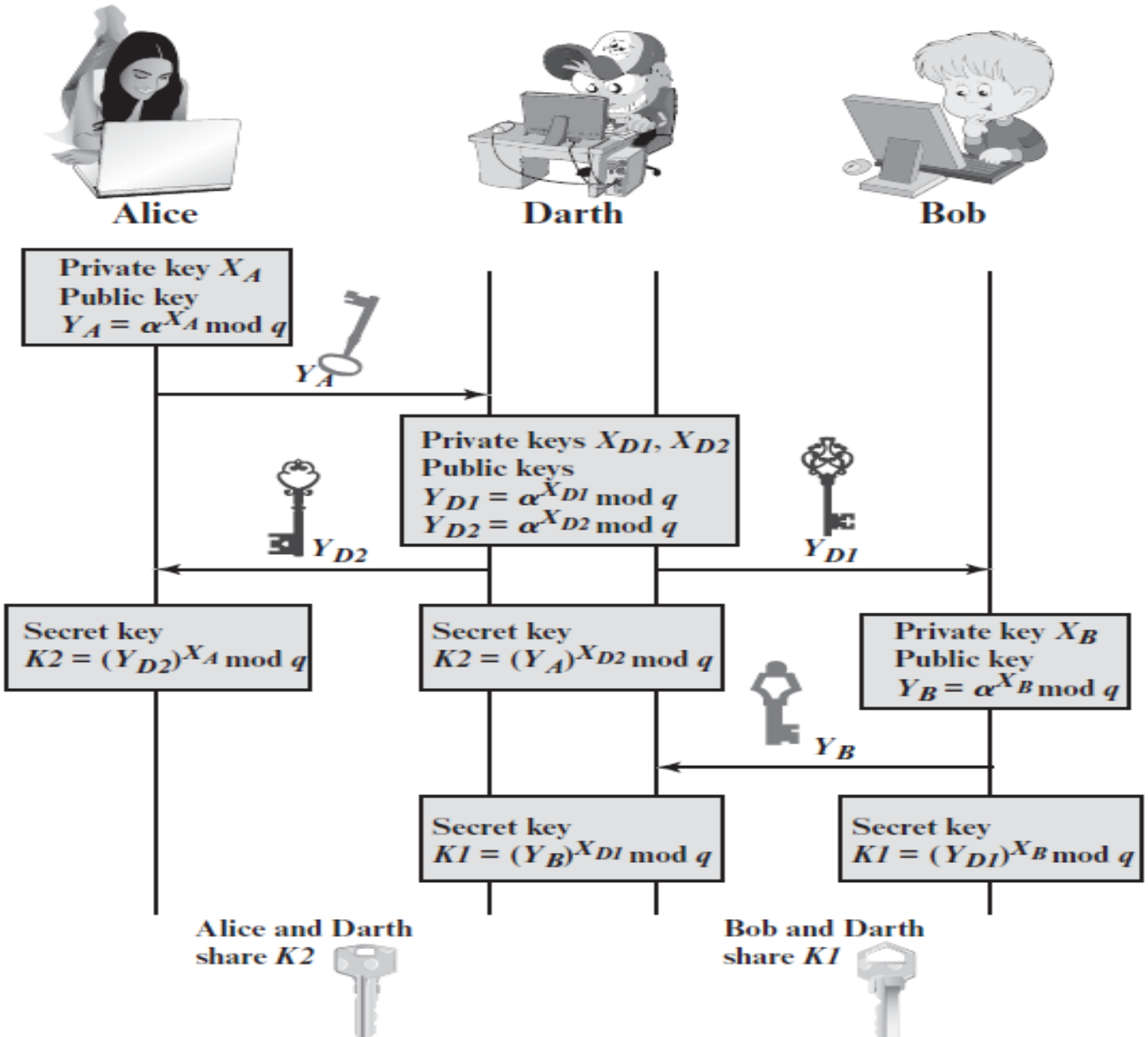
Man-in-the-Middle Attack

- One problem with Diffie-Hellman is that there is no authentication, and so the protocol is subject to a ***man-in-the-middle attack***:
  - Alice generates g^x and sends it to "Bob"
  - Eve intercepts the message:
    - Generates g^v, and sends it to Bob in place of Alice's message
  - Bob receives g^v, generates g^y, and sends it to "Alice"
  - Eve intercepts the message:
    - Generates g^w, and sends it to Alice in place of Bob's message
  - Alice computes $k = (g^w)^x$
  - Bob computes $k' = (g^v)^y$
  - Eve computes $k = (g^x)^w$ and $k' = (g^y)^v$

Man-in-the-Middle Attack

- There are a number of techniques to defend against such an attack:
  - Each person can have a "somewhat permanent" public and secret number, instead of creating one for each message exchange. This can be considered to be a kind of *Digital Phonebook*.
  - If Alice and Bob share some kind of secret which then can use to authenticate each other, then they can use this secret to verify each other's messages indeed came from the person they expected.

Man-in-the-Middle Attack



Diffie-Hellman and Safe Primes

- Diffie-Hellman works with any prime *p* and any number *g*
- However, it it less secure if *p* and *g* don't have additional mathematical properties
  - It turns out that things work better if (*p* – 1)/2 is also a prime.
    - Such a prime is called a ***Safe Prime***
  - It's also better if :

    $g \neq -1 \bmod p$, for which $g^{(p-1)/2} = -1 \bmod p$

    (true for almost half of all mod p numbers)