Professional Masters in Information and Cyber Security (PMICS)

Computer Science and Engineering,
University of Dhaka

CSE 802

Information Security Fundamentals

Content of the Lecture

- Basic concepts of Cryptography
- Symmetric key encryption
  - Different types of classical encryption
  - Random and pseudo-random number
  - Stream and Block cipher
  - Block cipher mode of operation
- Asymmetric key encryption
  - RSA algorithm
  - Diffie-Hellman Key exchange
- DNSSEC

Cryptography

- "Cryptography" comes from the Greek words ***kryptos,*** meaning "concealed, hidden, veiled, secret, or mysterious," and ***graphia,*** meaning "writing"; thus, cryptography is "the art of secret writing.“

- The practice of concealing information by converting plaintext (readable format) into ciphertext (unreadable format) using a key or encryption scheme.

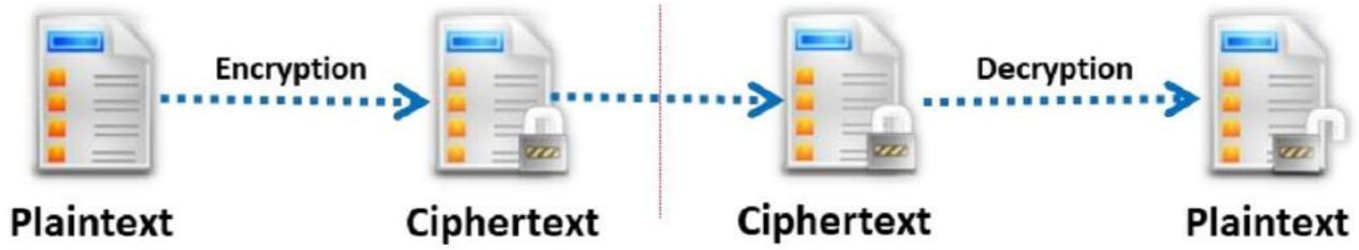
- Protects confidential data such as email messages, chat sessions, web transactions, personal data, corporate data, e-commerce applications, and many other types of communication.

Objective of Cryptography

- **Confidentiality:** Assurance that the information is accessible only to those authorized to access it.
- **Integrity:** Trustworthiness of data or resources in terms of preventing improper and unauthorized changes.
- **Authentication:** Assurance that the communication, document, or data is genuine.
- **Nonrepudiation:** Guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Cryptography

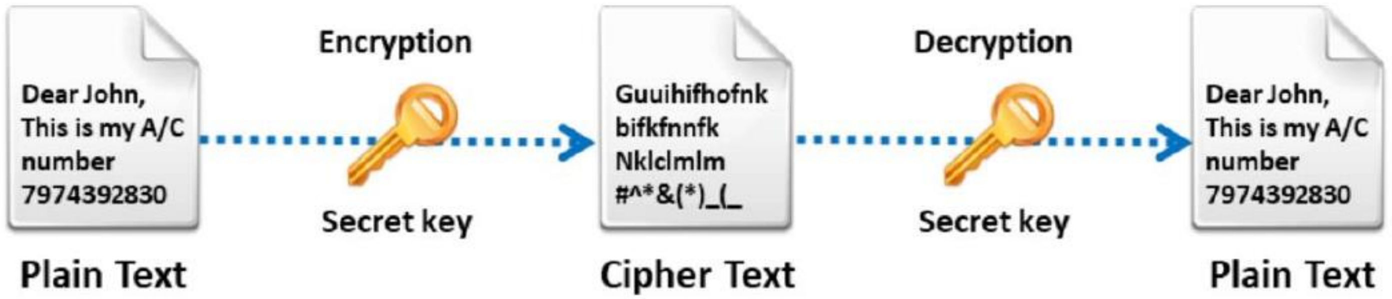
- Cryptography process



- Types of Cryptography
  - Symmetric key encryption
  - Asymmetric key encryption

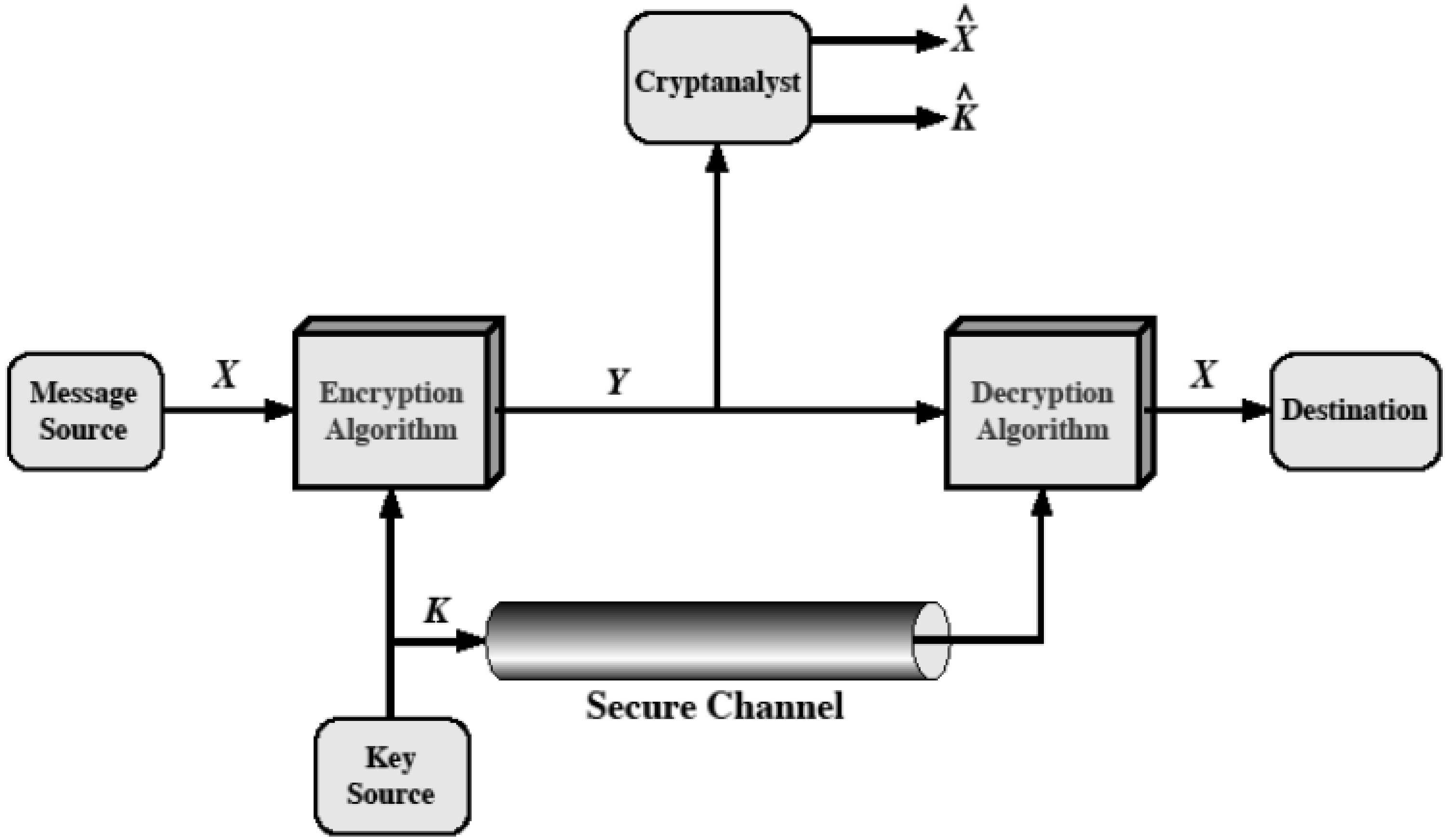
Symmetric Key Encryption

- Both sender and receiver use the same key to encrypt and decrypt data.



- Needs a mechanism to exchange key.

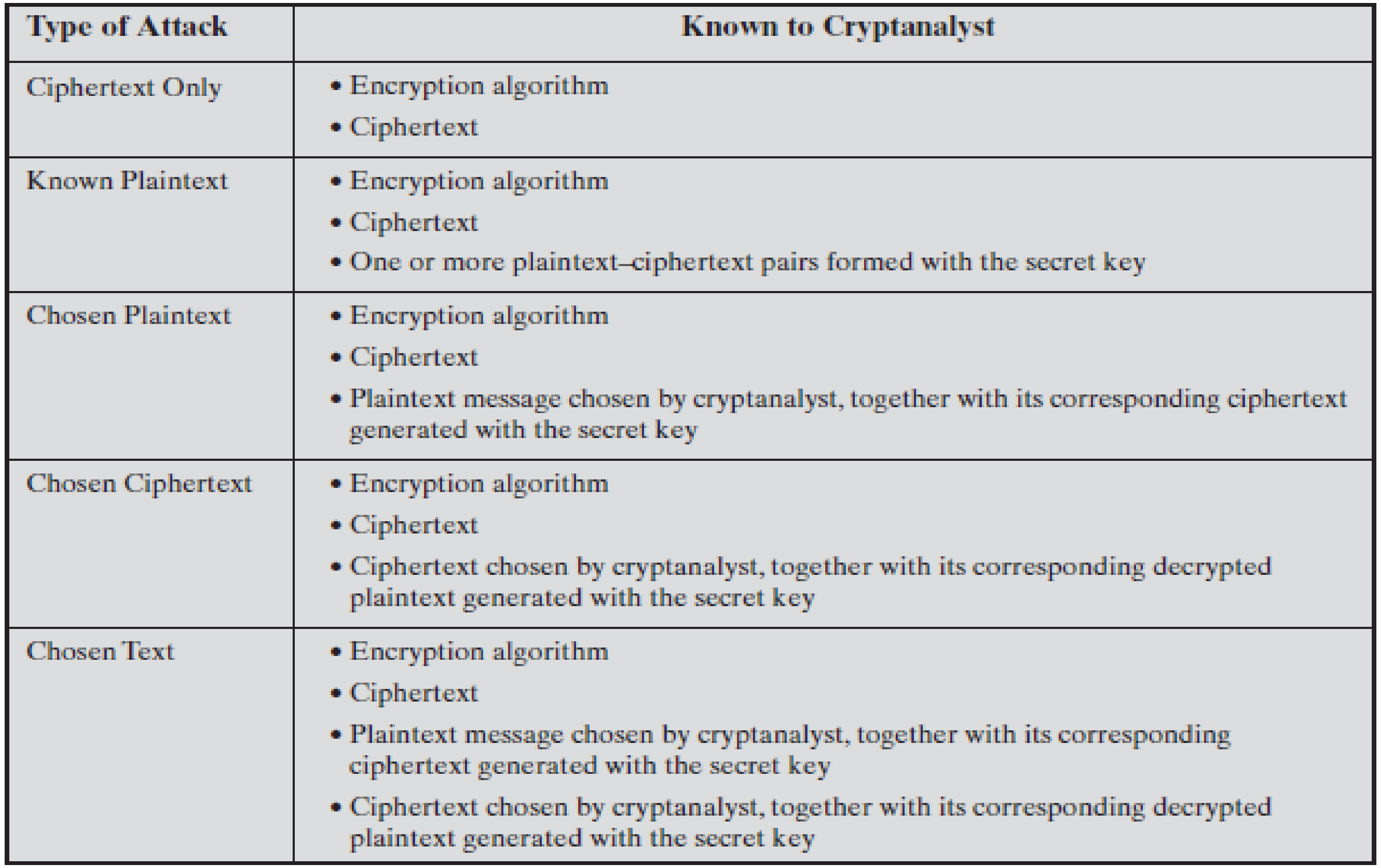
Model of Symmetric Cryptosystem



Attacks on Conventional Encryption Scheme

- **Cryptanalysis**
  - Exploits the nature of algorithm based on some knowledge of plaintext or some plaintext-ciphertext pair.
- **Brute-force Attack**
  - Tries every possible keys on a piece of ciphertext.
  - On average, half of all possible keys must be tried.

Types of Cryptanalytic Attacks



Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

More Definitions

- **Unconditional security**

  -- An encryption scheme is unconditionally secure if the ciphertext provides ***insufficient information*** to uniquely determine the corresponding plaintext.

  -- no matter how much computer power or time is available, the cipher cannot be broken.

- **Computational security**

  -- An algorithm needs to meet one or both criteria:

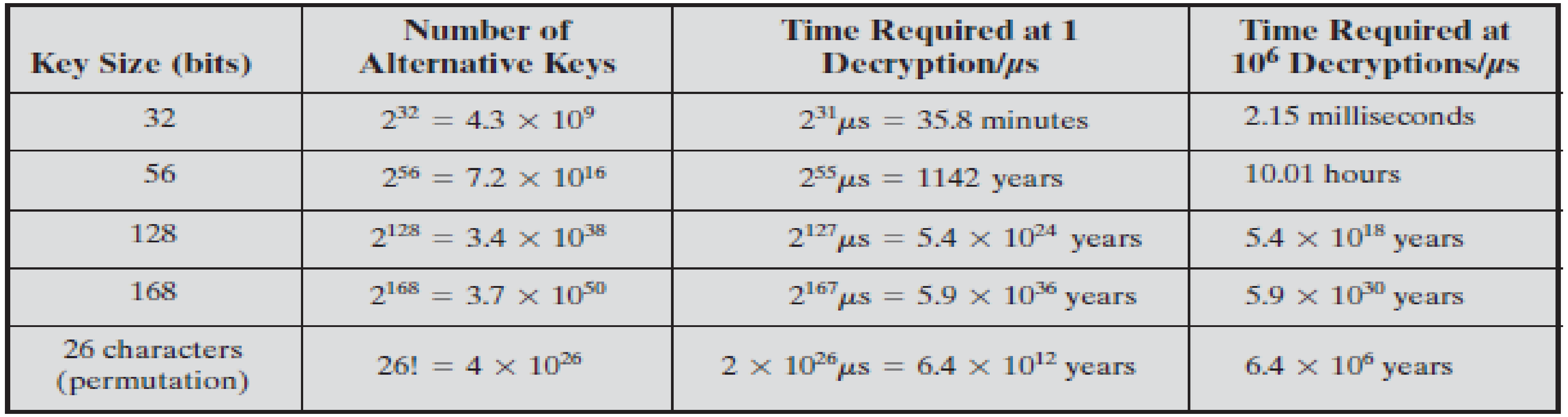
  1. The cost of breaking the cipher exceeds the value of the encrypted information.
  2. The time required to break the cipher exceeds the useful lifetime of the information

--- given limited computing resources the cipher cannot be broken.

Brute-Force Attack

- always possible to simply try every key
- most basic attack, proportional to key size
- assume able to know / recognize plaintext

Table 2.2 Average Time Required for Exhaustive Key Search



Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

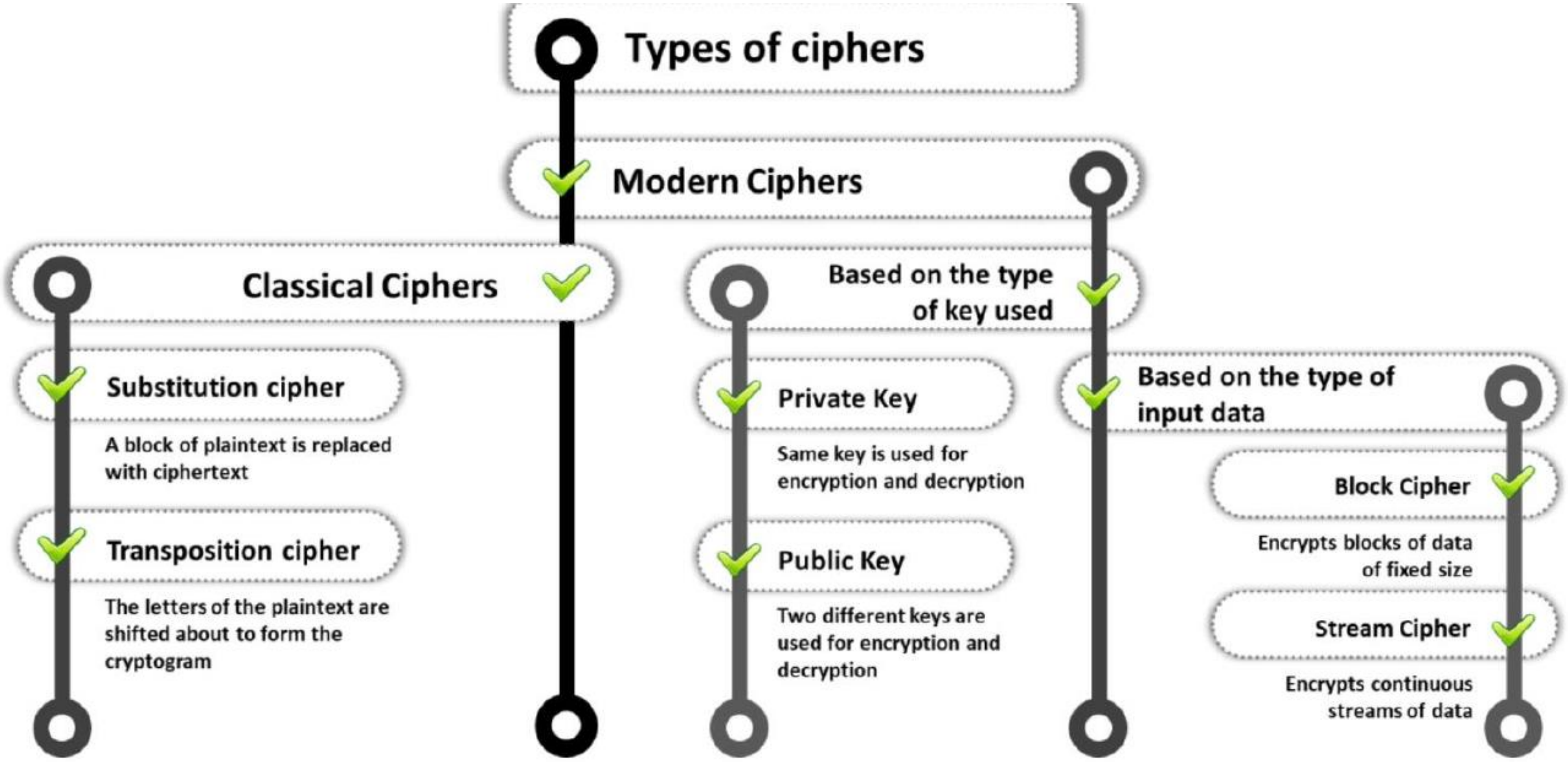
Encryption Algorithm

- Encryption is the process of converting readable plaintext into an unreadable ciphertext using a set of complex algorithms that transform the data into blocks or streams of random alphanumeric characters.
- Cipher:
  - a cipher is an algorithm (a series of well-defined steps) for performing encryption and decryption.
- Encipherment:
  - Convert plaintext to ciphertext.
- Decipherment:
  - Convert ciphertext to plaintext.

- Cipher may be open-source or closed-source (the process is developed for use in specific domains, such as the military).

- Furthermore, ciphers may be free for public use or licensed.

Types of Cipher


Types of ciphers
Modern Ciphers
Classical Ciphers
Substitution cipher
A block of plaintext is replaced with ciphertext
Transposition cipher
The letters of the plaintext are shifted about to form the cryptogram
Based on the type of key used
Private Key
Same key is used for encryption and decryption
Public Key
Two different keys are used for encryption and decryption
Based on the type of input data
Block Cipher
Encrypts blocks of data of fixed size
Stream Cipher
Encrypts continuous streams of data

Classical Cipher

- Most basic type of ciphers.
- Operate on letters of the alphabet (A-Z).
- Because these ciphers are easily deciphered, they are generally unreliable.
- **Types of classical ciphers**
  - **Substitution cipher:**
  - The user replaces units of plaintext with ciphertext according to a regular system.
  - The units may be single letters, pairs of letters, or combinations of them, and so on.
  - The recipient performs inverse substitution to decipher the text. Examples include the Ceaser cipher, Vigenere cipher, Play fair cipher, and Hill cipher.
  - For example, **"HELLO WORLD"** can be encrypted as **"PSTER HGFST"** (i.e., H=P, E=S, etc.).

Classical Cipher

- **Transposition cipher:**
  - letters in the plaintext are rearranged according to a regular system to produce the ciphertext.
  - For example, **''CRYPTOGRAPHY''** when encrypted becomes **''AOYCRGPTYRHP."**
  - Examples include the rail fence cipher.

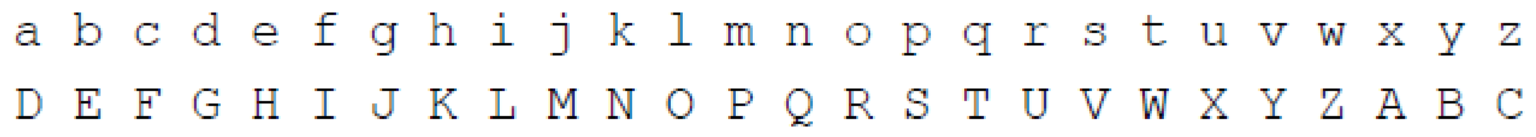
Caesar Cipher

- earliest known substitution cipher
- first attested use in military affairs
- replaces each letter by 3rd letter
- example:

```
plain:  meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

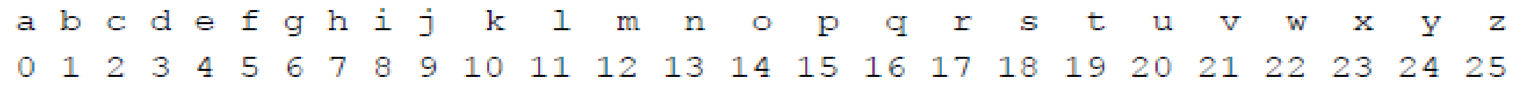
Caesar Cipher

Can define transformation as:



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
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

- mathematically give each letter a number
-



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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

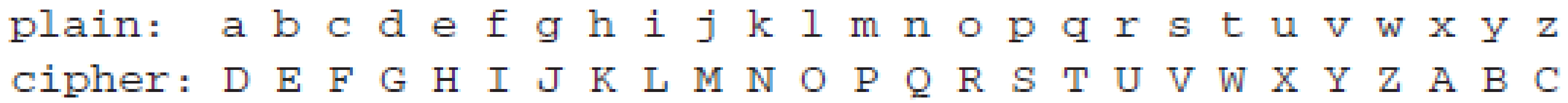
$$c = \mathrm{E}(\mathrm{k}, p) = (p + k) \bmod (26)$$
$$p = \mathrm{D}(\mathrm{k}, \mathrm{c}) = (\mathrm{c} - k) \bmod (26)$$

Brute-Force Attack on Caesar Cipher

- Brute-force attack on Caesar cipher is possible:
  - The encryption and decryption algorithm are known.
  - For each ciphertext component, only 25 keys to try.
  - The language of plaintext is known and easily recognizable.

Monoalphabetic Cipher

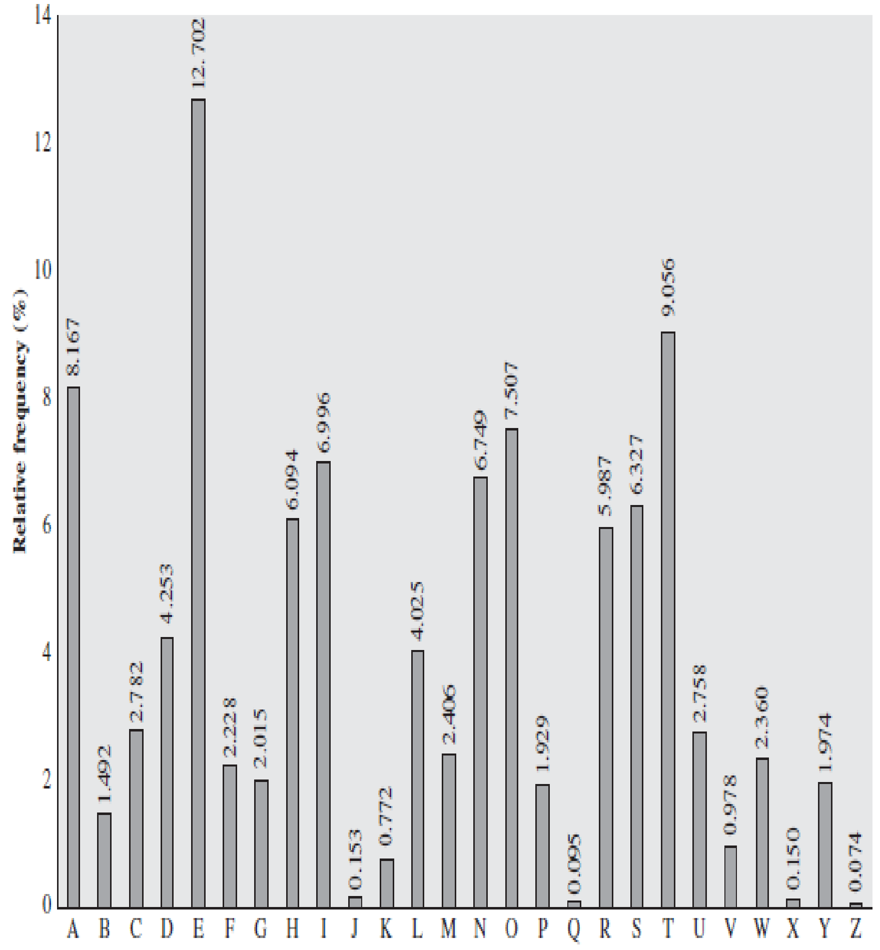
- A single plain alphabet is mapped to an individual cipher alphabet (per message).
- rather than just shifting the alphabet could shuffle (jumble) the letters arbitrarily



plain:	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
cipher:	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

- now have a total of 26! = 4 x 10^{26} keys
- with so many keys, might think is secure
- but would be **!!WRONG!!**
- problem is language characteristics

Language Redundancy and Cryptanalysis



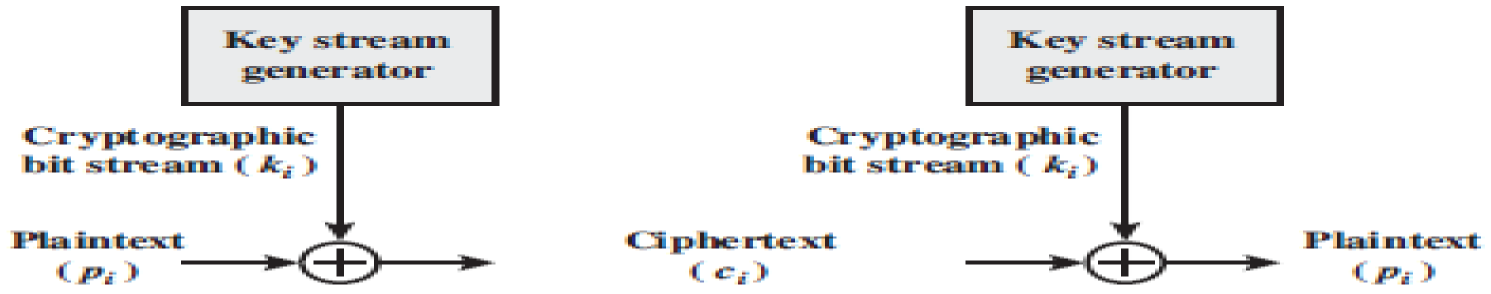
- key concept - monoalphabetic substitution ciphers do not change relative letter frequencies.

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

Vernam Cipher

- Invented by AT&T engineer Gilbert Vernam in 1918



- Equation for encryption: $c_i = p_i \oplus k_i$

- Equation for decryption: $p_i = c_i \oplus k_i$

One-Time Pad

- A truly random key as long as the message is used, that makes the cipher unbreakable.
- The key is used only once.
- for **any plaintext** & **any ciphertext** there exists a key mapping one to other
- is unbreakable since ciphertext bears no statistical relationship to the plaintext
- problems in generation & safe distribution of key

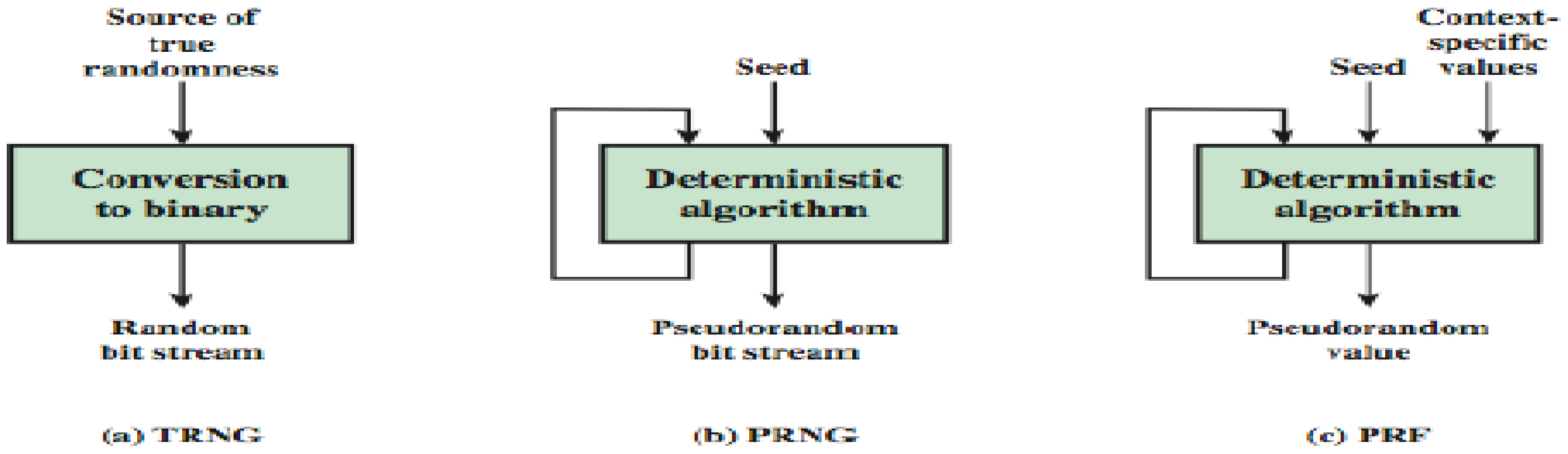
Random Number

- Criteria of Random Number:
  - Uniform Distribution.
  - Independence
  - Unpredictability

- Many uses of **random numbers** in cryptography
  - Nonces in authentication protocols to prevent replay attack.
  - session keys
  - public key generation
  - keystream for a one-time pad
- Care should be taken during the generation of pseudo random number.

(Pseudo) Random Number Generators

- Often use deterministic algorithmic techniques to create "random numbers"
  - not truly random
  - can pass many tests of "randomness"
- known as "pseudorandom numbers"
- created by "Pseudorandom Number Generators (PRNGs)"

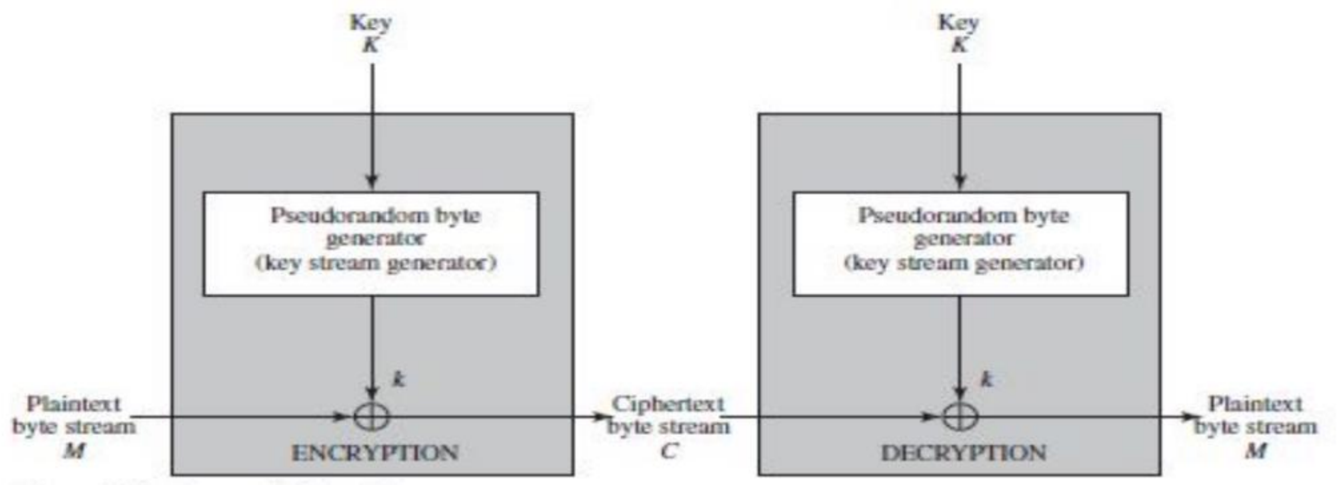


(Pseudo) Random Number Generators

- **Entropy Source:**
  - Source of true randomness.
  - Drawn from the physical environment of the computer such as keystroke timing patterns, disk electrical activity, mouse movements and instantaneous values of the system clock.
- **Seed:** Often generated by TRNG.
- If the adversary knows the algorithm and the seed, then s/he can reproduce the entire bit stream.

Stream Cipher

- process message bit by bit (as a stream)
- have a pseudo random keystream
- combined (XOR) with plaintext bit by bit $C_i = M_i$ XOR StreamKey_i



Stream Cipher

- Never reuse stream key
  - otherwise can recover messages (via XOR of msgs)
- Stream cipher can be used in data communication channel or a browser/web link.

Stream Cipher - RC4

- a proprietary cipher owned by RSA security.
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP/WPA)

RC4 Key Schedule

- Starts with an array **S** of numbers: 0....255
- Use key to well and truly shuffle **S**
- **S** forms **internal state** of the cipher

```
for i = 0 to 255 do
    S[i] = i
    T[i] = K[i mod keylen]
j = 0
for i = 0 to 255 do
    j = (j + S[i] + T[i]) (mod 256)
    swap (S[i], S[j])
```

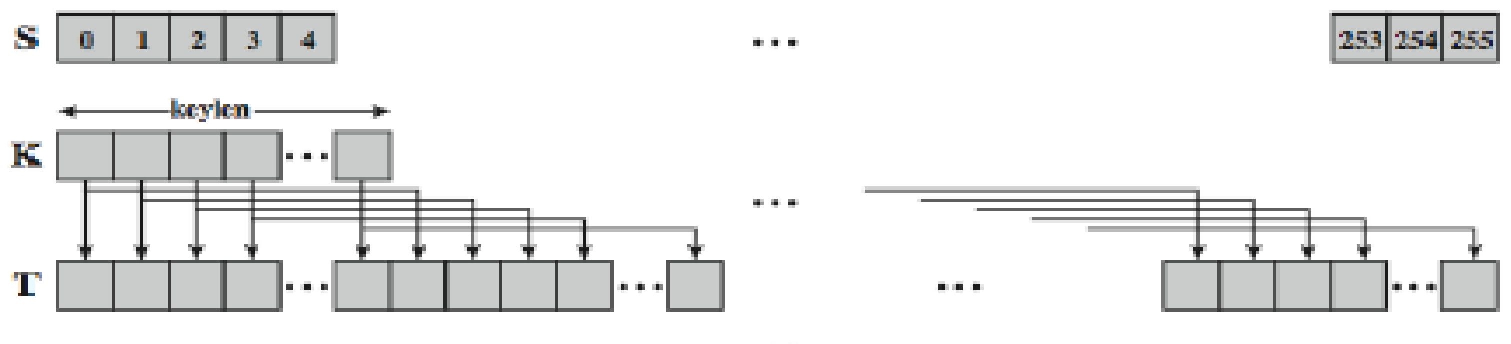
RC4 Encryption

- Encryption continues shuffling array values
- Sum of shuffled pair selects "stream key" value from permutation
- XOR **S[t]** with next byte of message to en/decrypt

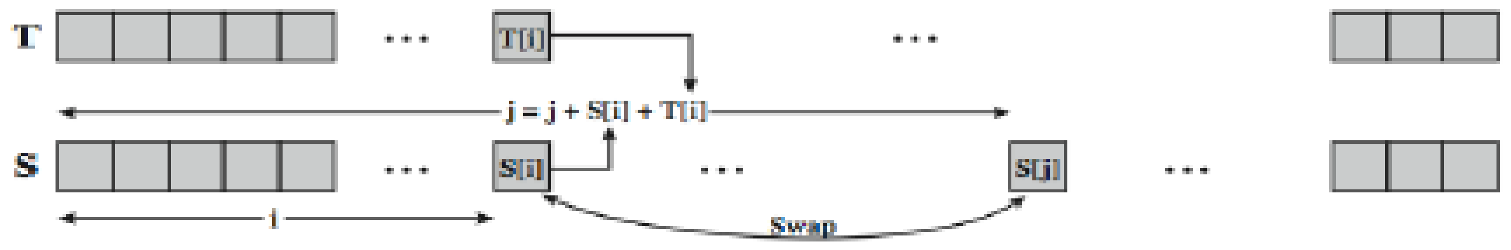
```
/* Stream Generation */
i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
```

- used in the WiFi Protected Access (WPA) .It is optional for use in Secure Shell (SSH) and Kerberos.

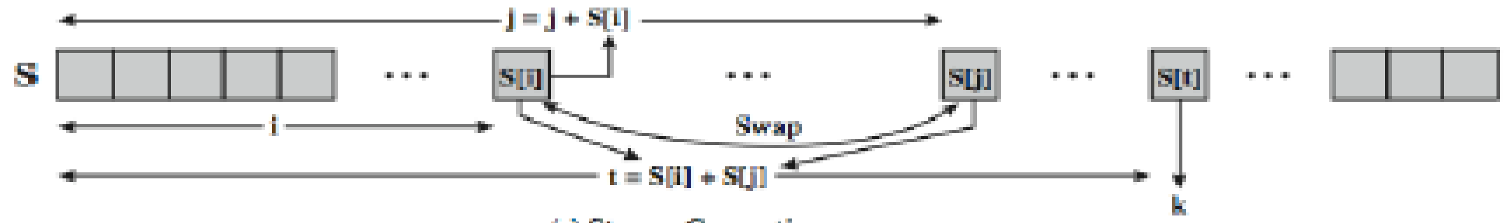
RC4 Overview



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream Generation

Block Cipher

- Data is encrypted in terms of a fixed size block.
- Data is divided into a number of block and each block is encrypted with a fixed size key.
- Last block may contain less data than a fixed size block. In this case, bytes are padded to this block to increase its size and match with the block size.
- AES and DES are used to encrypt individual block.

Data Encryption Standard (DES)

- is a symmetric key encryption system.
- Encrypts 64 bits data using 56 bits key.
- **Triple DES (3DES):**
- Eventually, it became obvious that DES would no longer be secure. The U.S. Federal Government began a contest seeking a replacement cryptography algorithm.
- In the meantime, 3DES was created as an interim solution. Essentially, it performs DES three times with three different keys.

•3DES uses a "key bundle" that comprises three DES keys, Kl, K2, and K3. Each key is a standard 56-bit DES key

Triple DES (3DES)

- DES *encrypt* with Kl, DES *decrypt* with K2, DES *encrypt* with K3
- There are three options for the keys.
    - All three keys are independent and different.
    - Kl and K3 are identical.
    - All three keys are the same;
    - The first option is the most secure, while the third is the least secure.

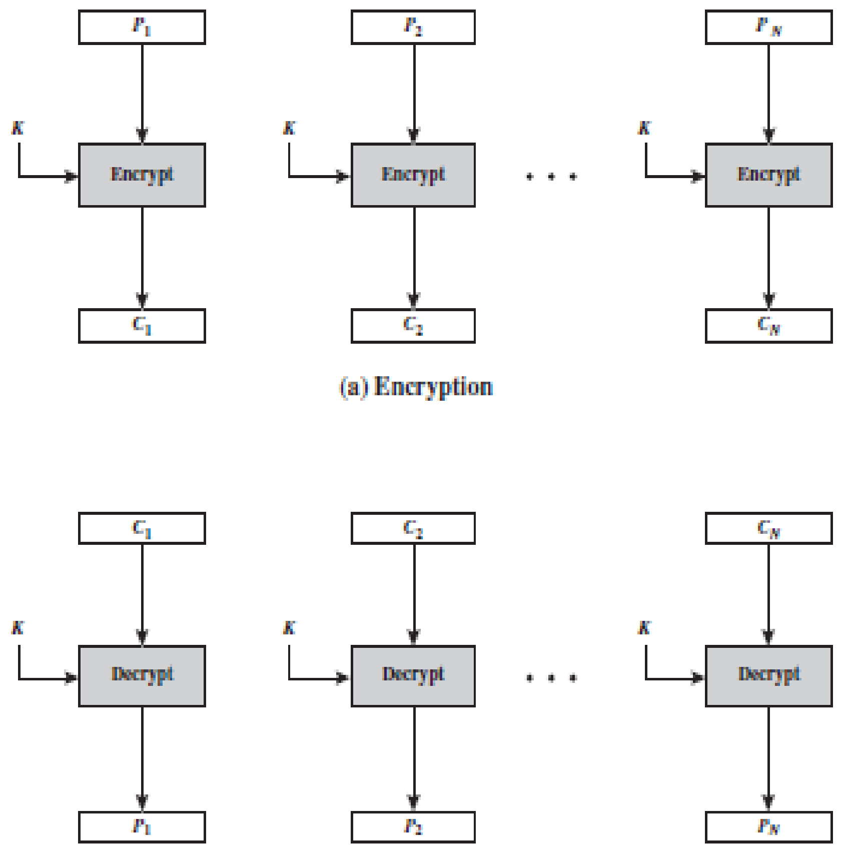
Advanced Encryption Scheme (AES)

- National Institute of Standards and Technology (NIST) specification for the encryption of electronic data.
- Helps to encrypt digital information such as telecommunications, financial, and government data.
- US government agencies have been using it to secure sensitive but unclassified material.
- AES is a symmetric-key algorithm: both encryption and decryption are performed using the same key.
- It has a 128-bit block size, with key sizes of 128, 192, and 256 bits for AES-128, AES-192, and AES-256, respectively.
- The design of AES makes its use efficient in both software and hardware.

Block Cipher Operation

- A block cipher takes a fixed-length block of text of length ***b*** and a key as input and produces a ***b***-bit block of ciphertext.
- If the amount of plaintext is greater than ***b*** bits, the plaintext is divided into blocks of ***b*** bits.
- When multiple blocks of plaintext is encrypted with same key, a number of security issues arises.
- To apply a block cipher in a variety of applications, five modes of operation have been defined:
    - Electronic Code Book
    - Cipher Block Chaining
    - Cipher Feedback
    - Output Feedback
    - Counter (CTR)

Electronic Code Book (ECB)

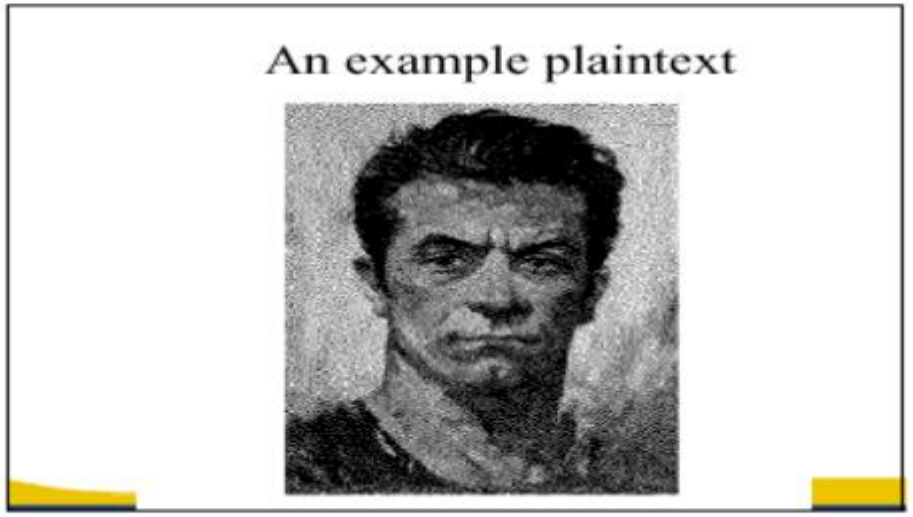


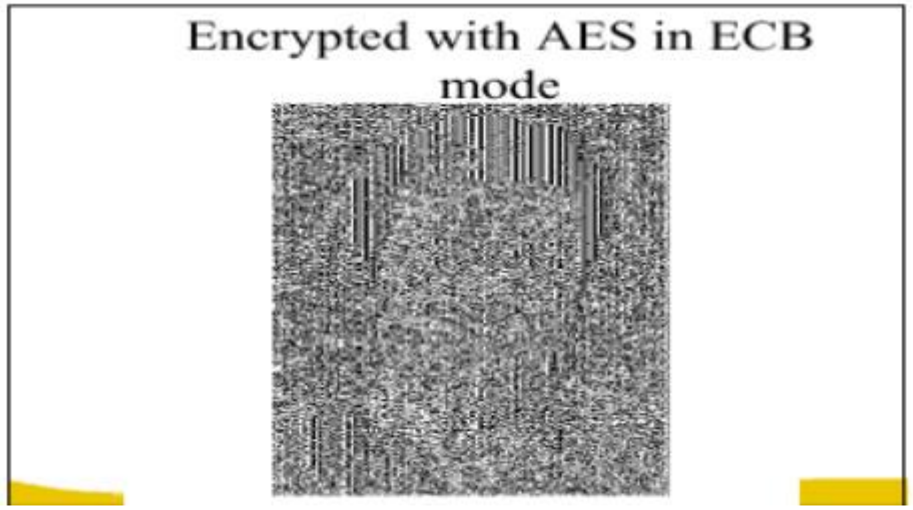
•Ideal for short amount of data, such as to transmit AES and DES key.

•If same b-bit blocks of plaintext appears more than once in the message, it always produce the same ciphertext.

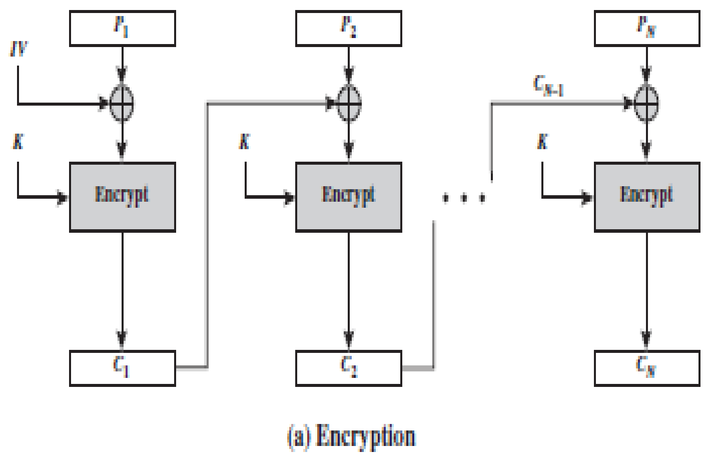
•For lengthy messages, if the message is highly structured, it may be possible to exploit these regularities.

Electronic Code Book (ECB)

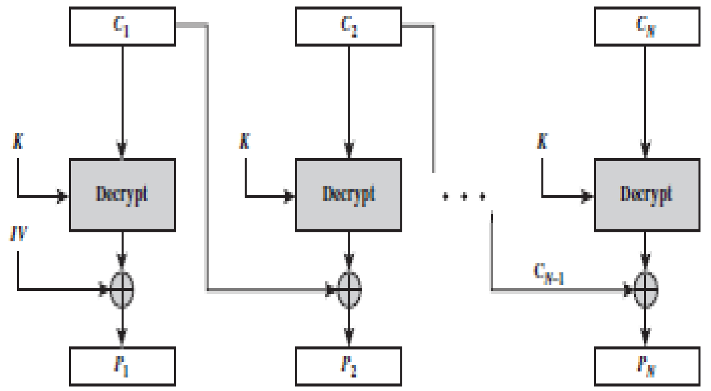




Cipher Block Chaining Mode



(a) Encryption



(b) Decryption

Figure 6.4 Cipher Block Chaining (CFB) Mode

- Overcome the security deficiencies of ECB.

$$C_j = E(K, [C_{j-1} \oplus P_j])$$

$$D(K, C_j) = D(K, E(K, [C_{j-1} \oplus P_j]))$$
$$D(K, C_j) = C_{j-1} \oplus P_j$$
$$C_{j-1} \oplus D(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

- IV must be protected against unauthorized changes. This could be done by sending IV using ECB encryption.

Cipher Block Chaining Mode

- IV can be generated in two ways:
  - Nonce: unique value unique for each execution of encryption algorithm. May be a counter, a timestamp or a message number.
  - Generate a random block using a random number generator.
- Applications:
  - Suitable for encrypting messages of length greater than b bits.
  - Used for achieving confidentiality and authentication.

Cipher Feedback Mode

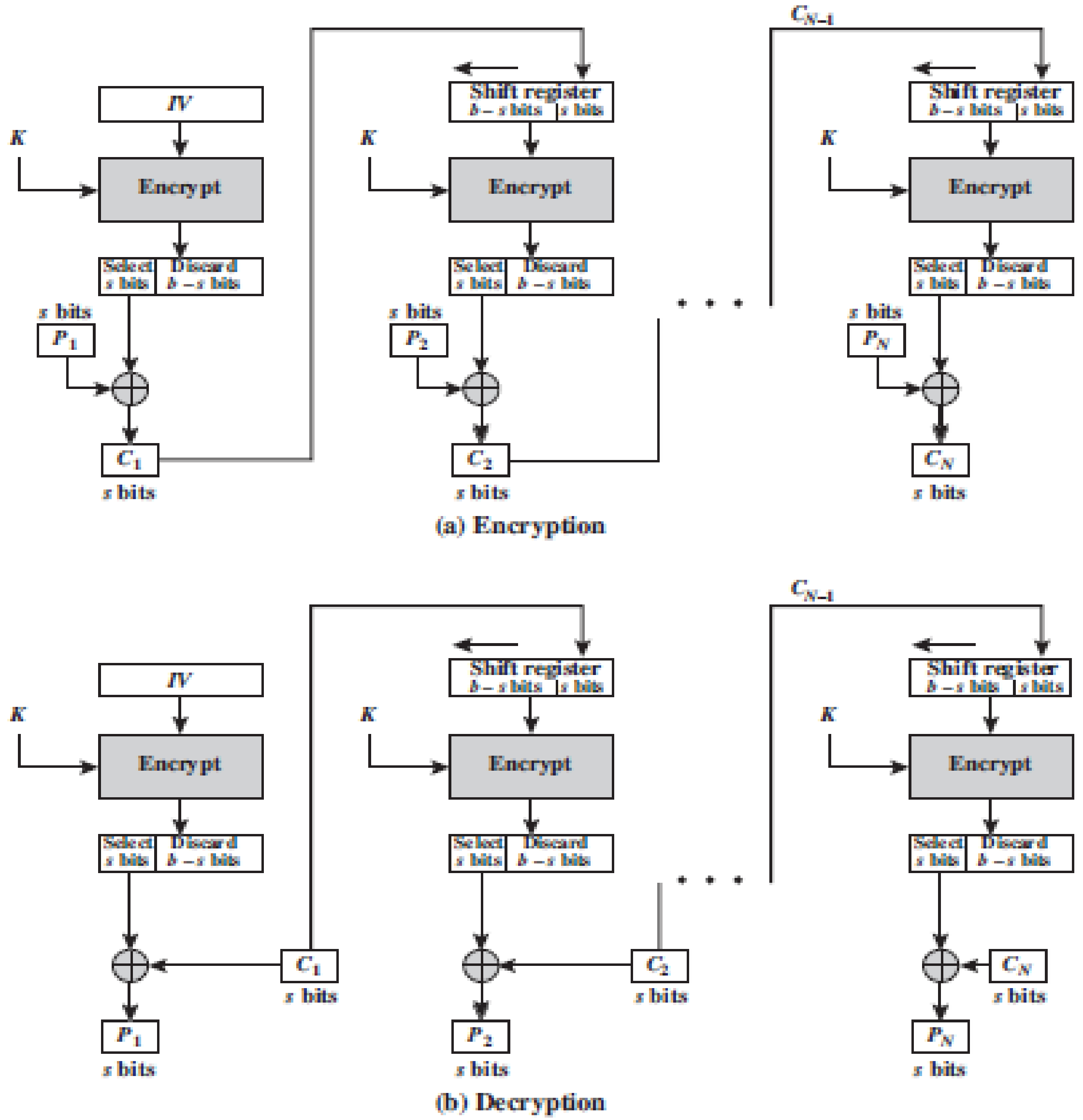


Figure 6.5 s-bit Cipher Feedback (CFB) Mode

- message is treated as a stream of bits.
  - Eliminates the need to pad a message to be an integral number of blocks.
  - If a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

Cipher Feedback Mode

- In encryption, each forward cipher function depends on the result of the previous forward cipher function. Thus multiple forward cipher operations cannot be performed in parallel.

- In decryption, the forward cipher functions can be performed in parallel if the input blocks are first constructed from the IV and ciphertext.

Output Feedback Mode

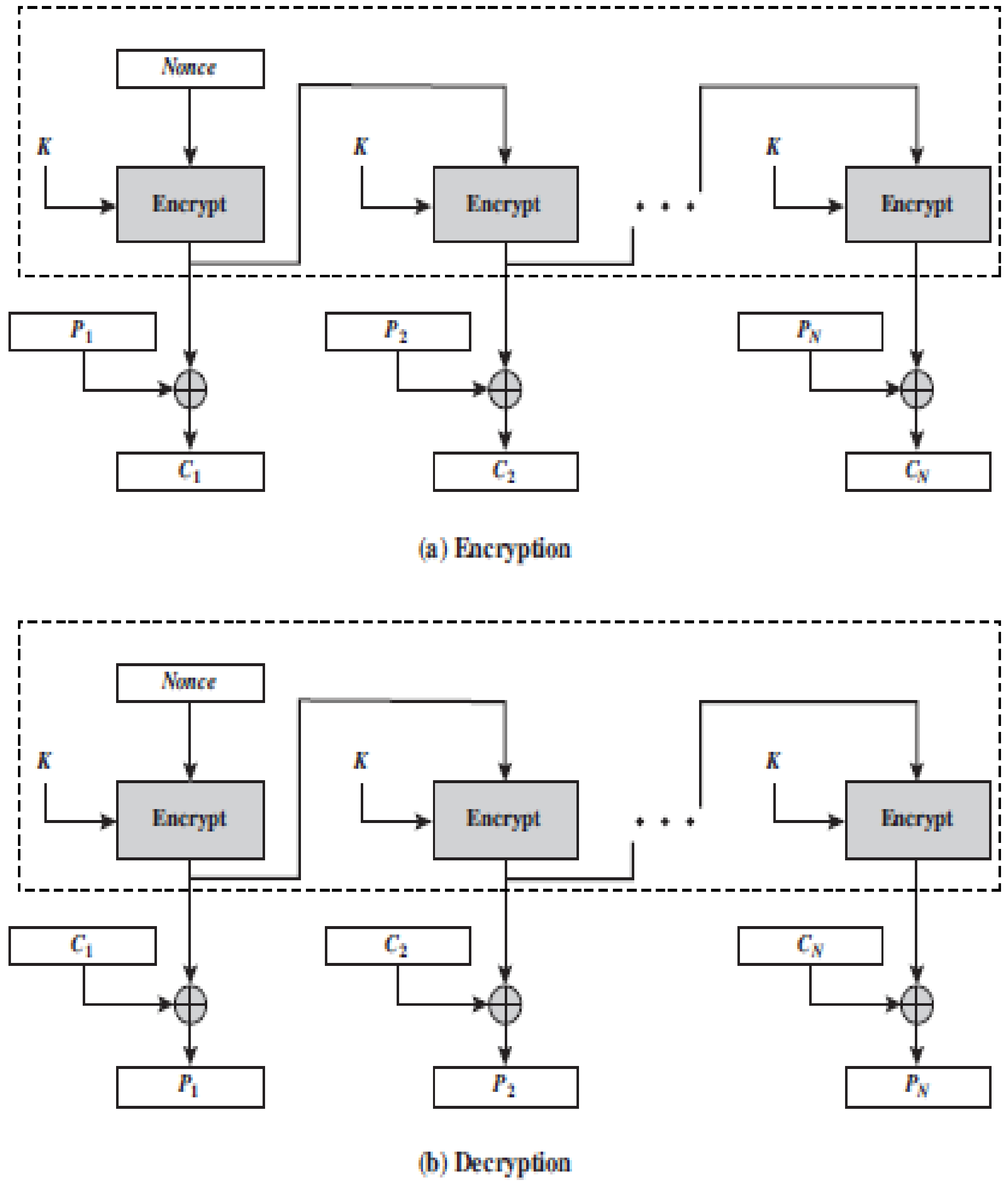


Figure 6.6 Output Feedback (OFB) Mode

- $C_j = P_j \text{ XOR } E(K, O_{j-1})$

  $O_{j-1} = E(K, O_{j-2})$
- $C_j = P_j \text{ XOR } E(K, C_{j-1} \text{ XOR } P_{j-1})$

  $P_j = C_j \text{ XOR } E(K, C_{j-1} \text{ XOR } P_{j-1})$

- If the size of the last block is ***u<b*** bits, the most significant ***u*** bits of the last output block $\boldsymbol{O_N}$ are used for XOR operation. Remaining bits of the last output block are discarded.

- In OFB, the IV must be a nonce which is unique for each execution of the encryption operation. Otherwise, identical plaintext at identical position of two different message will produce same ciphertext.

- In OFB, bit errors in transmission do not propagate. If bit error occurs in $\boldsymbol{C_1}$, the recovered value of $\boldsymbol{P_1}$ is affected.

Counter Mode

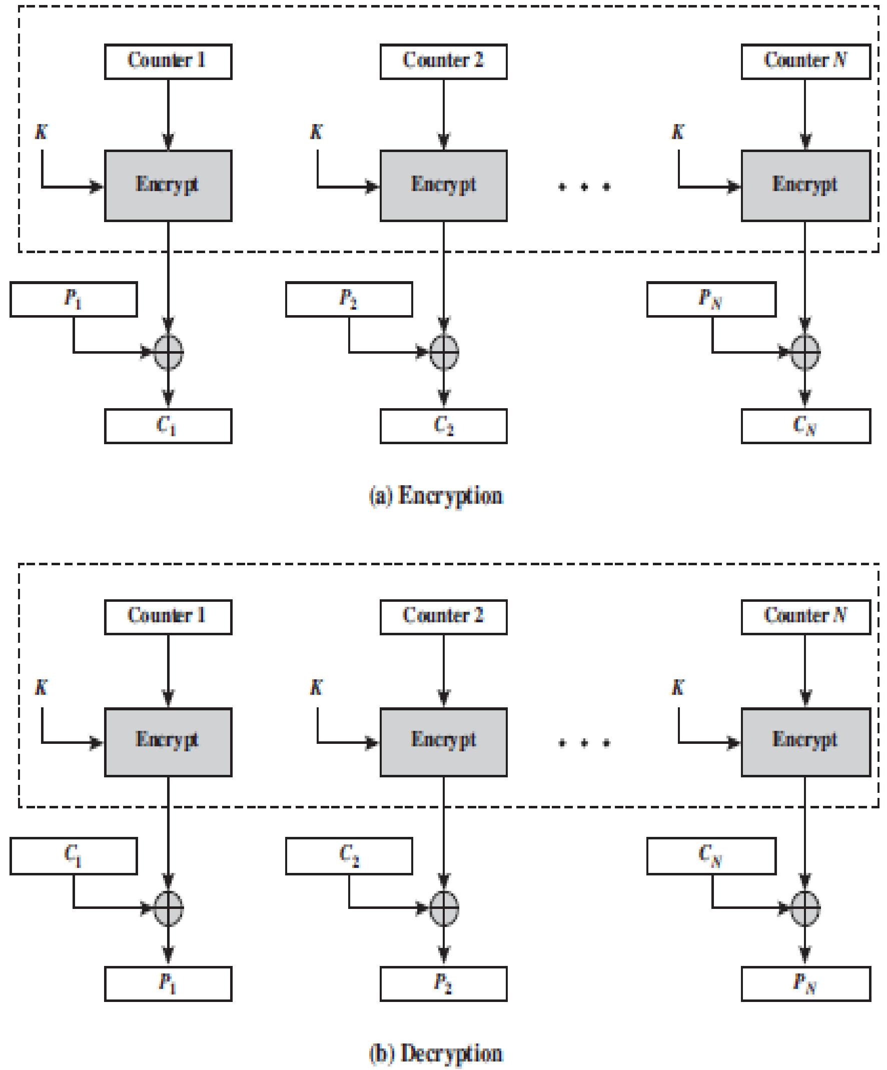
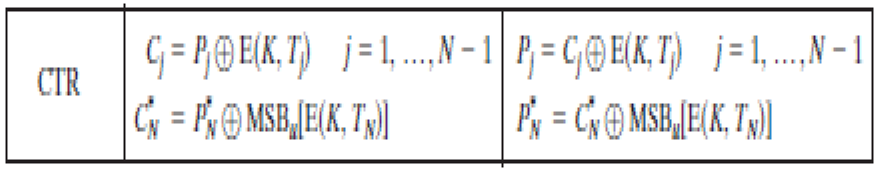


Figure 6.7 Counter (CTR) Mode



CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$

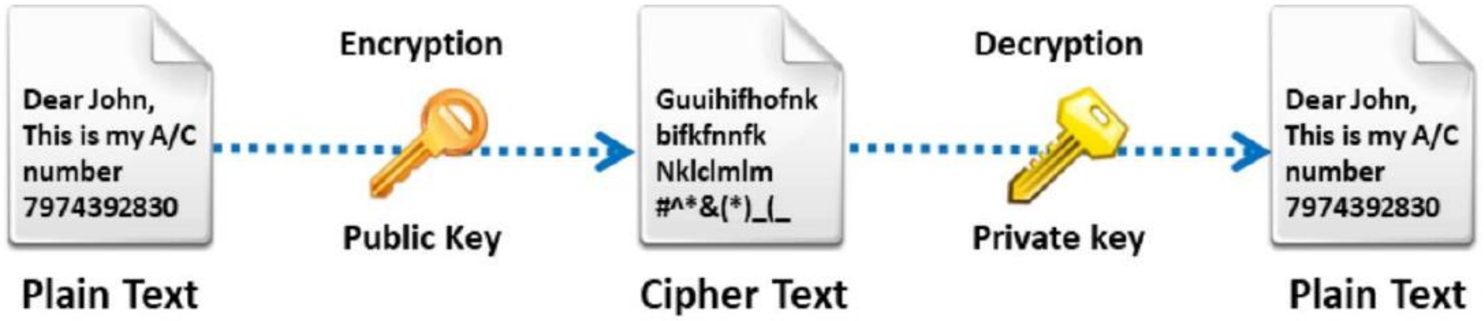
- The counter value must be a nonce. That is, all $\boldsymbol{T_i}$ across all messages must be unique.
- If a plaintext block is encrypted using a given counter value is known, then the output of the encryption block is known from the associated ciphertext. This output allows other plaintext blocks that are encrypted using the same counter value to be easily recovered from the associated ciphertext blocks.

Counter Mode

- Encryption and decryption can be performed in parallel.
- Preprocessing of the encryption algorithm are possible.
- Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm.

Asymmetric Key Encryption/Public Key Encryption

- Developed to solve key management problem.
- an encryption method that uses a key pair comprising a public key available to anyone and a private key held only by the key owner.



Asymmetric Key Encryption/ Public Key Encryption

- Asymmetric encryption uses the following sequence to send a message:

  -- An individual finds the public key of the person he or she wants

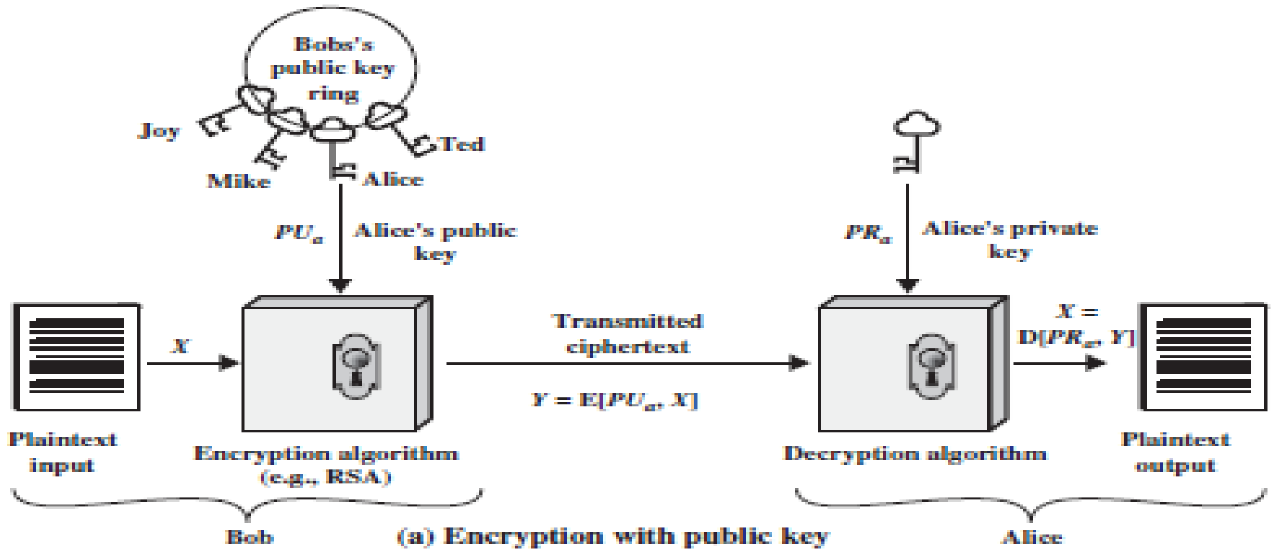
  to contact in a directory.

  --This public key is used to encrypt a message that is then sent to the

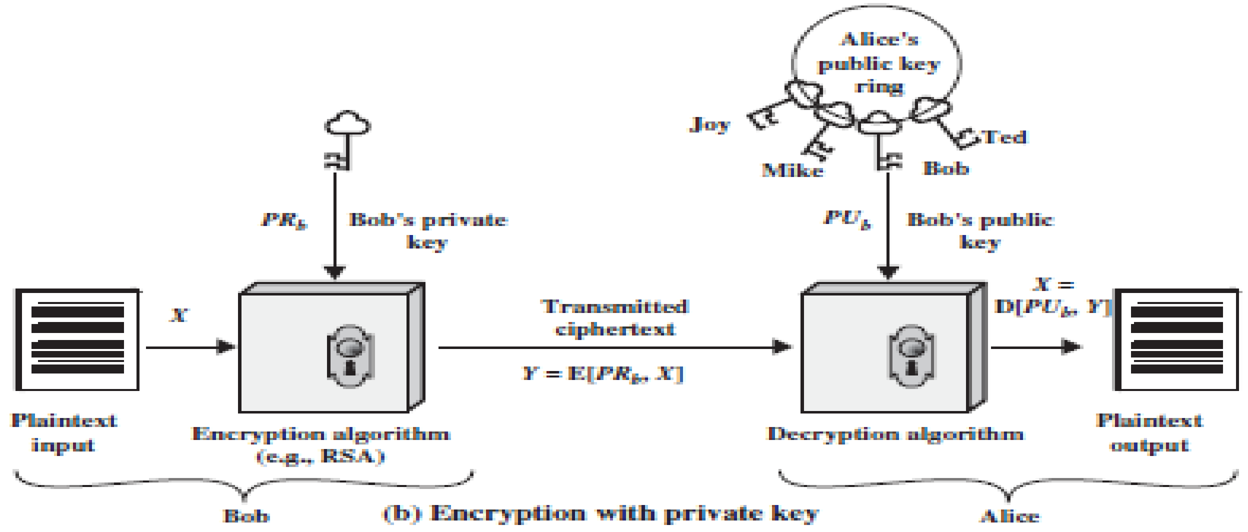
  intended recipient.

  --The receiver uses the private key to decrypt the message and reads it.

Public-Key Crypto Systems



(a) Encryption with public key

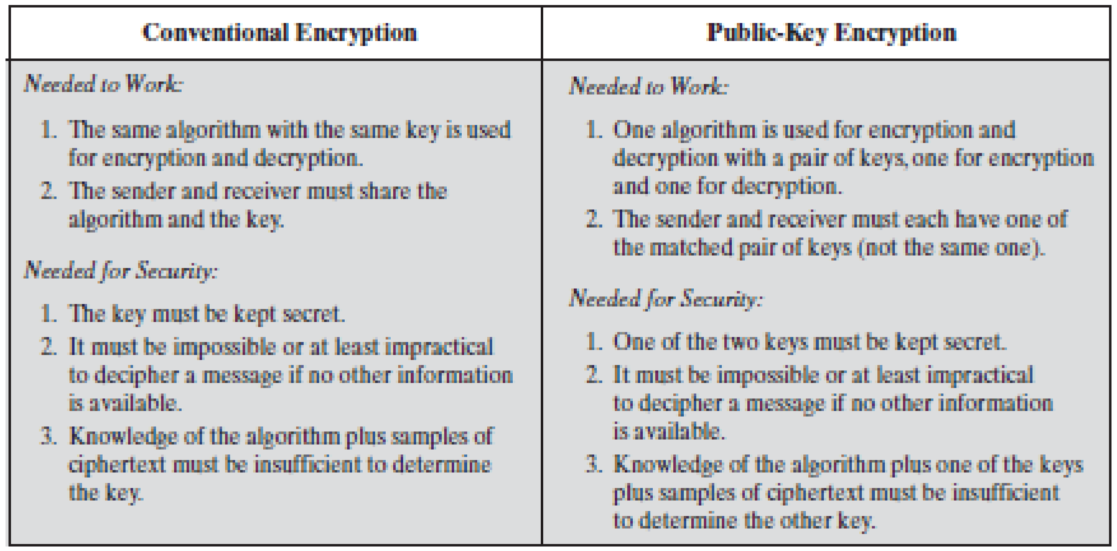


(b) Encryption with private key

Public-Key Crypto Systems

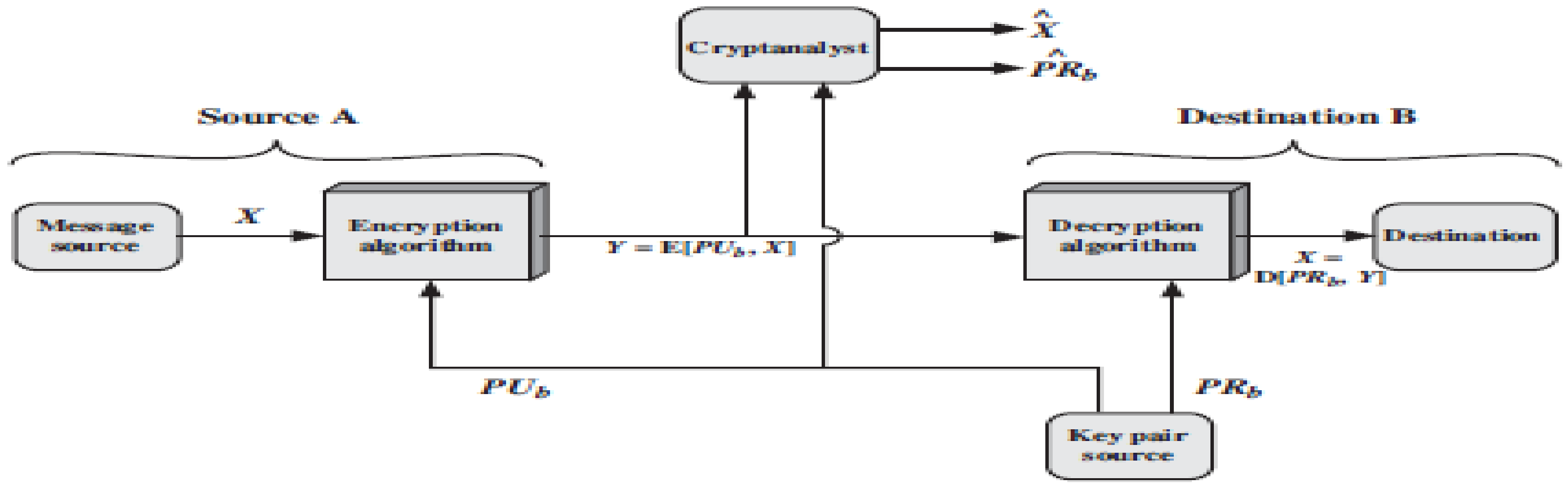
- Essential Steps:
  - Each user generates a pair of keys to be used for the encryption and decryption of messages.
  - Each user places one key (public key) in a public register or other accessible file. The other related key is kept private.

Table 9.2 Conventional and Public-Key Encryption



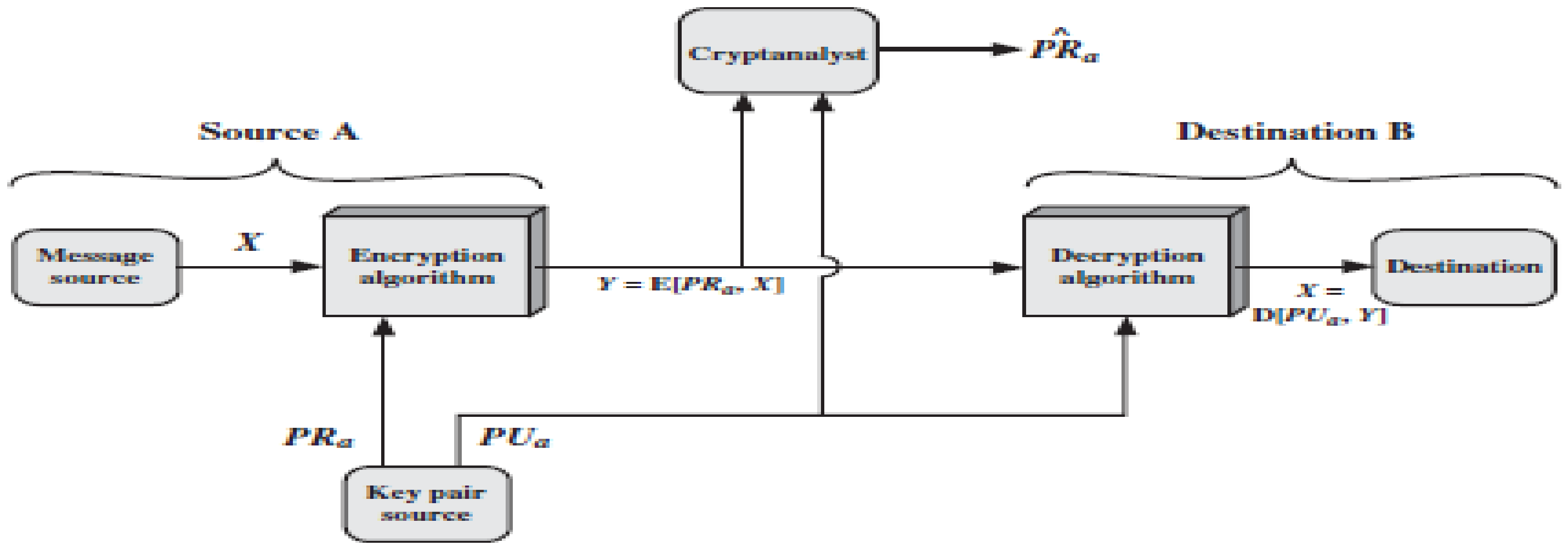
Conventional Encryption	Public-Key Encryption
Needed to Work:	*Needed to Work:*
1. The same algorithm with the same key is used for encryption and decryption.	1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.
2. The sender and receiver must share the algorithm and the key.	2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security:	*Needed for Security:*
1. The key must be kept secret.	1. One of the two keys must be kept secret.
2. It must be impossible or at least impractical to decipher a message if no other information is available.	2. It must be impossible or at least impractical to decipher a message if no other information is available.
3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public-Key Crypto System Secrecy



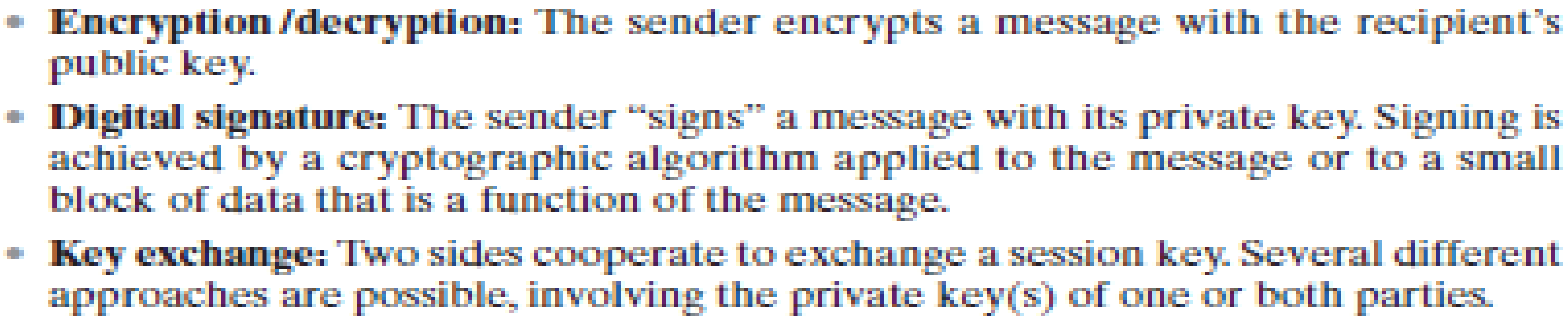
- ✓ An adversary observing Y and having access to PU_b but not having access to PR_b or X, must attempt to recover X and/ or PR_b.
- ✓ It is assumed that adversary have the knowledge of the encryption (E) and decryption (D) algorithms.

Public-Key Crypto System: Authentication



- ✓ Used for digital signature.
- ✓ In order to save storage and computational cost, a small block of bits (Authenticator) that is a function of the document, is encrypted with the private key.
- ✓ It is infeasible to change the document without changing the authenticator.
- ✓ Encryption with private key does not provide confidentiality.

Applications for Public-Key Crypto Systems

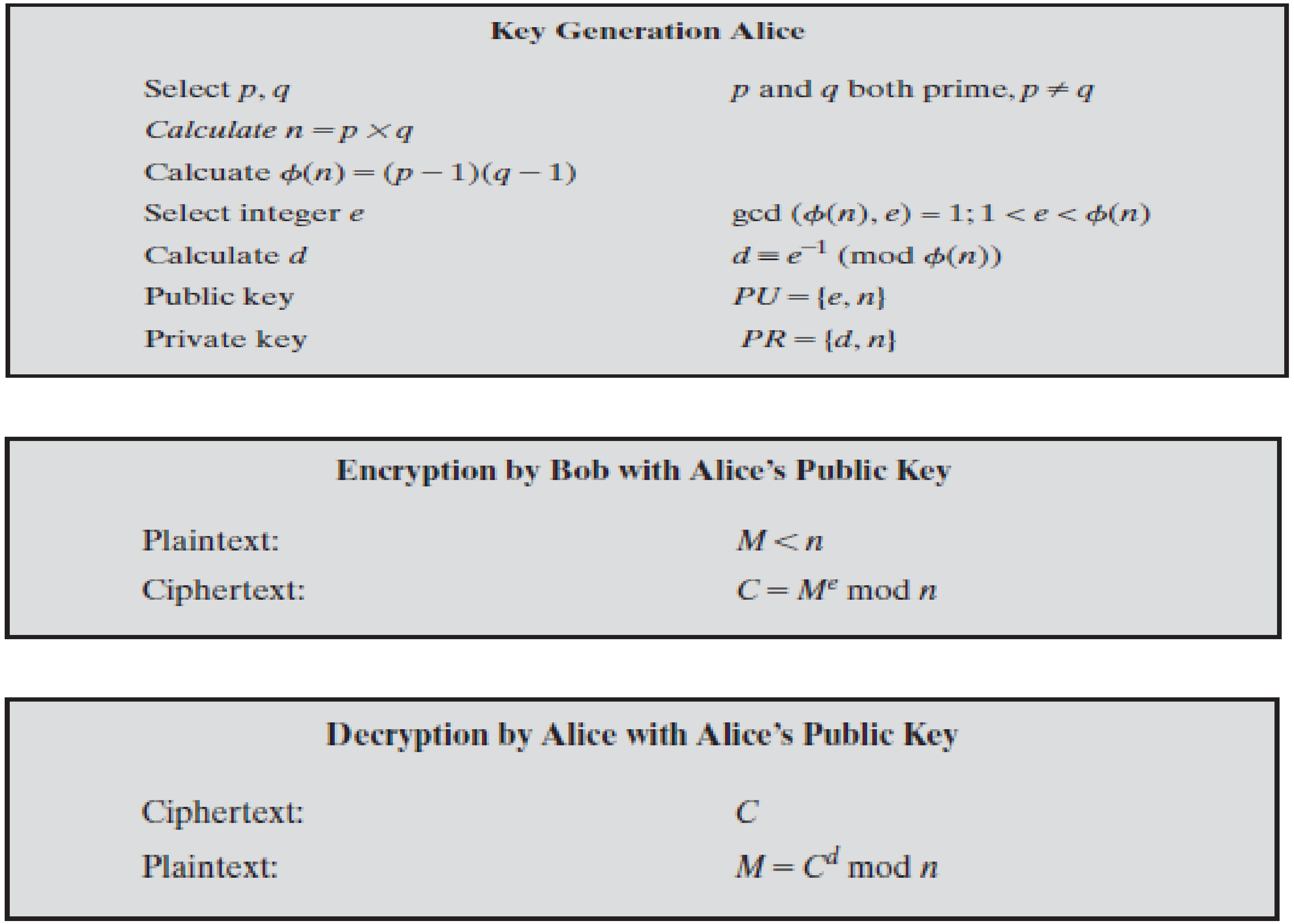


- **Encryption /decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

RSA Algorithm

- By Ron Rivest, Adi Shamir & Leonard Adleman of MIT in 1977
- Best known & widely used public-key scheme
- Uses large integers (eg. 1024 bits)
  - Plaintext and ciphertext are integers between ***0*** and ***n-1*** for some ***n***.
- Plaintext is encrypted in blocks with each block having binary value less than some number ***n***.

RSA Algorithm



Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

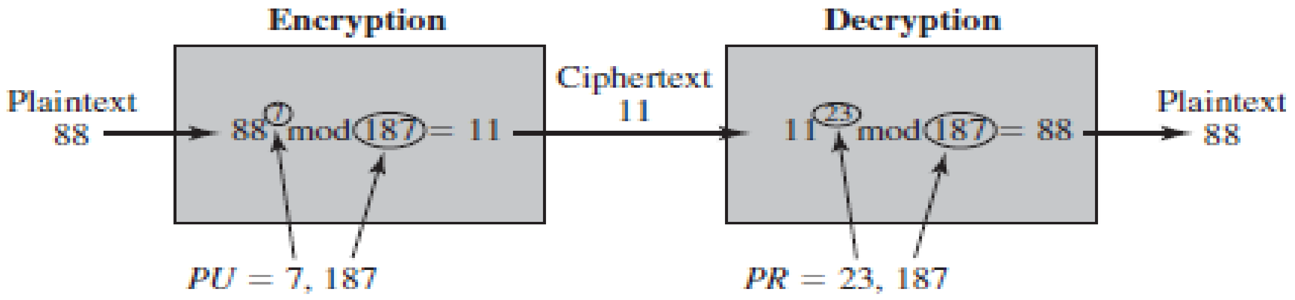
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

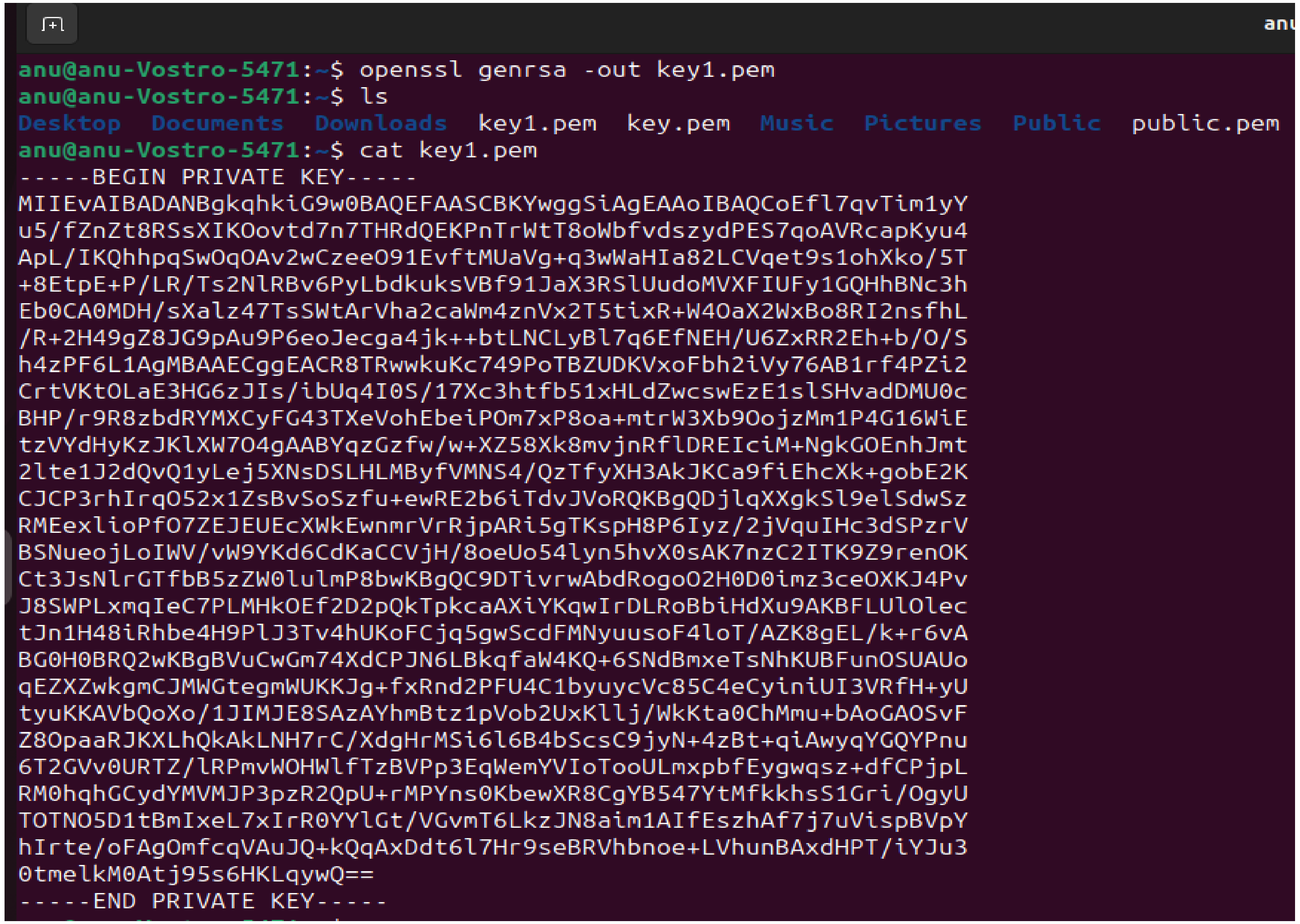
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example of RSA Algorithm

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 4).

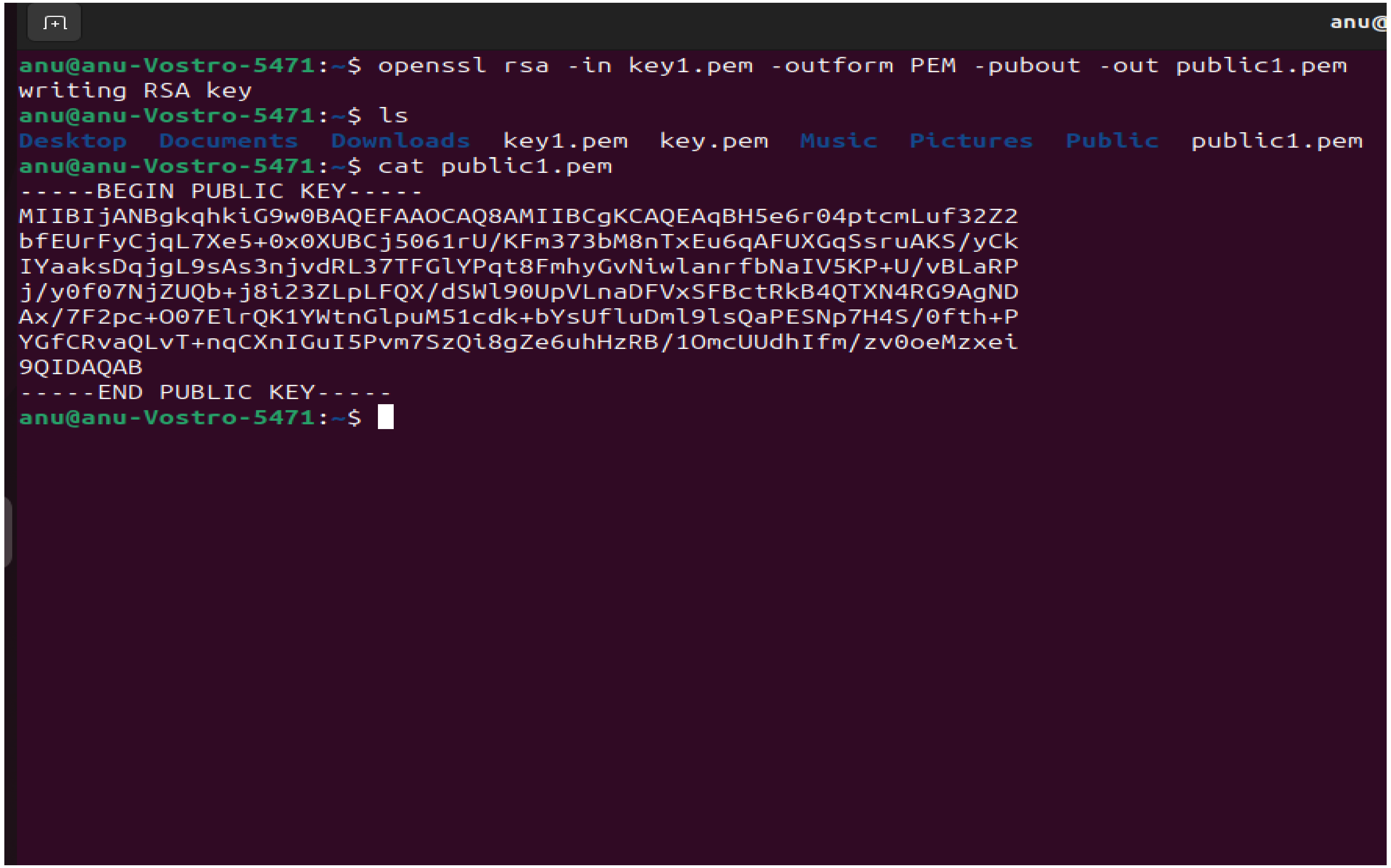


Creating RSA Private Key using Openssl



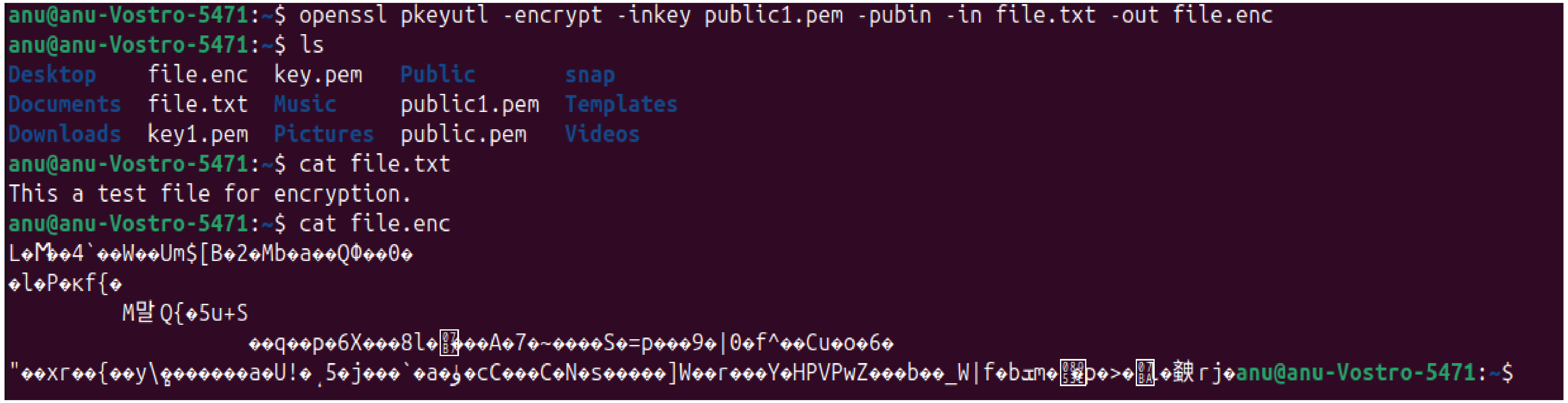
```
anu@anu-Vostro-5471:~$ openssl genrsa -out key1.pem
anu@anu-Vostro-5471:~$ ls
Desktop  Documents  Downloads  key1.pem  key.pem  Music  Pictures  Public  public.pem
anu@anu-Vostro-5471:~$ cat key1.pem
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCoEfl7qvTim1yY
u5/fZnZt8RSsXIKOovtd7n7THRdQEKPnTrWtT8oWbfvdszydPES7qoAVRcapKyu4
ApL/IKQhhpqSwOqOAv2wCzeeO91EvftMUaVg+q3wWaHIa82LCVqet9s1ohXko/5T
+8EtpE+P/LR/Ts2NlRBv6PyLbdkuksVBf91JaX3RSlUudoMVXFIUFy1GQHhBNc3h
Eb0CA0MDH/sXalz47TsSWtArVha2caWm4znVx2T5tixR+W4OaX2WxBo8RI2nsfhL
/R+2H49gZ8JG9pAu9P6eoJecga4jk++btLNCLyBl7q6EfNEH/U6ZxRR2Eh+b/O/S
h4zPF6L1AgMBAAECggEACR8TRwwkuKc749PoTBZUDKVxoFbh2iVy76AB1rf4PZi2
CrtVKtOLaE3HG6zJIs/ibUq4I0S/17Xc3htfb51xHLdZwcswEzE1slSHvadDMU0c
BHP/r9R8zbdRYMXCyFG43TXeVohEbeiPOm7xP8oa+mtrW3Xb9OojzMm1P4G16WiE
tzVYdHyKzJKlXW7O4gAABYqzGzfw/w+XZ58Xk8mvjnRflDREIciM+NgkGOEnhJmt
2lte1J2dQvQ1yLej5XNsDSLHLMByfVMNS4/QzTfyXH3AkJKCa9fiEhcXk+gobE2K
CJCP3rhIrqO52x1ZsBvSoSzfu+ewRE2b6iTdvJVoRQKBgQDjlqXXgkSl9elSdwSz
RMEexlioPfO7ZEJEUEcXWkEwnmrVrRjpARi5gTKspH8P6Iyz/2jVquIHc3dSPzrV
BSNueojLoIWV/vW9YKd6CdKaCCVjH/8oeUo54lyn5hvX0sAK7nzC2ITK9Z9renOK
Ct3JsNlrGTfbB5zZW0lulmP8bwKBgQC9DTivrwAbdRogoO2H0D0imz3ceOXKJ4Pv
J8SWPLxmqIeC7PLMHkOEf2D2pQkTpkcaAXiYKqwIrDLRoBbiHdXu9AKBFLUlOlec
tJn1H48iRhbe4H9PlJ3Tv4hUKoFCjq5gwScdFMNyuusoF4loT/AZK8gEL/k+r6vA
BG0H0BRQ2wKBgBVuCwGm74XdCPJN6LBkqfaW4KQ+6SNdBmxeTsNhKUBFunOSUAUo
qEZXZwkgmCJMWGtegmWUKKJg+fxRnd2PFU4C1byuycVc85C4eCyiniUI3VRfH+yU
tyuKKAVbQoXo/1JIMJE8SAzAYhmBtz1pVob2UxKllj/WkKta0ChMmu+bAoGAOSvF
Z8OpaaRJKXLhQkAkLNH7rC/XdgHrMSi6l6B4bScsC9jyN+4zBt+qiAwyqYGQYPnu
6T2GVv0URTZ/lRPmvWOHWlfTzBVPp3EqWemYVIoTooULmxpbfEygwqsz+dfCPjpL
RM0hqhGCydYMVMJP3pzR2QpU+rMPYns0KbewXR8CgYB547YtMfkkhsS1Gri/OgyU
TOTNO5D1tBmIxeL7xIrR0YYlGt/VGvmT6LkzJN8aim1AIfEszhAf7j7uVispBVpY
hIrte/oFAgOmfcqVAuJQ+kQqAxDdt6l7Hr9seBRVhbnoe+LVhunBAxdHPT/iYJu3
0tmelkM0Atj95s6HKLqywQ==
-----END PRIVATE KEY-----
```

Creating RSA Public Key using Openssl

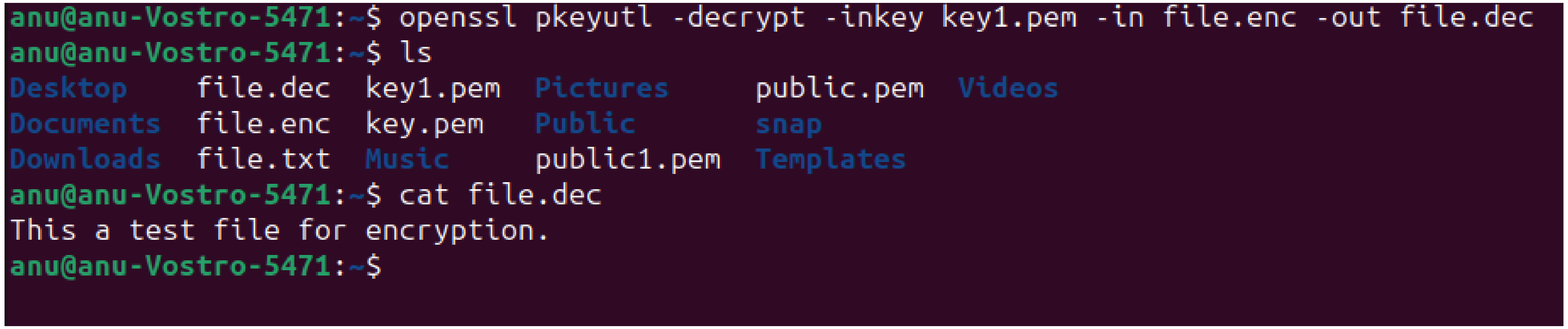


```
anu@anu-Vostro-5471:~$ openssl rsa -in key1.pem -outform PEM -pubout -out public1.pem
writing RSA key
anu@anu-Vostro-5471:~$ ls
Desktop  Documents  Downloads  key1.pem  key.pem  Music  Pictures  Public  public1.pem
anu@anu-Vostro-5471:~$ cat public1.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqBH5e6r04ptcmLuf32Z2
bfEUrFyCjqL7Xe5+0x0XUBCj5061rU/KFm373bM8nTxEu6qAFUXGqSsruAKS/yCk
IYaaksDqjgL9sAs3njvdRL37TFGlYPqt8FmhyGvNiwlanrfbNaIV5KP+U/vBLaRP
j/y0f07NjZUQb+j8i23ZLpLFQX/dSWl90UpVLnaDFVxSFBctRkB4QTXN4RG9AgND
Ax/7F2pc+O07ElrQK1YWtnGlpuM51cdk+bYsUfluDml9lsQaPESNp7H4S/0fth+P
YGfCRvaQLvT+nqCXnIGuI5Pvm7SzQi8gZe6uhHzRB/1OmcUUdhIfm/zv0oeMzxei
9QIDAQAB
-----END PUBLIC KEY-----
anu@anu-Vostro-5471:~$ 
```

Encrypt and Decrypt a File



```
anu@anu-Vostro-5471:~$ openssl pkeyutl -encrypt -inkey public1.pem -pubin -in file.txt -out file.enc
anu@anu-Vostro-5471:~$ ls
Desktop    file.enc  key.pem   Public       snap
Documents  file.txt  Music     public1.pem  Templates
Downloads  key1.pem  Pictures  public.pem   Videos
anu@anu-Vostro-5471:~$ cat file.txt
This a test file for encryption.
anu@anu-Vostro-5471:~$ cat file.enc
L�M��4`��W��Um$[B�2�Mb�a��QΦ��0�
�l�P�κf{�
         M말Q{�5u+S
                   ��q��p�6X���8l��[illegible]��A�7�~����S�=p���9�|0�f^��Cu�o�6�
"��xr��{��y\ǫ������a�U!� ̧5�j���`�a�ʝ�cC���C�N�s�����]W��r���Y�HPVPwZ���b��_W|f�bꞁm�[illegible]p�>�[illegible]�鋉rj�anu@anu-Vostro-5471:~$
```



```
anu@anu-Vostro-5471:~$ openssl pkeyutl -decrypt -inkey key1.pem -in file.enc -out file.dec
anu@anu-Vostro-5471:~$ ls
Desktop    file.dec  key1.pem  Pictures     public.pem  Videos
Documents  file.enc  key.pem   Public       snap
Downloads  file.txt  Music     public1.pem  Templates
anu@anu-Vostro-5471:~$ cat file.dec
This a test file for encryption.
anu@anu-Vostro-5471:~$
```

Diffie-Hellman Key Exchange

- Proposed by Whitfield Diffie and Martin Hellman in 1976
- Enables two users to securely exchange a key that can be used for subsequent symmetric key encryption of messages.
- Used in a number of commercial products such as SSL/TLS and SSH.
- Effectiveness depends on the difficulty of computing discrete logarithms.

Theory Behind Diffie-Hellman Key Exchange

- ***a*** is a primitive root of the prime number ***p***, if

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

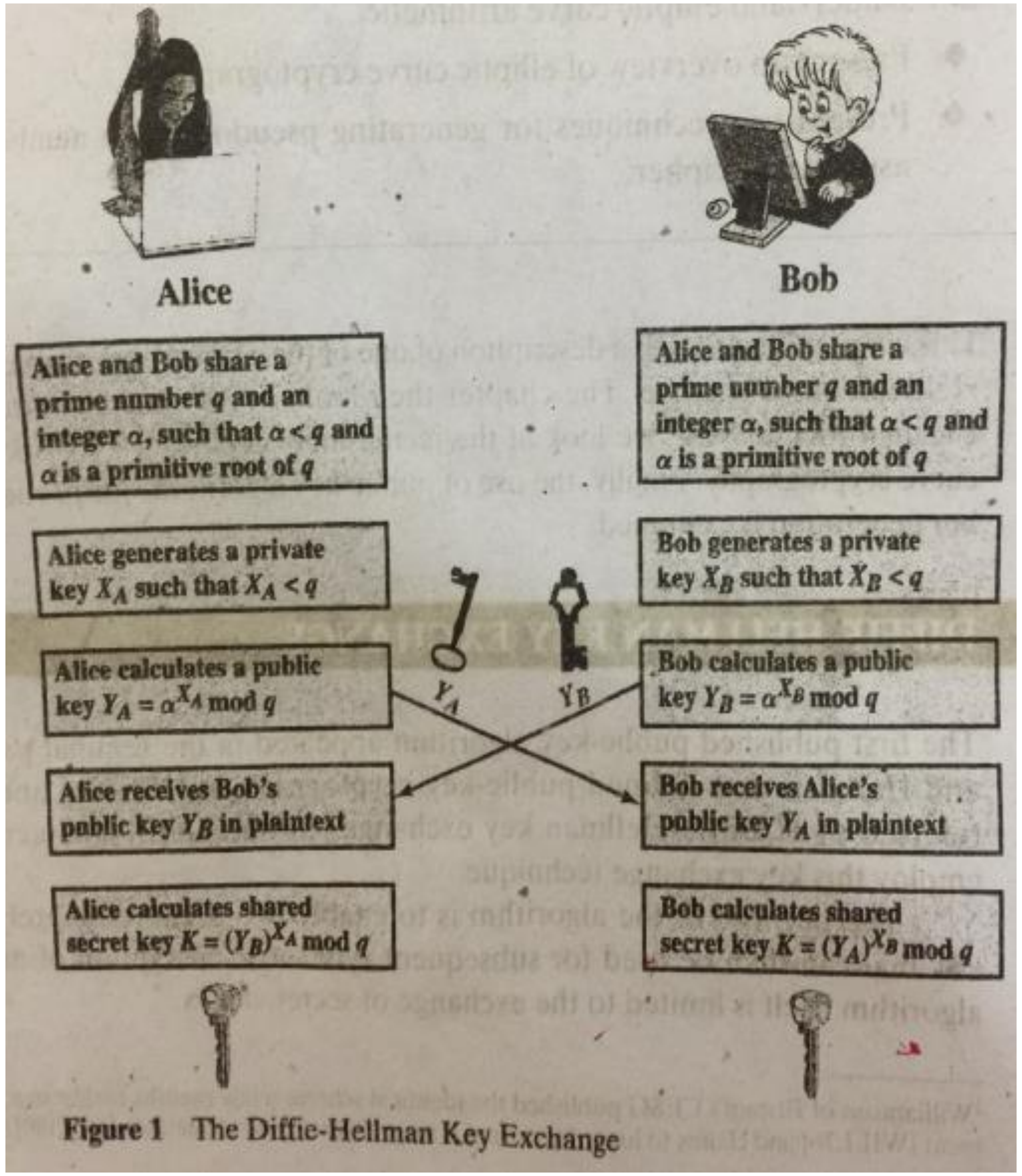
  are distinct and consists of the integers from ***1*** through ***p-1*** in some permutation.

- For any integer ***b*** and a primitive root ***a*** of prime number ***p***, there is a unique exponent ***i*** such that

  $$\boldsymbol{b} \equiv \boldsymbol{a}^{i}\ (\boldsymbol{mod\ p}) \quad \text{where } \boldsymbol{0} \leq \boldsymbol{i} \leq (\boldsymbol{p}-\boldsymbol{1})$$

  Exponent ***i*** is referred to as the discrete logarithm of ***b*** for base ***a, mod p*** i.e, $\boldsymbol{dlog}_{a,p}(\boldsymbol{b}) = \boldsymbol{i}$.

Diffie-Hellman Key Exchange Algorithm



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

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

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

Alice receives Bob's public key Y_B in plaintext

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

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

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

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

Bob receives Alice's public key Y_A in plaintext

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

Figure 1 The Diffie-Hellman Key Exchange

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

✓ To determine the private key of user ***B***, an adversary must compute

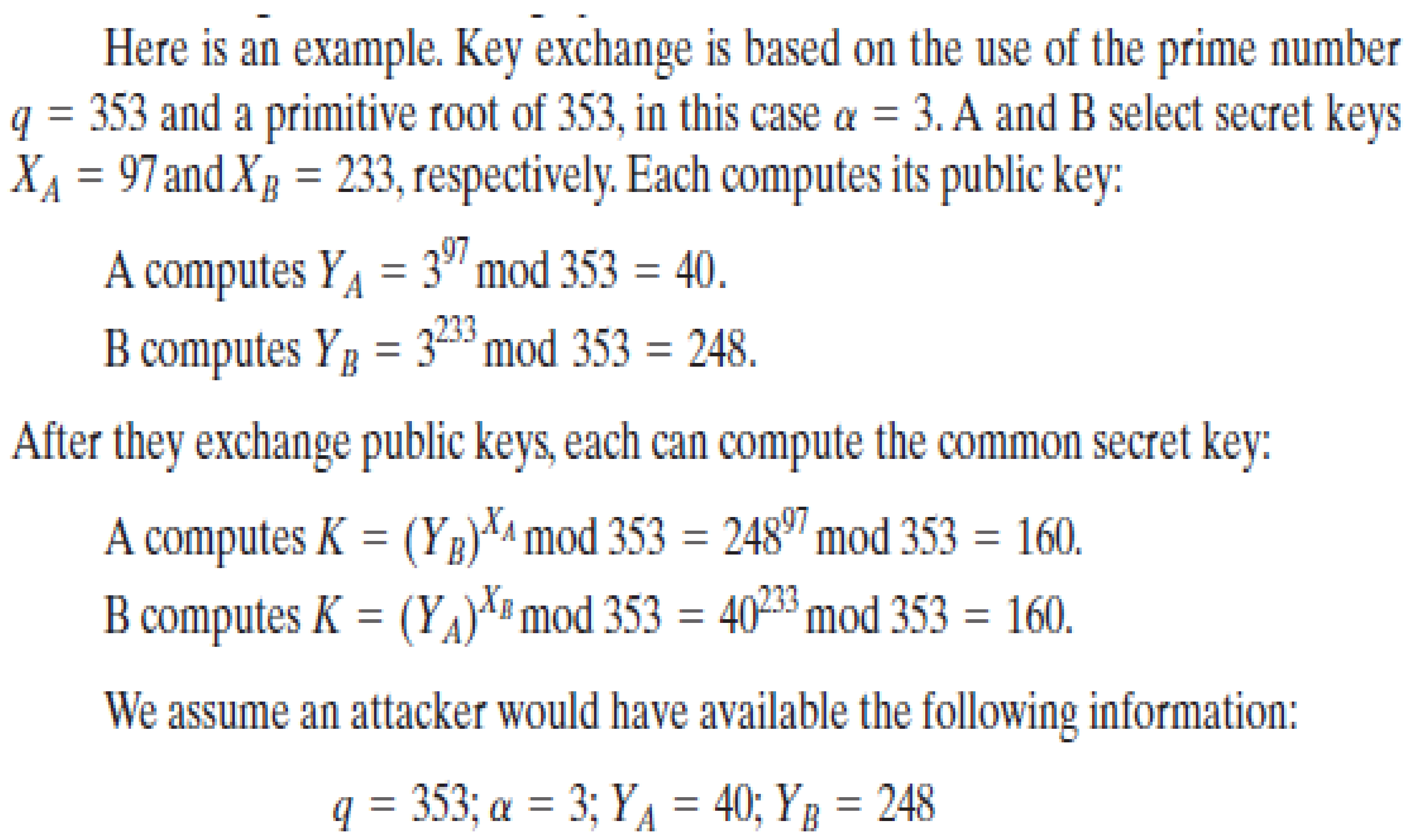
$$X_B = dlog_{\alpha,a}(Y_B)$$

Then the adversary can calculate ***K*** as

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

✓ $b \equiv a^i \pmod p$ is easy, but $dlog_{a,p}(b) = i$ is difficult for large ***p***.

Example



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

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

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

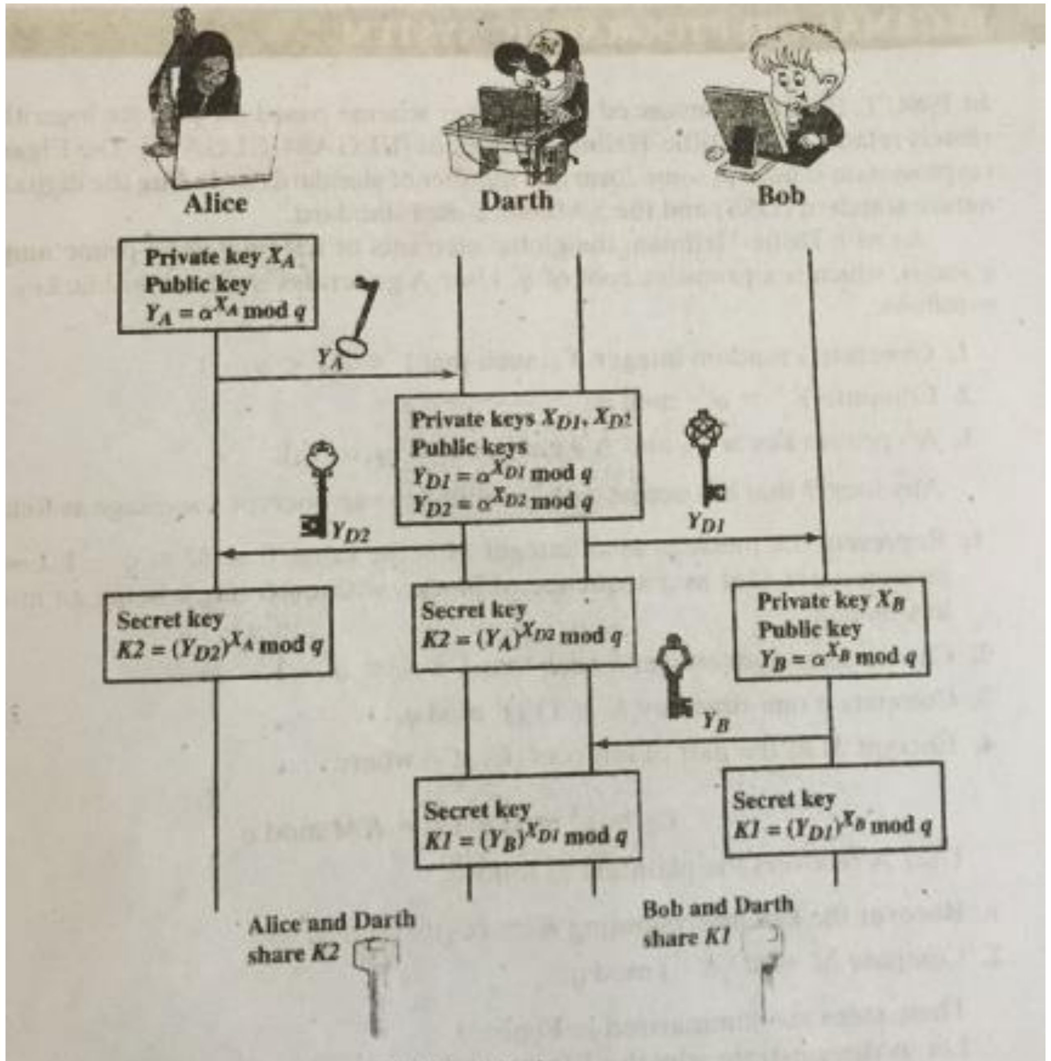
A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

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

Man-in-the-Middle Attack



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

Figure 2 Man-in-the-Middle Attack

DNSSEC (DSN Security Extensions)

- DNS accepts an answer without verifying credential.
- The solution is DNSSEC, adding a layer of trust on top of DNS by providing authentication.
- Example:
- When a DNS resolver is looking for web.example.com
  - the root DNS name servers help verify .com
  - the .com name servers help the resolver verify the records returned for example
  - example helps verify the records returned for web.
  - information published by the root is vetted by a thorough security procedure, including the Root Signing Ceremony.

DNSSEC

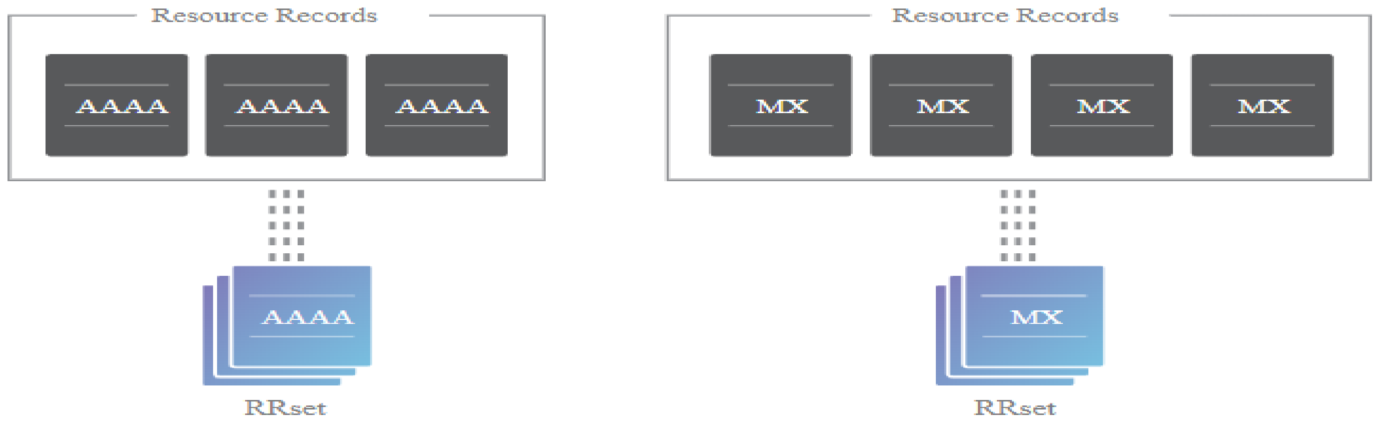
- DNSSEC adds cryptographic signatures to existing DNS records.
- These digital signatures are stored in DNS name servers alongside common DNS record types.
- By checking the associated signature, it can ensured that a requested DNS record comes from its authoritative name server and not altered en-route.

- DNSSEC adds a few new DNS record types:
  - **RRSIG** - Contains a cryptographic signature
  - **DNSKEY** - Contains a public signing key
  - **DS** - Contains the hash of a DNSKEY record
  - **NSEC** and **NSEC3** - For explicit denial-of-existence of a DNS record
  - **CDNSKEY** and **CDS** - For a child zone requesting updates to DS record(s) in the parent zone.

RRSets

- Security should be imposed on zone level.
- Group all the records with the same type into a resource record set (RRset).
  - If you have three AAAA records in your zone on the same label (i.e. label.example.com), they would all be bundled into a single AAAA RRset.

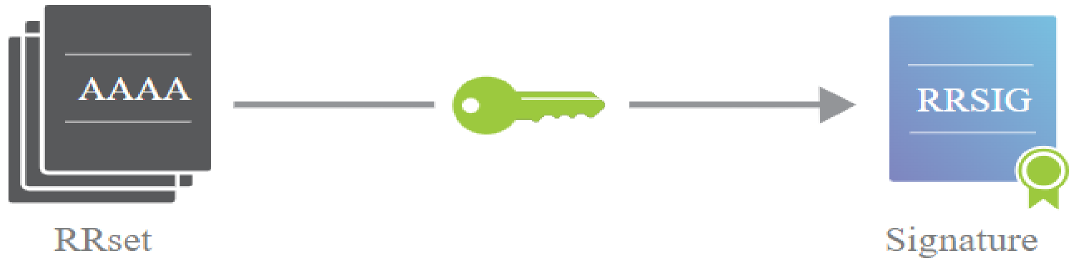
.



- The full RRset is digitally signed.
- You must request and validate all of the AAAA records from a zone with the same label instead of validating only one of them.

Zone-Signing Key (ZSK)

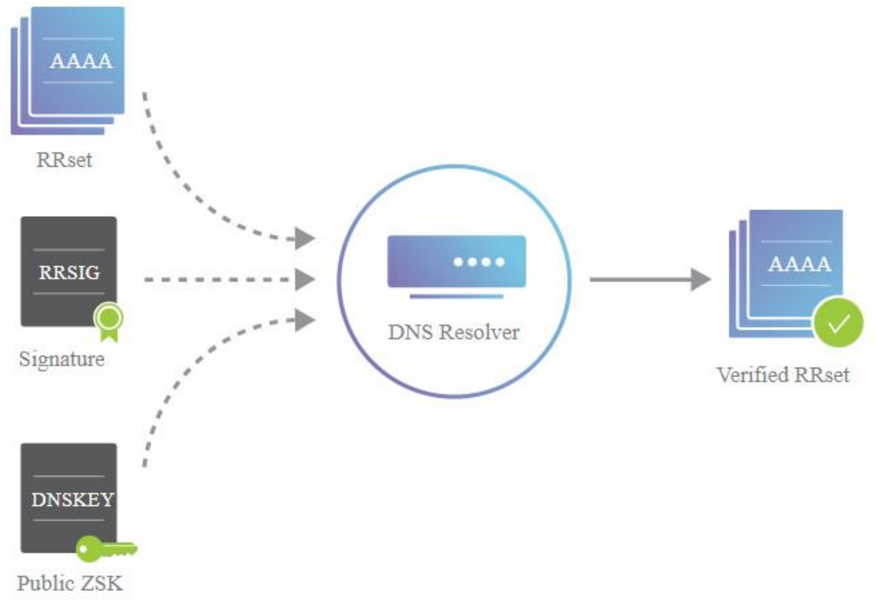
- Each zone in DNSSEC has a zone-signing key pair (ZSK):
  - the private portion of the key digitally signs each RRset in the zone.
  - the public portion verifies the signature.
- A zone operator creates digital signatures for each RRset using the private ZSK and stores them in their name server as RRSIG records.



- The public ZSK is made available by adding it to the name server in a DNSKEY record.

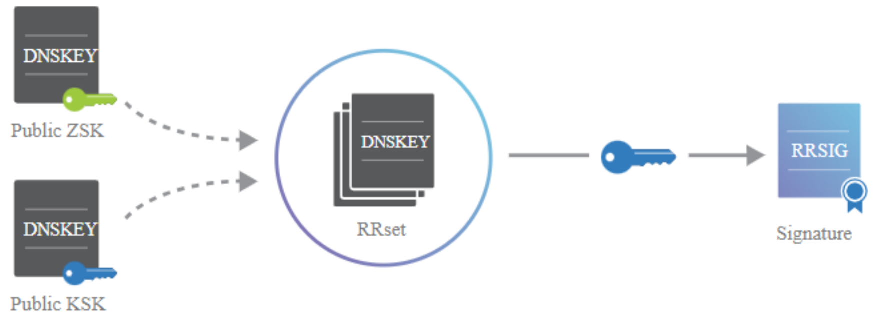
Zone-Signing Key (ZSK)

- When a DNSSEC resolver requests a particular record type, the name server also returns the corresponding RRSIG.
- The resolver can then pull the DNSKEY record containing the public ZSK from the name server. Together, the RRset, RRSIG, and public ZSK can validate the response.
- Need to validate public ZSK.



Key-Signing Keys

- DNSSEC name servers also have a key-signing key (KSK).
- It signs the public ZSK (which is stored in a DNSKEY record), creating an RRSIG for the DNSKEY.



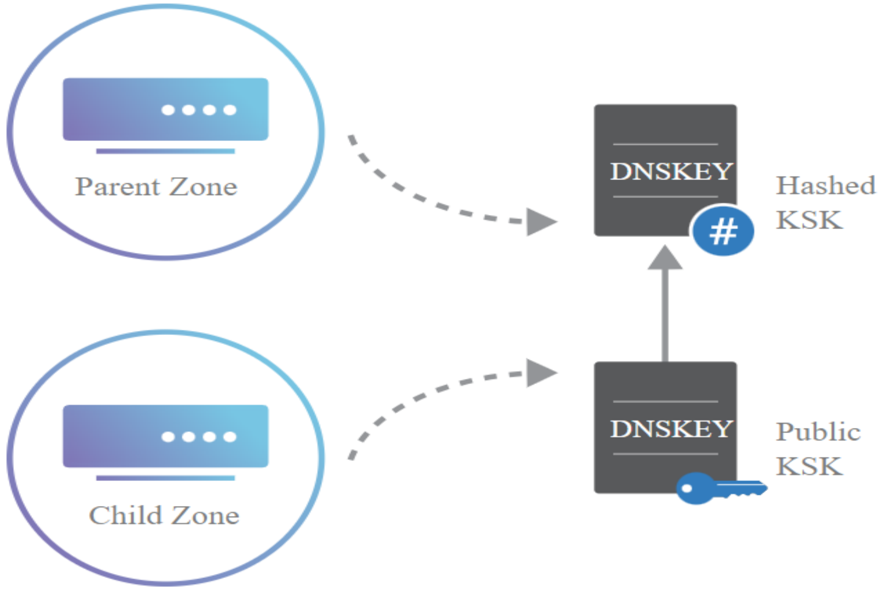
- The name server publishes the public KSK in another DNSKEY record, creating a DNSKEY Rrset.
- Both the public KSK and public ZSK are signed by the private KSK.

Key-Signing Keys

- Validation for resolvers now looks like this:
  - Request the desired RRset, which also returns the corresponding RRSIG record.
  - Request the DNSKEY records containing the public ZSK and public KSK, which also returns the RRSIG for the DNSKEY RRset.
  - Verify the RRSIG of the requested RRset with the public ZSK.
  - Verify the RRSIG of the DNSKEY RRset with the public KSK.

Delegation Signer Records

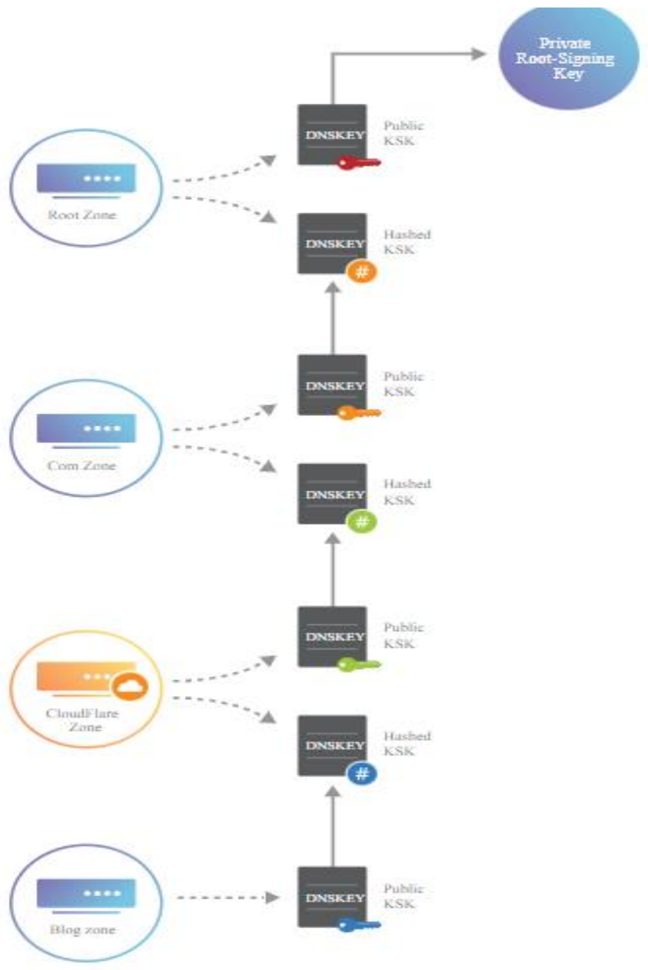
- A delegation signer (DS) record allows the transfer of trust from a parent zone to a child zone.
- A zone operator hashes the DNSKEY record containing the public KSK and gives it to the parent zone to publish as a DS record.



Delegation Signer Records

- Every time a resolver is referred to a child zone, the parent zone also provides a DS record.
- This DS record indicates that the child zone is DNSSEC-enabled.
- To check the validity of the child zone's public KSK, the resolver hashes it and compares it to the DS record from the parent. If they match, the resolver can assume that the public KSK hasn't been tampered with.
- any change in the KSK also requires a change in the parent zone's DS record. The parent needs to add the new DS record, then they need to wait until the TTL for the original DS record to expire before removing it.

Trust of Chain



- the DS record is signed with the private KSK of the parent and the corresponding RRSIG is stored in the parent.
- In the root DNS zone, there is no parent DS record to validate.
- Solution: Root Signing Procedure
- In this procedure, several selected individuals from around the world come together and sign the root DNSKEY RRset in a very public and highly audited way.
- The ceremony produces an RRSIG record that can be used to verify the root name server’s public KSK and ZSK.

Resources

- Cryptography elements:

  Cryptography and Network Security (Principles and Practice)

  --- William Stalling

- DNSSEC:

  https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

(Lecture slides 64-73 are adapted from this source)