Cryptography

Introduction

- Encryption is the process of encoding a message in such a way that only authorized parties can read the content of the original message
- History of encryption dates back to 1900 BC
- Two types of encryption
  - symmetric encryption : same key for encryption and decryption
  - asymmetric (pubic-key) encryption : different keys for encryption and decryption

Terminology

- **Plaintext**: This is what you want to encrypt
- **Ciphertext:** This is the encrypted output
- **Enciphering or Encryption:** The process of converting plaintext into ciphertext
- **Deciphering or Decryption:** Recovering plaintext from ciphertext
- **Block Cipher:** A block cipher processes a block of input data at a time and produces a ciphertext block of the same size.
- **Stream cipher:** Encrypts data on the fly, usually one byte at at time.

Terminology

- **Key Space:** The total number of all possible keys that can be used in a cryptographic system. For example, DES uses a 56-bit key. So the key space is of size 2^56, which is approximately the same as 7.2 × 10^16 .

- **Brute-force Attack:** When encryption and decryption algorithms are publicly available, as they generally are, a brute-force attack means trying every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

- **Codebook Attack:** In general, a codebook is a mapping from the plaintext symbols to the ciphertext symbols. In old times, the two endpoints of a military communication link would have the same codebook that would be composed of sheets, with a different sheet to be used for each day.

- **Algebraic Attack:** You express the plaintext-to-ciphertext relationship as a system of equations. Given a set of (plaintext, ciphertext) pairs, you try to solve the equations for the encryption key.

Terminology

- Two building blocks of all classical encryption techniques are substitution and transposition.

- Substitution means replacing an element of the plaintext with an element of ciphertext.

- The same overall substitution rule may be applied to every element of the plaintext, or the substitution rule may vary from position to position in the plaintext.

- Transposition means rearranging the order of appearance of the elements of the plaintext. Transposition is also referred to as permutation.

Caesar Cipher

- This is the earliest known example of a substitution cipher.

- Each character of a message is replaced by a character three position down in the alphabet (case insensitive).
  - plaintext: are you ready
  - ciphertext: duh brx uhdgb

- If we represent each letter of the alphabet by an integer that corresponds to its position in the alphabet, the formula for replacing each character p of the plaintext with a character c of the ciphertext can be expressed as
  - $c = E(3, p) = (p + 3) \bmod 26$

Caesar Cipher

- A more general version of this cipher that allows for any degree of shift would be expressed by
  - $c = E(k, p) = (p + k) \bmod 26$
- The formula for decryption would be
  - $p = D(k, c) = (c - k) \bmod 26$
- In these formulas, k would be the secret key. E() stands for encryption. By the same token, D() stands for decryption.

Monoalphabetic Substitution Cipher

- The Caesar cipher is an example of a monoalphabetic cipher.

- In a monoalphabetic cipher, you use the same substitution rule to find the replacement ciphertext letter for each letter of the alphabet in the plaintext message.

- Consider a monoalphabetic cipher that uses a random permutation of the 26 letters of the alphabet
  - plaintext letters: a b c d e f .....
  - substitution letters: t h i j a b .....
  - The encryption key in this case is the sequence of substitution letters (i.e. thijab…).

Monoalphabetic Substitution Cipher

- Encryption and decryption

```
# Encryption
$ tr 'a-z' 'vgapnbrtmosicuxejhqyzflkdw' < plaintext  > ciphertext

# Decryption
$ tr 'vgapnbrtmosicuxejhqyzflkdw' 'a-z' < ciphertext > plaintext_new
```

Monoalphabetic Substitution Cipher

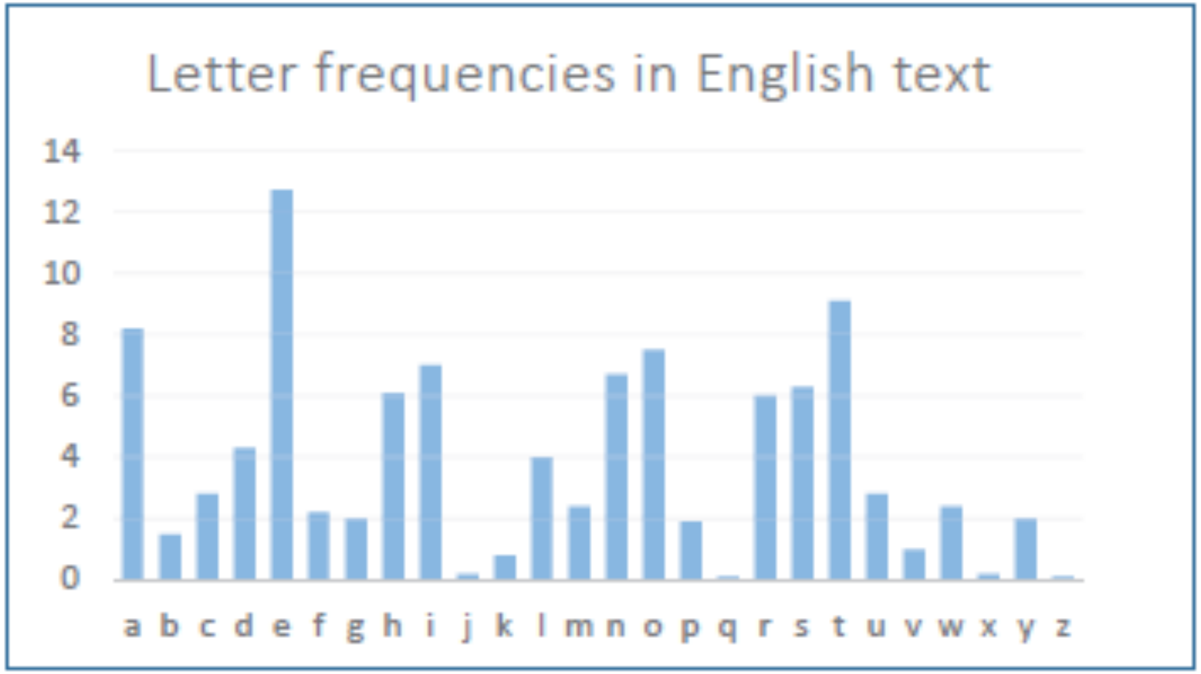
- There are 26! Permutations of the alphabet.
- This is a very large key space
- Does it make this make the cipher difficult to break?

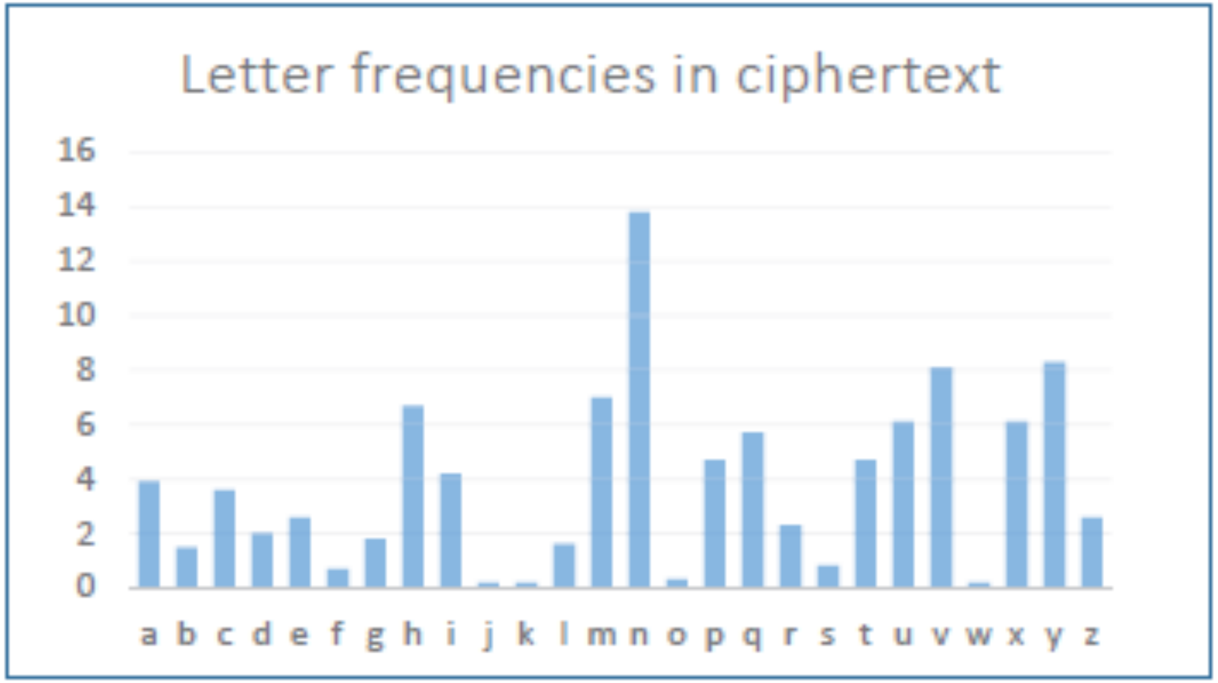
Breaking Monoalphabetic Substitution Cipher

- Frequency analysis is the study of the frequency of letters or groups of letters in a ciphertext.
- Common letters : T, A, E, I, O
- Common 2-letter combinations (bigrams): TH, HE, IN, ER
- Common 3-letter combinations (trigrams): THE, AND, and ING

Breaking Monoalphabetic Substitution Cipher

- **Letter** Frequency Analysis results:





Breaking Monoalphabetic Substitution Cipher

- **Bigram** Frequency Analysis results:

Bigram frequency in English

TH	:	2.71	EN	:	1.13	NG	:	0.89
HE	:	2.33	AT	:	1.12	AL	:	0.88
IN	:	2.03	ED	:	1.08	IT	:	0.88
ER	:	1.78	ND	:	1.07	AS	:	0.87
AN	:	1.61	TO	:	1.07	IS	:	0.86
RE	:	1.41	OR	:	1.06	HA	:	0.83
ES	:	1.32	EA	:	1.00	ET	:	0.76
ON	:	1.32	TI	:	0.99	SE	:	0.73
ST	:	1.25	AR	:	0.98	OU	:	0.72
NT	:	1.17	TE	:	0.98	OF	:	0.71

Bigram frequency in chiphertext (The top-10 patterns)

tn	:	77	np	:	50
yt	:	76	hn	:	45
nh	:	61	nu	:	44
nq	:	51	mu	:	42
vu	:	51	cv	:	42

Breaking Monoalphabetic Substitution Cipher

- **Trigram** Frequency analysis results:

Trigram frequency in English

THE	:	1.81	ERE	:	0.31	HES	:	0.24
AND	:	0.73	TIO	:	0.31	VER	:	0.24
ING	:	0.72	TER	:	0.30	HIS	:	0.24
ENT	:	0.42	EST	:	0.28	OFT	:	0.22
ION	:	0.42	ERS	:	0.28	ITH	:	0.21
HER	:	0.36	ATI	:	0.26	FTH	:	0.21
FOR	:	0.34	HAT	:	0.26	STH	:	0.21
THA	:	0.33	ATE	:	0.25	OTH	:	0.21
NTH	:	0.33	ALL	:	0.25	RES	:	0.21
INT	:	0.32	ETH	:	0.24	ONT	:	0.20

Trigram frequency in chiphertext (The top-10 patterns)

ytn	:	60	tnh	:	13
vup	:	26	pyt	:	13
nhc	:	16	hcv	:	13
nhn	:	15	tne	:	13
nuy	:	14	mrc	:	13

Playfair Cipher

- You choose an encryption key, making sure that there are no duplicate characters in the key.
- You then enter the characters in the key in the cells of a 5 × 5 matrix in a left-to-right and top-to-down fashion starting with the first cell at the top-left corner.
- You fill the rest of the cells of the matrix with the remaining characters in the alphabet and do so in alphabetic order. The letters I and J are assigned the same cell. In the following example, the key is "**smythework**":

Playfair Cipher

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

- You scan the plaintext in pairs of consecutively occurring characters. And, for any given pair of plaintext characters, you use the following three rules to determine the corresponding pair of ciphertext characters:
- 1) Two plaintext letters that fall in the same row of the 5 × 5 matrix are replaced by letters to the right of each in the row. The “rightness” property is to be interpreted circularly in each row, meaning that the first entry in each row is to the right of the last entry. Therefore, the pair of letters “bf” in plaintext will get replaced by “CA” in ciphertext.

Playfair Cipher

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

- 2) Two plaintext letters that fall in the same column are replaced by the letters just below them in the column. The “belowness” property is to be considered circular, in the sense that the topmost entry in a column is below the bottom-most entry. Therefore, the pair “ol” of plaintext will get replaced by “CV” in ciphertext.
- 3) Otherwise, for each plaintext letter in a pair, replace it with the letter that is in the same row but in the column of the other letter. Consider the pair “gf” of the plaintext. We have ‘g’ in the fourth row and the first column; and ‘f’ in the third row and the fifth column. So we replace ‘g’ by the letter in the same row as ‘g’ but in the column that contains ‘f’. This given us ‘P’ as a replacement for ‘g’. And we replace ‘f’ by the letter in the same row as ‘f’ but in the column that contains ‘g’. That gives us ‘A’ as replacement for ‘f’. Therefore, ‘gf’ gets replaced by ‘PA’.

Playfair Cipher

- Before the substitution rules are applied, you must insert a chosen “filler” letter (let’s say it is ‘x’) between any repeating letters in the plaintext. So a plaintext word such as “hurray” becomes “hurxray”

- Although the Playfair cipher was used in WW1 and WW2, it is extremely easy to break through frequency analysis

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

The Hill Cipher

- The Hill cipher uses a more mathematical
- You assign an integer to each letter of the alphabet. Typically, integers 0 through 25 are assigned to the letters ‘a’ through ‘z’ of the plaintext.
- The encryption key, call it K, consists of a 3 × 3 matrix of integers:

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

The Hill Cipher

- The Hill cipher uses a more mathematical
- You assign an integer to each letter of the alphabet. Typically, integers 0 through 25 are assigned to the letters 'a' through 'z' of the plaintext.
- The encryption key, call it K, consists of a 3 × 3 matrix of integers:

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

The Hill Cipher

- Encryption is done through the following equation 3 letters at a time:

$$\vec{\mathbf{C}} \;=\; [\mathbf{K}]\,\vec{\mathbf{P}}\; mod\ 26$$

- Decryption would require the inverse of K matric

$$\vec{\mathbf{P}} \;=\; \left[\mathbf{K}^{-1}\right]\vec{\mathbf{C}}\; mod\ 26$$

The Hill Cipher

- Example:
  - Input : Plaintext: ACT
  - Key: GYBNQKURP
  - Output : Ciphertext: POH

- We have to encrypt “ACT” (n = 3), the key is GYBNQKURP which can be written nxn matrix (3x3)

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

The Hill Cipher

- The "ACT" message is written as a vector:

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix}$$

- The encrypted vector is given as

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \pmod{26}$$

- This corresponds to "POH"

The Hill Cipher

- Decryption:
  - Obtain inverse Key:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}^{-1} \equiv \begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \pmod{26}$$

- The ciphertext "POH" is:

$$\begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \equiv \begin{bmatrix} 260 \\ 574 \\ 539 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} \pmod{26}$$

- This gives back "ACT"

The Hill Cipher

- The ”ACT” message is written as a vector:

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix}$$

- The encrypted vector is given as

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \pmod{26}$$

- This corresponds to ”POH”

How Secure is the Hill Cipher

- The cipher is extremely secure against ciphertext only attacks.
- That is because the keyspace can be made extremely large by choosing the matrix elements from a large set of integers. (The key space can be made even larger by generalizing the technique to larger matrices.)

- However, the cipher has zero security when the plaintext–ciphertext pairs are known. The key matrix can be calculated easily from a set of known <P> , <C> pairs.

Data Encryption Standard (DES)

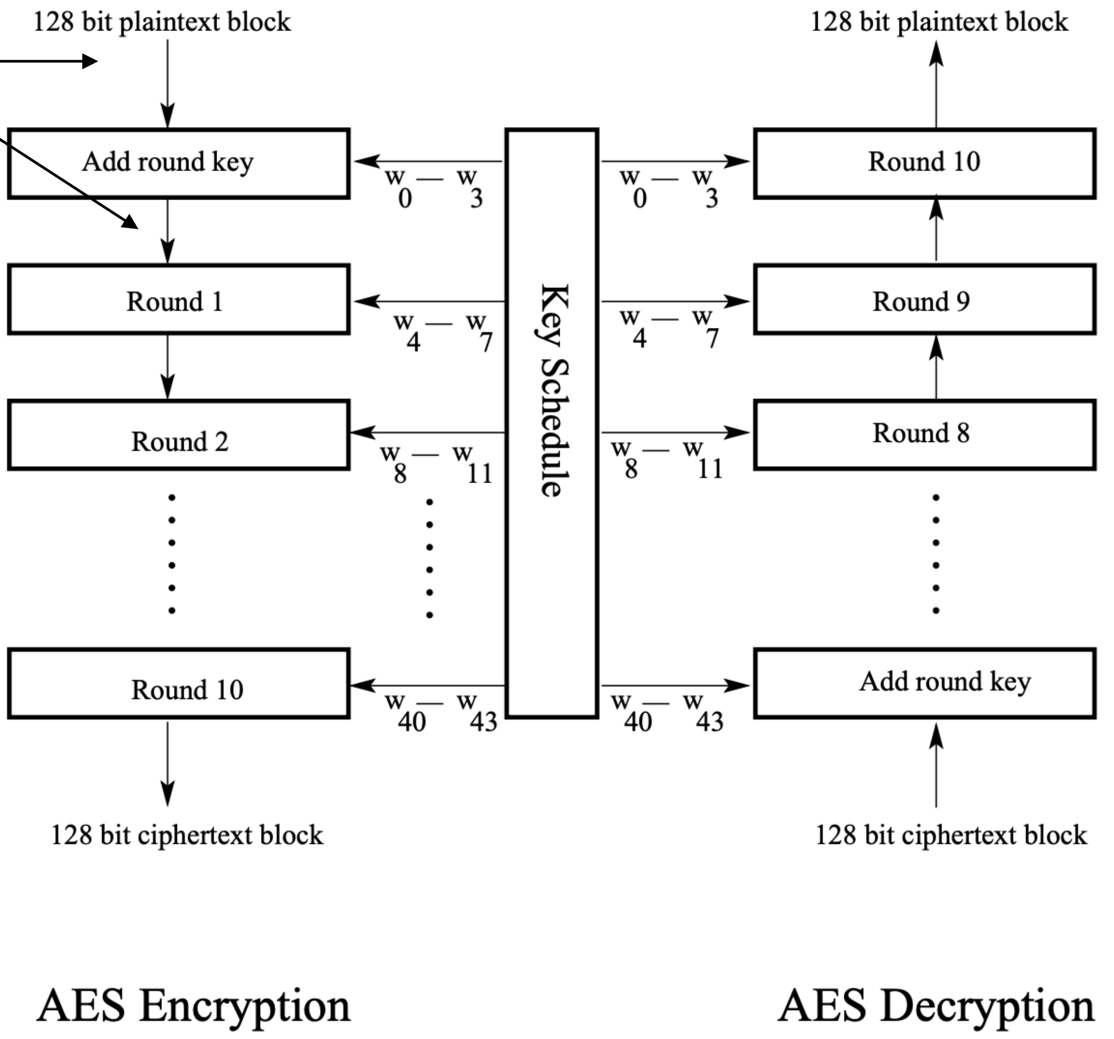
- DES is a block cipher - can only encrypt a block of data
- Block size for DES is 64 bits
- DES uses 56-bit keys although a 64-bit key is fed into the algorithm
- Theoretical attacks were identified. None was practical enough to cause major concerns.
- Triple DES can solve DES’s key size problem

Advanced Encryption Standard (AES)

- AES is a block cipher
- 128-bit block size.
- Three different key sizes: 128, 192, and 256 bits
- Uses symmetric encryption

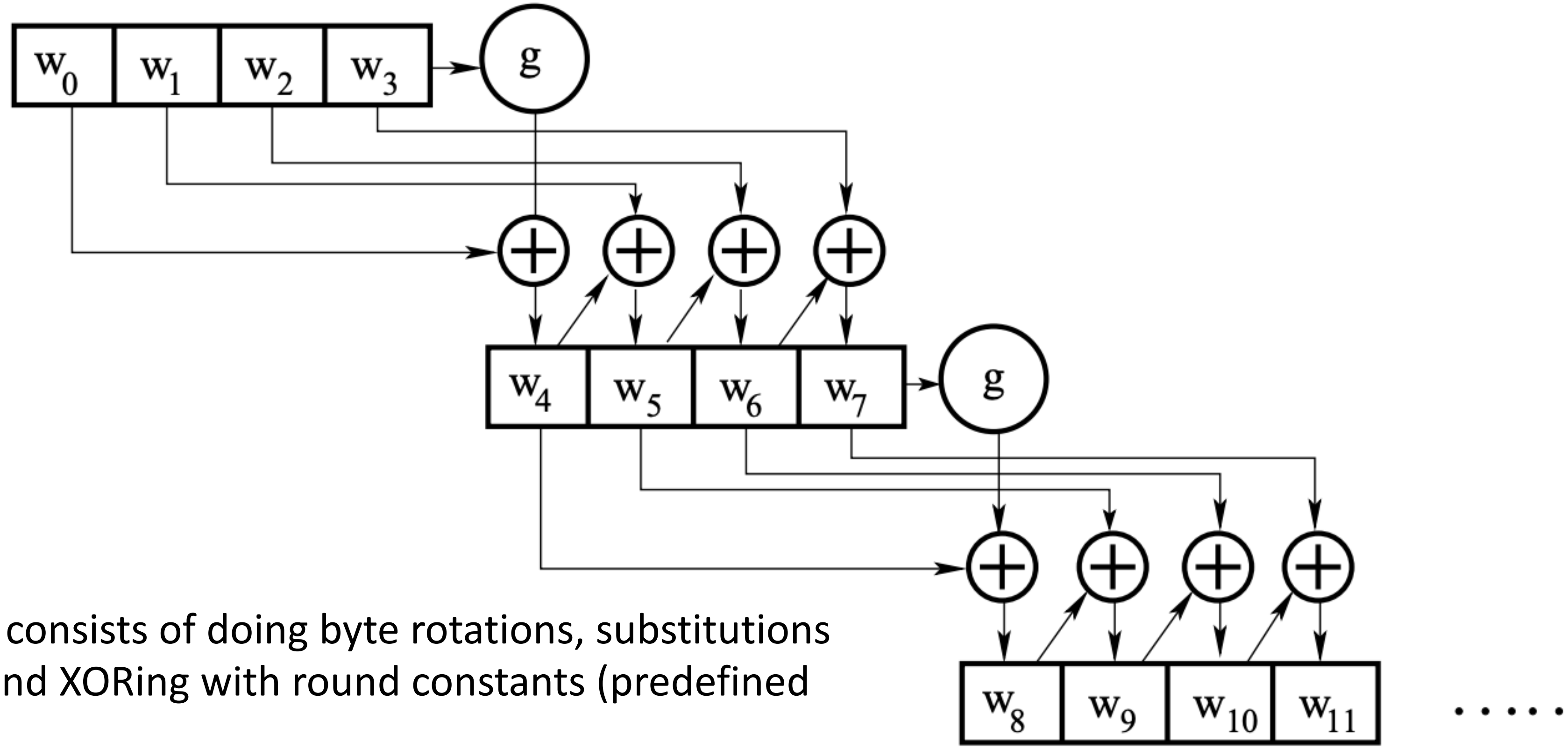
Advanced Encryption Standard (AES)

Known as **state array**



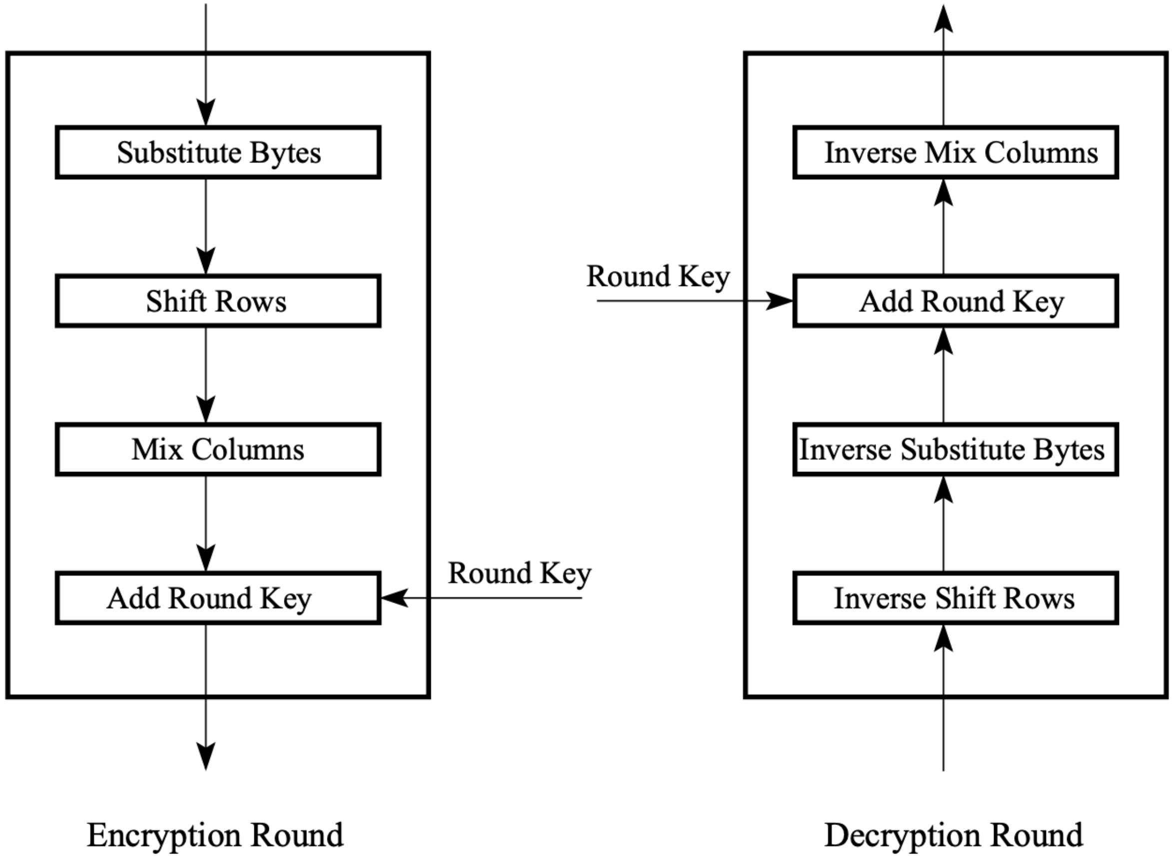
- Add round key simply XORs input state array with the original key in the case of encryption.
- Same happens for decryption except the last 4 words of the key schedule are used.

AES Key Expansion (Schedule)



The g() function consists of doing byte rotations, substitutions from the sbox, and XORing with round constants (predefined constant values)

AES Round



Substitute Bytes
Shift Rows
Mix Columns
Add Round Key
Round Key
Inverse Mix Columns
Round Key
Add Round Key
Inverse Substitute Bytes
Inverse Shift Rows
Encryption Round
Decryption Round

Byte Substitution (Substitute Bytes)

- We will focus on encryption (forward direction). Decryption is simply the reverse direction with inverse operations

- This step consists of using a 16×16 lookup table to find a replacement byte for a given byte in the input state array.

- The entries in the lookup table are created by using the notions of multiplicative inverses in Galois Fields (GF) and bit scrambling **to destroy the bit-level correlations inside each byte**

Byte Substitution (Substitute Bytes)

- To find the substitute byte for a given input byte, we divide the input byte into two 4-bit patterns, each yielding an integer value between 0 and 15. (We can represent these by their hex values 0 through F.)

- One of the hex values is used as a row index and the other as a column index for reaching into the 16×16 lookup table.

Byte Substitution (Substitute Bytes)

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Inverse S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Byte Substitution (Substitute Bytes)

$$\begin{pmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{pmatrix}$$

State array

$$\begin{pmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{pmatrix}$$

New state array after substitution

Shift Rows

- The shift row transformation consists of:

1. Not shifting the first row of the state array at all;
2. Circularly shifting the second row by one byte to the left;
3. Circularly shifting the third row by two bytes to the left; and
4. Circularly shifting the last row by three bytes to the left.

$$\begin{bmatrix} s_{0.0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1.0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2.0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3.0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} ===> \begin{bmatrix} s_{0.0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1.1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2.2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3.3} & s_{3,0} & s_{3,1} & s_{3,2} \end{bmatrix}$$

State array

New state array after shifting rows

Mix Columns

- This step replaces each byte of a column by a function of all the bytes in the same column.

- More precisely, each byte in a column is replaced by two times that byte, plus three times the next byte, plus the byte that comes next, plus the byte that follows.

- The words ‘next’ and ‘follow’ refer to bytes in the same column, and their meaning is circular, in the sense that the byte that is next to the one in the last row is the one in the first row.

State array

New state array after shifting rows

Mix Columns

- For the bytes in the first row in the state array apply:

$$s'_{0,j} \quad = \quad (0\mathbf{x}02 \times s_{0,j}) \otimes (0\mathbf{x}03 \times s_{1,j}) \otimes s_{2,j} \otimes s_{3,j}$$

- For the bytes in the first row in the state array apply:

$$s'_{1,j} \quad = \quad s_{0,j} \otimes (0\mathbf{x}02 \times s_{1,j}) \otimes (0\mathbf{x}03 \times s_{2,j}) \otimes s_{3,j}$$

- For the third the bytes in the first row in the state array apply:

$$s'_{2,j} \quad = \quad s_{0,j} \otimes s_{1,j} \otimes (0\mathbf{x}02 \times s_{2,j}) \otimes (0\mathbf{x}03 \times s_{3,j})$$

- For the fourth the bytes in the first row in the state array apply:

$$s'_{3,j} \quad = \quad (0\mathbf{x}03 \times s_{0,j}) \otimes s_{1,j} \otimes s_{2,j} \otimes (0\mathbf{x}02 \times s_{3,j})$$

Mix Columns

- More compactly, the column operations can be shown as

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_{0.0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1.0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2.0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3.0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0.0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1.0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2.0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3.0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- Where, on the left hand side, when a row of the leftmost matrix multiples a column of the state array matrix, additions involved are meant to be XOR operations.

Add Round Key

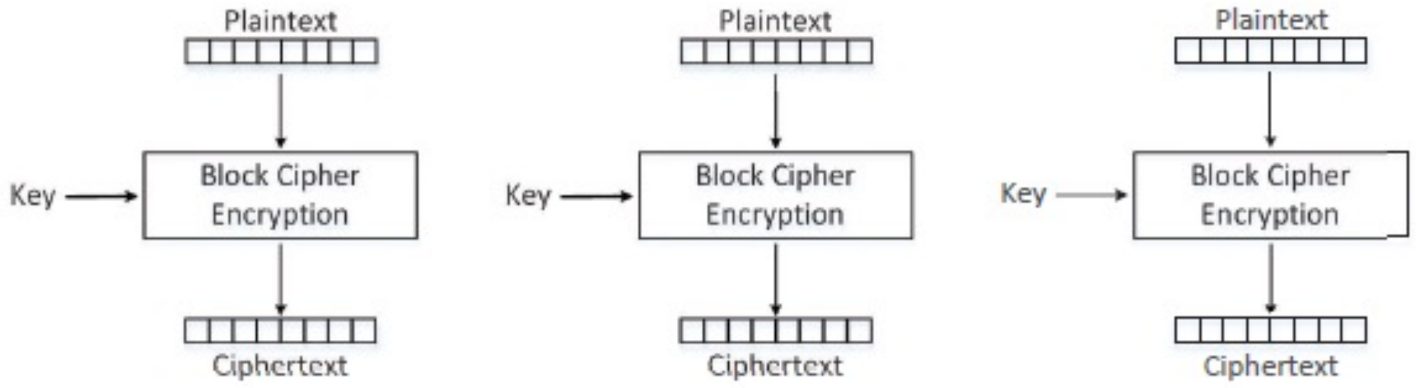
- This is the last step which simply consists of XORing the output of the Mix Columns step with four words from the key schedule.

- See for a full encryption example:
  - https://www.kavaliro.com/wp-content/uploads/2014/03/AES.pdf

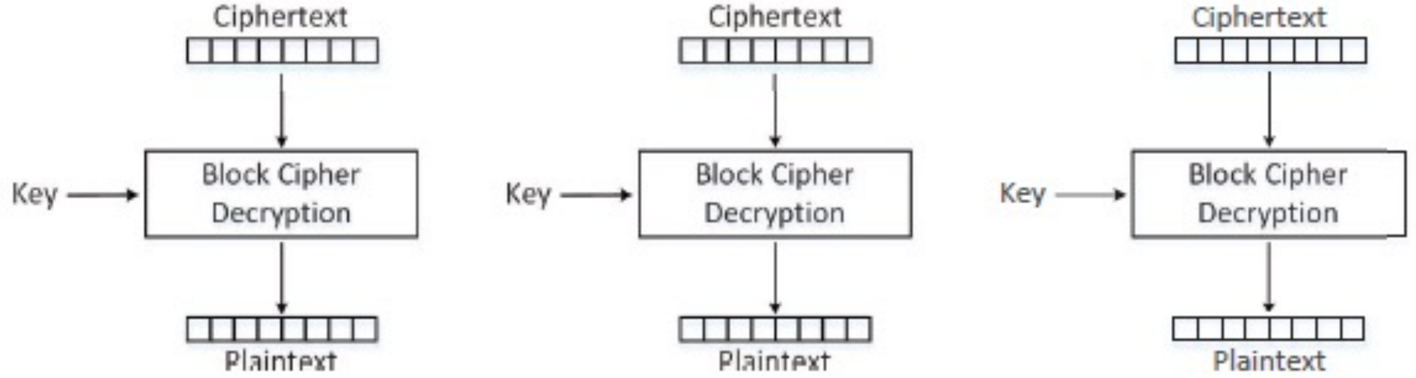
Encryption Modes

- Encryption mode or mode of operation refers to the many ways to make the input of an encryption algorithm different.
- Examples include:
  - Electronic Codebook (ECB)
  - Cipher Block Chaining (CBC)
  - Propagating CBC (PCBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)

Electronic Codebook (ECB) Mode



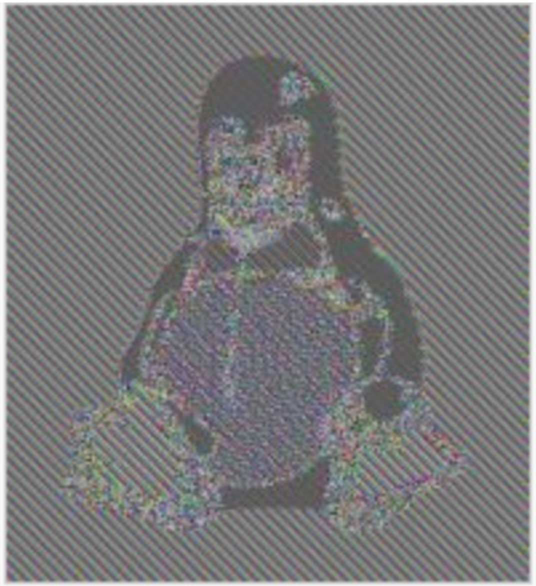
(a) Electronic Codebook (ECB) mode encryption



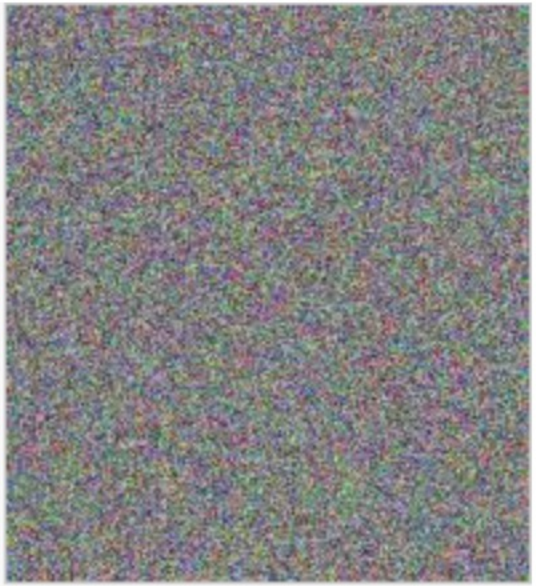
(b) Electronic Codebook (ECB) mode decryption

Weakness of ECB Mode

Original image

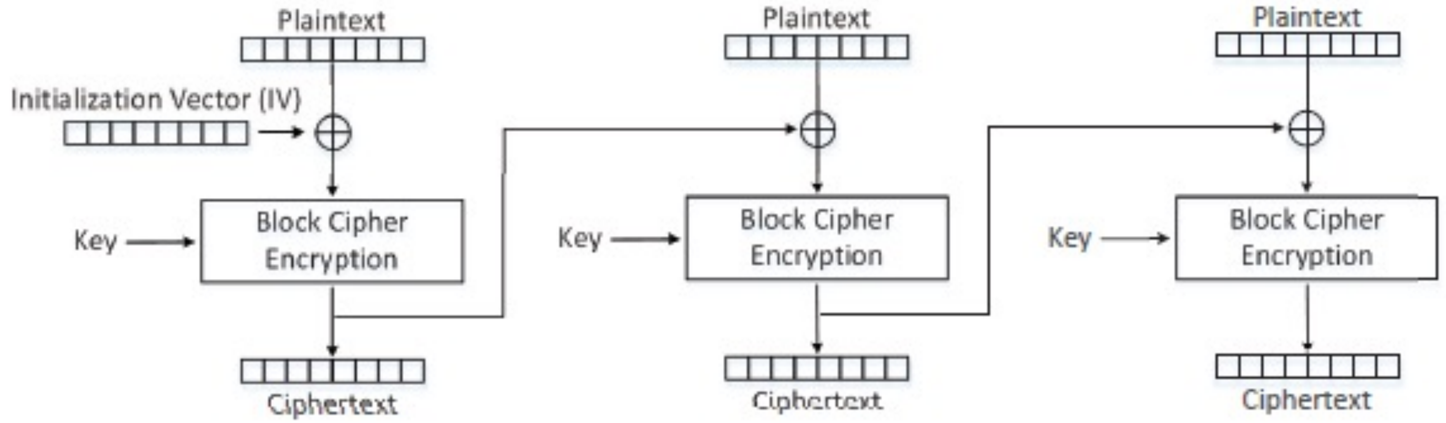


Encrypted using ECB mode

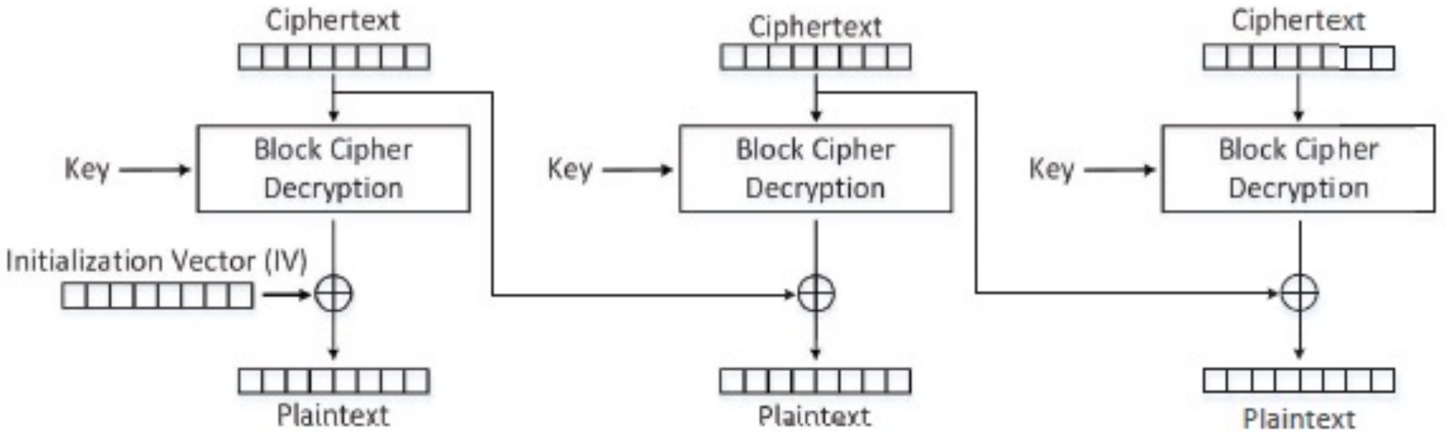


Modes other than ECB result in pseudo-randomness

Cipher Block Chaining (CBC) Mode



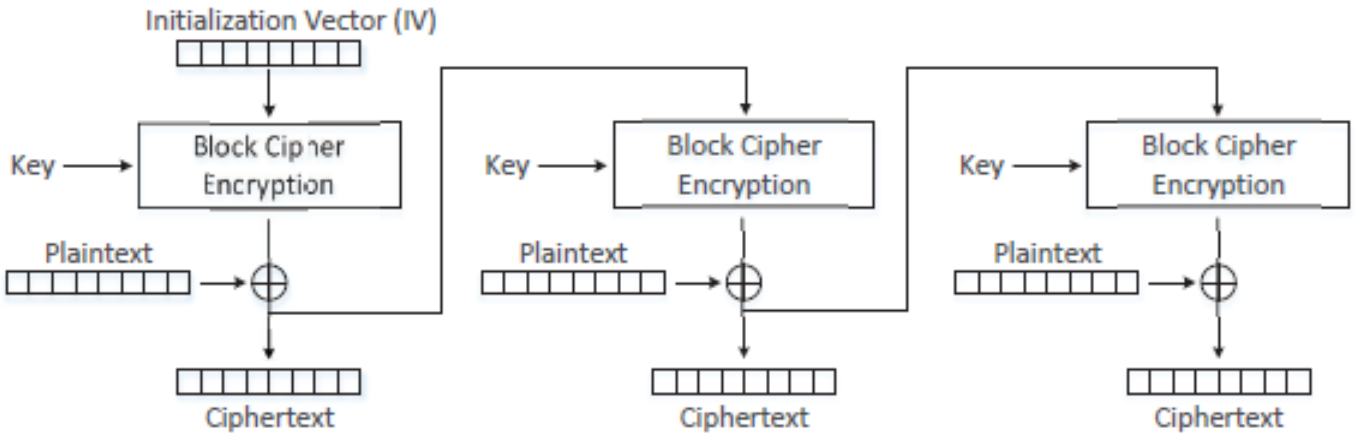
(a) Cipher Block Chaining (CBC) mode encryption



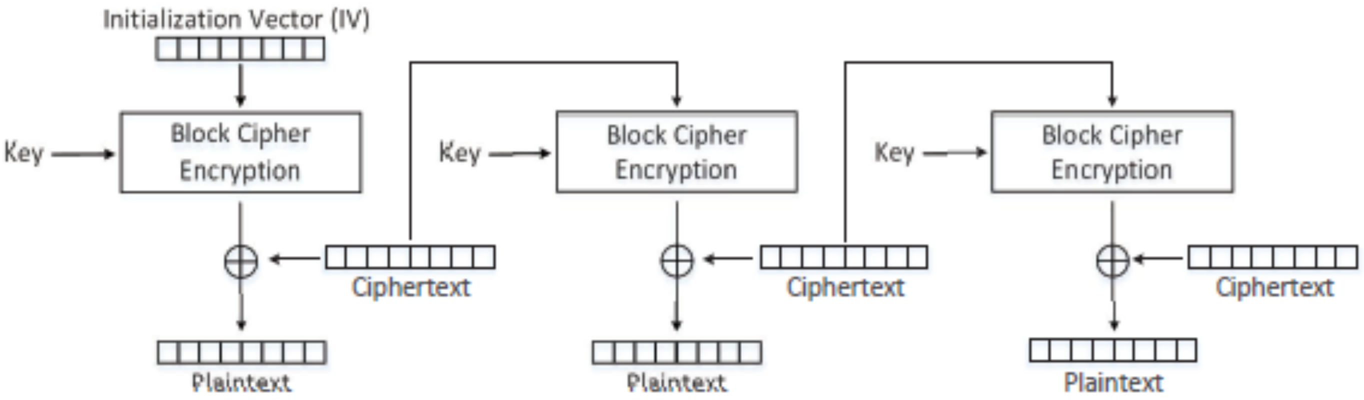
(b) Cipher Block Chaining (CBC) mode decryption

- The main purpose of **IV** is to ensure that even if two plaintexts are identical, their ciphertexts are still different, because different IVs will be used.
- Decryption **can** be parallelized
- Encryption **cannot** be parallelized

Cipher Feedback (CFB) Mode



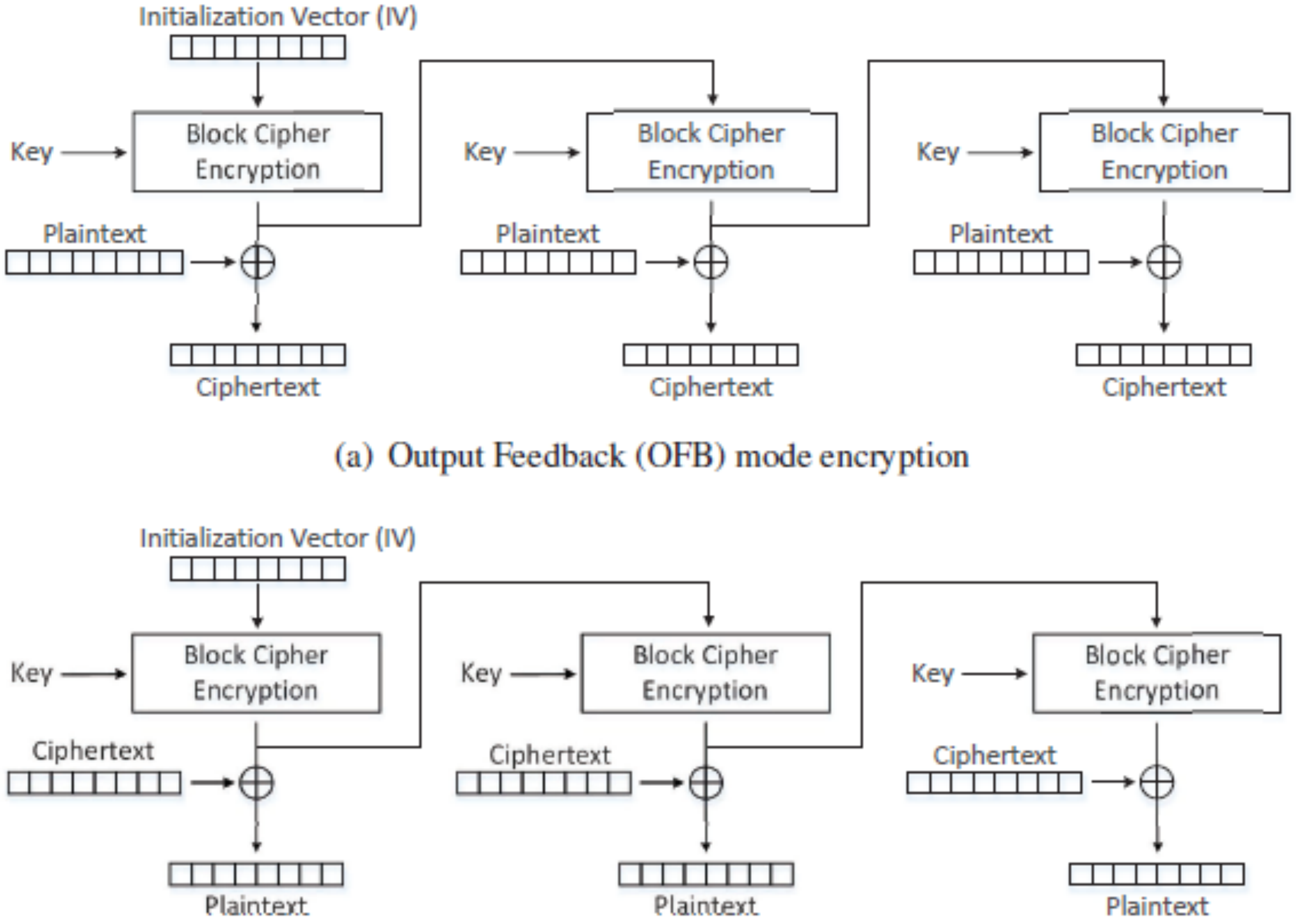
(a) Cipher Feedback (CFB) mode encryption



(b) Cipher Feedback (CFB) mode decryption

- A block cipher is turned into a stream cipher.
- Ideal for encrypting real-time data.
- Padding not required for the last block.
- decryption using the CFB mode can be parallelized, while encryption can only be conducted sequentially

Output Feedback (OFB) Mode

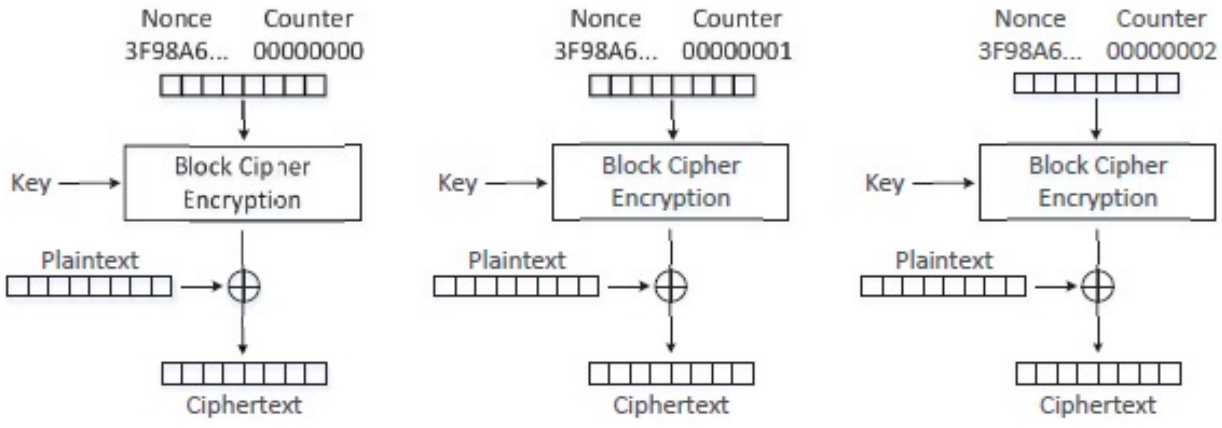


(a) Output Feedback (OFB) mode encryption

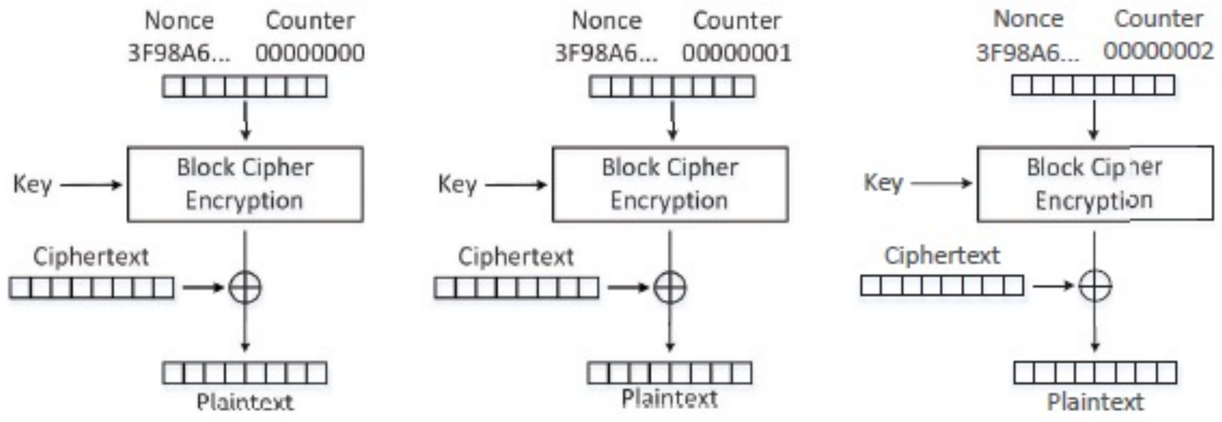
(b) Output Feedback (OFB) mode decryption

- Similar to CFB
  - Used as stream cipher
  - Does not need padding
  - Decryption can parallelized
- Encryption in the OFB mode can be parallelized

Counter (CTR) Mode



(a) Counter (CTR) mode encryption



(b) Counter (CTR) mode decryption

- It basically uses a counter to generate the key streams
- no key stream can be reused, hence the counter value for each block is prepended with a randomly generated value called *nonce*
- This nonce serves the same role as the IV does to the other encryption modes.
- both encryption and decryption can be parallelized
- the key stream in the CTR mode can be calculated in parallel during the encryption

Padding

- Block cipher encryption modes divide plaintext into blocks and the size of each block should match the cipher's block size.
- No guarantee that the size of the last block matches the cipher's block size.
- Last block of the plaintext needs **padding** i.e. before encryption, extra data needs to be added to the last block of the plaintext, so its size equals to the cipher's block size.
- Padding schemes need to clearly mark where the padding starts, so decryption can remove the padded data.
- Commonly used padding scheme is PKCS#5

Initial Vector

- Initial vectors have the following requirements:
  - IV is supposed to be stored or transmitted in plaintext
  - IV should not repeat (uniqueness).
  - IV should not be predictable.

Public Key Cryptography

Introduction

- Foundation of today’s secure communication
- Allows communicating parties to obtain a shared secret key
- Public key (for encryption) and Private key (for decryption) – the algorithm allows for this to be reversed
- Private key (for digital signature) and Public key (to verify signature)

Modulo Operation

- The RSA algorithm is based on modulo operations
- `a mod n` is the remainder after division of a by the modulus n
- Second number is called modulus
- For example, (10 mod 3) equals to 1 and (15 mod 5) equals to 0
- Modulo operations are distributive:

$$
\begin{aligned}
(a+b) \mod n &= [(a \mod n) + (b \mod n)] \mod n \\
a * b \mod n &= [(a \mod n) * (b \mod n)] \mod n \\
a^x \mod n &= (a \mod n)^x \mod n
\end{aligned}
$$

Euler’s Theorem

- Euler’s totient function $\varphi(n)$ counts the positive integers up to a given integer n that are relatively prime to n
- $\varphi(n) = n - 1$, if n is a prime number. (numbers < n not divisible by n)
- Euler’s totient function property:
  - if m and n are relatively prime, $\varphi(mn) = \varphi(m) * \varphi(n)$
- Euler’s theorem states:
  - $a^{\varphi(n)} = 1 \pmod{n}$

Extended Euclidean Algorithm

- Euclid's algorithm: efficient method for computing GCD
- Extended Euclidean algorithm:
  - computes GCD of integers a and b
  - finds integers x and y, such that: $ax + by = gcd(a, b)$
- RSA uses extended Euclidean algorithm:
  - e and n are components of public key
  - Find solution to equation:
    $e * x + \varphi(n) * y = gcd(e, \varphi(n)) = 1$
  - x is private key (also referred as d)
  - Equation results: $e * d \bmod \varphi(n) = 1$

RSA Algorithm

Consists of:

- Key generation
- Encryption
- Decryption

RSA: Key Generation

- Choose two prime numbers p and q and compute n = p * q (should be large)
  - Calculate the totient $\varphi(n)$
- Select e (there may be multiple values) such that gcd(e, $\varphi(n)$) = 1 and 1 < e < $\varphi(n)$ (e is a relative prime)
  - In other words, e is relatively prime to ϕ(n)
  - Find d, $ed \bmod \varphi(n) = 1$
- Find d such that e * d mod $\varphi(n) = 1$
  - In other words e & d are relative primes of n

RSA: Key Generation

- (e,n) is public key
- d is private key, for completion we include (d, n)

RSA: Encryption/Decryption

- Encryption of plaintext M, where M < n:
  - $C = M^e \mod n$

- Decryption of ciphertext C:
  - $M = C^d \mod n$

RSA: Example

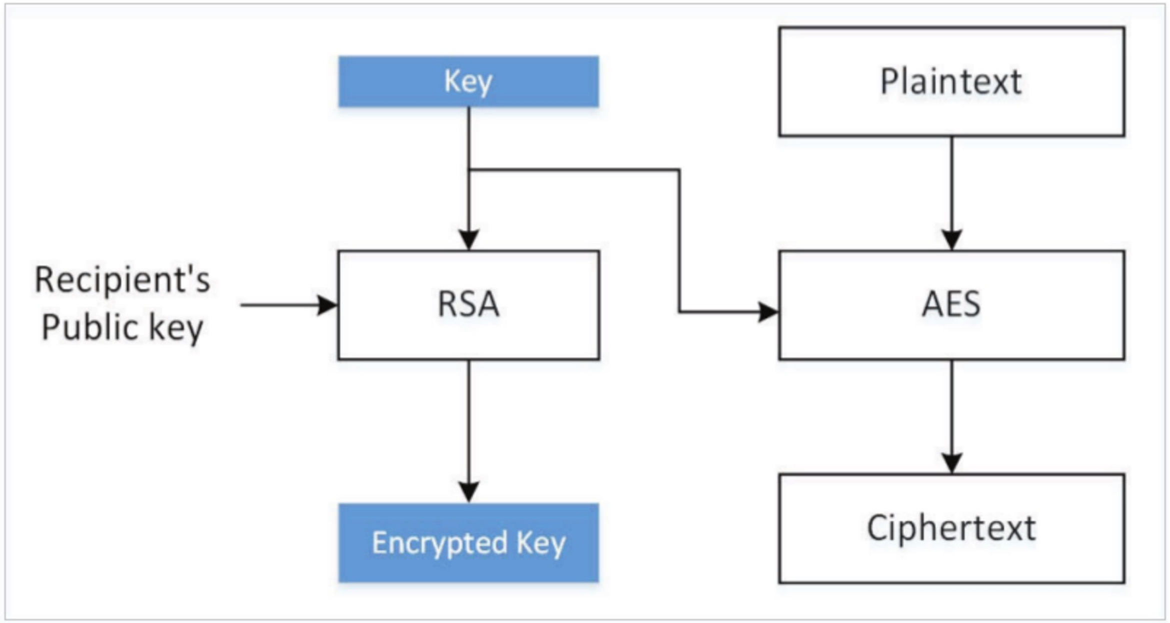
- $p = 13$, $q = 23$
- $n = 13 * 23 = 299$
- $\varphi(n) = (p - 1) * (q - 1) = 12 * 22 = 264$
- Choose e: $\gcd(e, 264) = 1$
  - Valid e values are 5, 7, etc. (satisfies above equation)
  - Assume $e = 5$ so public key is $PU = (e = 5, n = 299)$
- Choose d that satisfies: $e * d \bmod 264 = 1$
  - $d = 53$ satisfies the above
  - Private key is $PR = (d = 53, n = 299)$

RSA: Example (Contd.)

- Let's now encrypt the message M = 15 using:
  - $C = M^e \mod n$
  - C = 15 ^ 5 mod 299 = 214
- Let's now decrypt the ciphertext C = 214 using:
  - $M = C^d \mod n$
  - M = 214 ^ 53 mod 299 = 15

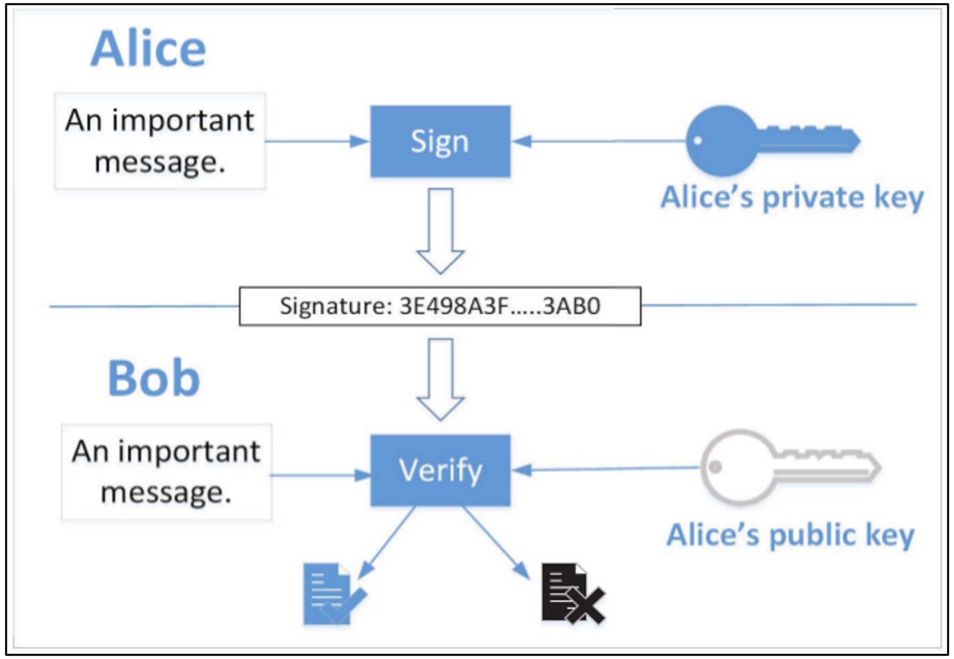
Hybrid Encryption

- High computation cost of public-key encryption
- Public key algorithms used to exchange a secret session key
- Key (content-encryption key) used to encrypt data using a symmetric-key algorithm



Digital Signature

- Goal: provide an authenticity proof by signing digital documents
- RSA authors developed the first digital signature algorithm



Digital Signature using RSA

- Apply private-key operation on m using private key, and get a number s, everybody can get the m back from s using our public key
- For a message m that needs to be signed:

  Digital signature = $m^d \bmod n$

- In practice, message may be long resulting in long signature and more computing time
- Instead, we generate a cryptographic hash value from the original message, and only sign the hash

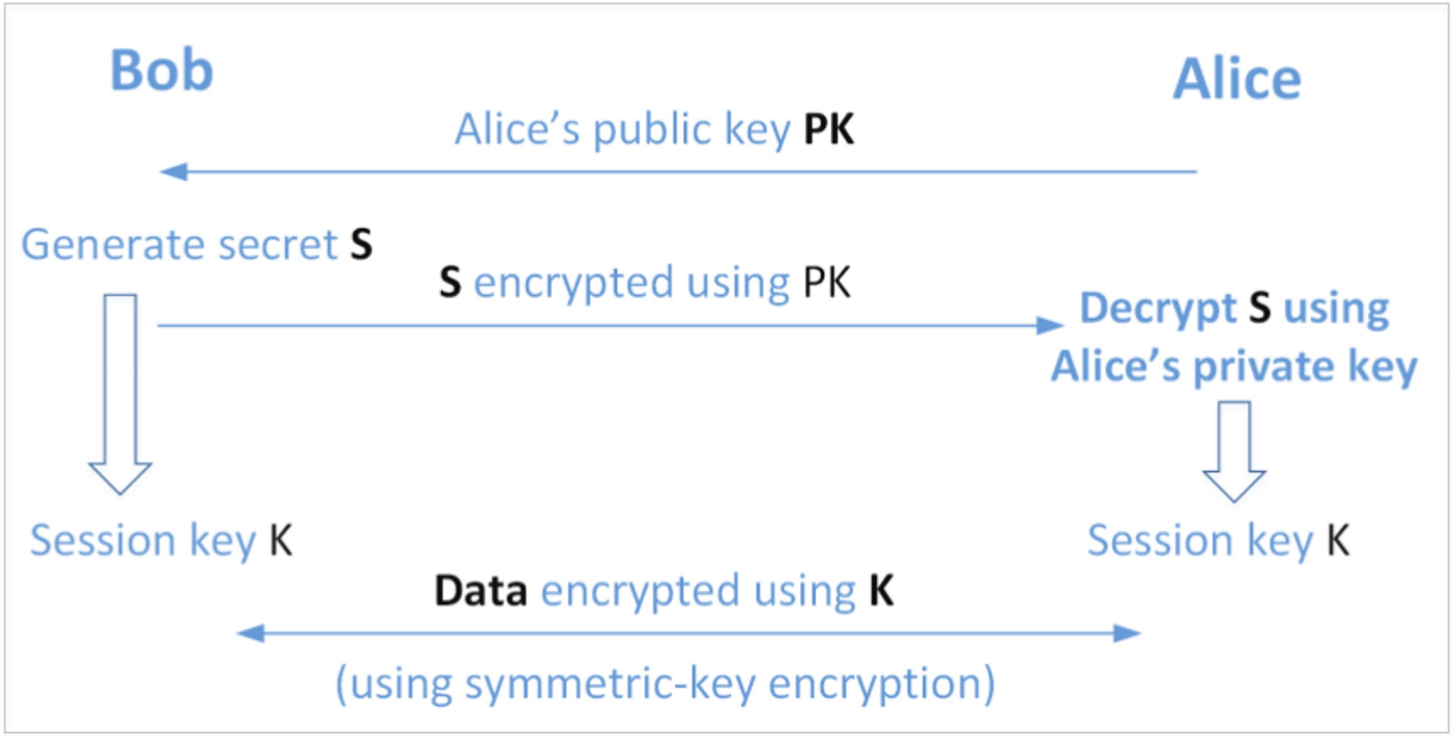
Applications

- Authentication
- HTTPS and TLS/SSL
- Chip Technology Used in Credit Cards

Applications: HTTPS and TLS/SSL

- HTTPS protocol is used to secure web services
- HTTPS is based on the TLS/SSL protocol (uses both public key encryption and signatures)
  - encryption is done through secret-key encryption algorithms (e.g. AES)
  - public key algorithms are mainly used for key exchange (e.g. RSA)

Applications: HTTPS and TLS/SSL (Contd.)

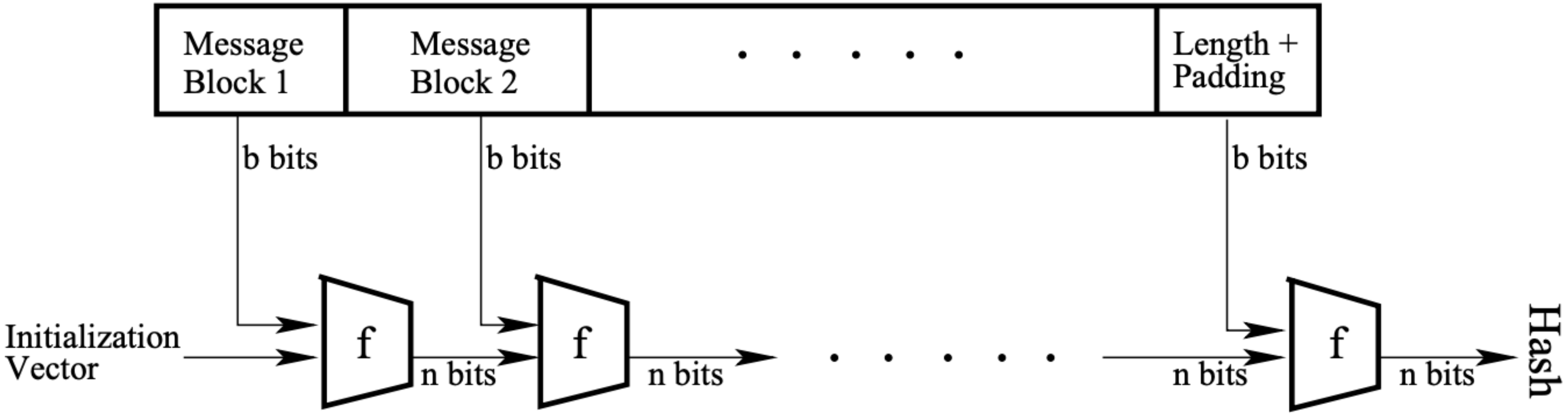


Hash MACs (HMACs)

Hash Function Properties

- Very fast to compute
- Takes arbitrary size and returns fixed-sized output
- Pre-image resistant
  - Given H(m), hard to determine m (one way function)
- Collision resistant
  - Given m and H(m), hard to find m’ != m s.t. H(m) = H(m’)
- Good hash functions: SHA family (SHA-256, SHA-512, etc.)

Hash Function Structure




```
Message:            "The quick brown fox jumps over the lazy dog"
SHA1 hashcode:      2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

Message:            "The quick brown  fox jumps over the lazy dog"
SHA1 hashcode:      8de49570b9d941fb26045fa1f5595005eb5f3cf2
```

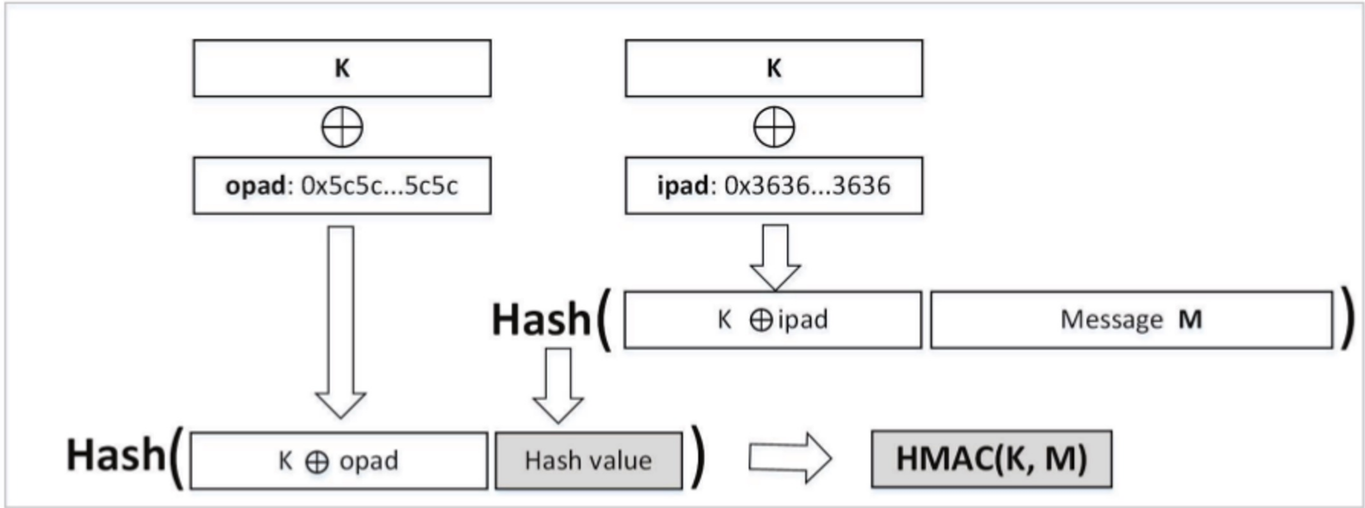
Hash MACs

- Sign(k,m)
- opad = 0x5c5c5c5c… (removes regularity in key)
- ipad = 0x36363636… (removes regularity in key)
- Append to the message H((k xor opad) || H((k xor ipad) || m))

- Then verify:
  - Recompute and compare to determine if message was tampered with

Keyed-Hash MAC (HMAC)

- Uses hash function H (compression function block size B) and a secret key K
- ipad = 0x36 (B times), opad = 0x5c (B times)
- Can be used with any one-way hash function



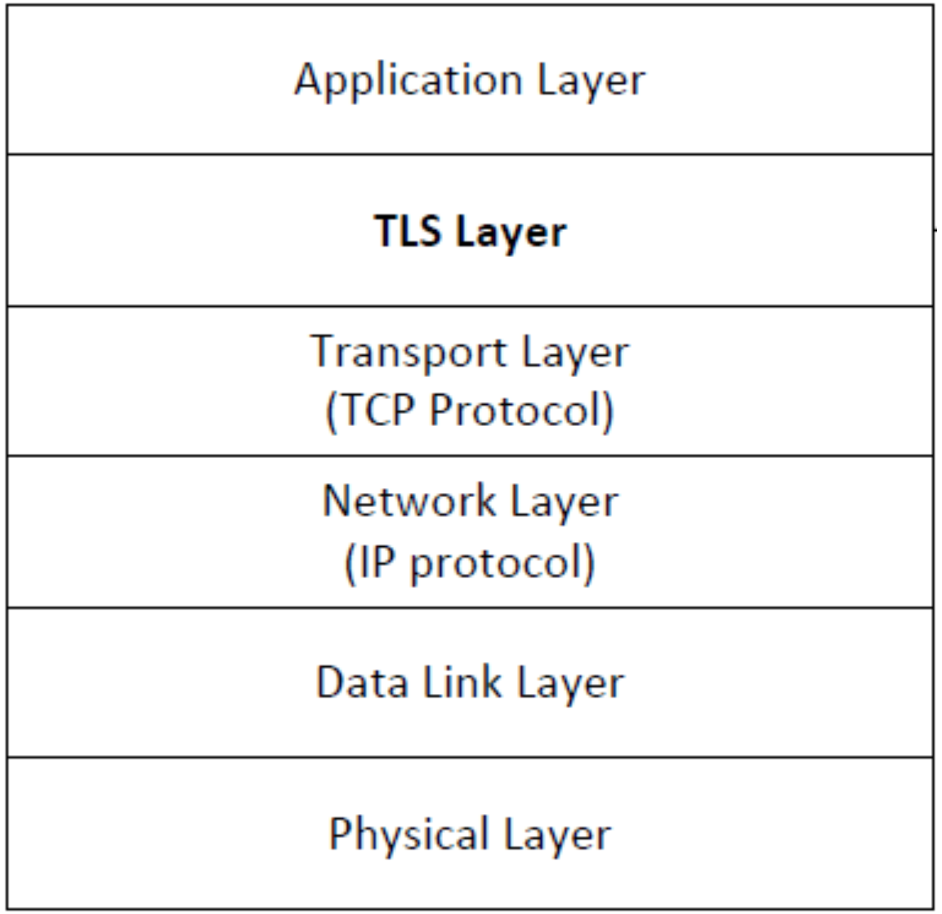
Transport Layer Security

Overview of TLS

- Transport Layer Security (TLS) is a protocol that provides a secure channel between two communicating applications.

TLS Layer

- TLS sits between the Transport and Application layer
  - Unprotected data is given to TLS by Application layer
  - TLS handles encryption, decryption and integrity checks
  - TLS gives protected data to Transport layer



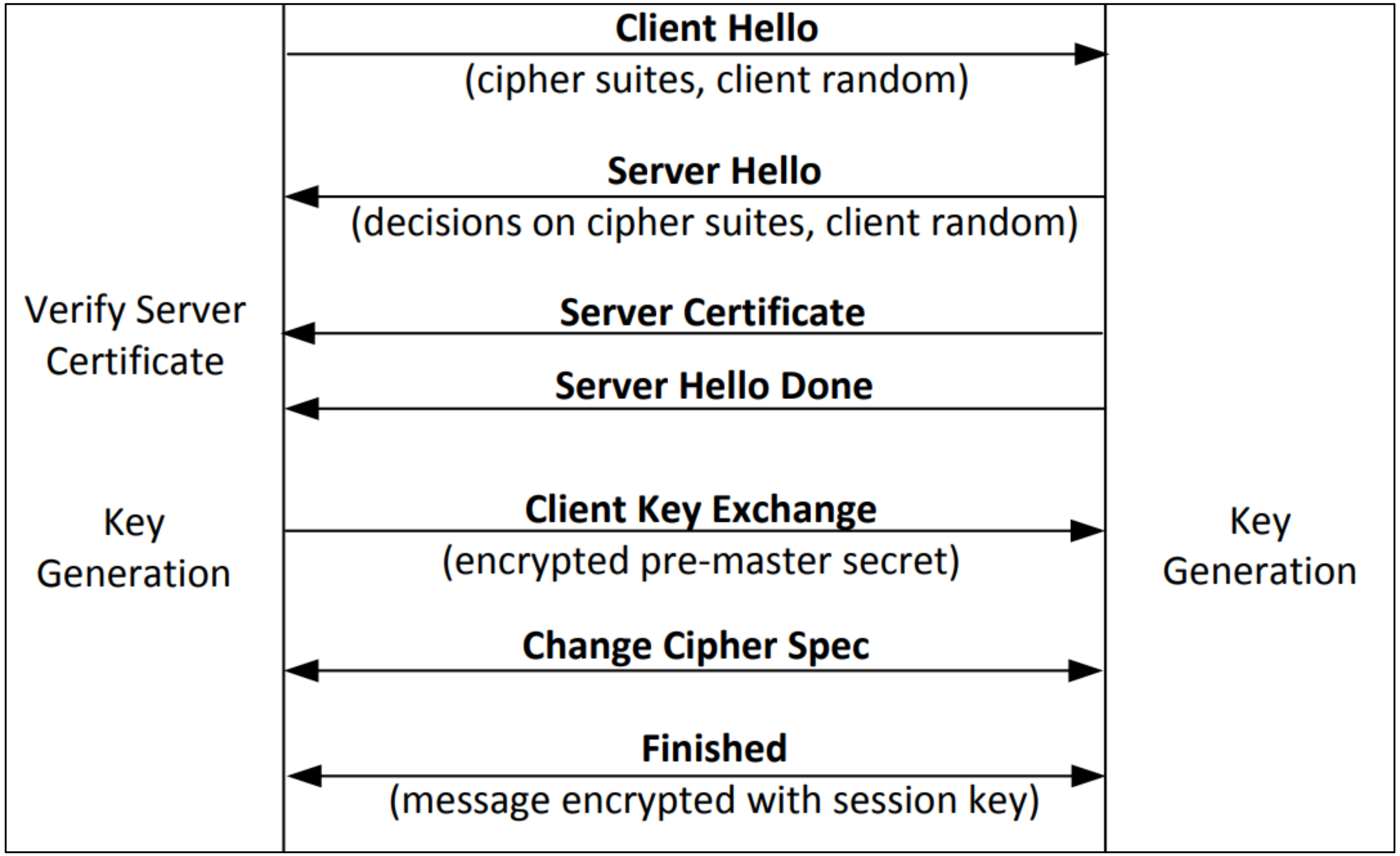
TLS Handshake

- Before a client and server can communicate securely, several things need to be set up first:
  - Encryption algorithm and key
  - MAC algorithm
  - Algorithm for key exchange
- These cryptographic parameters need to be agreed upon by the client and server
- This is the primary purpose of the handshake protocol

TLS Protocols

- TLS is a layered protocol which has 5 message layers
  - Handshake Protocol: responsible for establishing a secure channel
  - Alert Protocol: used for reporting cause of failure between peers
  - Change Cipher Spec Protocol: Used to change the encryption method used between the client and server. Normally used in the handshake protocol to switch to symmetric key encryption, e.g. AES
  - Heartbeat Protocol: Used to keep an established TLS session alive
  - Application Protocol: Used for data transmission

TLS Handshake Protocol

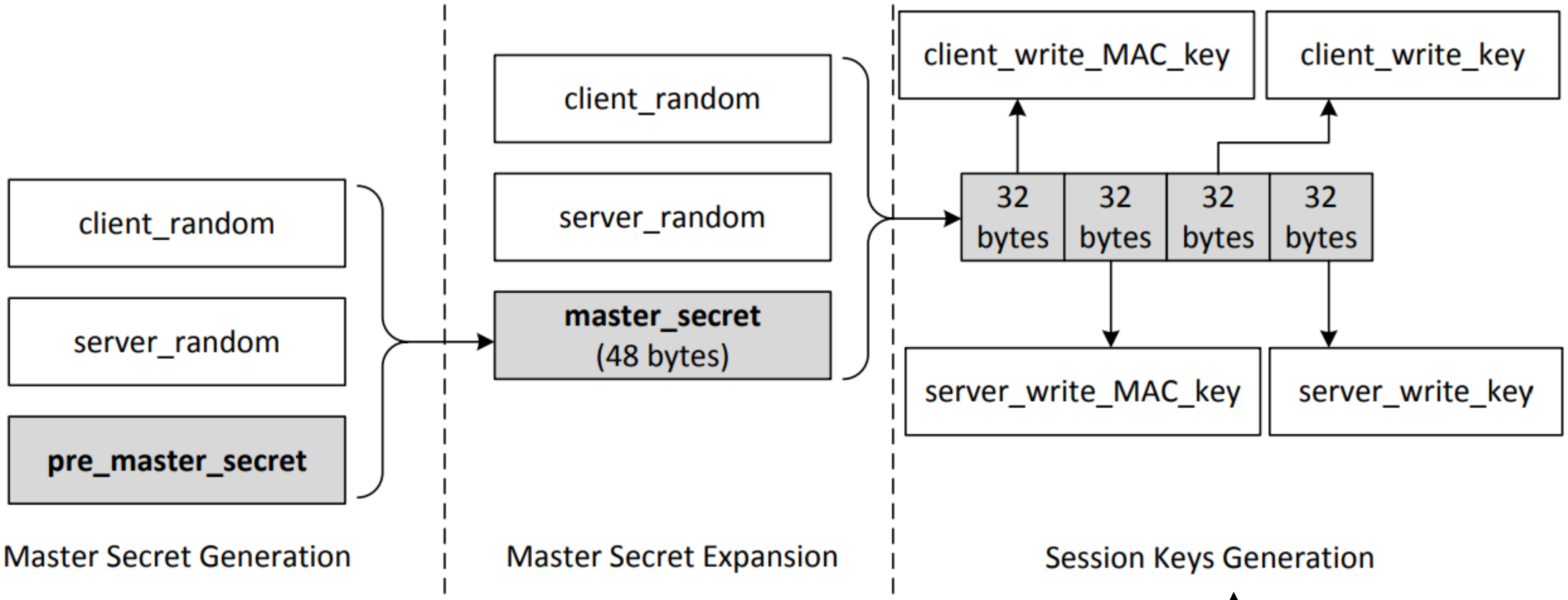


Key Generation and Exchange

- Although public-key algorithms can be used to encrypt data, it is much more expensive than secret-key algorithms.
  - TLS uses PKI for key exchange.
  - After that, server and client switch to secret-key encryption algorithm

- The entire key generation consists of three steps:
  - Step 1: Generating pre-master secret
  - Step 2: Generating master secret
  - Step 3: Generating session keys

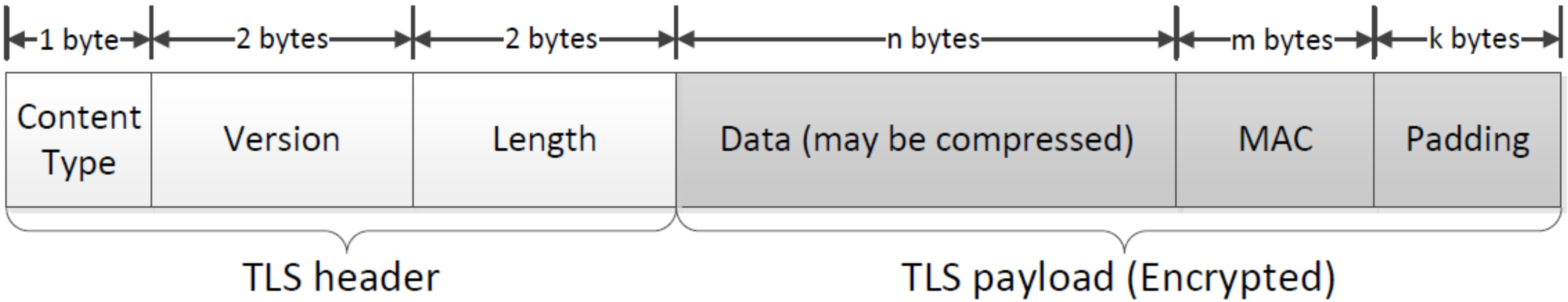
Key Generation and Exchange



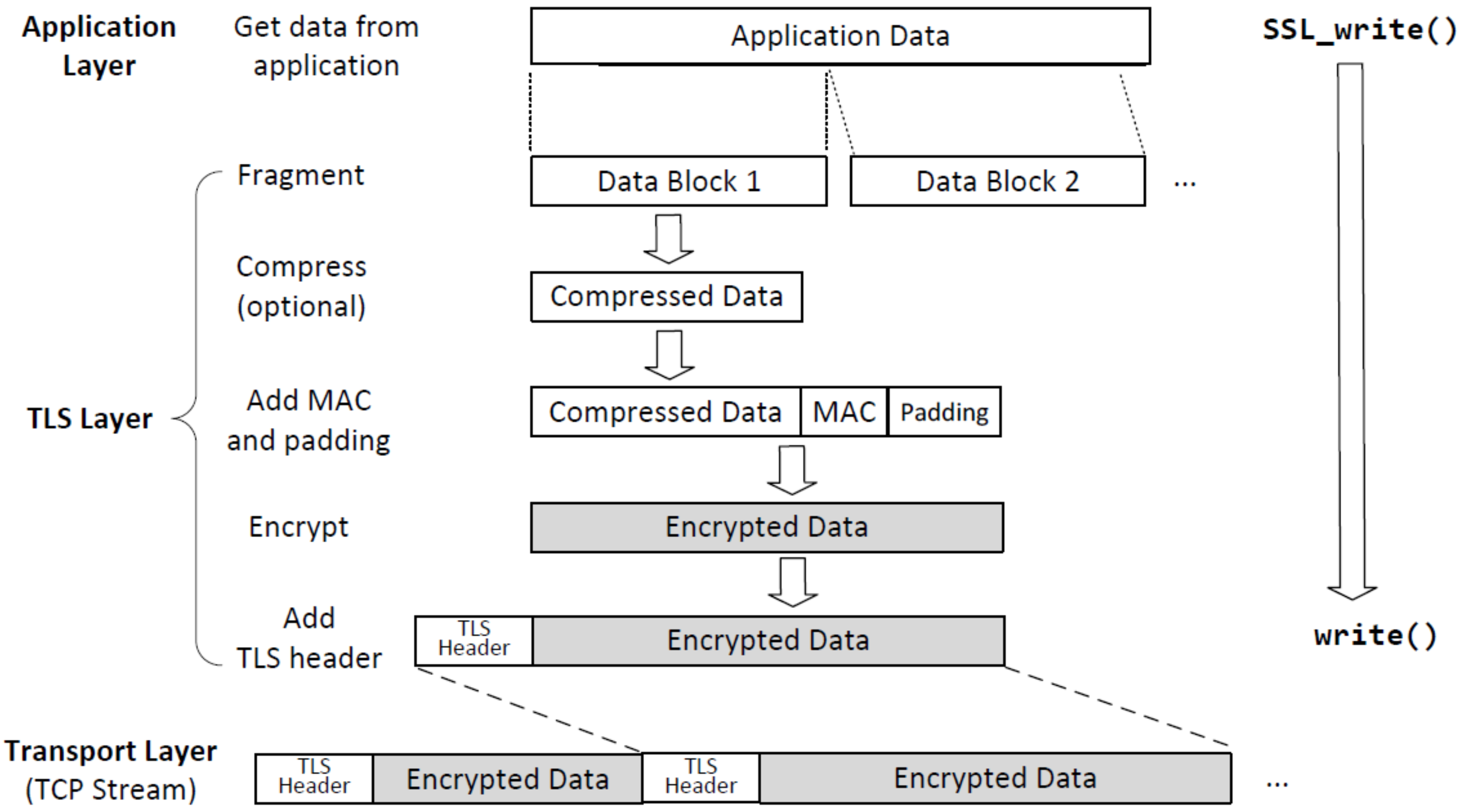
These keys are used to protect an SSL session

TLS Data Transmission

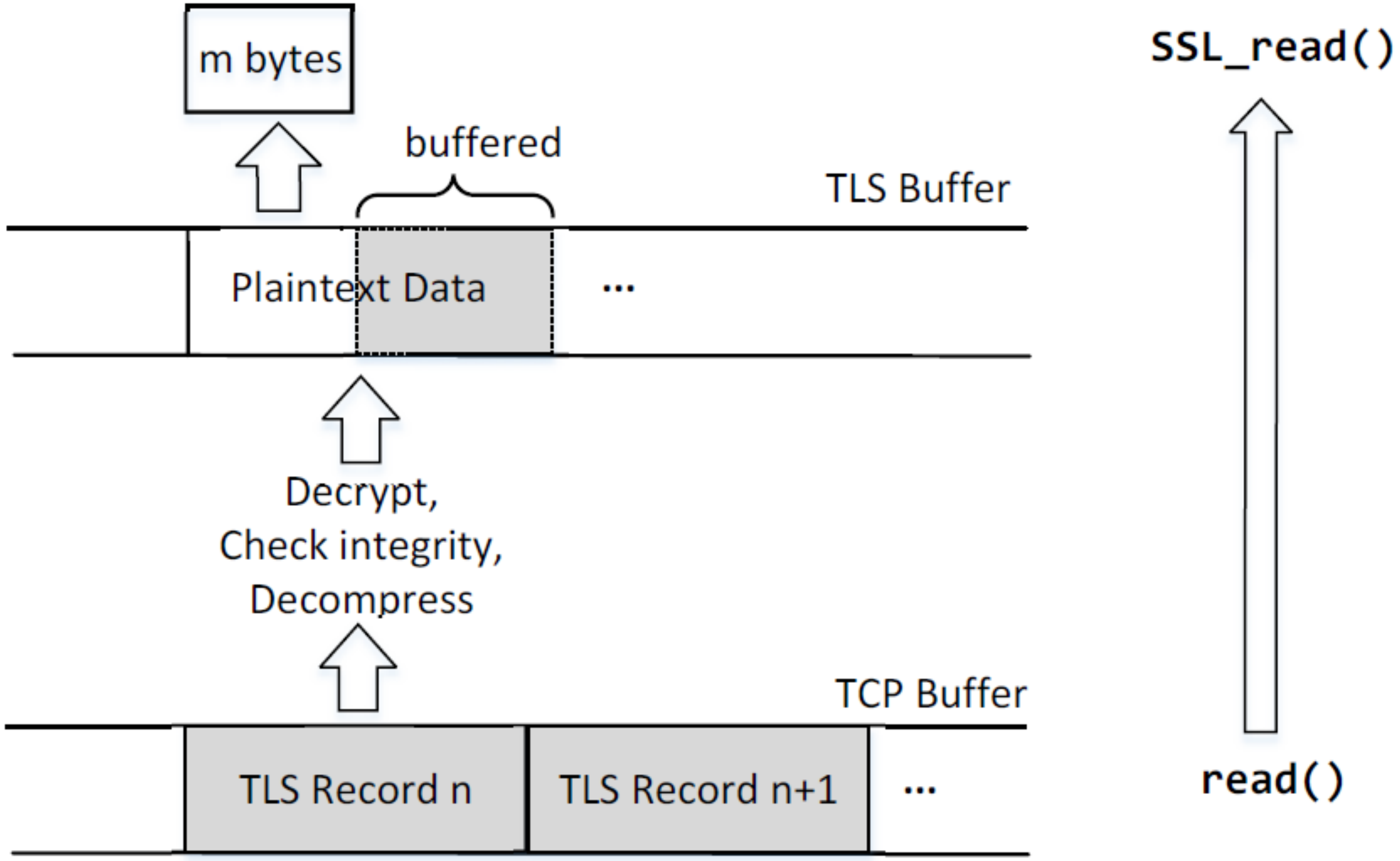
- Once the handshake protocol is finished, client and server can start exchanging data.
- Data is transferred using records.
- Each record contains a header and a payload



Sending Data with the TLS Record Protocol



Receiving Data with the TLS Record Protocol



Network Traffics During TLS Handshake

Since TLS runs top of TCP, a TCP connection needs to be established before the handshake protocol. This is how the packet exchange looks between a client and server during a TLS handshake protocol captured using Wireshark:

No.	Source	Destination	Protocol	Info
1	10.0.2.45	10.0.2.35	TCP	59930 –> 11110 [SYN] Seq=0 Win=14600 Len=0 MSS=1460...
2	10.0.2.35	10.0.2.45	TCP	11110 –> 59930 [SYN, ACK] Seq=0 Ack=1 Win=14480...
3	10.0.2.45	10.0.2.35	TCP	59930 –> 11110 [ACK] Seq=1 Ack=1 Win=14720 Len=0...
4	10.0.2.45	10.0.2.35	TLSv1.2	Client Hello
6	10.0.2.35	10.0.2.45	TLSv1.2	Server Hello, Certificate, Server Hello Done
8	10.0.2.45	10.0.2.35	TLSv1.2	Client Key Exchange, Change Cipher Spec, Finished
9	10.0.2.35	10.0.2.45	TLSv1.2	New Session Ticket, Change Cipher Spec, Finished

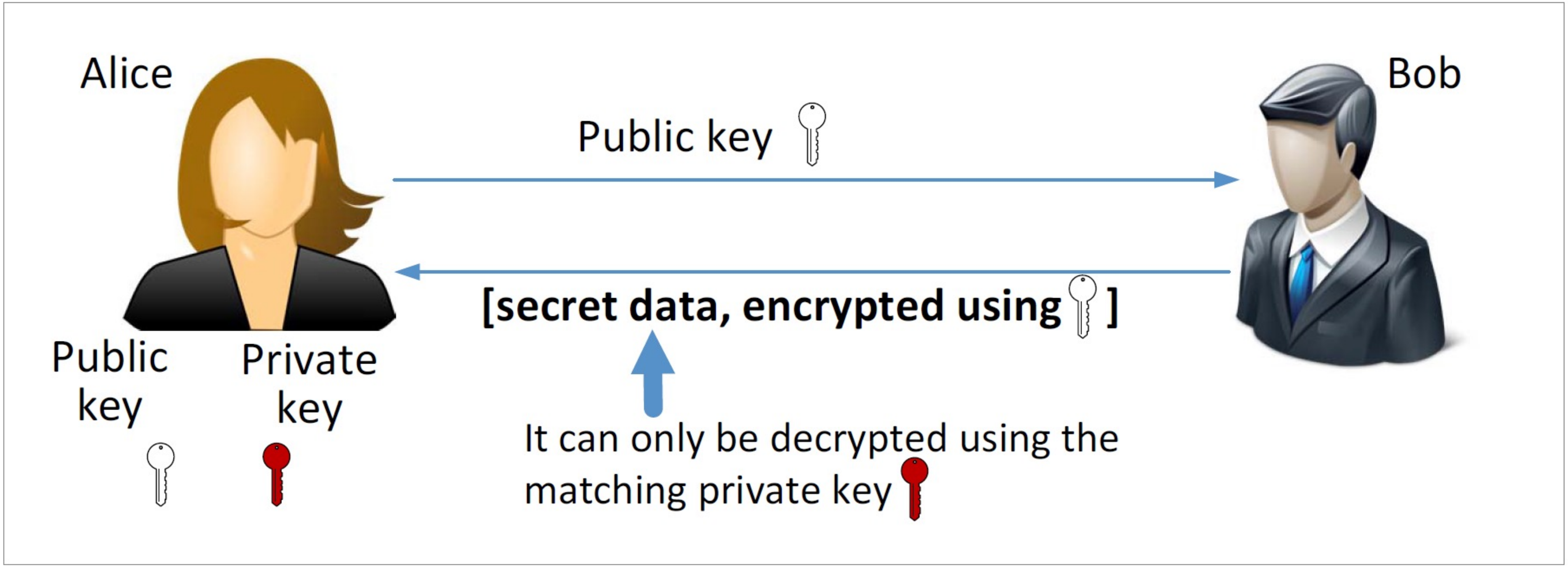
Key Generation and Exchange

- Although public-key algorithms can be used to encrypt data, it is much more expensive than secret-key algorithms.
  - TLS uses PKI for key exchange.
  - After that, server and client switch to secret-key encryption algorithm

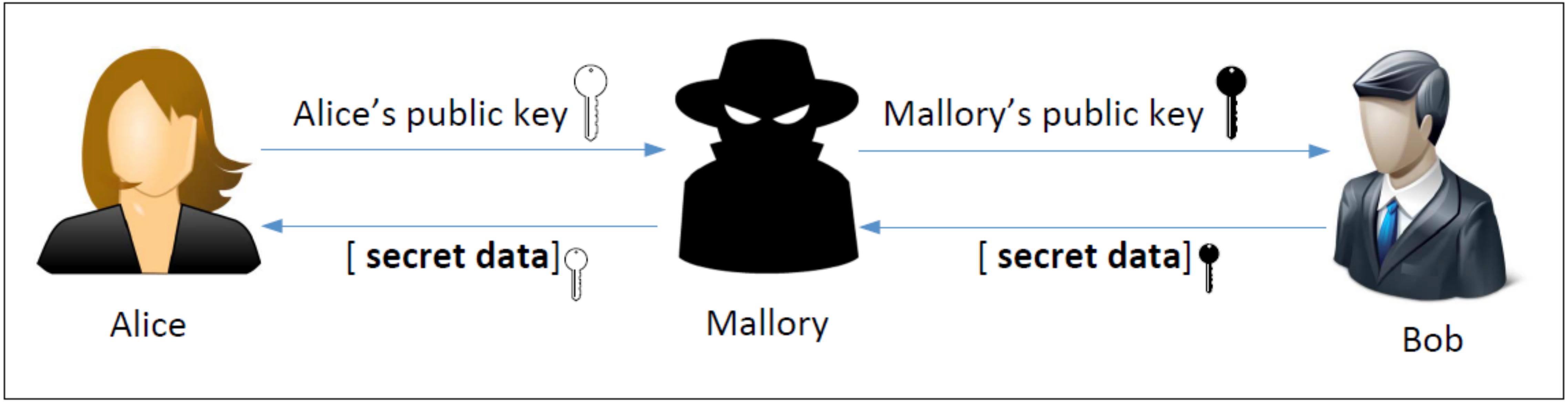
- The entire key generation consists of three steps:
  - Step 1: Generating pre-master secret
  - Step 2: Generating master secret
  - Step 3: Generating session keys

Public Key Infrastructure

Public Key Cryptography



Man-in-the-Middle (MITM) Attack



What Is the Fundamental Problem?

Fundamental Problem: Bob has no way to tell whether the public key he has received belongs to Alice or not.

Solution:

- Find a trusted party to verify the identity
- Bind an identity to a public key in a certificate
- The certificate cannot be forged or tampered with (using digital signature)

Public Key Infrastructure

- **Certificate Authority (CA):** a trusted party, responsible for verifying the identity of users, and then bind the verified identity to a public keys.
- **Digital Certificates:** A document certifying that the public key included inside does belong to the identity described in the document.
    - X.509 standard

Example of X.509 Certificate (1st Part)

The CA's identity (Symantec)

The owner of the certificate (paypal)

```
Certificate:
Data:
  Serial Number:
            2c:d1:95:10:54:37:d0:de:4a:39:20:05:6a:f6:c2:7f
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, O=Symantec Corporation, OU=Symantec Trust Network,
          CN=Symantec Class 3 EV SSL CA - G3
  Validity
     Not Before: Feb  2 00:00:00 2016 GMT
     Not After : Oct 30 23:59:59 2017 GMT
  Subject: 1.3.6.1.4.1.311.60.2.1.3=US/
           1.3.6.1.4.1.311.60.2.1.2=Delaware/
           businessCategory=Private Organization/
           serialNumber=3014267, C=US/
           postalCode=95131-2021, ST=California,
           L=San Jose/street=2211 N 1st St,
           O=PayPal, Inc., OU=CDN Support, CN=www.paypal.com
```

Example of X.509 Certificate (2nd Part)

Public key

```
 Subject Public Key Info:
     Public Key Algorithm: rsaEncryption
         Public-Key: (2048 bit)
         Modulus:
           00:da:43:c8:b3:a6:33:5d:83:c0:63:14:47:fd:6b:22:bd:
           bf:4e:a7:43:11:55:eb:20:8b:e4:61:13:ee:de:fe:c6:e2:
           ... (omitted) ...
           7a:15:00:c5:01:69:b5:10:16:a5:85:f8:fd:07:84:9a:c9:
         Exponent: 65537 (0x10001)
Signature Algorithm: sha256WithRSAEncryption
4b:a9:64:20:cc:77:0b:30:ab:69:50:d3:7f:de:dc:7c:e2:fb:93:84:fd:
78:a7:06:e8:14:03:99:c0:e4:4a:ef:c3:5d:15:2a:81:a1:b9:ff:dc:3a:
... (omitted) ...
fb:00:3e:7d:6a:de:cb:9f:ff:ef:8c:65:35:e4:22:b5:88:b2:48:32:1e:
```

CA's signature

The Core Functionalities of CA

- **Verify the subject**
  - Ensure that the person applying for the certificate either owns or represents the identity in the subject field.

- **Signing digital certificates**
  - CA generates a digital signature for the certificate using its private key.
  - Once the signature is applied, the certificate cannot be modified.
  - Signatures can be verified by anyone with the CA’s public key.