NETWORK SECURITY ESSENTIALS

BCA – IV

Credits – 4

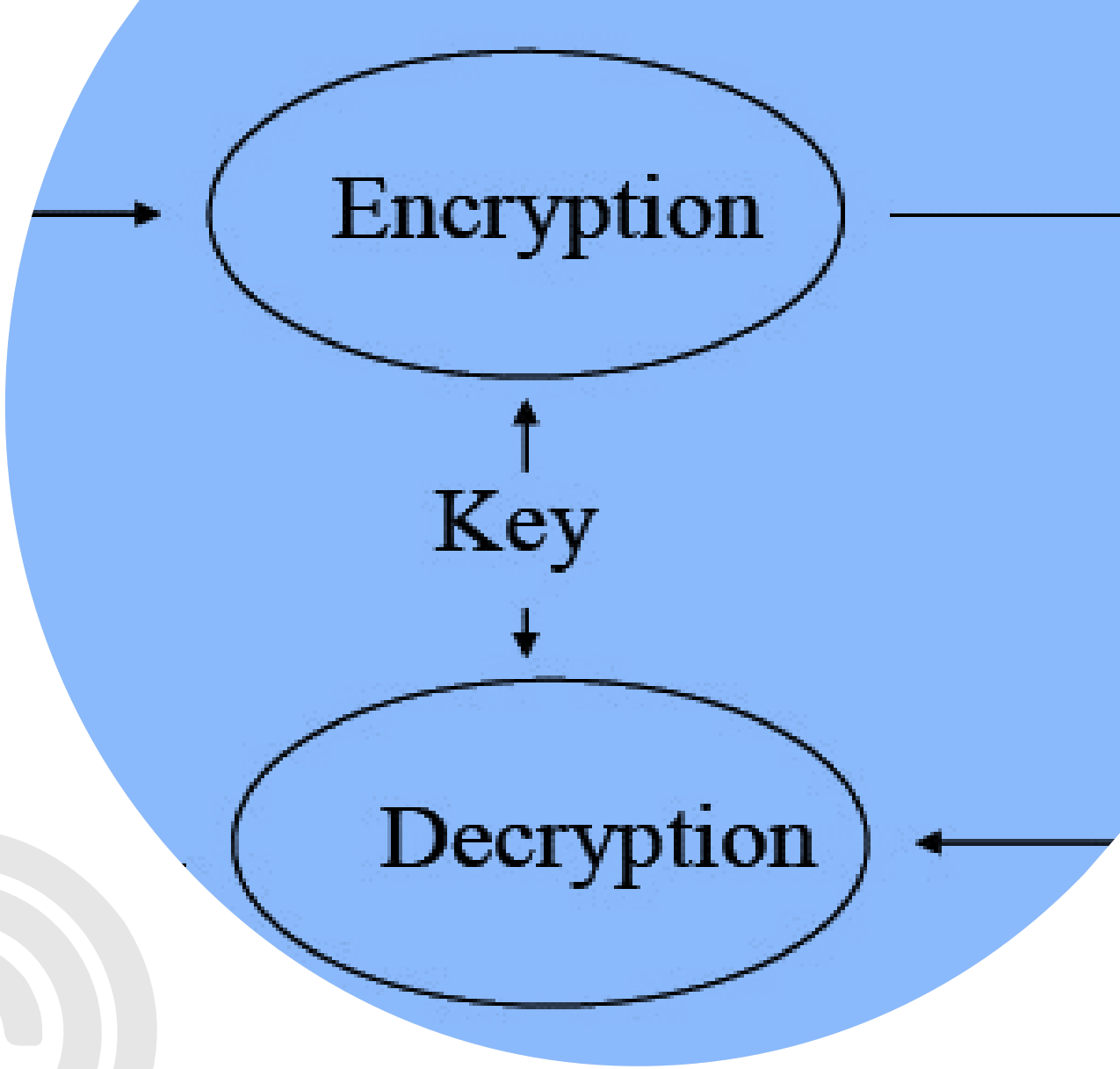
Evaluations – 5

The course is designed to build an understanding of various network security components, protocols and creating the awareness about the issues due to security.

Pre-requisites: An understanding of Basic Computer Networking and security

UNIT 2

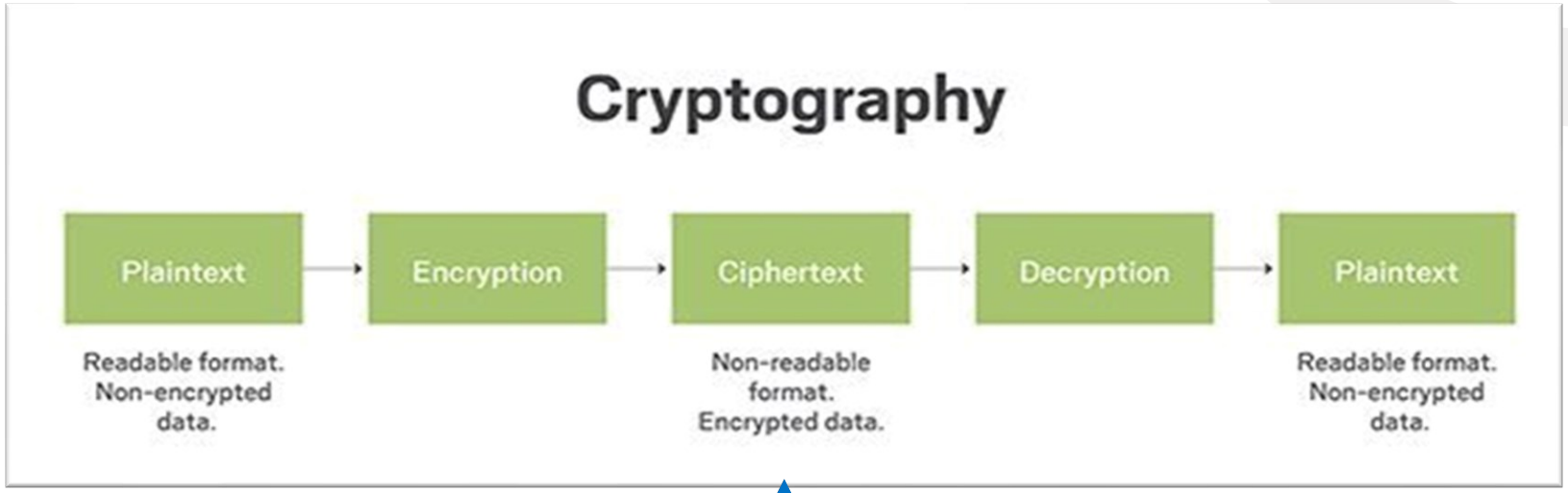
- Cryptography
- Different types of ciphers
- Concepts of Public-Key Cryptography, cryptographic hash functions, symmetric and public-key encryption, Cryptographic Checksums, HMAC, offset Codebook Mode of Operations, Birthday Attacks,
- Digital Signature Standard, Dual Signatures and Electronic Transactions, Blind Signatures and Electronic Cash

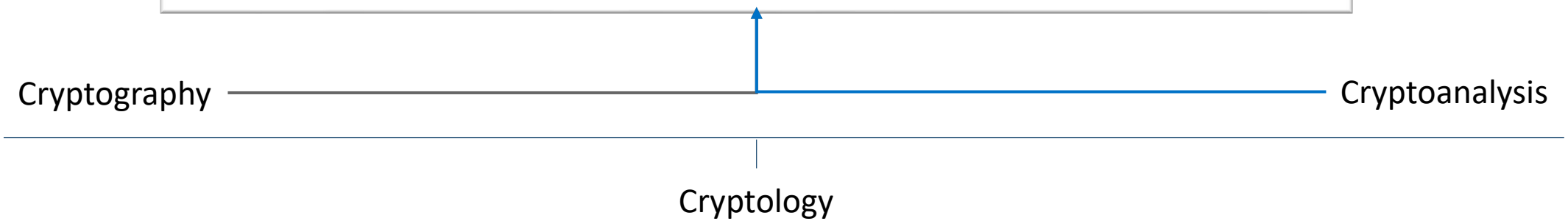


WHAT IS CRYPTOGRAPHY?

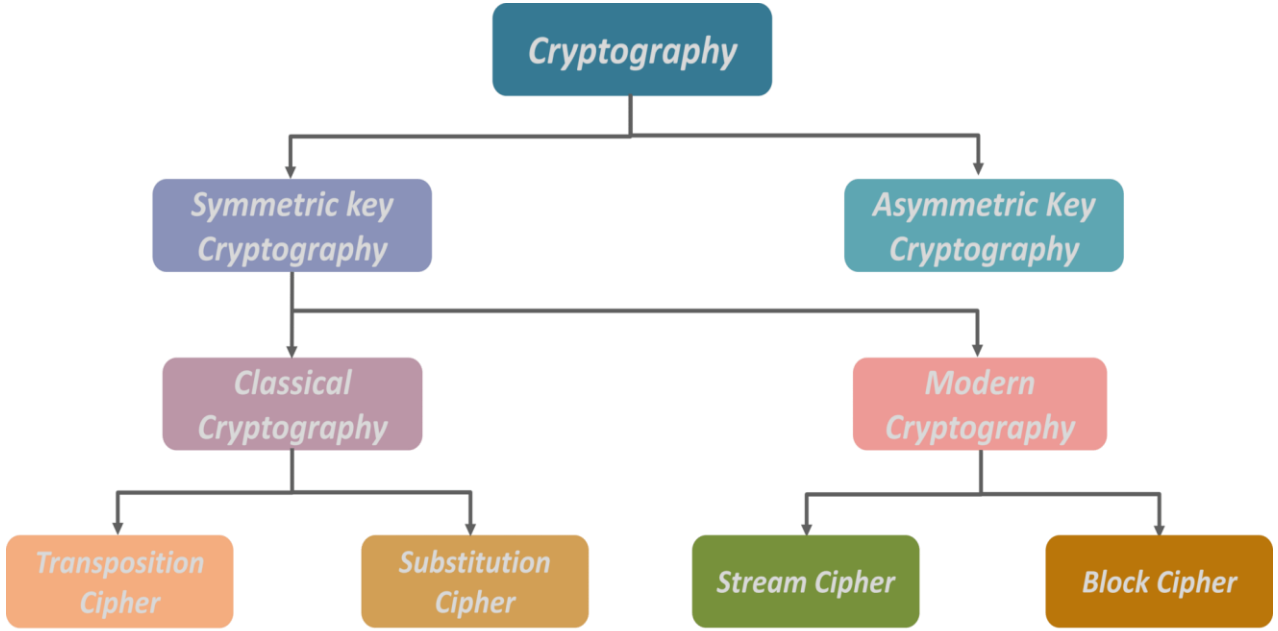
Kryptos – hidden/vault Graphien – writing

- It is an art & science of achieving security by encoding messages to make them non-readable
- Cryptography is associated with the process of converting ordinary plain text into unintelligible text and vice-versa. It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it.



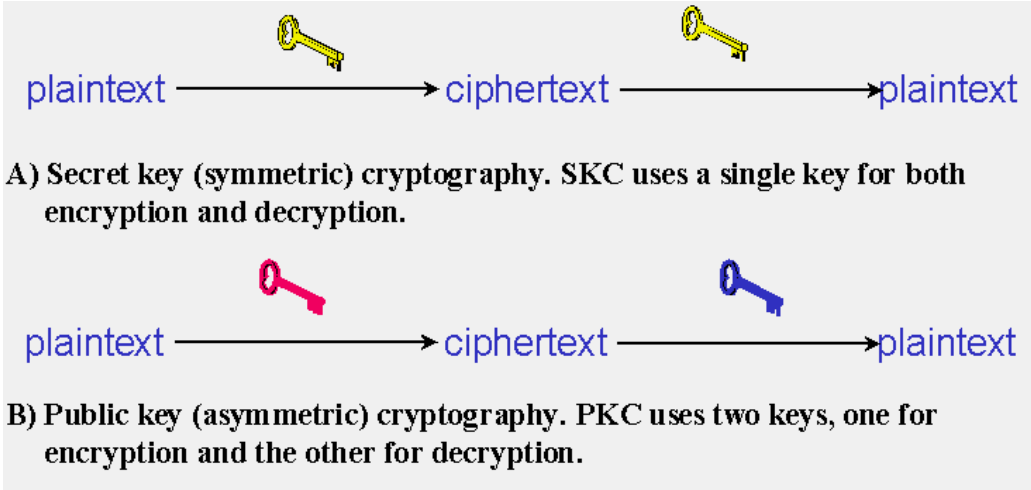


TYPES OF CRYPTOGRAPHY

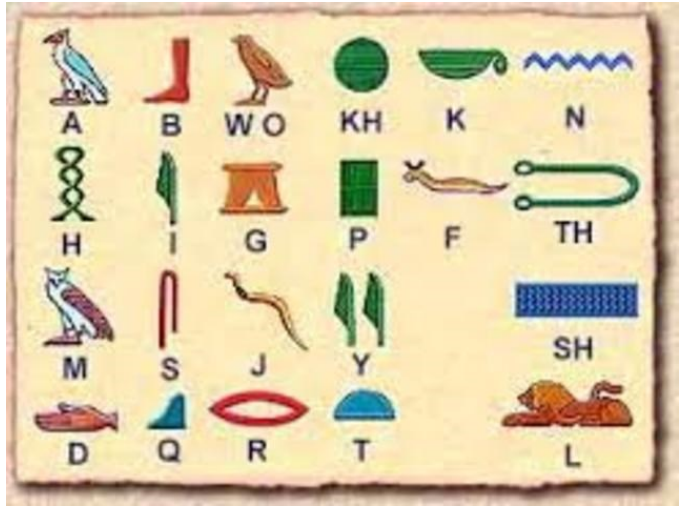


Symmetric Key Cryptography also known as Symmetric Encryption is when a secret key is leveraged for both encryption and decryption functions.

In an **asymmetric key encryption** scheme, anyone can encrypt messages using a public key, but only the holder of the paired private key can decrypt such a message. The security of the system depends on the secrecy of the private key, which must not become known to any other.



EARLIEST CRYPTOGRAPHIES (1900 BC)



	1	2	3	4	5
1	Α	Β	Γ	Δ	Ε
2	Ζ	Η	Θ	Ι	Κ
3	Λ	Μ	Ν	Ξ	Ο
4	Π	Ρ	Σ	Τ	Υ
5	Φ	Χ	Ψ	Ω	

As early as 1900 B.C., Egyptian scribes used hieroglyphs in a non-standard fashion, presumably to hide the meaning from those who did not know the meaning.

The Greek's idea was to wrap a tape around a stick, and then write the message on the wound tape. When the tape was unwound, the writing would be meaningless. The receiver of the message would of course have a stick of the same diameter and use it to decipher the message.

The Roman method of cryptography was known as the Caesar Shift Cipher. It utilized the idea of shifting letters by an agreed upon number (three was a common historical choice), and thus writing the message using the letter-shift.

CAESAR CYPHER – SUBSTITUTION TECHNIQUES

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

It's simply a type of substitution cipher, i.e., each **letter of a given text is replaced by a letter some fixed number of positions down the alphabet**. For example with a shift of 1, A would be replaced by B, B would become C, and so on.

Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.
The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

$$E_n(x) = (x + n) mod\ 26$$

(Encryption Phase with shift n)

$$D_n(x) = (x - n) mod\ 26$$

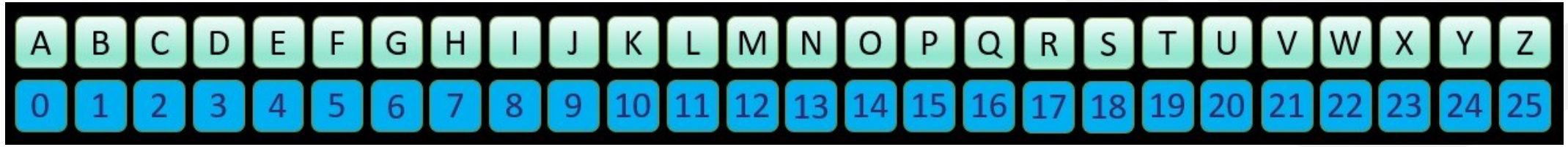
(Decryption Phase with shift n)

Algorithm for Caesar Cipher:

Input:
1.A String of lower case letters, called Text.
2.An Integer between 0-25 denoting the required shift.

Procedure:
•Traverse the given text one character at a time .
•For each character, transform the given character as per the rule ***(Modulo 26)***, depending on whether we're encrypting or decrypting the text.
•Return the new string generated.
Program that receives a Text (string) and Shift value(integer) and returns the encrypted text.

EXERCISE



1. Write “My name is (your name)” as encrypted in Caesar Cypher (consider n=3)
2. Decrypt the following using Caesar Cypher and find the value of “n”:

   “ai evi wxyhcmek gvctxskvetlc”

3. What is ROT-13? What are its applications?

within=13

Decipher the following:
“ck gxk yzajeogm ixevzumxgvne”

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

The Vigenère square or Vigenère table, also known as the *tabula recta*, can be used for encryption and decryption.

2 step process of encryption:

Plain text - We are studying Cryptography

Caesar Cypher 1 (n=4)
“ai evi wxyhcmek gvctxskvetlc”

Caesar Cypher 2 (n=2)
“ck gxk yzajeogm ixevzumxgvne”

Does this work?

The major **drawbacks of Caesar cipher** is that it can easily be broken, even in **cipher**-text only scenario. Various methods have been detected which crack the **cipher** text using frequency analysis and pattern words. One of the approaches is using brute force to match the frequency distribution of letters.

VIGENÈRE CIPHER – POLYALPHABETIC CYPHER

The **Vigenère cipher** is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword.

The Vigenère cipher uses a 26×26 table with **A** to **Z** as the row heading and column heading. This table is usually referred to as the ***Vigenère Table*** or ***Vigenère Square***. We shall use ***Vigenère Table***.

In addition to the plaintext, the Vigenère cipher also requires a keyword, which is repeated so that the total length is equal to that of the plaintext.

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

The Vigenère square or Vigenère table, also known as the *tabula recta*, can be used for encryption and decryption.

ENCRYPTION & DECRYPTION USING VIGENÈRE CYPHER

Suppose the plaintext is
“**MICHIGAN TECHNOLOGICAL UNIVERSITY**” and the keyword is “ H**OUGHTON**”.
Then, the keyword must be repeated as follows:

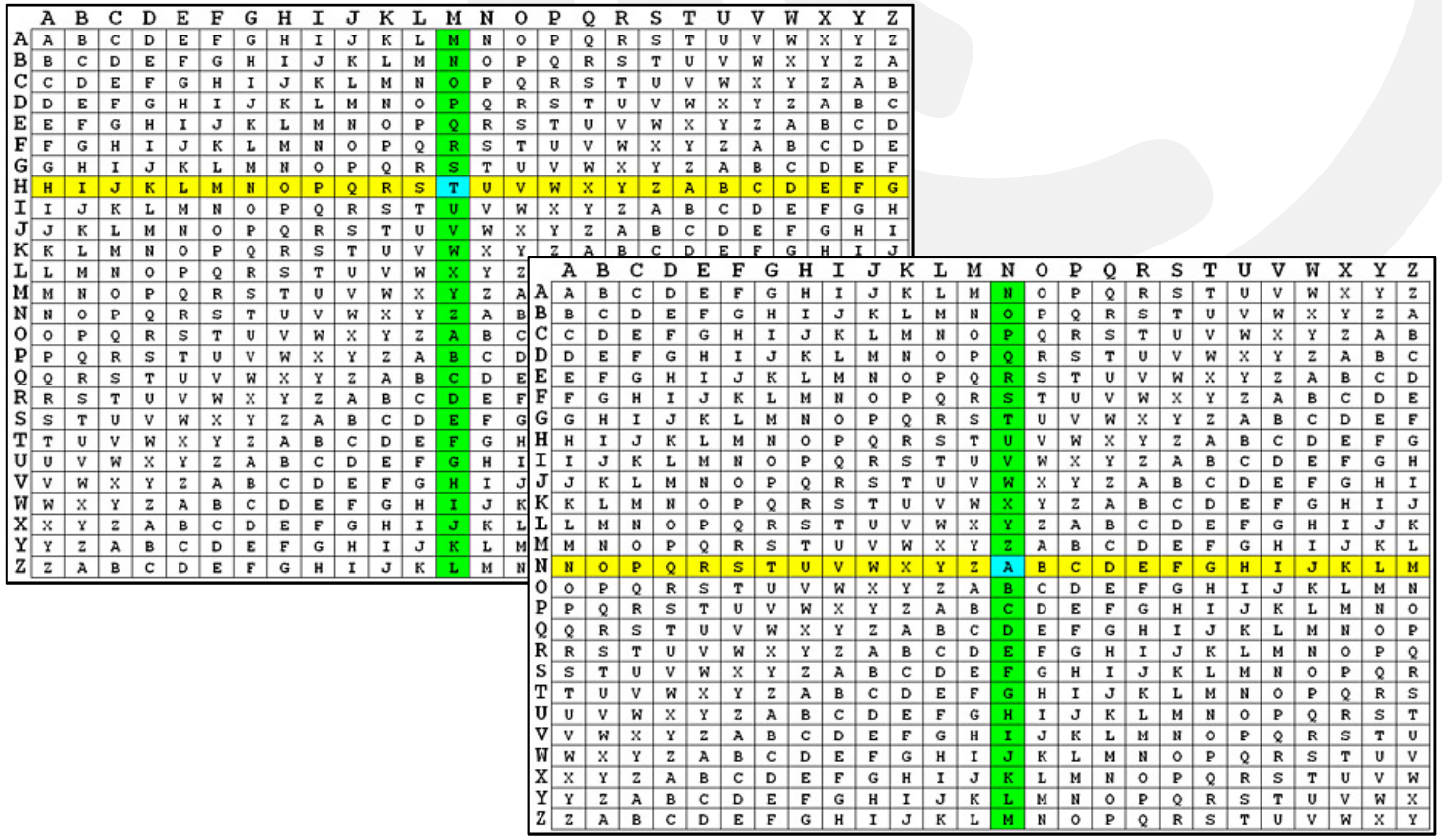
```
MICHIGAN TECHNOLOGICAL UNIVERSITY
HOUGHTON HOUGHTONHOUGH TONHOUGNTO
```

We follow the tradition by removing all spaces and punctuation, converting all letters to upper case, and dividing the result into 5-letter blocks. As a result, the above plaintext and keyword become the following:

```
MICHI GANTE CHNOL OGICA LUNIV ERSIT Y
HOUGH TONHO UGHTO NHOUG HTONH OUGHT O
```

To encrypt, pick a letter in the plaintext and its corresponding letter in the keyword, use the keyword letter and the plaintext letter as the row index and column index, respectively, and the entry at the row-column intersection is the letter in the ciphertext.

Do it for rest of the plain text and share your results



Repeating this process until all plaintext letters are processed, the ciphertext is **TWWNPZOA ASWNUHZBNWWGS NBVCSLYPMM**.

TRANSPOSITION TECHNIQUES

The order of alphabets in the plaintext is rearranged to form a cipher text.

Rail fence cypher

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

- In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence.
- When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner.
- After each alphabet has been written, the individual rows are combined to obtain the cipher-text.

Plaintext T H I S I S A S E C R E T M E S S A G E

Rail Fence Encoding
key = 4

T						A						T						G	
	H				S		S				E		M				A		E
		I		I				E		R				E		S			
			S						C						S				

Ciphertext T A T G H S S E M A E I I E R E S S C S

Write “Symbiosis Institute of Computer Studies and Research” using rail fence cypher where key is 3

VERNAM CIPHER (ONE TIME PAD)

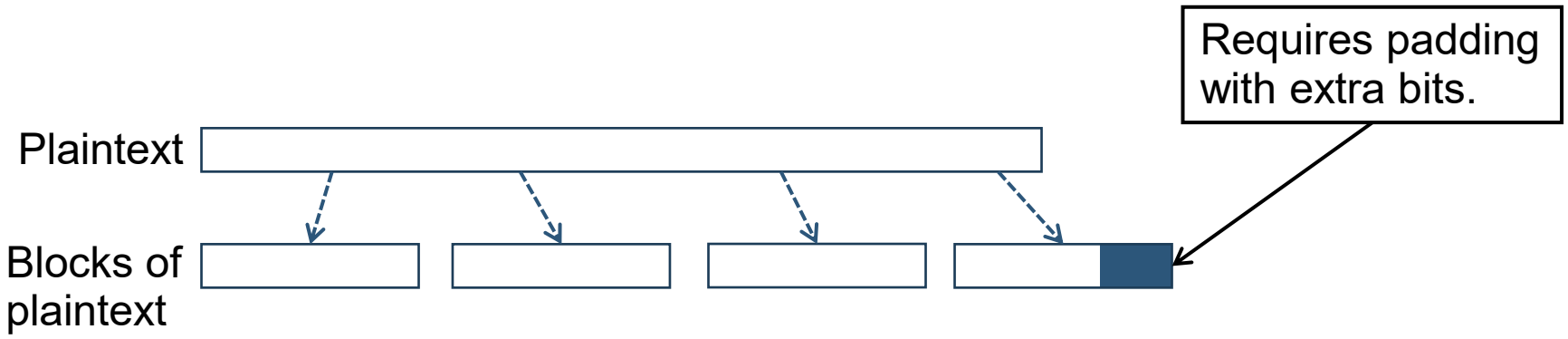
Vernam Cipher is a method of encrypting alphabetic text. It is one of the Transposition techniques for converting a plain text into a cipher text. In this mechanism we assign a number to each character of the Plain-Text, like (a = 0, b = 1, c = 2, … z = 25).

Method to take key:
In Vernam cipher algorithm, we take a key to encrypt the plain text which length should be equal to the length of the plain text.

Plaintext	V	E	R	N	A	M	C	I	P	H	E	R
Numeric Equivalent	21	4	17	13	0	12	2	8	15	7	4	17
+ Random Number	76	48	16	82	44	3	58	11	60	5	48	88
= Sum	97	52	33	95	44	15	60	19	75	12	52	105
= mod 26	19	0	7	17	18	15	8	19	23	12	0	1
Ciphertext	t	a	h	r	s	p	i	t	x	m	a	b

BLOCK CYPHER

- In a **block cipher:**
  - Plaintext and ciphertext have fixed length b (e.g., 128 bits)
  - A plaintext of length n is partitioned into a sequence of m **blocks**, P[0], …, P[m–1], where $n \leq bm < n + b$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks.



PADDING

- Block ciphers require the length n of the plaintext to be a multiple of the block size b
- Padding the last block needs to be unambiguous (cannot just add zeroes)
- When the block size and plaintext length are a multiple of 8, a common padding method (PKCS5) is a sequence of identical bytes, each indicating the length (in bytes) of the padding
- Example for b = 128 (16 bytes)
  - Plaintext: “Roberto” (7 bytes)
  - Padded plaintext: “Roberto999999999” (16 bytes), where 9 denotes the number and not the character
- We need to always pad the last block, which may consist only of padding

STREAM CIPHERS

In stream cipher, one byte is encrypted at a time while in block cipher ~128 bits are encrypted at a time. Initially, a key(k) will be supplied as input to pseudorandom bit generator and then it produces a random 8-bit output which is treated as keystream.

The resulted keystream will be of size 1 byte, i.e., 8 bits.
1.Stream Cipher follows the sequence of pseudorandom number stream.
2.One of the benefits of following stream cipher is to make cryptanalysis more difficult, so the number of bits chosen in the Keystream must be long in order to make cryptanalysis more difficult.
3.By making the key more longer it is also safe against brute force attacks.
4.The longer the key the stronger security is achieved, preventing any attack.
5.Keystream can be designed more efficiently by including more number of 1s and 0s, for making cryptanalysis more difficult.
6.Considerable benefit of a stream cipher is, it requires few lines of code compared to block cipher.

Encryption :

For Encryption,

•Plain Text and Keystream produces Cipher Text (Same keystream will be used for decryption.).
•The Plaintext will undergo XOR operation with keystream bit-by-bit and produces the Cipher Text.

DIFFERENCE BETWEEN SYMMETRIC AND ASYMMETRIC KEY ENCRYPTION

Symmetric Key Encryption	Asymmetric Key Encryption
It only requires a single key for both encryption and decryption.	It requires two key one to encrypt and the other one to decrypt.
The size of cipher text is same or smaller than the original plain text.	The size of cipher text is same or larger than the original plain text.
The encryption process is very fast.	The encryption process is slow.
It is used when a large amount of data is required to transfer.	It is used to transfer small amount of data.
It only provides confidentiality.	It provides confidentiality, authenticity and non-repudiation.
Examples: 3DES, AES, DES and RC4	Examples: Diffie-Hellman, ECC, El Gamal, DSA and RSA
In symmetric key encryption, resource utilization is low as compared to asymmetric key encryption.	In asymmetric key encryption, resource utilization is high.

SYMMETRIC KEY ENCRYPTION

Also known as secret-key, single-key, shared-key, one-key, and private-key encryption

Symmetric encryption is generally more efficient than asymmetric encryption and therefore preferred when large amounts of data need to be exchanged.

Establishing the shared key is difficult using only symmetric encryption algorithms, so in many cases, an asymmetric encryption is used to establish the shared key between two parties.

Examples for symmetric key cryptography include AES, DES, and 3DES. Key exchange protocols used to establish a shared encryption key include Diffie-Hellman (DH), elliptic curve (EC) and RSA.

PUBLIC KEY CRYPTOGRAPHY

- **Public key cryptography** (PKC) is an **encryption** technique that uses a paired **public** and private **key** algorithm for secure data communication.
- A message sender uses a recipient's **public key** to encrypt a message.
- To decrypt the sender's message, only the recipient's **private key** may be used.

Principles of Public-Key Cryptosystems

- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

Key distribution

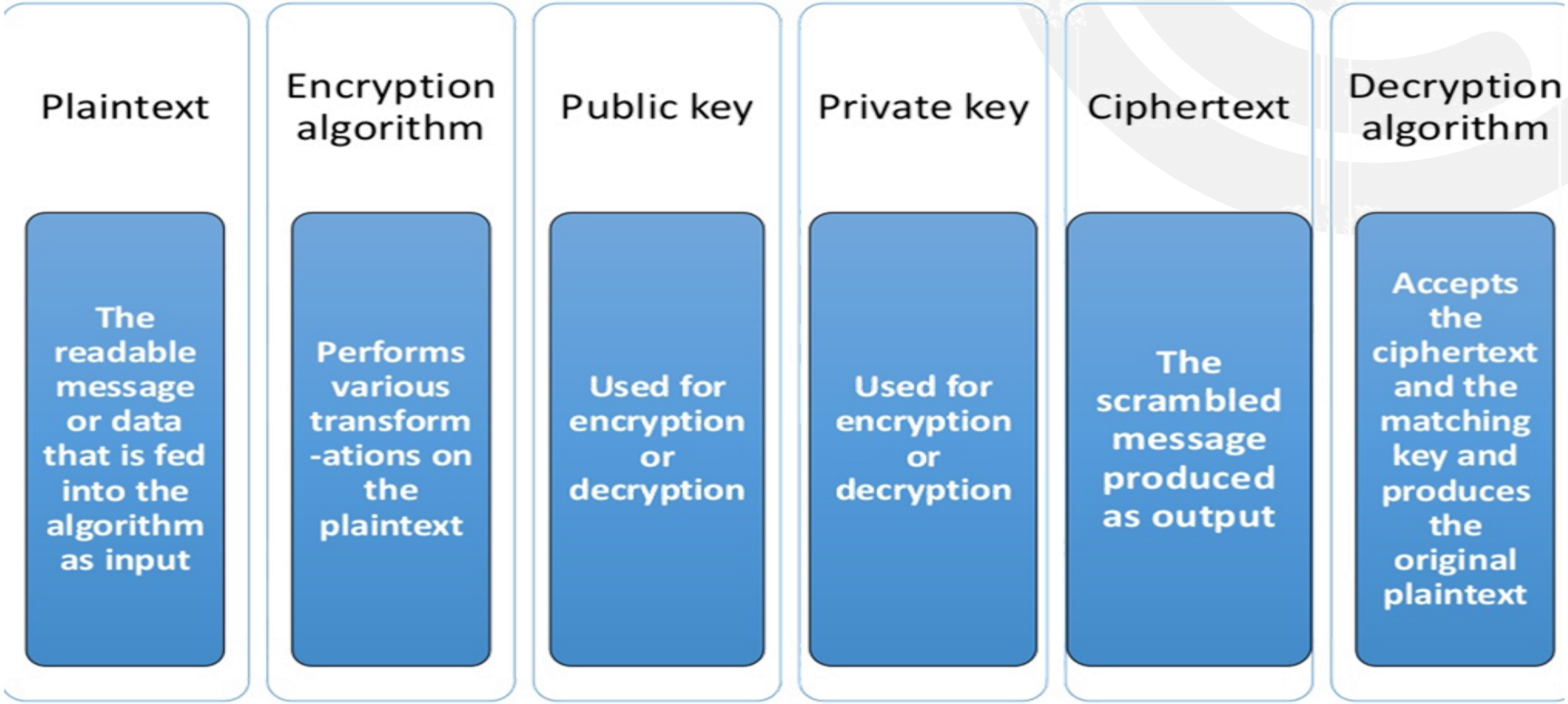
- **The communicants already shares a key or someone has been distributed the key.**
- **How to secure communications in general without having to trust a KDC with your key**

Digital signatures

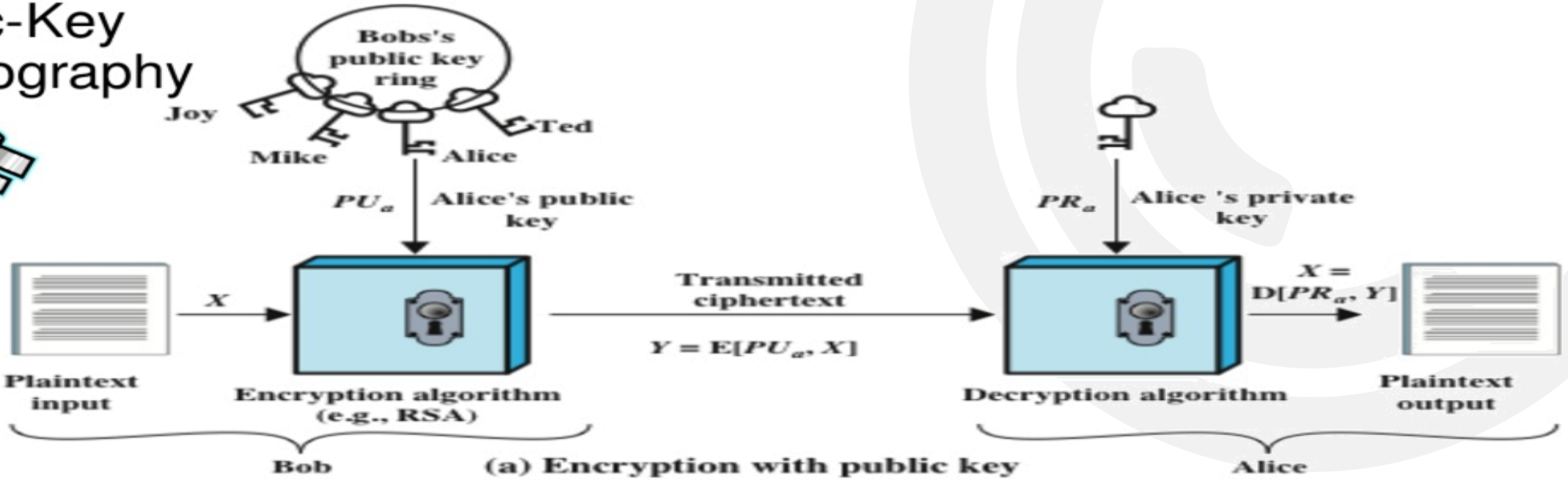
- **How to verify that a message comes intact from the claimed sender**

Public-Key Cryptosystems

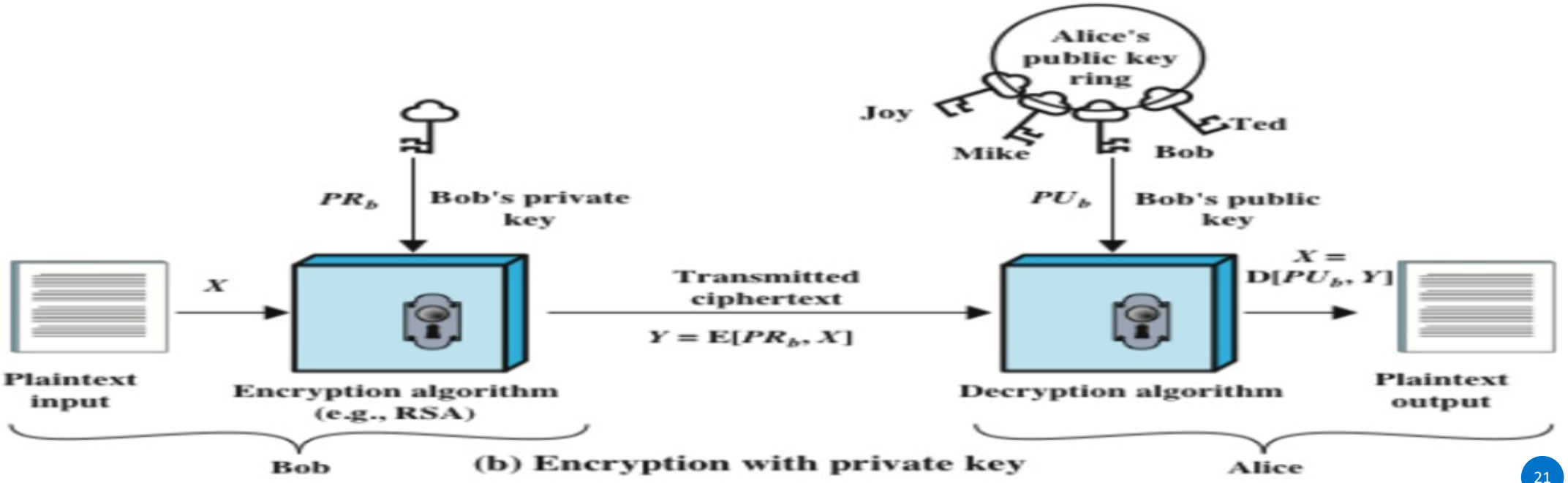
- A public-key encryption scheme has six ingredients:



Public-Key Cryptography



(a) Encryption with public key



(b) Encryption with private key

CONVENTIONAL & PUBLIC KEY ENCRYPTION

CONVENTIONAL ENCRYPTION

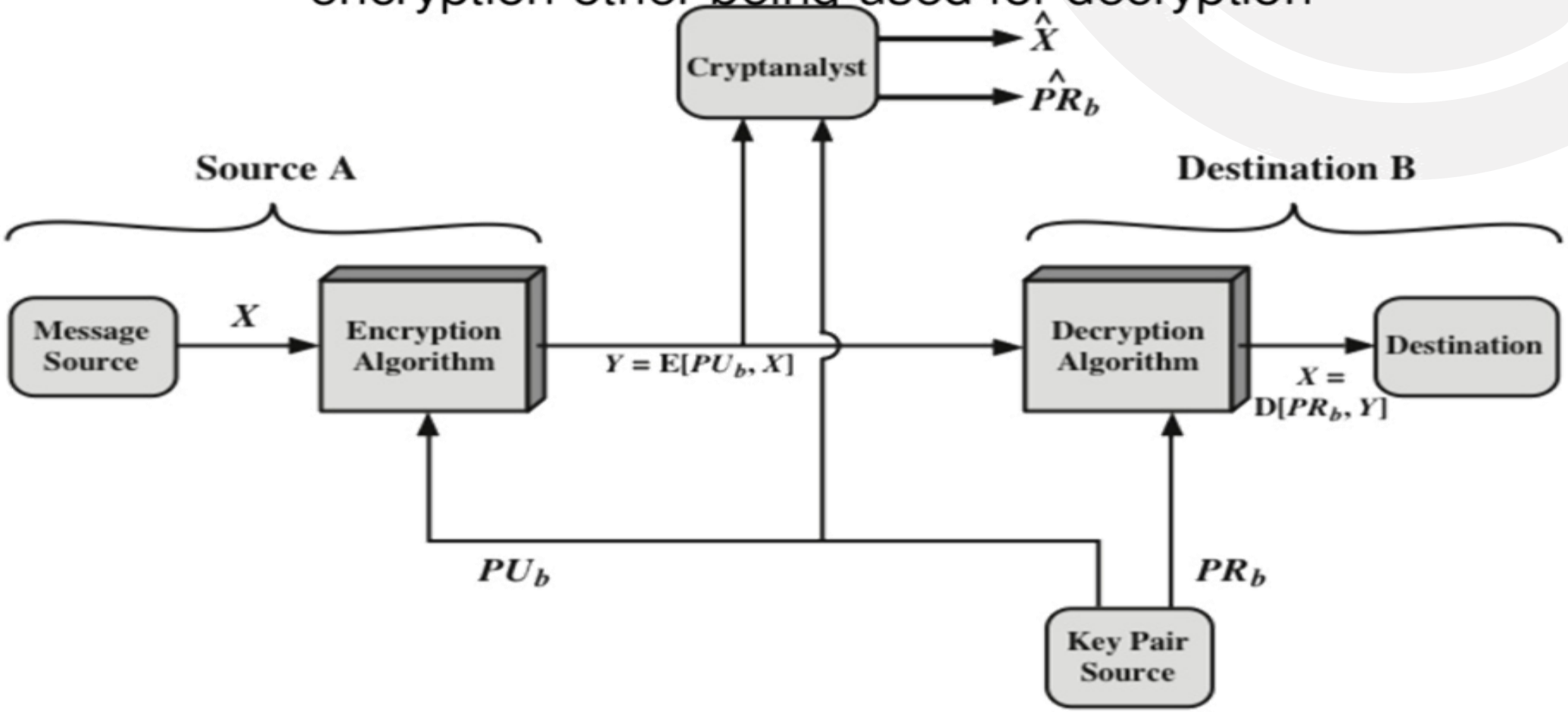
- The same algorithm with the same key is used for encryption and decryption
- The sender and receiver must share the algorithm and the key
- For Security:
  - They key must be kept secret
  - It must be impossible to decipher the message if the key is kept secret
  - Knowledge of algorithm and the sample of cyphertext must be insufficient to determine the key

- One algorithm is used for encryption and related algorithm is used for decryption, with a pair of keys
- The sender and receiver must have one of the matched pairs, not the same key
- For Security:
  - One of the two keys should be kept secret
  - It must be impossible to decipher the message if one of the keys is kept secret
  - Knowledge of algorithm plus one key should be insufficient to determine the second key

PUBLIC KEY ENCRYPTION

Public-Key Cryptosystem: encryption using public key -Secrecy

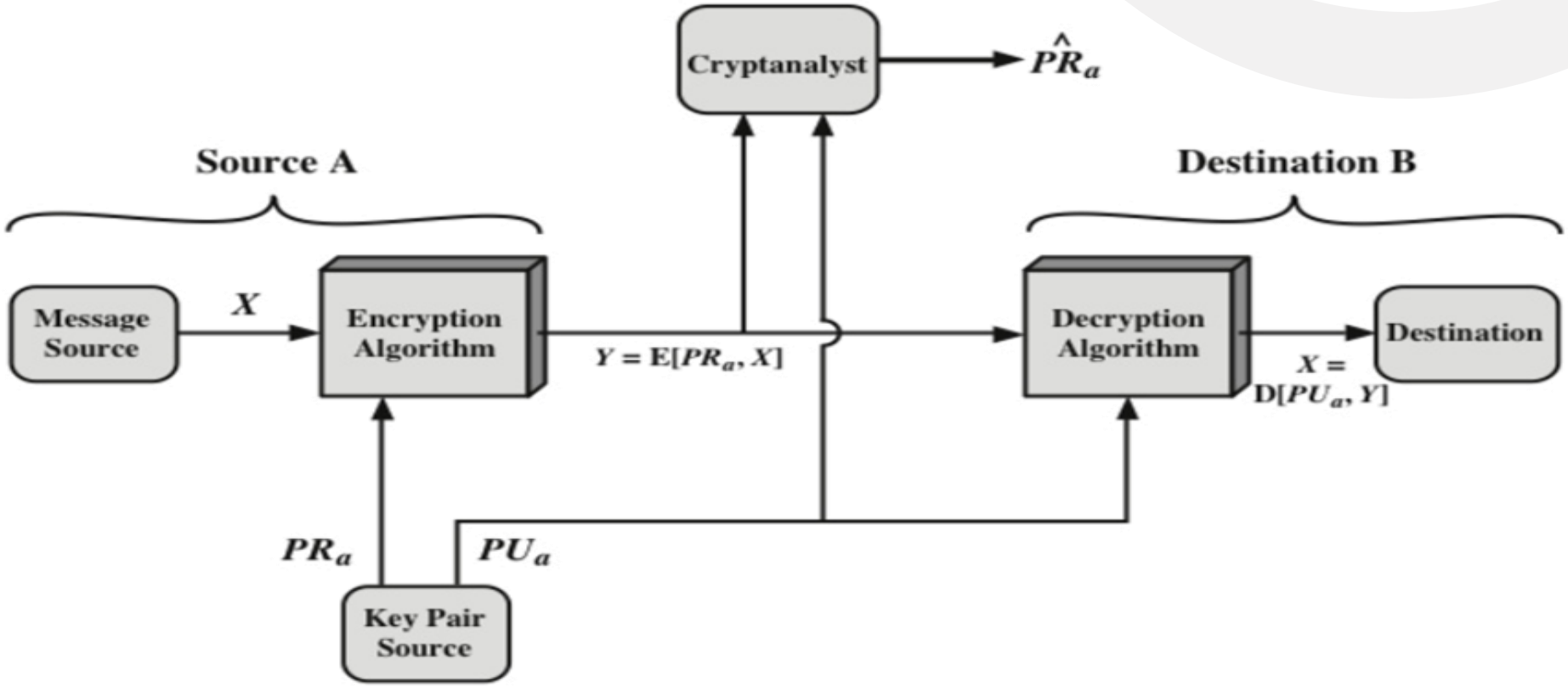
This figure provides confidentiality because two related key used for encryption other being used for decryption



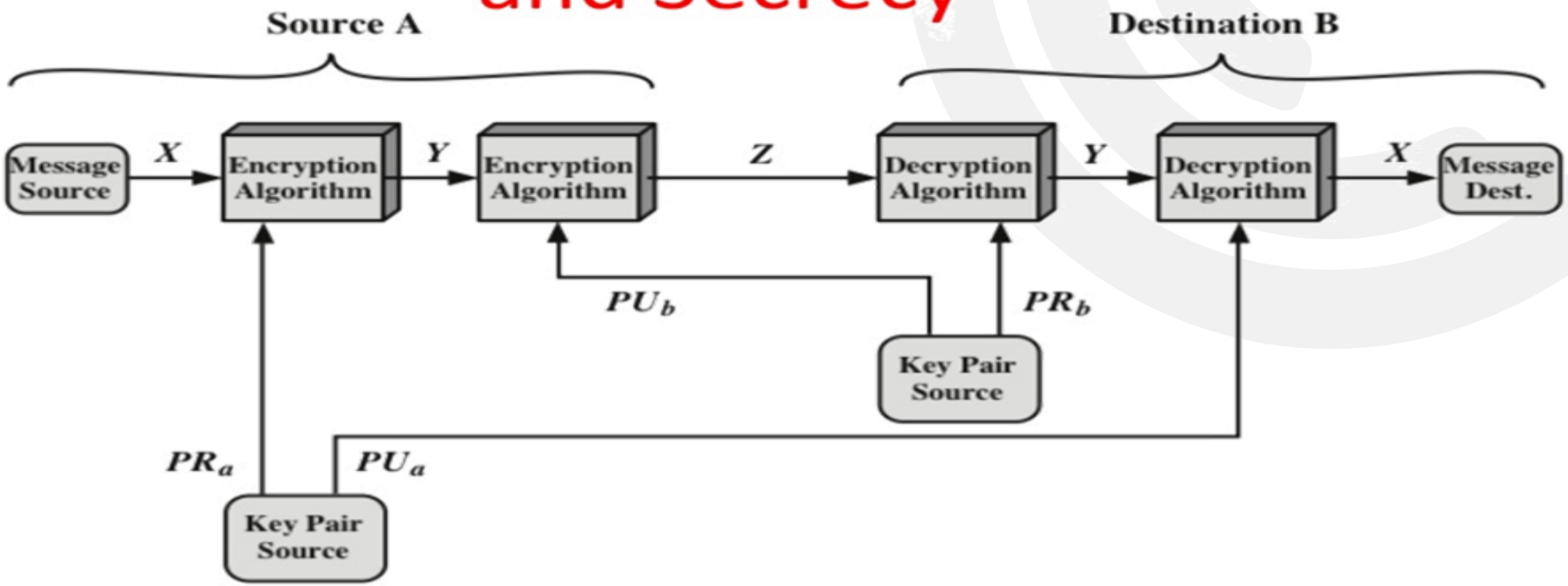
Public-Key Cryptosystem:

Encryption using private key -Authentication

There is no protection of confidentiality because any observer can decrypt the message by using the sender's public key



Public-Key Cryptosystem: Authentication and Secrecy



we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided

Public-Key Requirements

Conditions that these algorithms must fulfil:

1. It is computationally easy for a party B to generate a pair (public-key PU_b, private key PR_b)
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message
4. It is computationally infeasible for an adversary, knowing the public key, to determine the private key.
5. It is computationally infeasible for an adversary, knowing the public key and a ciphertext, to recover the original message.
6. The two keys can be applied in either order.

trap-door one-way function

- A **trapdoor function** is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction (finding its inverse) without special information, called the "trapdoor". Trapdoor functions are widely used in cryptography.

  - $Y = f(X)$ easy
  - $X = f^{-1}(Y)$ infeasible

A trap-door one-way function is a family of invertible functions f_k, such that

$Y = f_k(X)$ easy, if k and X are known
$X = f_k^{-1}(Y)$ easy, if k and Y are known
$X = f_k^{-1}(Y)$ infeasible, if Y known but k not known

A practical public-key scheme depends on a suitable trap-door one-way function

RSA (RIVEST-SHAMIR-ADLEMAN) SCHEME

- Developed by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT in 1977.
- Based on the difficulty of factoring large numbers
- Factorization so far is unsolvable in polynomial-time
- In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm: It provides a method to assure the confidentiality, integrity, authenticity, and non-repudiation of electronic communications and data storage.

RSA EXAMPLE

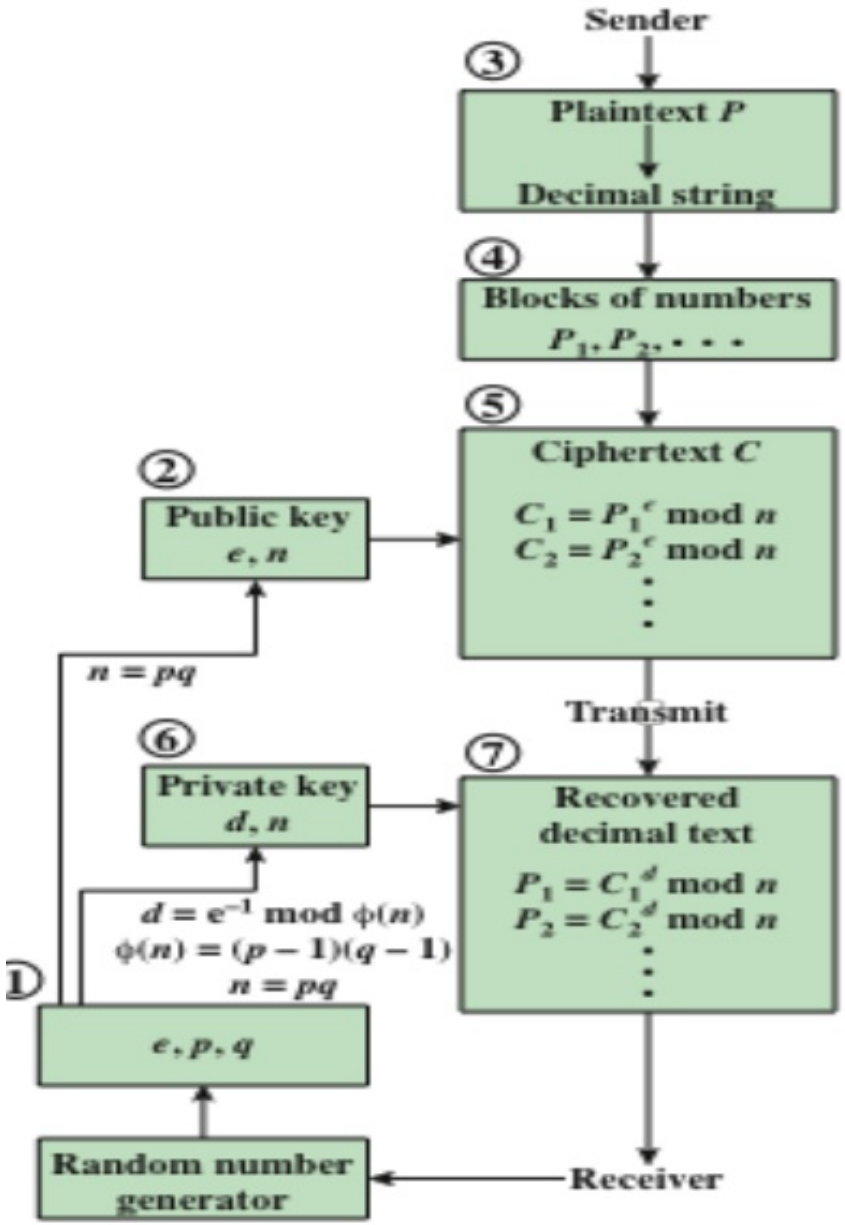
- Find two large prime integers, p and q, and form product $n = pq$
- Find a random integer, e, that is relatively prime to $\Phi(n) = (p-1)(q-1)$
- p and q are kept private, (n,e) are the public key
- Message is partitioned into blocks, b, such that $b < n$
- Each block is encrypted using the equation: $c = b^e \bmod n$
- For the private key, calculate integer d which is the modular inverse of e for $\Phi(n)$, or $e * d \bmod \Phi(n) = 1$
- Once d is calculated it becomes your private key and all records of p and q should be destroyed
- Each encrypted block, c, is decrypted using the equation: $b = c^d \bmod n$
- p = 61, q = 53, n = 3233, $\Phi(n) = 3120$, e = 17, d = 2753
- encrypt(123) = $123^{17} \bmod 3233 = 855$
- decrypt(855) = $855^{2753} \bmod 3233 = 123$

Key Generation by Alice	
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	$d = e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

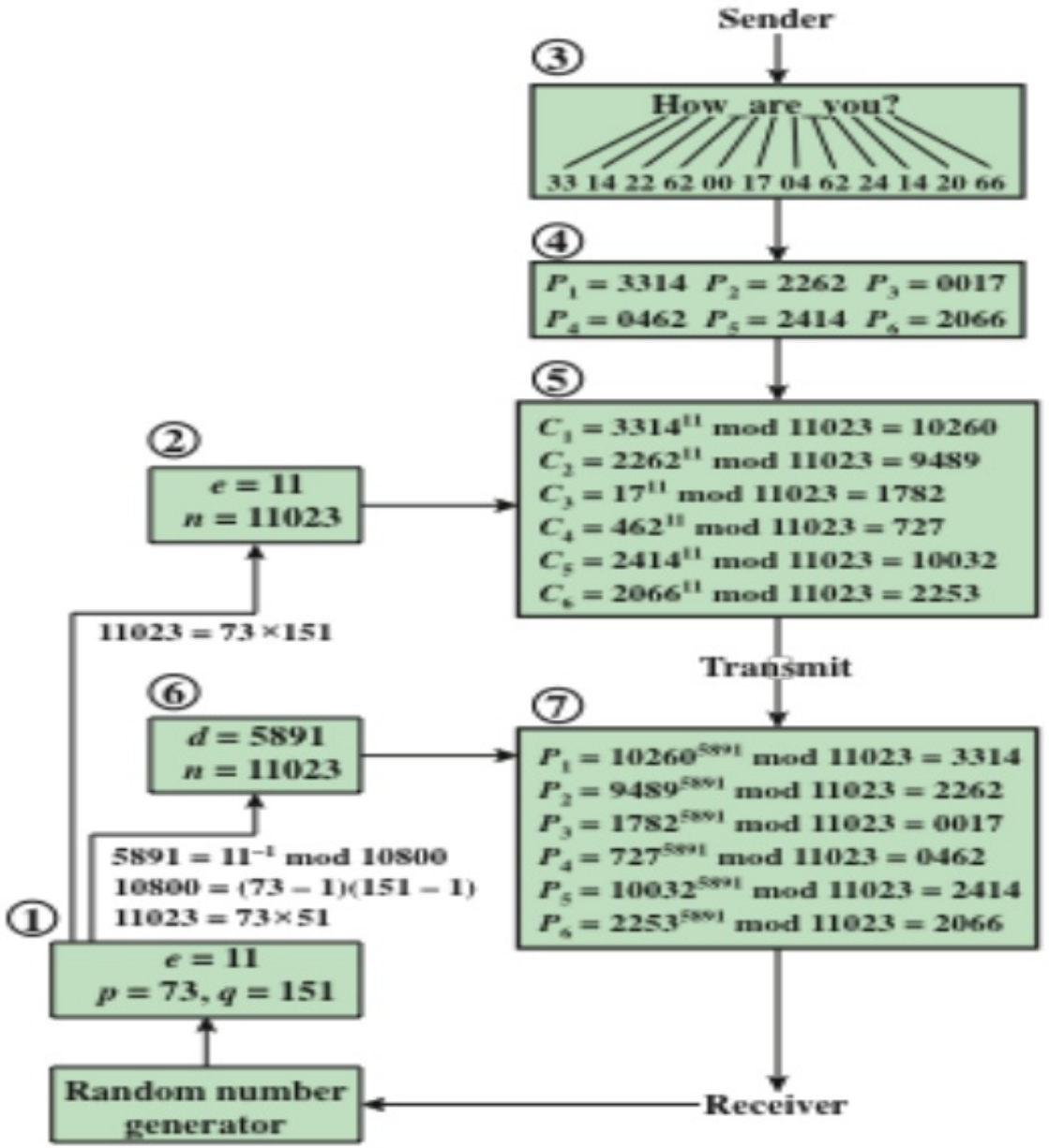
Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

RSA Algorithm



(a) General approach



(b) Example

RSA: Algorithm Notes

- How should we choose e?
  - It doesn't matter for security; everybody could use the same e.
  - It matters for performance: 3, 17, or 65537 are good choices.
- n is referred to as the modulus, since it's the n of mod n.
- You can only encrypt messages $M < n$. Thus, to encrypt larger messages you need to break them into pieces, each $< n$.
- Throw away p, q, and $\phi(n)$ after the key generation stage.
- Encrypting and decrypting requires a single modular exponentiation.

RSA Example: Key Generations

1. Select two primes: $p = 47$ and $q = 71$.
2. Compute $n = pq = 3337$.
3. Compute $\phi(n) = (p-1)(q-1) = 3220$.
4. Select $e = 79$.
5. Compute

$$\begin{aligned} d &= e^{-1} \bmod \phi(n) \\ &= 79^{-1} \bmod 3220 \\ &= 1019 \end{aligned}$$

6. $P = (79, 3337)$ is the RSA public key.
7. $S = (1019, 3337)$ is the RSA private key.

RSA Example: Encryption

1. Encrypt $M = 6882326879666683$.
2. Break up M into 3-digit blocks:

$$m = \langle 688, 232, 687, 966, 668, 003 \rangle$$

Note the padding at the end.

3. Encrypt each block:

$$\begin{aligned} c_1 &= m_1^e \bmod n \\ &= 688^{79} \bmod 3337 \\ &= 1570 \end{aligned}$$

We get:

$$c = \langle 1570, 2756, 2091, 2276, 2423, 158 \rangle$$

RSA Example: Decryption

1. Decrypt each block:

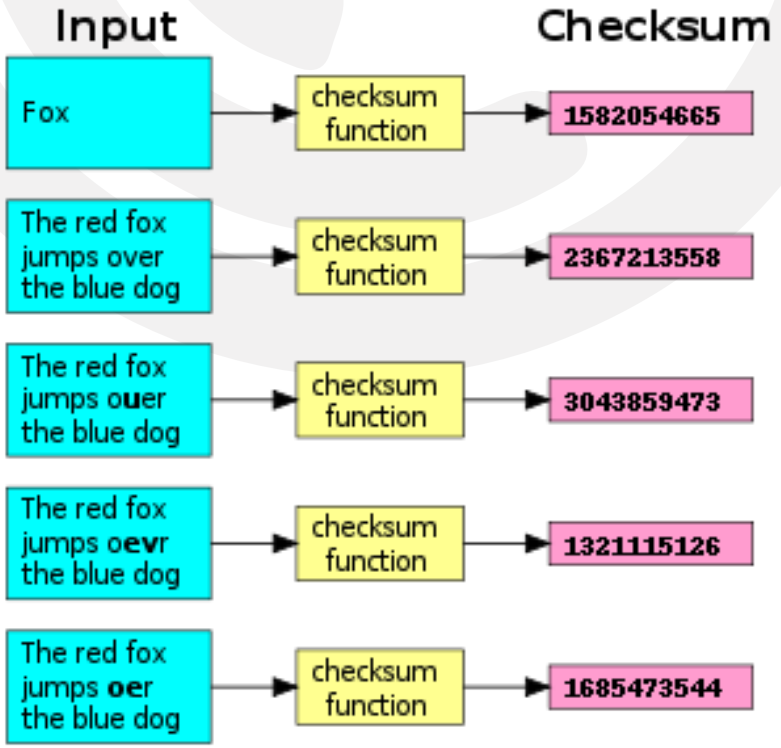
$$\begin{aligned} m_1 &= c_1^d \bmod n \\ &= 1570^{1019} \bmod 3337 \\ &= 688 \end{aligned}$$

CRYPTOGRAPHIC CHECKSUMS

A **checksum** is a small-sized block of data derived from another block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage. By themselves, checksums are often used to verify data integrity but are not relied upon to verify data authenticity.

The procedure which generates this checksum is called a **checksum function** or **checksum algorithm**. Depending on its design goals, a good checksum algorithm usually outputs a significantly different value, even for small changes made to the input. This is used to detect many data corruption errors and verify overall data integrity.

If the *computed checksum for the current data input matches the stored value of a previously computed checksum*, there is a very high probability the data has not been accidentally altered or corrupted.



Parity byte or parity word

The simplest checksum algorithm is the so-called longitudinal parity check, which breaks the data into "words" with a fixed number *n* of bits, and then computes the (XOR) of all those words. The result is appended to the message as an extra word. To check the integrity of a message, the receiver computes the exclusive or of all its words, including the checksum; if the result is not a word consisting of *n* zeros, the receiver knows a transmission error occurred.

With this checksum, any transmission error which flips a single bit of the message, or an odd number of bits, will be detected as an incorrect checksum. However, an error which affects two bits will not be detected if those bits lie at the same position in two distinct words. Also swapping of two or more words will not be detected.

Sum complement

A variant of the previous algorithm is to add all the "words" as unsigned binary numbers, discarding any overflow bits, and append the two's complement of the total as the checksum. To validate a message, the receiver adds all the words in the same manner, including the checksum; if the result is not a word full of zeros, an error must have occurred. This variant, too, detects any single-bit error, but the promodular sum is used in SAE J1708.

Position-dependent

The simple checksums described above fail to detect some common errors which affect many bits at once, such as changing the order of data words, or inserting or deleting words with all bits set to zero. The checksum algorithms most used in practice, such as Fletcher's checksum, Adler-32, and cyclic redundancy checks (CRCs), address these weaknesses by considering not only the value of each word but also its position in the sequence. This feature generally increases the cost of computing the checksum.

Fuzzy checksum

The idea of fuzzy checksum was developed for detection of email spam by building up co-operative databases from multiple ISPs of email suspected to be spam. The content of such spam may often vary in its details, which would render normal checksumming ineffective. By contrast a "fuzzy checksum" reduces the body text to its characteristic minimum, then generates a checksum in the usual manner. This greatly increases the chances of slightly different spam emails producing the same checksum. The ISP spam detection software, such as SpamAssassin, of co-operating ISPs submits checksums of all emails to the centralised service such as DCC. If the count of a submitted fuzzy checksum exceeds a certain threshold, the database notes that this probably indicates spam. ISP service users similarly generate a fuzzy checksum on each of their emails and request the service for a spam likelihood.

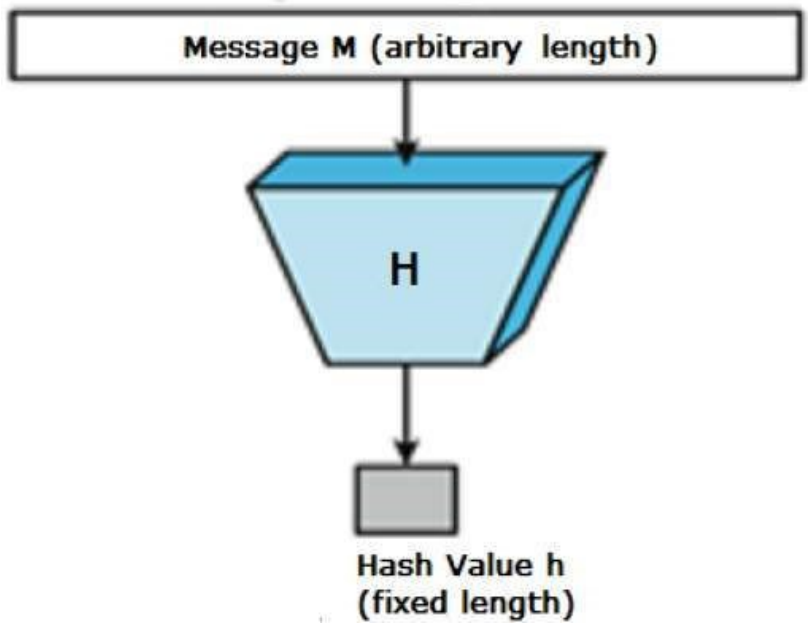
CRYPTOGRAPHIC HASH FUNCTION

When you hash data, the resulting digest is typically smaller than the input that it started with. (Probably the exception here is when you're hashing passwords.) With hashing, it doesn't matter if you have a one-sentence message or an entire book — the result will still be a fixed-length chunk of bits (1s and 0s). This prevents unintended parties from figuring out how big (or small) the original input message was.
Hash functions are primarily used for authentication but also have other uses.

- A hash function is a unique identifier for any given piece of content. It's also a process that takes plaintext data of any size and converts it into a unique ciphertext of a specific length.

- The first part of the definition tells you that no two pieces of content will have the same hash digest, and if the content changes, the hash digest changes as well. Basically, hashing is a way to ensure that any data you send reaches your recipient in the same condition that it left you, completely intact and unaltered.

Properties of a Strong Hash Algorithm



•**Determinism** — A hash algorithm should be **deterministic**, meaning that it always gives you an output of identical size regardless of the size of the input you started with. This means that if you're hashing a single sentence, the resulting output should be the same size as one you'd get when hashing an entire book.

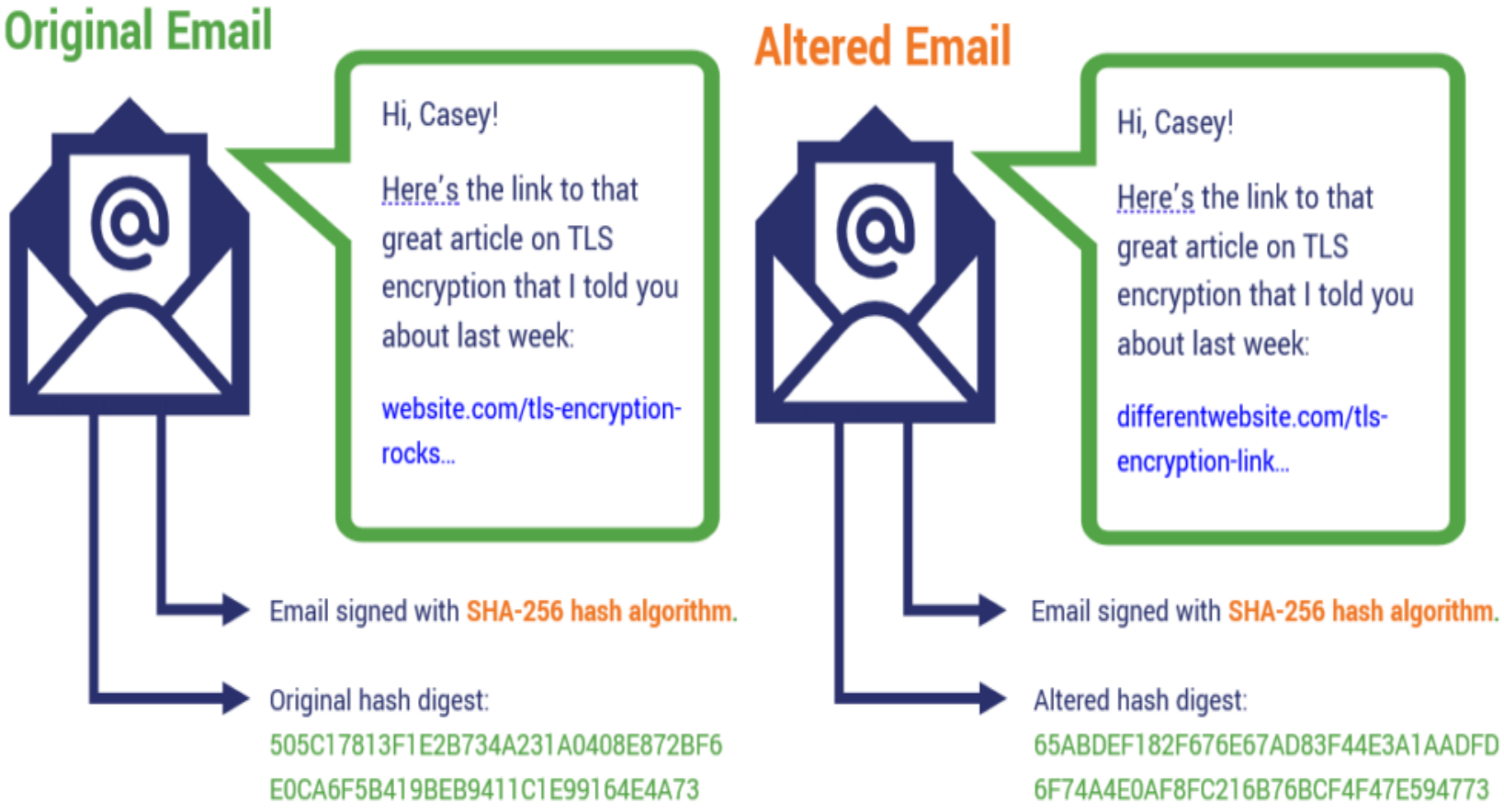
•**Pre-Image Resistance** — The idea here is that a strong hash algorithm is one that's preimage resistance, meaning that it's infeasible to reverse a hash value to recover the original input plaintext message. Hence, the concept of hashes being irreversible, one-way functions.

•**Collision Resistance** — A collision occurs when two objects collide. Well, this concept carries over in cryptography with hash values. If two unique samples of input data result in identical outputs, it's known as a collision. This is bad news and means that the algorithm you're using to hash the data is broken and, therefore, insecure. Basically, the concern here is that someone could create a malicious file with an artificial hash value that matches a genuine (safe) file and pass it off as the real thing because the signature would match. So, a good and trustworthy hashing algorithm is one that is resistant to these collisions.

•**Avalanche Effect** — What this means is that any change made to an input, no matter how small, will result in a massive change in the output. Essentially, a small change (such as adding a comma) snowballs into something much larger, hence the term "avalanche effect."

•**Hash Speed** — Hash algorithms should operate at a reasonable speed. In many situations, hashing algorithms should compute hash values quickly; this is considered an ideal property of a cryptographic hash function. However, this property is a little more subjective. You see, faster isn't always better because the speed should depend on how the hashing algorithm is going to be used. Sometimes, you want a faster hashing algorithm, and other times it's better to use a slower one that takes more time to run through. The former is better for website connections and the latter is better for password hashing.00

HASHING TO ENSURE DATA INTEGRITY



Hash functions are a way to ensure data integrity in public key cryptography. What I mean by that is that hash functions serve as a check-sum, or a way for someone to identify whether data has been tampered with after it's been signed. It also serves as a means of identity verification.

One of the best aspects of a cryptographic hash function is that it helps you to ensure data integrity. But if you apply a hash to data, does it mean that the message can't be altered? No. But what it does is inform the message recipient that the message has been changed. That's because even the smallest of changes to a message will result in the creation of an entirely new hash value.

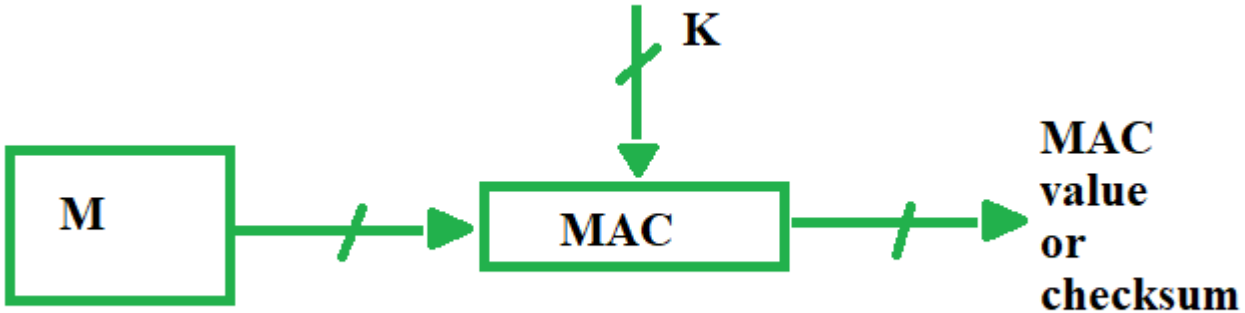
Hashes in Use

Hashing is useful for everything from signing new software and verifying digital signatures to securing the website connections in your computer and mobile web browsers.

It’s also great for indexing and retrieving items in online databases. For example, hashing is used for verifying:

- Data blocks in cryptocurrencies and other blockchain technologies.
- Data integrity of software, emails, and documents.
- Passwords and storing password hashes (rather than the passwords themselves) in online databases.
- (Note: This process requires a little “dash” of something special to make those hashes more secure — a salt).

MAC – MESSAGE AUTHENTICATION CODE



These are components:

- Message
- Key
- MAC algorithm
- MAC value

There are different types of models Of Message Authentication Code (MAC) as following below:

1. MAC without encryption
2. Internal error code
3. External error code

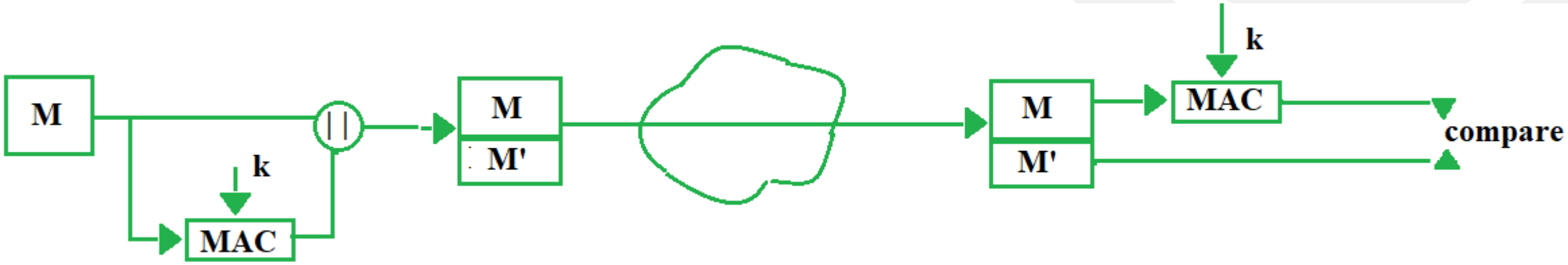
MAC stands for **Message Authentication Code.**

Here in MAC, sender and receiver share same key where sender generates a fixed size output called Cryptographic checksum or Message Authentication code and appends it to the original message.

On receiver's side, receiver also generates the code and compares it with what he/she received thus ensuring the originality of the message.

MAC WITHOUT ENCRYPTION

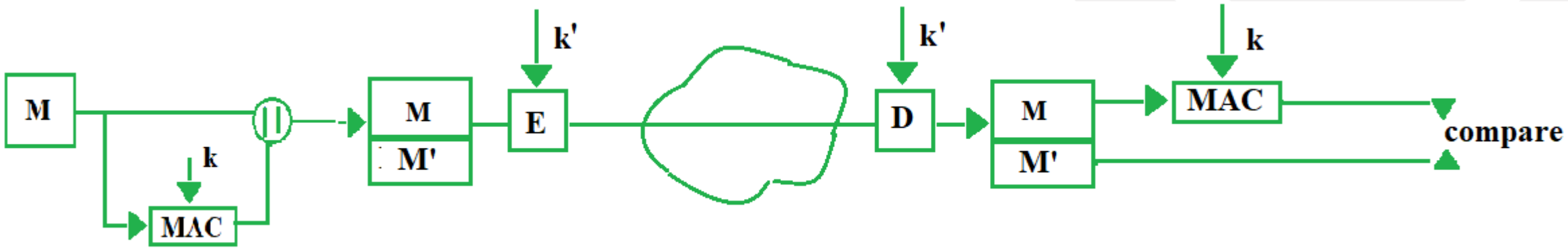
This model can provide authentication but not confidentiality as anyone can see the message.



INTERNAL ERROR CODE

In this model of MAC, sender encrypts the content before sending it through network for confidentiality.
Thus this model provides confidentiality as well as authentication

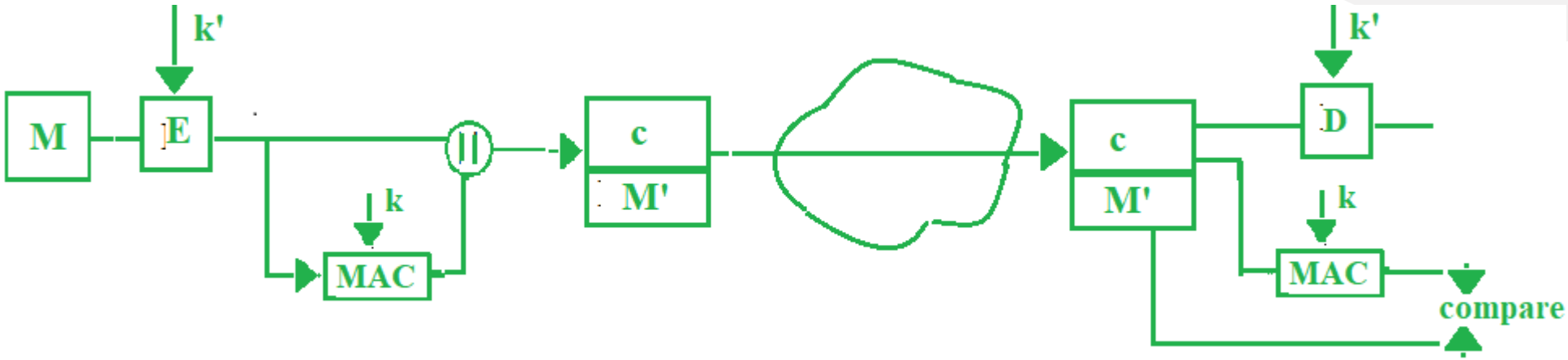
M' = MAC(M, k)



EXTERNAL ERROR CODE

For cases when there is an alteration in message, we decrypt it for waste, to overcome that problem, we opt for external error code. Here we first apply MAC on the encrypted message ‘c’ and compare it with received MAC value on the receiver’s side and then decrypt ‘c’ if they both are same, else we simply discard the content received. Thus it saves time.

c = E(M, k’)
M' = MAC(c, k)



HMAC – HASHED MESSAGE AUTHENTICATION CODE

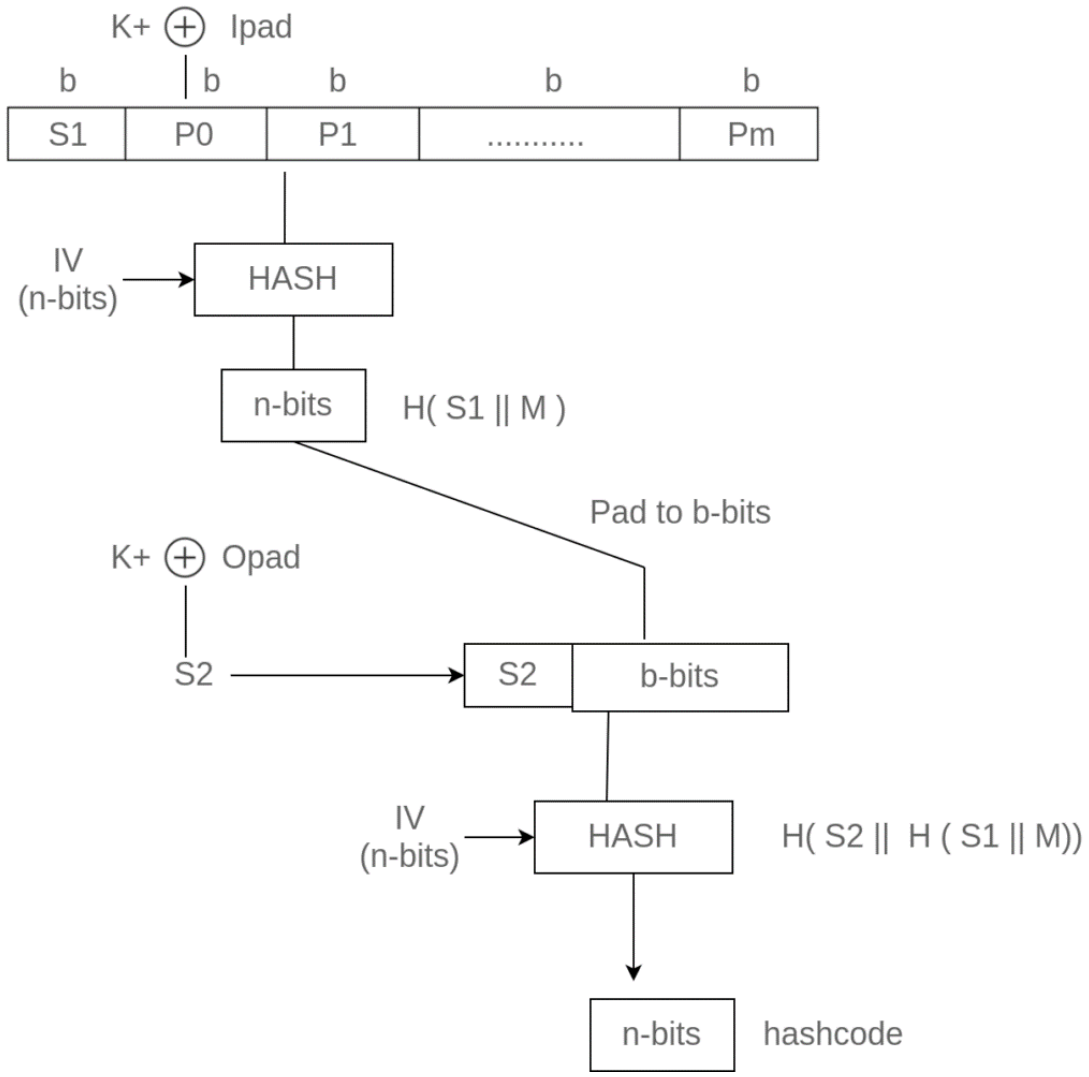
Limitation of MAC –
Cryptoanalyst can easily reach plain text or even the key. To overcome this we move on to hash functions which are “One way”.

HMAC (Hash-based Message Authentication Code) is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key. Like any of the MAC, it is used for both data integrity and authentication.

Working of HMAC
HMACs provides client and server with a shared private key that is known only to them. The client makes a unique hash (HMAC) for every request. When the client requests the server, it hashes the requested data with a private key and sends it as a part of request. Both the message and key are hashed in separate steps making it secure. When the server receives request, it makes its own HMAC. Both the HMACS are compared and if both are equal, the client is considered legitimate.

```
HMAC = hashFunc(secret key + message)
```

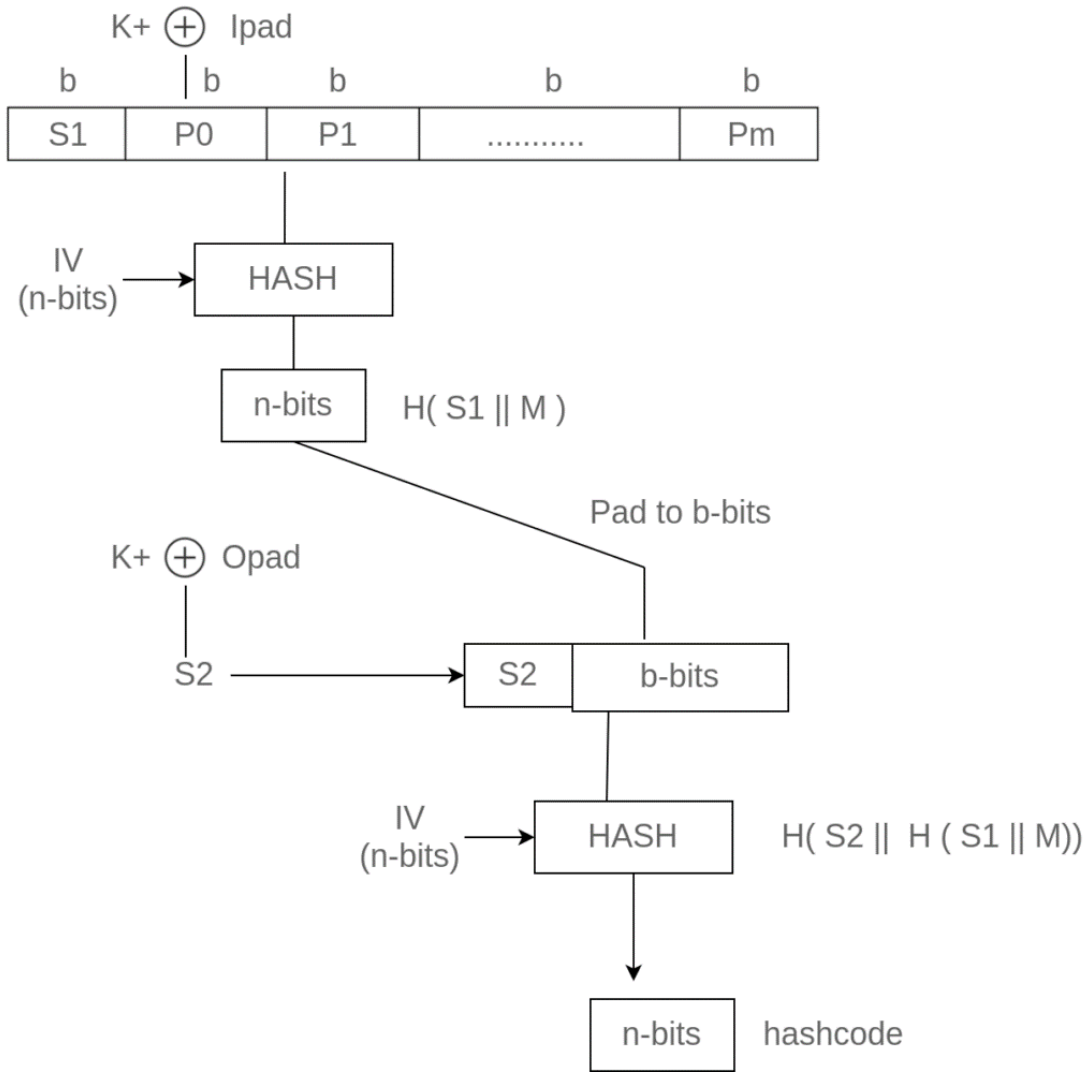
Note: Digital signatures are nearly similar to HMACs i.e they both employ a hash function and a shared key. The difference lies in the keys i.e HMACs use symmetric key(same copy) while Signatures use asymmetric (two different keys).



In HMAC we have to apply the hash function along with a key on the plain text. The hash function will be applied to the plain text message. But before applying, we have to compute S bits and then append it to plain text and after that apply hash function. For generating those S bits we make use of a key that is shared between the sender and receiver.

Using key K (0 < K < b), K+ is generated by padding O's on left side of key K until length becomes b bits. The reason why it's not padded on right is change(increase) in the length of key. b bits because it is the block size of plain text. There are two predefined padding bits called ipad and opad. All this is done before applying hash function to the plain text message.

```
ipad - 00110110
opad - 01011100
```



Now we have to calculate S bits

K+ is EXORed with ipad and the result is S1 bits which is equivalent to b bits since both K+ and ipad are b bits. We have to append S1 with plain text messages. Let P be the plain text message. S1, p0, p1 upto Pm each is b bits. m is the number of plain text blocks. P0 is plain text block and b is plain text block size. After appending S1 to Plain text we have to apply HASH algorithm (any variant). Simultaneously we have to apply initialisation vector (IV) which is a buffer of size n-bits. The result produced is therefore n-bit hashcode i.e H(S1 || M).

Similarly, n-bits are padded to b-bits And K+ is EXORed with opad producing output S2 bits. S2 is appended to the b-bits and once again hash function is applied with IV to the block. This further results into n-bit hashcode which is H(S2 || H(S1 || M)).

SUMMARY

1.Select K.
If K < b, pad 0's on left until k=b. K is between 0 and b (0 < K < b)

2.EXOR K+ with ipad equivalent to b bits producing S1 bits.

3.Append S1 with plain text M

4.Apply SHA-512 on (S1 || M)

5.Pad n-bits until length is equal to b-bits

6.EXOR K+ with opad equivalent to b bits producing S2 bits.

7.Append S2 with output of step 5.

8.Apply SHA-512 on step 7 to output n-bit hashcode.

OFFSET CODEBOOK MODE

Offset codebook mode (OCB mode) is an authenticated encryption mode of operation for cryptographic block ciphers.

OCB mode was designed to provide both message authentication and privacy. It is essentially a scheme for integrating a message authentication code (MAC) into the operation of a block cipher. In this way, OCB mode avoids the need to use two systems: a MAC for authentication and encryption for privacy. This results in lower computational cost compared to using separate encryption and authentication functions.

DIGITAL SIGNATURE STANDARD (DSS)

Digital Signature Standard (DSS) is a Federal Information Processing Standard(FIPS) which defines algorithms that are used to generate digital signatures with the help of Secure Hash Algorithm(SHA) for the authentication of electronic documents. DSS only provides us with the digital signature function and not with any encryption or key exchanging strategies.

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

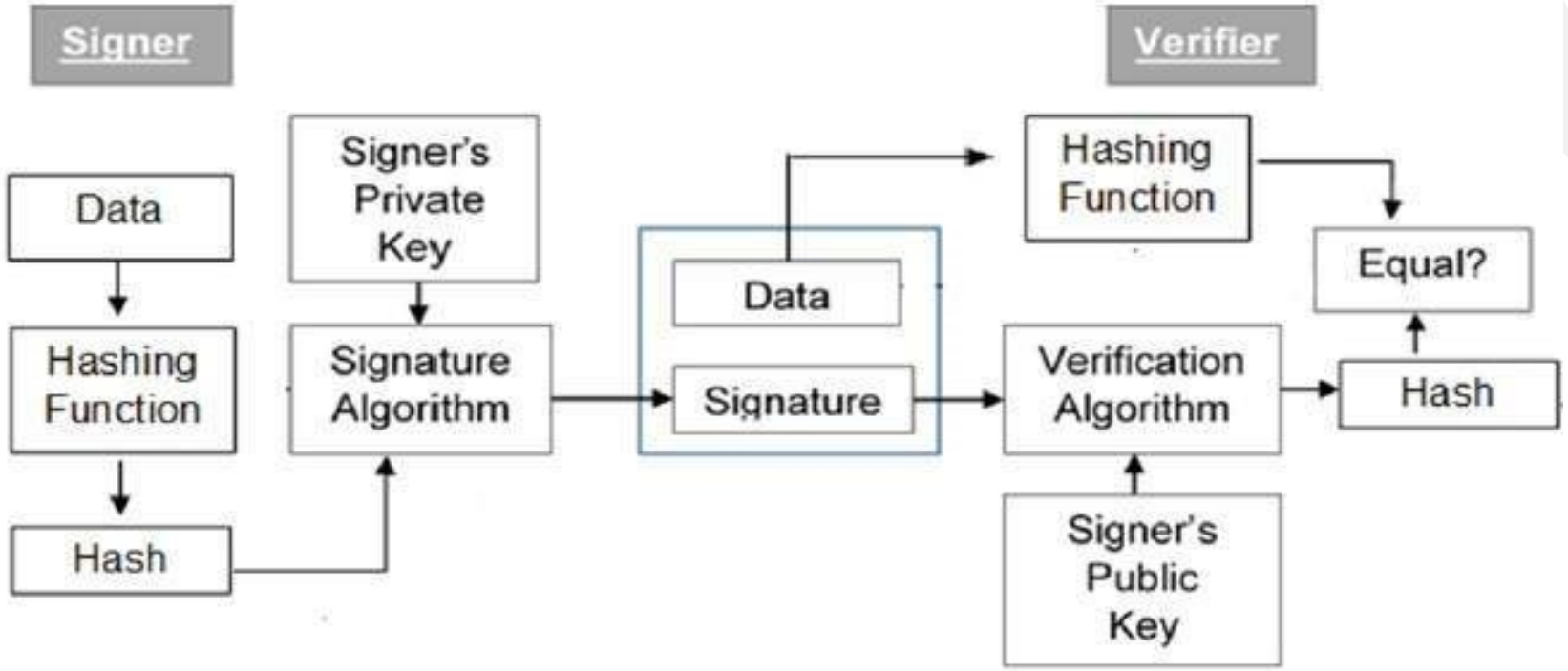
Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

The receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature

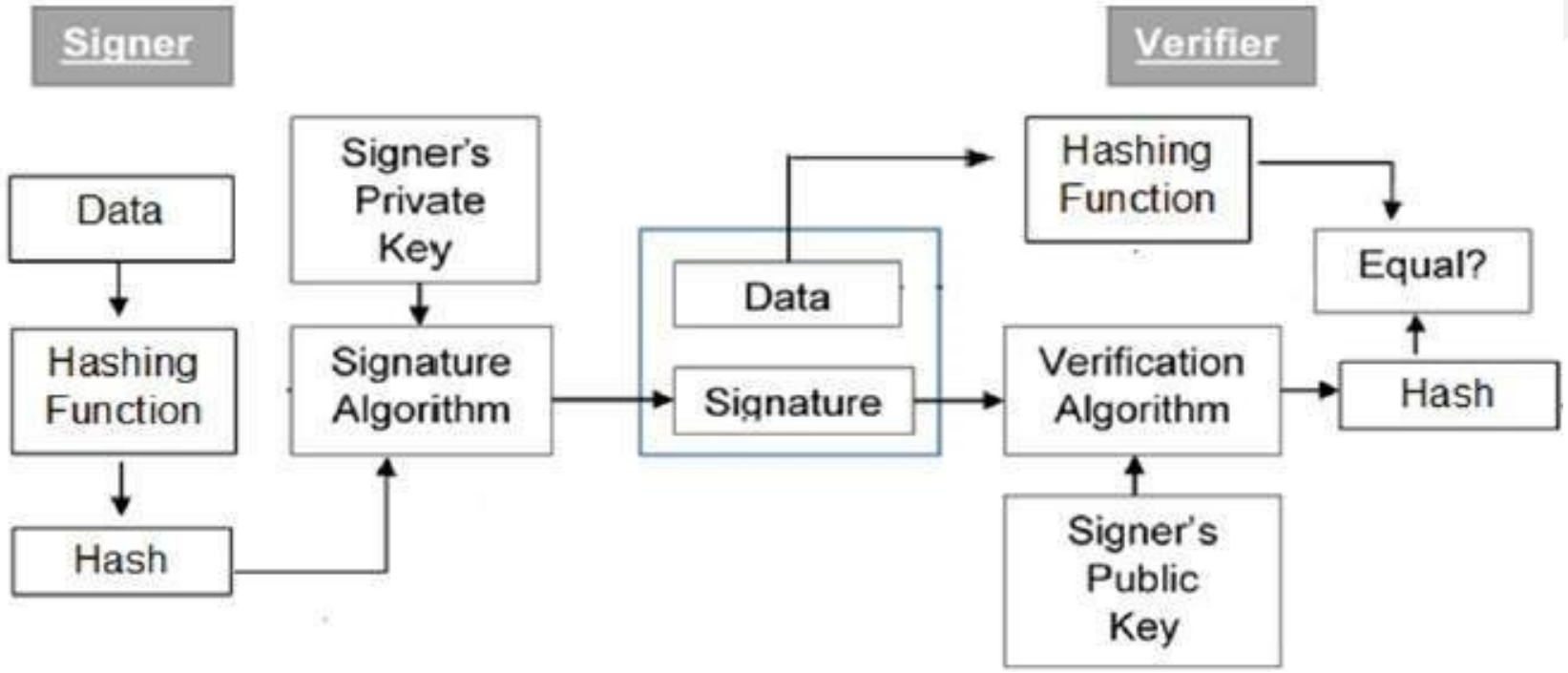
As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration −



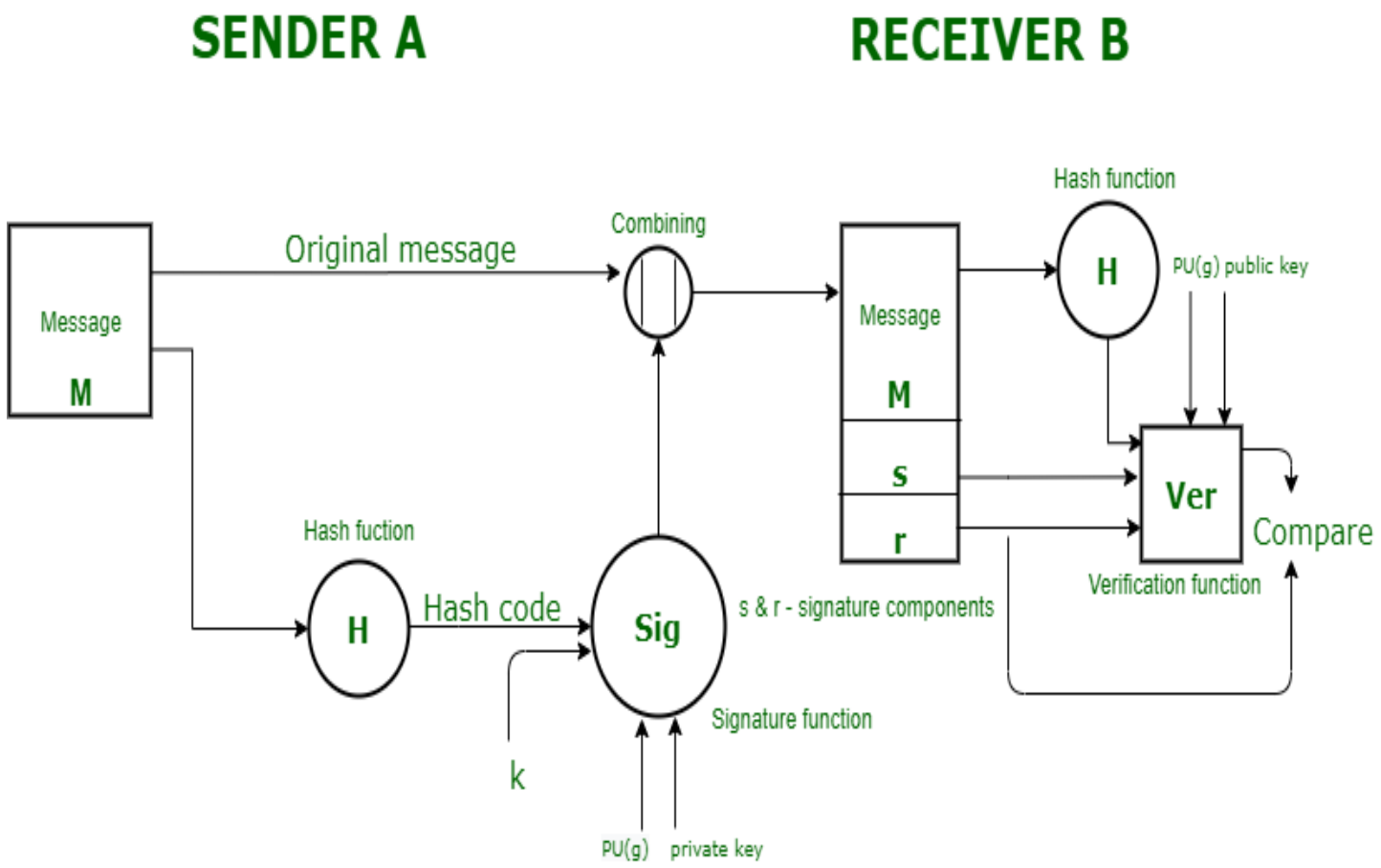
- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.

Model of Digital Signature

•Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
•Verifier also runs same hash function on received data to generate hash value.
•For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
•Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.



DSS ON SENDER'S SIDE



Sender Side :

In DSS Approach, a hash code is generated out of the message and following inputs are given to the signature function –

1.The hash code.

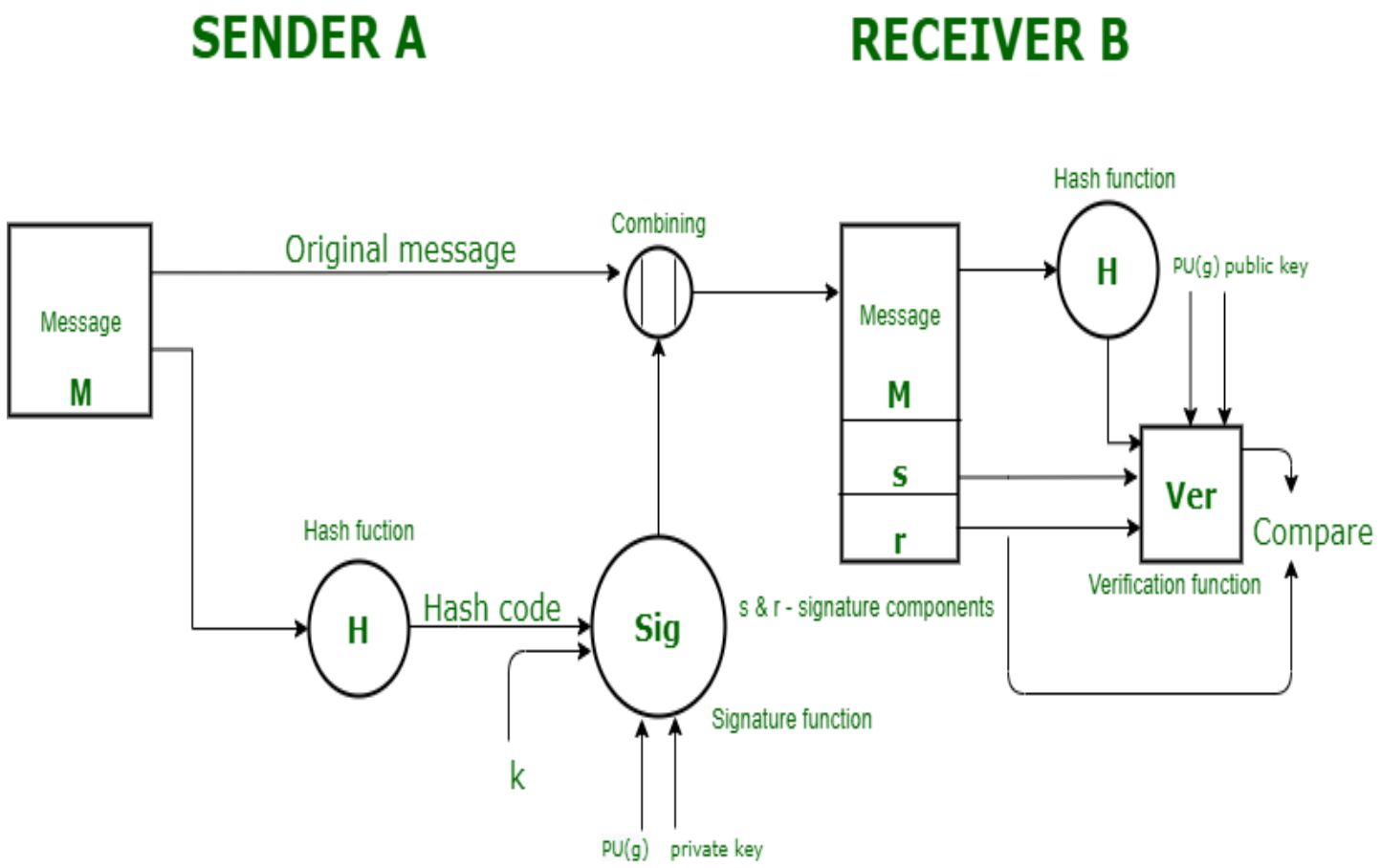
2.The random number 'k' generated for that particular signature.

3.The private key of the sender i.e., PR(a).

4.A global public key(which is a set of parameters for the communicating principles) i.e., PU(g).

These input to the function will provide us with the output signature containing two components – 's' and 'r'. Therefore, the original message concatenated with the signature is sent to the receiver.

DSS ON THE RECEIVER'S SIDE



Receiver Side :

At the receiver end, verification of the sender is done. The hash code of the sent message is generated. There is a verification function which takes the following inputs –

1.The hash code generated by the receiver.
2.Signature components 's' and 'r'.
3.Public key of the sender.
4.Global public key.

The output of the verification function is compared with the signature component 'r'. Both the values will match if the sent signature is valid because only the sender with the help of it private key can generate a valid signature.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data**.

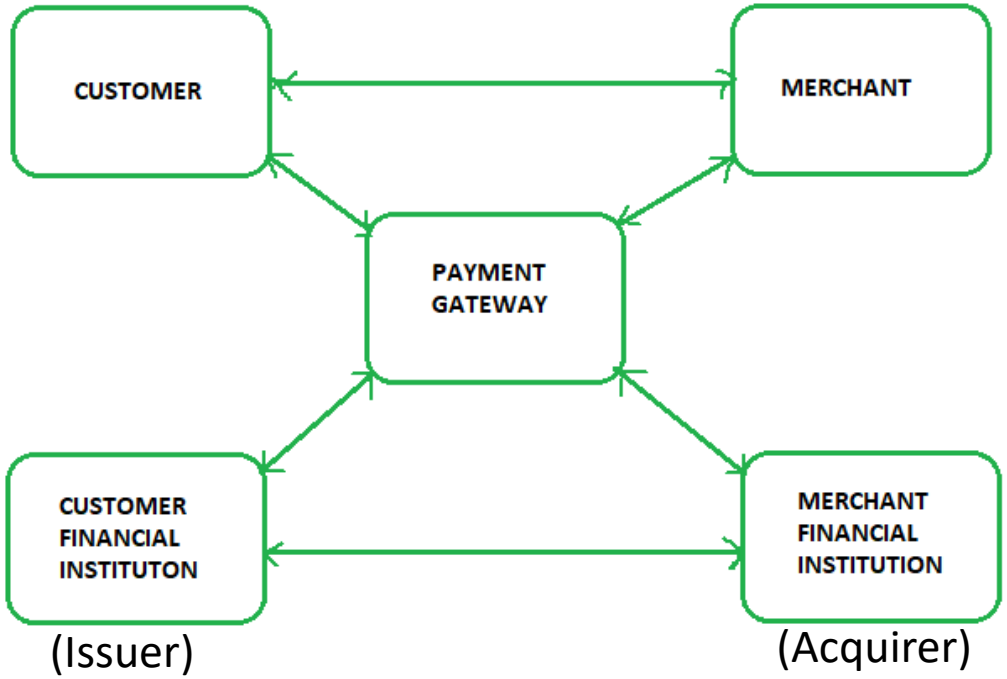
IMPORTANCE OF DIGITAL SIGNATURE

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.
Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature −

•**Message authentication** − When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
•**Data Integrity** − In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
•**Non-repudiation** − Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.
By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely − Privacy, Authentication, Integrity, and Non-repudiation.

SECURE ELECTRONIC TRANSACTION - SET

Secure Electronic Transaction or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard



Requirements in SET :

SET protocol has some requirements to meet, some of the important requirements are :

•It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.

•It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.

•It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.

SET FUNCTIONALITIES

•**Provide Authentication**

- **Merchant Authentication** – To prevent theft, SET allows customers to check previous relationships between merchant and financial institution. Standard X.509V3 certificates are used for this verification.
- **Customer / Cardholder Authentication** – SET checks if use of credit card is done by an authorized user or not using X.509V3 certificates.

•**Provide Message Confidentiality** : Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purpose.

•**Provide Message Integrity** : SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1,

HOW SET WORKS?

Both cardholders and merchants must register with the CA (certificate authority) first, before they can buy or sell on the Internet. Once registration is done, cardholder and merchant can start to do transactions, which involve nine basic steps in this protocol:

1.Customer browses the website and decides on what to purchase
2.Customer sends order and payment information, which includes two parts in one message:
a. Purchase order – this part is for merchant
b. Card information – this part is for merchant's bank only.
3.Merchant forwards card information (part b) to their bank
4.Merchant's bank checks with the issuer for payment authorization
5.Issuer sends authorization to the merchant's bank
6.Merchant's bank sends authorization to the merchant
7.Merchant completes the order and sends confirmation to the customer
8.Merchant captures the transaction from their bank
9.Issuer prints credit card bill (invoice) to the customer

An important innovation introduced in SET is the *dual signature*. The purpose of the dual signature is to link two messages that are intended for two different recipients.

In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit-card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. The link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service.

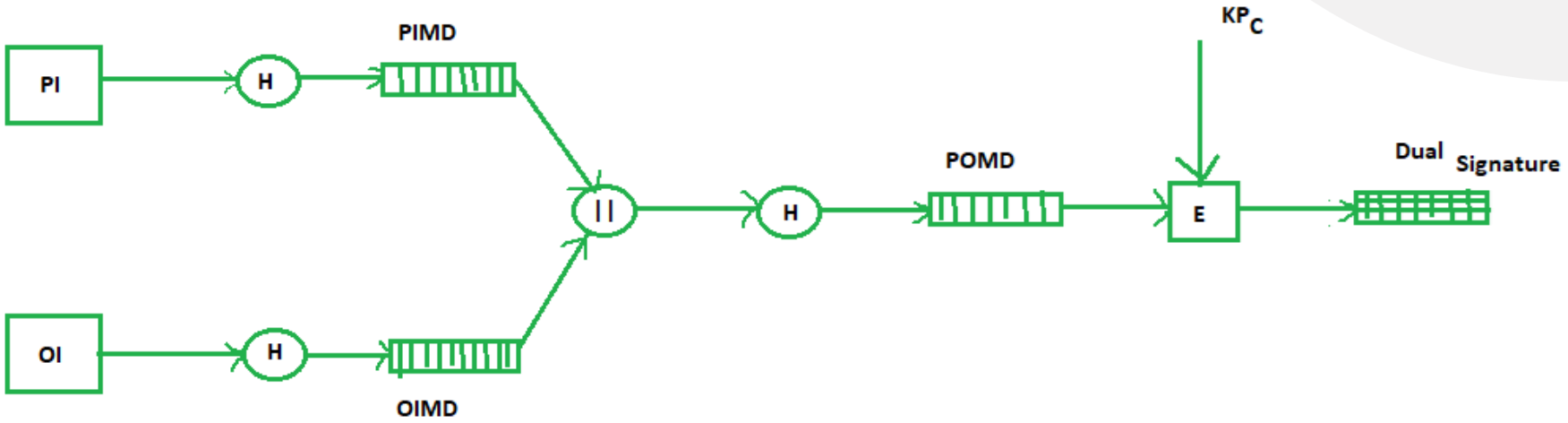
The message digest (MD) of the OI and the PI are independently calculated by the customer. These are concatenated and another MD is calculated from this. Finally, the dual signature is created by encrypting the MD with the customer's secret key. The dual signature is sent to both the merchant and the bank. The protocol arranges for the merchant to see the MD of the PI without seeing the PI itself, and the bank sees the MD of the OI but not the OI itself. The dual signature can be verified using the MD of the OI or PI, without requiring either the OI or PI. Privacy is preserved as the MD can't be reversed, which would reveal the contents of the OI or PI.

DUAL SIGNATURE

The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers :

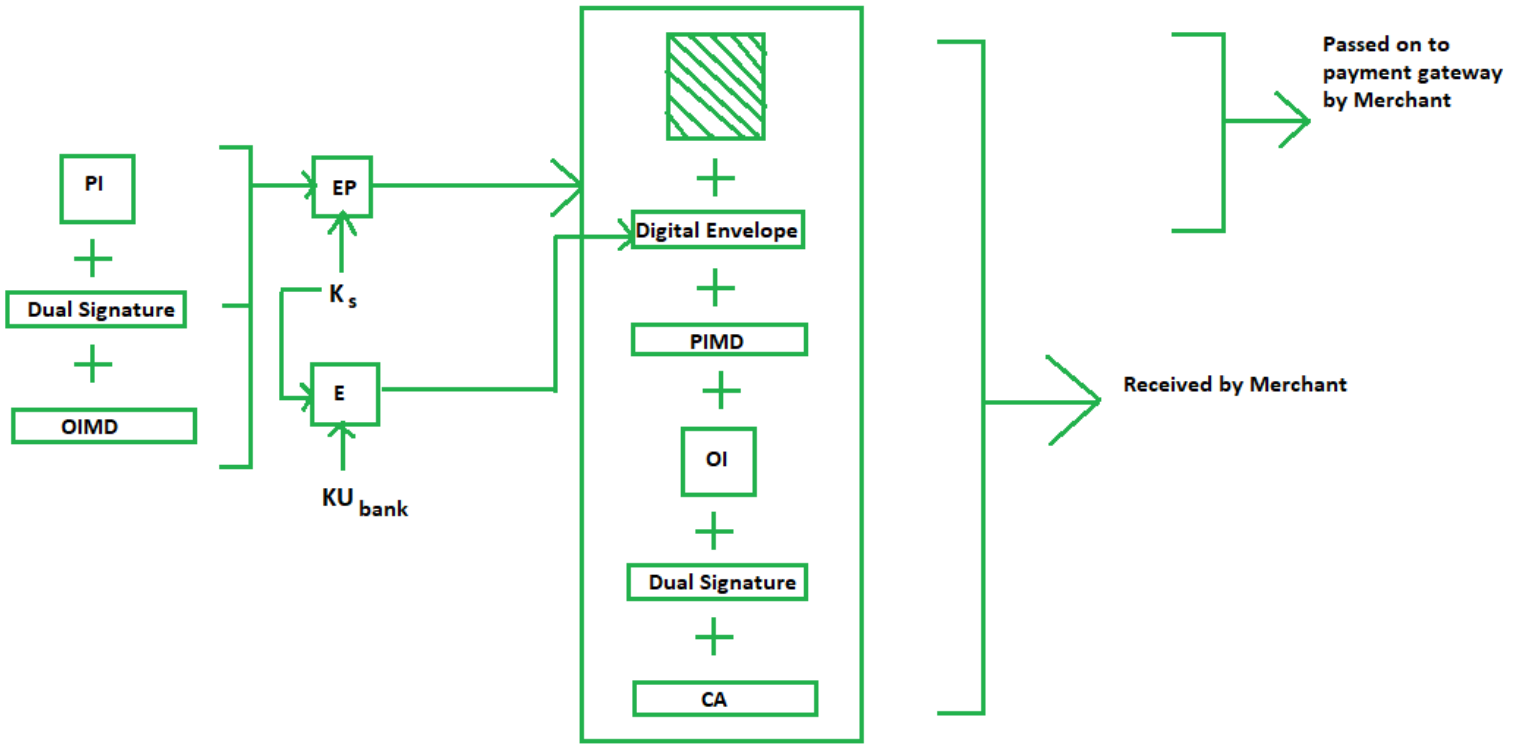
1. Order Information (OI) for merchant and 2. Payment Information (PI) for bank

You might think sending them separately is an easy and more secure way, but sending them in a connected form resolves any future dispute possible. Here is the generation of dual signature:



PI - payment information OI -order information PIMD - Payment Information Message Digest
OIMD - Order Information Message Digest POMD - Payment Order Message Digest H - Hashing E - public key encryption
KPc - customer's private key || stands for append operation Dual signature DS= E(KPc, [H(H(PI)||H(OI))])

PURCHASE REQUEST GENERATION

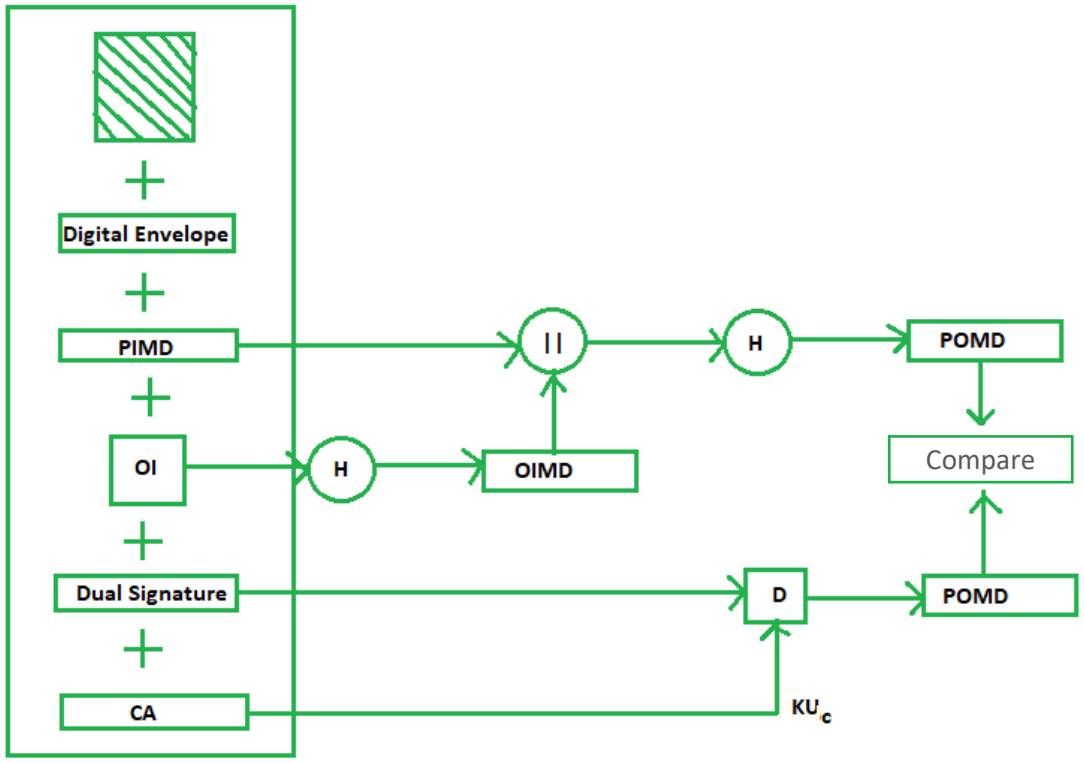


The process of purchase request generation requires three inputs:

- Payment Information (PI)
- Dual Signature
- Order Information Message Digest (OIMD)

EP which is symmetric key encryption Ks is a temporary symmetric key KUbank is public key of bank CA is Cardholder or customer Certificate Digital Envelope = E(KUbank, Ks)

PURCHASE REQUEST VALIDATION



The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as shown in the diagram.

Since we used Customer private key in encryption here we use KUc which is public key of customer or cardholder for decryption ‘D’.

Payment Authorization and Payment Capture :
Payment authorization as the name suggests is the authorization of payment information by merchant which ensures payment will be received by merchant. Payment capture is the process by which merchant receives payment which includes again generating some request blocks to gateway and payment gateway in turn issues payment to merchant.