University of BATNA 2
Department of Computer Science

Chapter 2

Cryptography Basics

Cryptography

- Cryptography is based on the use of the mathematical principles in storing and transmitting data in a particular form so that only those whom it is intended can read and process it.
- The main purpose of the cryptography, as science, is to preserve :
  - ***Confidentiality*** *: Information is kept secret from all but authorized parties.*
  - ***Integrity*** *: Messages have not been modified or altered in transit.*
  - ***Authentication*** *: The sender of a message is authentic. An alternative term is data origin authentication.*
  - ***Non-repudiation*** *: The sender cannot deny later that they sent the message. This means that an entity cannot refuse the ownership of a previous commitment or an action.*

Encryption function

- The general principle of cryptography is based on the use of mathematical functions and algorithms to encrypt data.
- An encryption function or algorithm, is a means of transforming plaintext into ciphertext under the control of a secret key, this process is called encryption or encipherment (تشفير).

$c = e_k(m)$

m is the plaintext,
e is the cipher function,
k is the secret key,
c is the ciphertext.

The decryption or decipherment is the reverse process :

$m = d_k(c)$

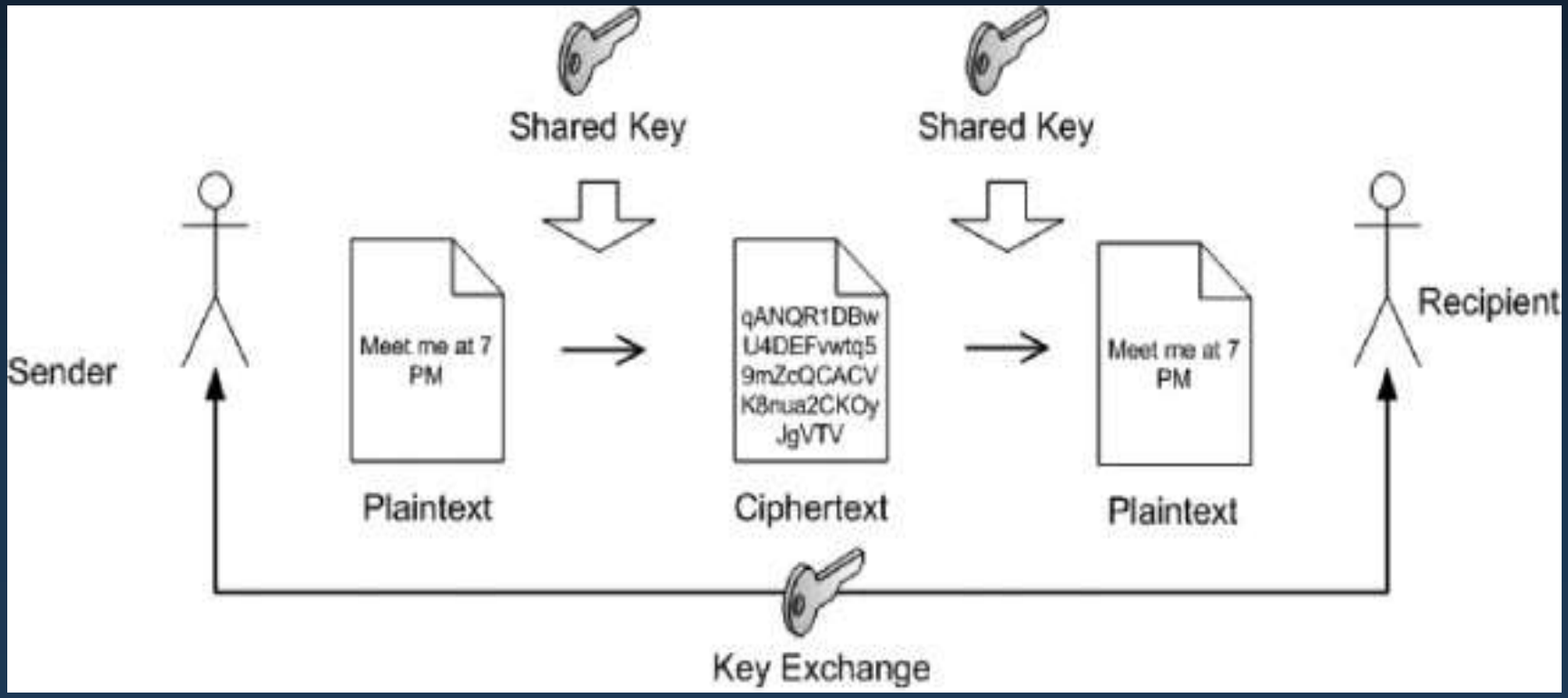
Note that e and d are public, the secrecy of m given c depends totally on the secrecy of k.

Types of cryptosystems

- Three cryptosystems can be distinguished : Symmetric cryptography, Asymmetric cryptography, Hashing
  - **Symmetric cryptosystem** (Secret key cryptosystem) : *The communicating party use the same key (Shared key) for encryption and decryption.*
  - **Asymmetric cryptosystem** (Public key cryptosystem) : *The communicating party use two different types of key, one is publicly available and used for encryption while the other is private and used for decryption.*
  - **Cryptographic Hashing** : *Function used to transform large random size data to small fixed size data. The hash functions does not need any key (0-key).*

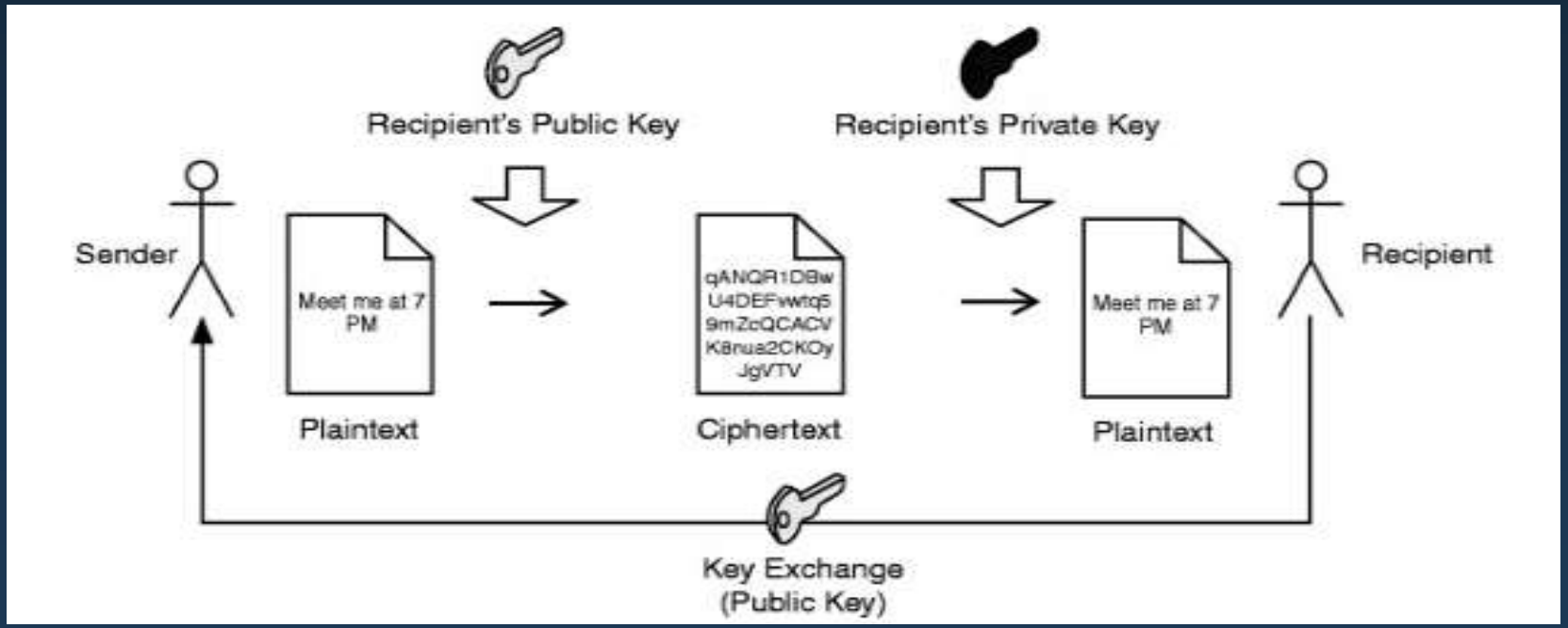
Symmetric cryptosystem : *One-Key*

- Symmetric cryptography is the oldest form of cryptosystem.
- Some cyptosystem : Shift cipher, Caeser, Affine, Vigenere, Hill, AES, DES, IDEA, Blowfish, RC4, RC5, RC6, …
- Example : *In shift cipher* $e_k = (m+k) \bmod 26$ *and* $d_k = (c-k) \bmod 26$



Asymmetric cryptosystem : *Two-Key*

- Based an idea proposed by Diffie, Hellman and Merkle : *It is not necessary that the encryption key is secret*.
- Asymmetric cryptography uses a pair of keys: a public key and a private key that are mathematically related to each other.
- The public key is made public without reducing the security of the process, but the private key is kept safe and private.



Asymmetric cryptosystem : *Two-Key*

- Families of asymmetric cryptograhy :
  - **Integer-Factorization Schemes :** *Several public-key schemes are based on the fact that it is difficult to factor large integers. The most prominent representative of this algorithm family is RSA (Rivest–Shamir–Adleman).*
  - **Discrete Logarithm Schemes :** *There are several algorithms which are based on what is known as the discrete logarithm (DL) problem in finite fields. The most prominent examples include the Diffie–Hellman key exchange, Elgamal encryption or the Digital Signature Algorithm (DSA).*
  - **Elliptic Curve (EC) Schemes :** *A generalization of the discrete logarithm algorithm is elliptic curve public-key schemes. The most popular examples include Elliptic Curve Diffie–Hellman key exchange (ECDH) and the Elliptic Curve Digital Signature Algorithm (ECDSA).*

RSA is the most widely used asymmetric cryptographic scheme.
Therefore, Blockchain is principally based on elliptic curve scheme.

Asymmetric cryptosystem : *RSA*

- Generally used for encryption of small pieces of data, especially for key transport, digital signatures, digital certificates, …
- The keys generation is done by following steps :

public key : $k_{pub} = (n,e)$ *and private key :* $k_{pr} = (d)$

Choose two large primes p *and* q

Compute $n = p \cdot q$

Compute $\varphi(n) = (p-1)(q-1)$

Select the public exponent $e \in \{1,2,\ldots,\varphi(n)-1\}$ *such that* $gcd(e,\varphi(n))=1$

Compute the private key d such that $d \cdot e \equiv 1 \bmod \varphi(n)$

Encryption function : $c = m^e \bmod n$

Decryption function : $m = c^d \bmod n$

Asymmetric cryptosystem : *Diffie–Hellman key exchange*

- The Diffie–Hellman key exchange DHKE (Symetric key agreement) scheme enables two parties to derive a common secret key (Symmetric key) by communicating over an insecure channel.
- It's widely used in cryptographic protocols, such SSL/TLS, …
- The basic idea behind the DHKE is the exponentiation in Zp, p prime, is commutative

$$(g^a)^b \bmod p = (g^b)^a \bmod p$$

- DHKE is based on the discrete logarithm problem in finite fields

  Knowing the value of $g^i \bmod p$, g and p it's very difficult to find i

- DHKE steps : Setup and main protocols

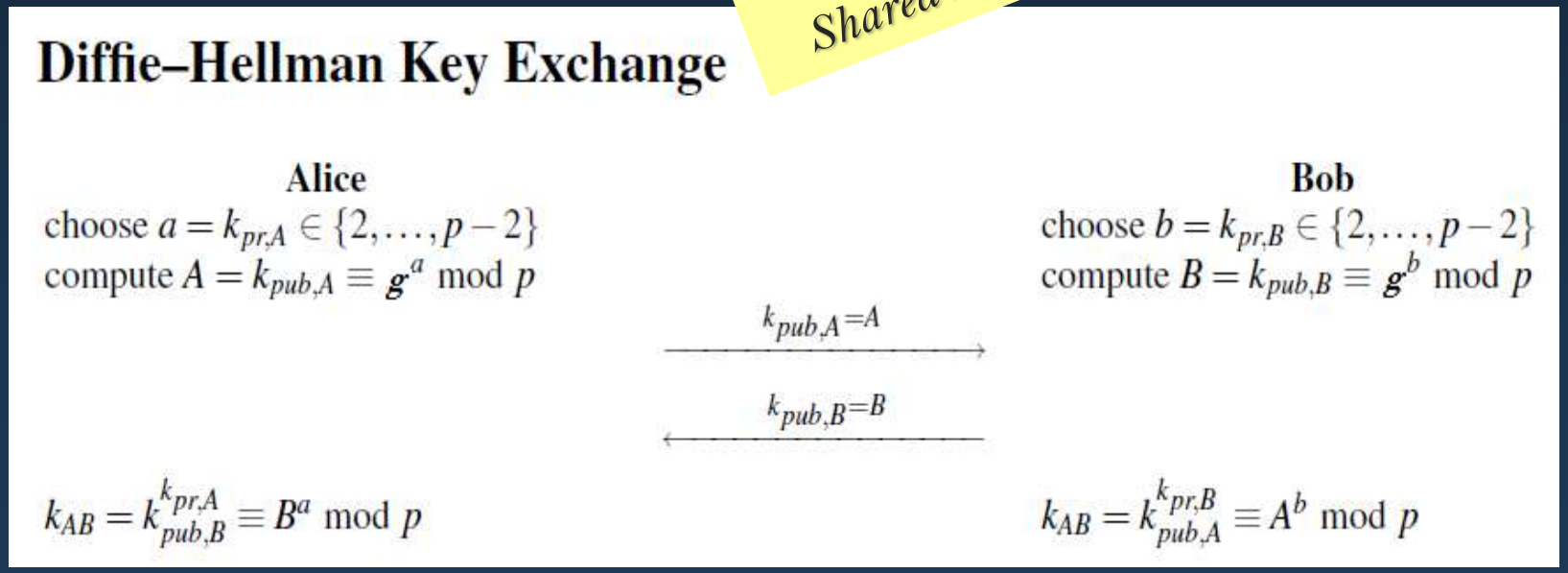
  Setup
  - *Alice and Bob choos two numbers p and g : Domain parameters, Public*
  - *p is a large prime number on the order of 300 decimal digits (1024 bits)*
  - *g is a primitive root of p named generator*

Asymmetric cryptosystem : *Diffie–Hellman key exchange*

Main

- *Alice chooses a large random number* a ($k_{pr,A}$) *such that* $1<a<p-1$ *and calculates* $A = g^a \bmod p$ ($k_{pub,A}$)
- *Bob chooses another large random number* b ($k_{pr,B}$) *such that* $1<b<p-1$ *and calculates* $B = g^b \bmod p$ ($k_{pub,B}$)
- *Alice sends* A *to Bob and Bob sends* B *to Alice.*
- *Alice calculates* $k_{A,B} = B^a \bmod p$
- *Bob calculates* $k_{A,B} = A^b \bmod p$

$k_{A,B}$ is the Shared Secret key



Asymmetric cryptosystem : *Diffie–Hellman key exchange*

Example : Let us give a more realistic example. We used a program to create a random integer of 512 bits (the ideal is 1024 bits). The integer p is a 159-digit number. We also choose g, a, and b as shown below :

p	764624298563493572182493765955030507476338096726949748923573772860925235666660755423637423309661180033338106194730130950414738700999178043654878580798758 1
g	2
a	557
b	273

The following shows the values of A, B and k_{AB} :

A	844920284205665505216172947491035094143433698520012660862863631067673619959280828586700802131859290945140217500319973312945836083821943065966020157955354
B	435262838709200379470747114895581627636389116262115557975123379218566310011435718208390040181876486841753831165342691630263421106721508589625520128859414 3
K	155638000664522290596225827523270765273218046944423678520320400146406500887936651204257426776608327911017153038674561252213151610976584200120408643361774 0

Asymmetric cryptosystem : *Elliptic Curve*

- Elliptic curves are cubic (x^3) mathematic equations, in contrast to ellipse which is formed by quadratic curves (x^2). The elliptic curves have favorable characteristics, and especially the X-axis symmetry, in the field of cryptography and therefore in security
- Elliptic Curve Cryptography (ECC) provides the same level of security as RSA or discrete logarithm systems with considerably shorter operands (approximately 160–256 bit vs. 1024–3072 bit).
- ECC is based on the generalized discrete logarithm (DL) problem, and thus DL-protocols such as the Diffie–Hellman key exchange can also be realized using elliptic curves
- In many cases, ECC has performance advantages (fewer computations) and bandwidth advantages (shorter signatures and keys) over RSA and DL schemes.

Security Level	1	2	3	4	5	6
ECC-key-Size	112	160	224	256	384	512
RSA-key-Size	512	1024	2048	3072	7680	15360

Asymmetric cryptosystem : *Elliptic Curve*

- Elliptic curve

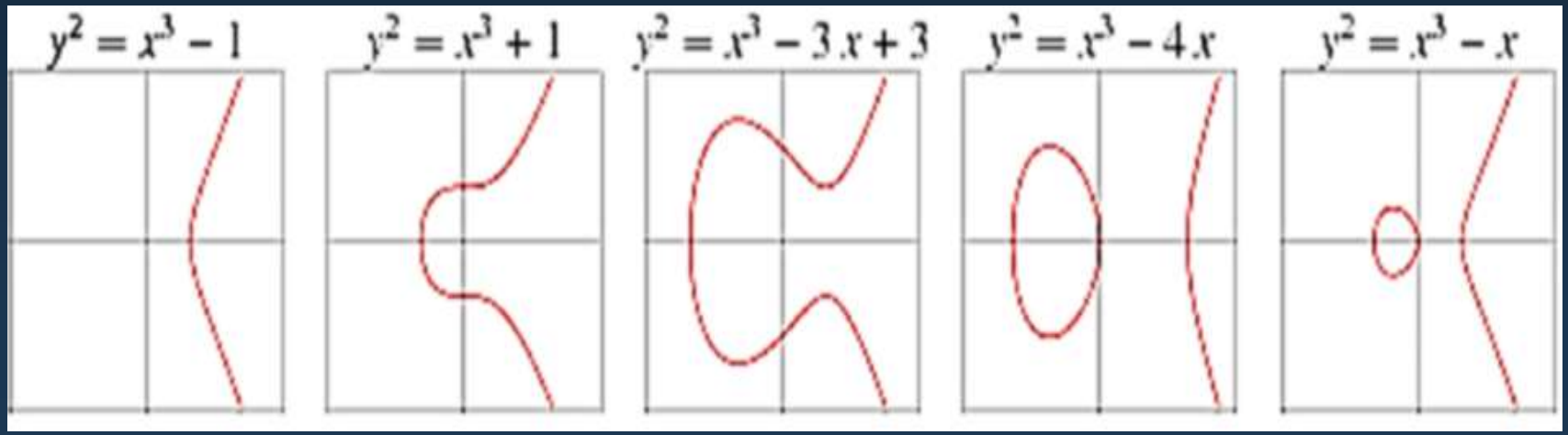
The elliptic curve over Z_p, $p > 3$, *is the set of all pairs* $(x,y) \in Z_p$ *which fulfill*

$$y^2 = x^3 + ax + b \bmod p$$

together with an imaginary or abstract point at infinity $\mathcal{O}$, *where*

$$a,b \in Z_p$$

and the condition $4a^3 + 27\,b^2 \neq 0 \bmod p$



Asymmetric cryptosystem : *Elliptic Curve*

- Group Operations on Elliptic Curves

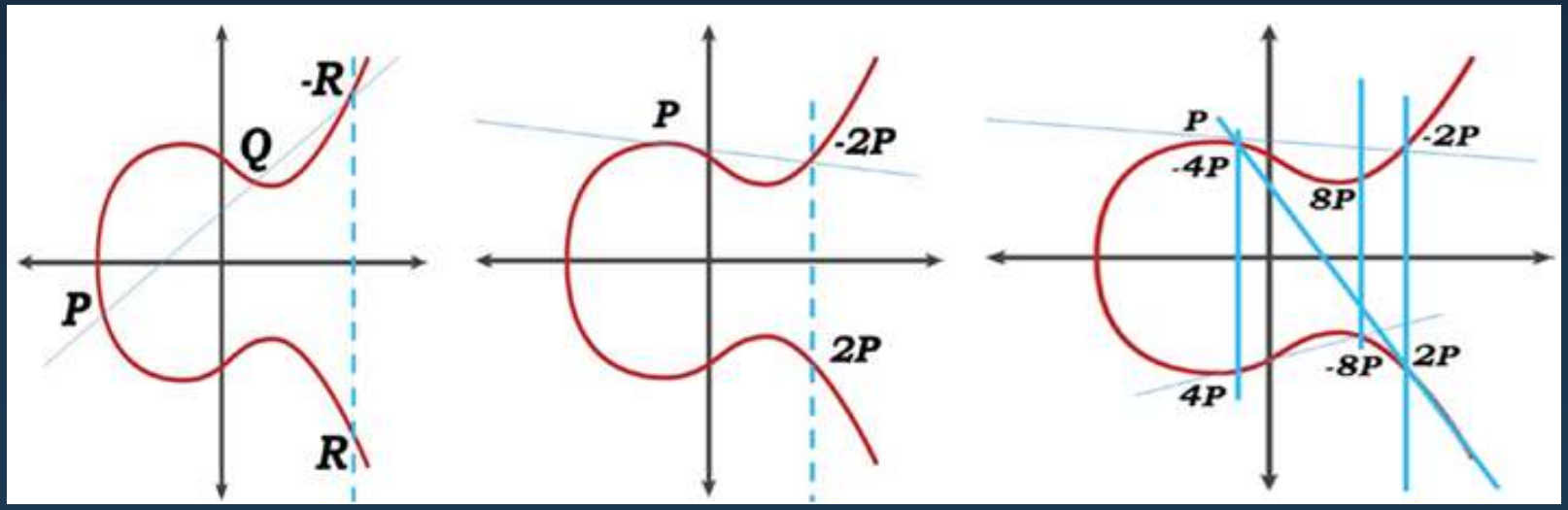
  Let's denote the group operation with the addition symbol "+". "Addition" means that given two points and their coordinates, say $P = (x1,y1)$ and $Q = (x2,y2)$, we have to compute the coordinates of a third point R such that :

  $$P+Q = R \Leftrightarrow (x1,y1)+(x2,y2) = (x3,y3)$$

  Point Addition $R = P+Q \,/\, P \neq Q$

  Point Doubling $2P = P+P$

  Identity or Neutral Element $P + 6 = P$



Asymmetric cryptosystem : *Elliptic Curve*

- In a cryptosystem we cannot perform geometric constructions. However, by applying simple coordinate geometry, we can express both of the geometric constructions from above through analytic expressions.

Elliptic Curve Point Addition and Point Doubling

$$x_3 = s^2 - x_1 - x_2 \bmod p$$
$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \bmod p \text{ ; if } P \neq Q \text{ (point addition)} \\ \frac{3x_1^2 + a}{2y_1} \bmod p \text{ ; if } P = Q \text{ (point doubling)} \end{cases}$$

Asymmetric cryptosystem : *Elliptic Curve*

Example : We consider a curve over the small field Z_{17}

$$E : y^2 = x^3 + 2x + 2 \bmod 17$$

We want to double the point P = (5,1)

$$2P = P + P = (5,1) + (5,1) = (x_3, y_3)$$

$$s = \frac{3x_1^2 + a}{2y_1} = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 \equiv 9 \cdot 9 \equiv 13 \bmod 17$$

$$x_3 = s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \bmod 17$$

$$y_3 = s(x_1 - x_3) - y_1 = 13(5 - 6) - 1 = -14 \equiv 3 \bmod 17$$

$$2P = (5,1) + (5,1) = (6,3)$$

- The points on an elliptic curve E together with 6 have cyclic sub groups. Under certain conditions all points on an elliptic curve form a cyclic group. Let's denote n the number of point operations on the curve until the resultant is : nP=6

Asymmetric cryptosystem : *Elliptic Curve*

Example : We want to find all points on the curve

$$E : y^2 = x^3 + 2x + 2 \bmod 17$$

Let's start with the primitive element $P=(5,1)$. We compute now all "powers" of P. More precisely, since the group operation is addition, we compute $P, 2P, \ldots, nP$. Here is a list of the elements that we obtain :

$2P = (5,1)+(5,1) = (6,3)$	$8P = (13,7)$	$14P = (9,1)$
$3P = 2P+P = (10,6)$	$9P = (7,6)$	$15P = (3,16)$
$4P = (3,1)$	$10P = (7,11)$	$16P = (10,11)$
$5P = (9,16)$	$11P = (13,10)$	$17P = (6,14)$
$6P = (16,13)$	$12P = (0,11)$	$18P = (5,16)$
$7P = (0,6)$	$13P = (16,4)$	$19P = \mathcal{O}$

From now on, the cyclic structure becomes visible since :

$20P = 19P+P = \mathcal{O} + P = P$ $\quad 21P = 2P$ $\quad \ldots\ldots\ldots\ldots$

It is also instructive to look at the last computation above, which yielded :

$18P+P = \mathcal{O}$

This means that $P=(5,1)$ is the inverse of $18P=(5,16)$, and vice versa.

Asymmetric cryptosystem : *Elliptic Curve*

- The ECC is a great technique to generate the keys, but is used along side other techniques for digital signatures and key exchange.
- For example, ECDH is quite popularly used for key exchange and ECDSA is used for digital signatures.
- The ECDSA is a type of DSA that uses ECC for key generation. Its purpose is digital signature, and not encryption. ECDSA can be a better alternative to RSA in terms of smaller key size, better security, and higher performance.
- The ECDSA, e.d secp256k, is an important cryptographic component used in Bitcoin and Ethereum blockchains.
- The ECDSA includes broadly three steps: key generation, signature generation, and signature verification.

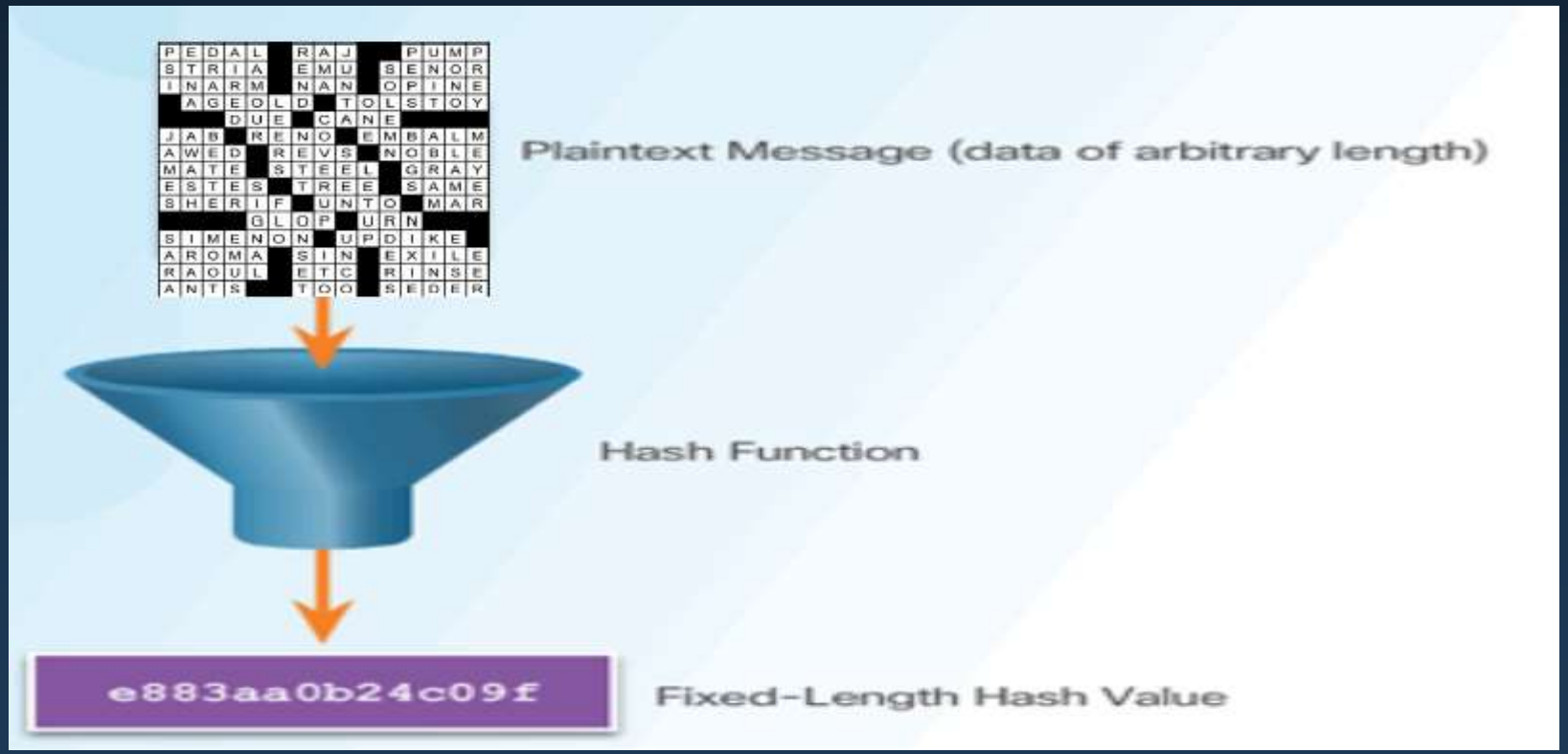
Elliptic Curve Diffie–Hellman Key Exchange (ECDH)

Alice		Bob
choose $k_{prA} = a \in \{2,3,\ldots,\#E-1\}$		choose $k_{prB} = b \in \{2,3,\ldots,\#E-1\}$
compute $k_{pubA} = aP = A = (x_A, y_A)$		compute $k_{pubB} = bP = B = (x_B, y_B)$
	$A \longrightarrow$	
	$\longleftarrow B$	
compute $aB = T_{AB}$		compute $bA = T_{AB}$

Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$.

Cryptographic Hash function

- Hash function is a form of cryptographic scheme, it is an integral part of blockchain data structure.
- It is a mathematical function or algorithm that maps data of arbitrary size to a bit array of a fixed size
- It is a one-way function, that is, a function which is practically infeasible to invert.



Cryptographic Hash function

- An ideal cryptographic h hash function should has the following main properties :

  Arbitrary message size *h(x) can be applied to messages x of any size.*

  Fixed output length *h(x) produces a hash value z of fixed length.*

  Efficiency *h(x) is relatively easy to compute.*

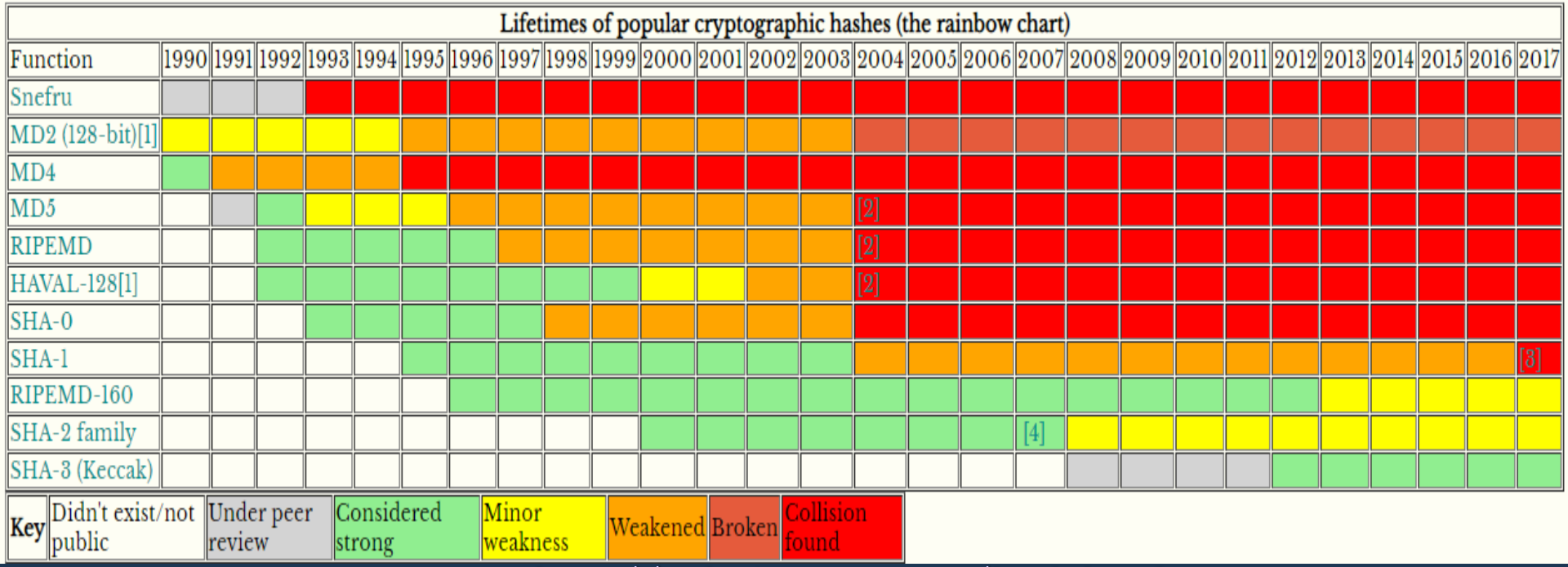
  Preimage resistance *For a given output z, it is impossible to find any input x such that h(x) = z, i.e, h(x) is one-way.*

  Second preimage resistance *Given x1, and thus h(x1), it is computationally infeasible to find any x2 such that h(x1) = h(x2).*

  Collision resistance *It is computationally infeasible to find any pairs x1 ≠x2 such that h(x1) = h(x2).*

Cryptographic Hash function

- Cryptographic hash functions have many information-security applications, notably in digital signatures, message authentication codes (MACs), blockchain data structure and mining, and other forms of authentication.
- There are many cryptographic hash algorithms, the table below enumerate some hash functions and it's life cycle. SHA-2 (Secure Hash Algorithm 2) is a family including SHA-224, SHA-256, SHA-384, SHA-512.

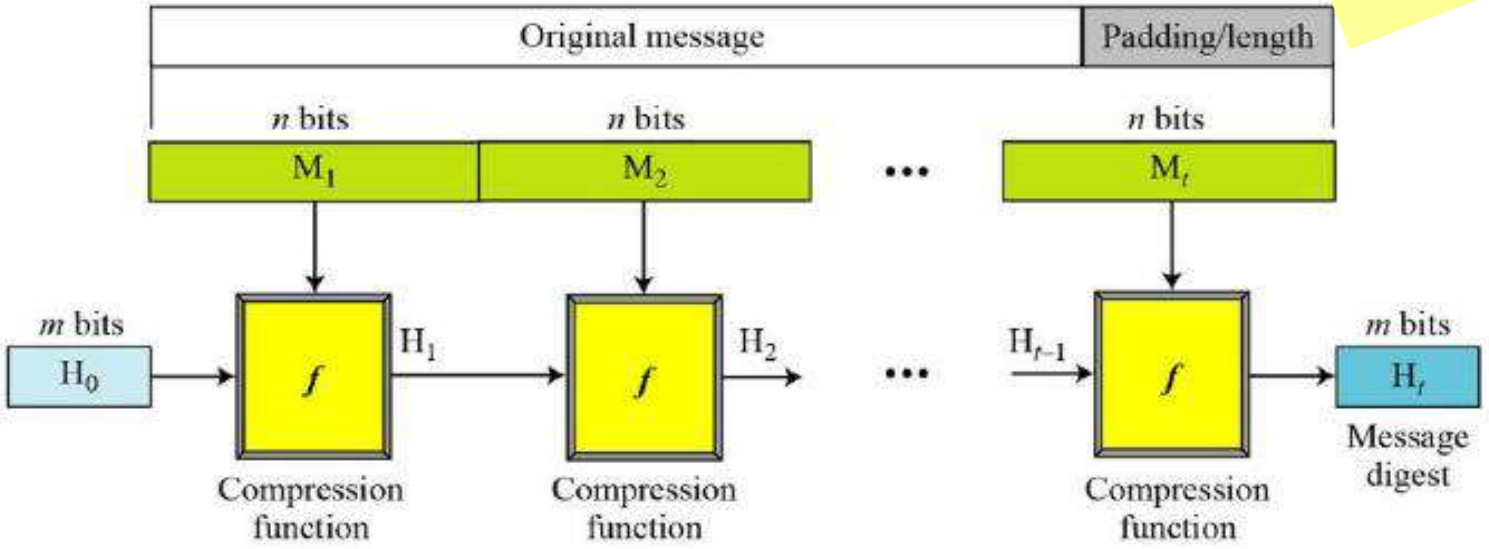


Source http://valerieaurora.org/hash.html

Cryptographic Hash function : *SHA-1*

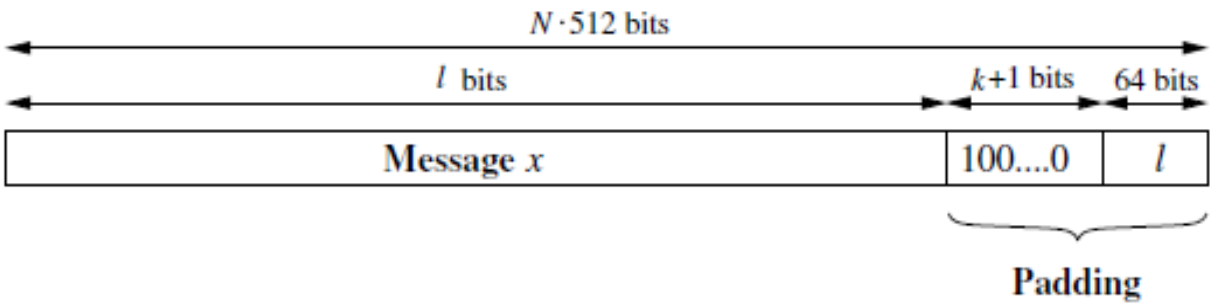
- The Secure Hash Algorithm (SHA-1) is the most widely used message digest function of the MD4 family. It is based on a Merkle Damgard scheme.
- An interpretation of the SHA-1 algorithm is that the compression function works like a block cipher, where the input is the previous hash value H_{i-1} and the key is formed by the message block x_i

Merkle Damgard Scheme



Cryptographic Hash function : *SHA-1*

- SHA-1 produces a 160-bit message digest output. Before the hash computation, the algorithm has to preprocess the message.
- The compression function processes the message in 512-bit chunks. The compression function consists of 80 rounds which are divided into four stages of 20 rounds each.
- Preprocessing : Before the actual hash computation, the message x has to be padded to fit a size of a multiple of 512 bit. For the internal processing, the padded message must then be divided into blocks. Also, the initial value H_0 is set to a predefined constant
- Padding : l is the length of the message x



Cryptographic Hash function : *SHA-1*

- Dividing the padded message : Prior to applying the compression function, we need to divide the message into 512-bit blocks $x_1, x_2, \ldots, x_n$. The i th block x_i is subdivided into 16 words $x_i^{(k)}$ of size of 32 bits :

$$x_i = (x_i^{(0)} \; x_i^{(1)} \; \ldots \; x_i^{(15)})$$

- Initial value H_0 : A 160-bit buffer is used to hold the initial hash value for the first iteration. The five 32-bit words are fixed and given in hexadecimal notation as:

$$A = H_0^{(0)} = 67452301$$
$$B = H_0^{(1)} = EFCDAB89$$
$$C = H_0^{(2)} = 98BADCFE$$
$$D = H_0^{(3)} = 10325476$$
$$E = H_0^{(4)} = C3D2E1F0.$$

Cryptographic Hash function : *SHA-1*

- Hash Computation : As mensionned before, each message block x_i is processed in four stages with 20 rounds each. The algorithm uses :
  - A message schedule which computes a 32-bit word $W_0, W_1, \ldots, W_{79}$ for each of the 80 rounds The words W_j are derived from the **512-bit** message block as follows :

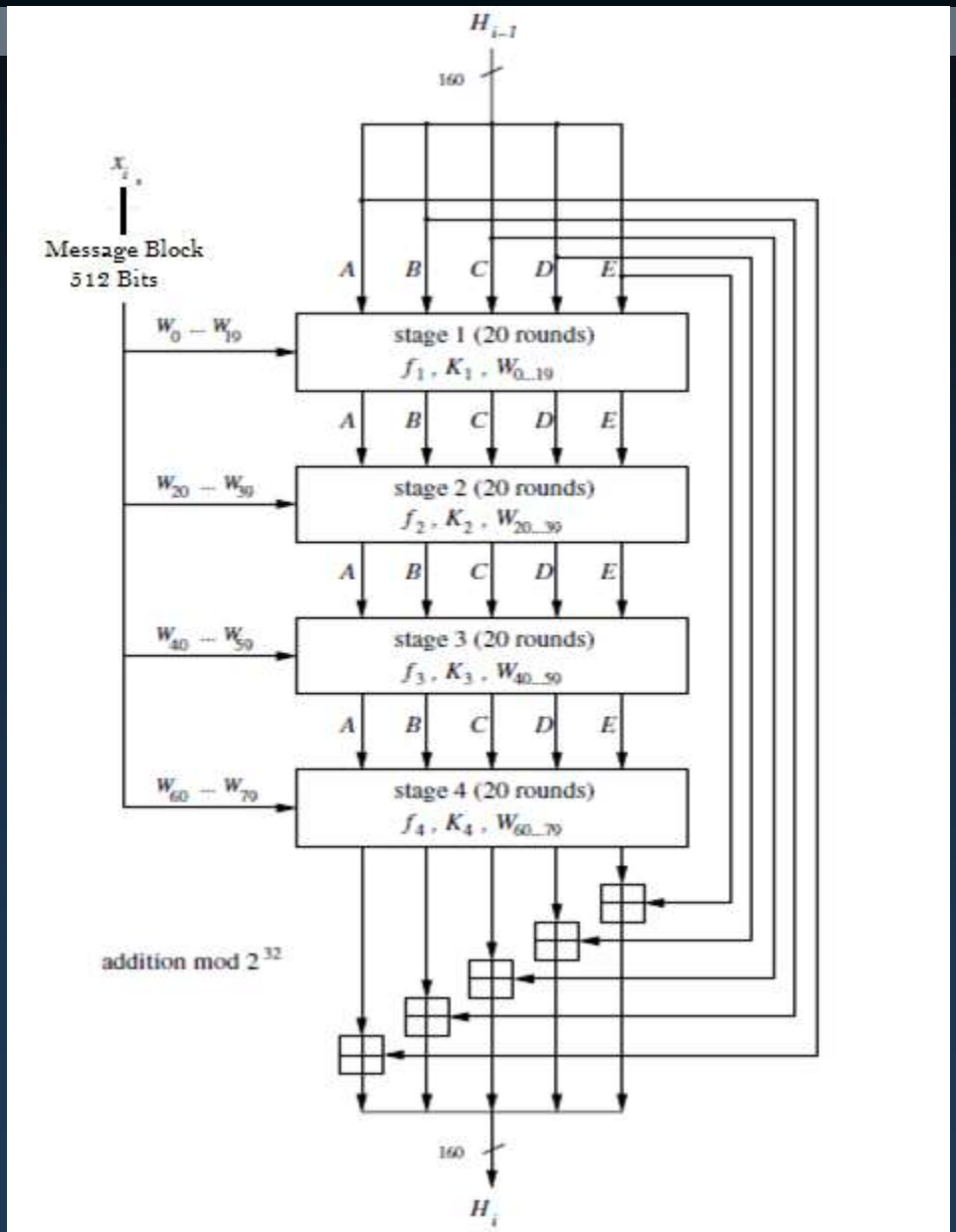
$$W_j = \begin{cases} x_i^{(j)} & 0 \le j \le 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \lll 1 & 16 \le j \le 79, \end{cases}$$

  Where $X_{\lll n}$ indicates a circular left shift of the word X by n bit positions.
  - Five working registers of size of 32 bits A,B,C,D,E.
  - A hash value H_i *consisting of five 32-bit words*

$$H_i^{(0)}, H_i^{(1)}, H_i^{(2)}, H_i^{(3)}, H_i^{(4)}$$

- In the beginning, the hash value holds the initial value H_0

Cryptographic Hash function : *SHA-1*

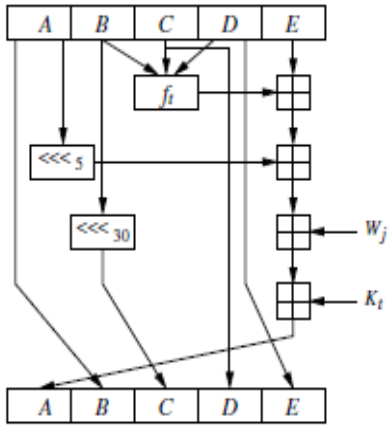


Cryptographic Hash function : *SHA-1*

- The four SHA-1 stages have a similar structure but use different internal functions f_t and stage-dependent constants K_t, where $1 \leq t \leq 4$.
- The output after 80 rounds is added to the input value H_{i-1} modulo 2^{32} .
- The operation within round j in stage t is given by the equation and depicted by the figure below.
- According to the table and depending on the stage every 20 rounds a new function and a new constant are being used.

Cryptographic Hash function : *SHA-1*

$$A,B,C,D,E = (E + f_t(B,C,D) + (A)_{\lll 5} + W_j + K_t), A, (B)_{\lll 30}, C, D$$



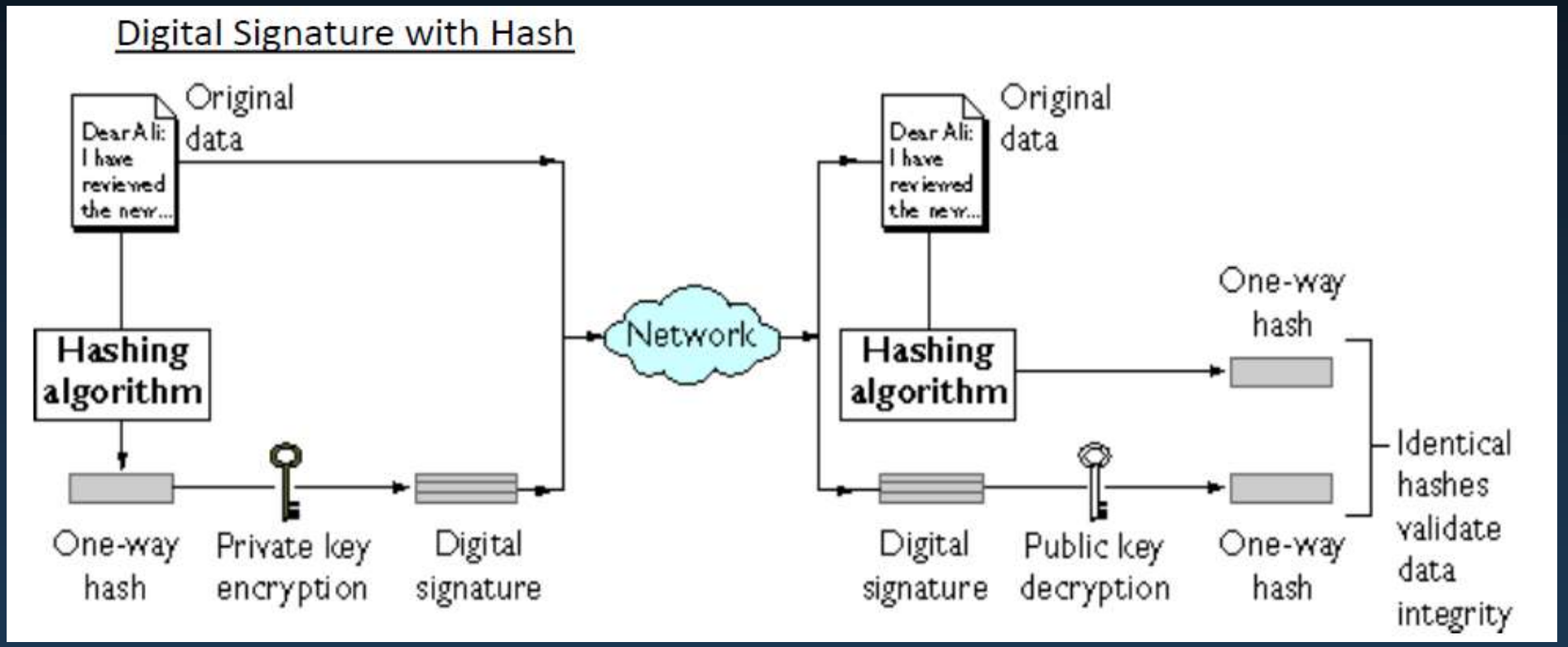
Stage t	Round j	Constant K_t	Function f_t
1	0...19	$K_1 = \texttt{5A827999}$	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20...39	$K_2 = \texttt{6ED9EBA1}$	$f_2(B,C,D) = B \oplus C \oplus D$
3	40...59	$K_3 = \texttt{8F1BBCDC}$	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60...79	$K_4 = \texttt{CA62C1D6}$	$f_4(B,C,D) = B \oplus C \oplus D$

$$W_j = \begin{cases} x_i^{(j)} & 0 \le j \le 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3})_{\lll 1} & 16 \le j \le 79, \end{cases}$$

Cryptographic Hash functions

Cryptographic Hash Functions		
APR1	N-hash	SHA-0
AR	PANAMA	SHA-1
Boognish	RadioGatún	SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)
FFT-hash	RIPEMD	SHA-3
HAS-160	RIPEMD-128	Snefru
Haval	RIPEMD-160	StepRightUp
MD2	RIPEMD-256	Streebog
MD4	Scrypt	Tiger
MD5		VSH
MD6		Whirlpool

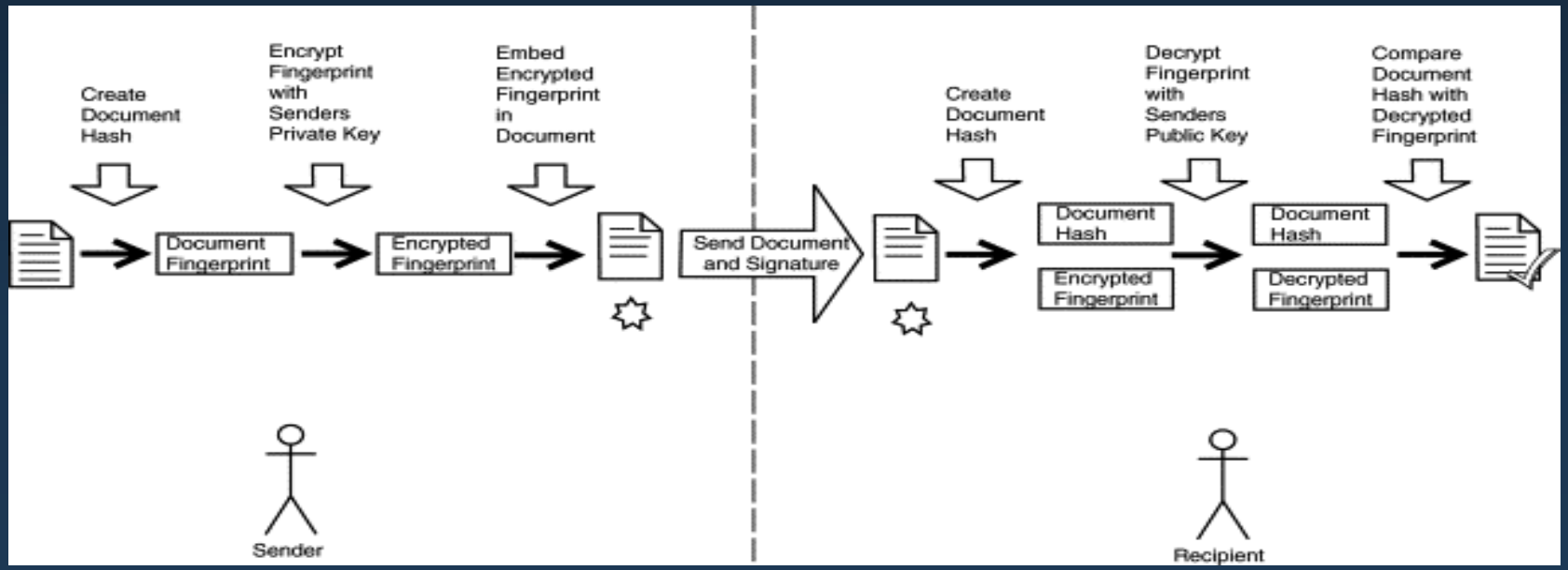
Digital Signature



Digital Signature

The digital signature process begins with creating a cryptographic hash of the message. This hash value is also known as a "digital fingerprint" and is a unique value. The digital fingerprint is then encrypted with the sender's private key, and the resulting value, Digital Signature, is appended to the message. This message is then sent to the recipient.

When the message arrives, the recipient decrypts the digital signqture appended to the message, creates a digital fingerprint of the message itself, and compares the two. If they match, the integrity of the message has been intact and the authentication of the sender is established



Digital Signature

Digital signature scheme. A digital signature scheme consists of the following three algorithms:

- **(sk, pk) := generateKeys(*keysize*)** The generateKeys method takes a key size and generates a key pair. The secret key *sk* is kept privately and used to sign messages. *pk* is the public verification key that you give to everybody. Anyone with this key can verify your signature.
- **sig := sign(*sk, message*)** The sign method takes a message and a secret key, *sk*, as input and outputs a signature for *message* under *sk*
- **isValid := verify(*pk, message, sig*)** The verify method takes a message, a signature, and a public key as input. It returns a boolean value, *isValid*, that will be ***true*** if *sig* is a valid signature for *message* under public key *pk*, and ***false*** otherwise.

We require that the following two properties hold:

- *Valid signatures must verify*

  verify(*pk*, *message*, **sign**(*sk*, *message*)) **== true**
- Signatures are ***existentially unforgeable***

Elliptic Curve Digital Signature Algorithm : ECDSA

- The ECC is a great technique to generate the keys, but is used alongside other techniques for digital signatures and key exchange.
- For example, Elliptic Curve Diffie-Hellman(ECDH) is quite popularly used for key exchange and ECDSA is used for digital signatures.
- The ECDSA is a type of DSA that uses ECC for key generation. Its purpose is digital signature, and not encryption. ECDSA can be a better alternative to RSA in terms of smaller key size, better security, and higher performance.
- It is one of the most important cryptographic components used in Bitcoin and Ethereum blockchains.
- The ECDSA includes broadly three steps: *key generation, signature generation, and signature verification.*