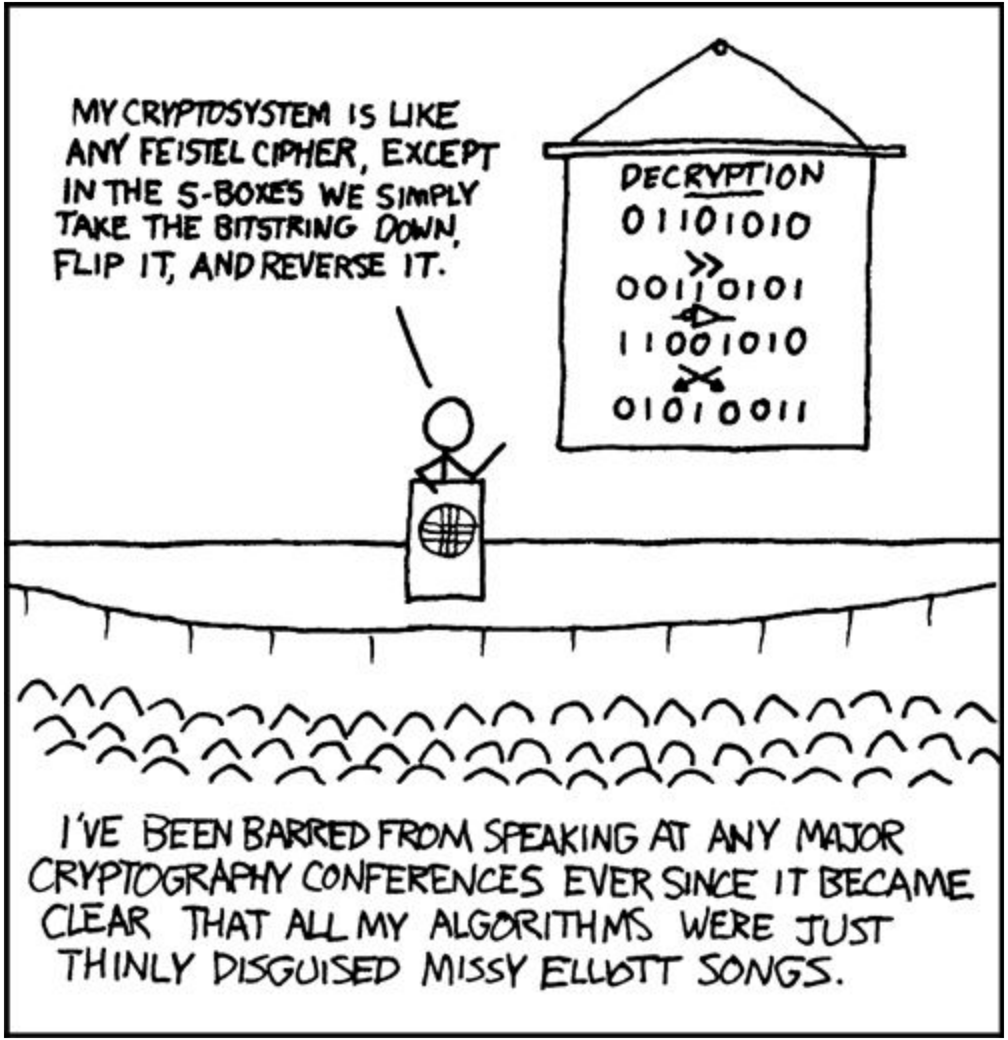
Cryptography

Introduction

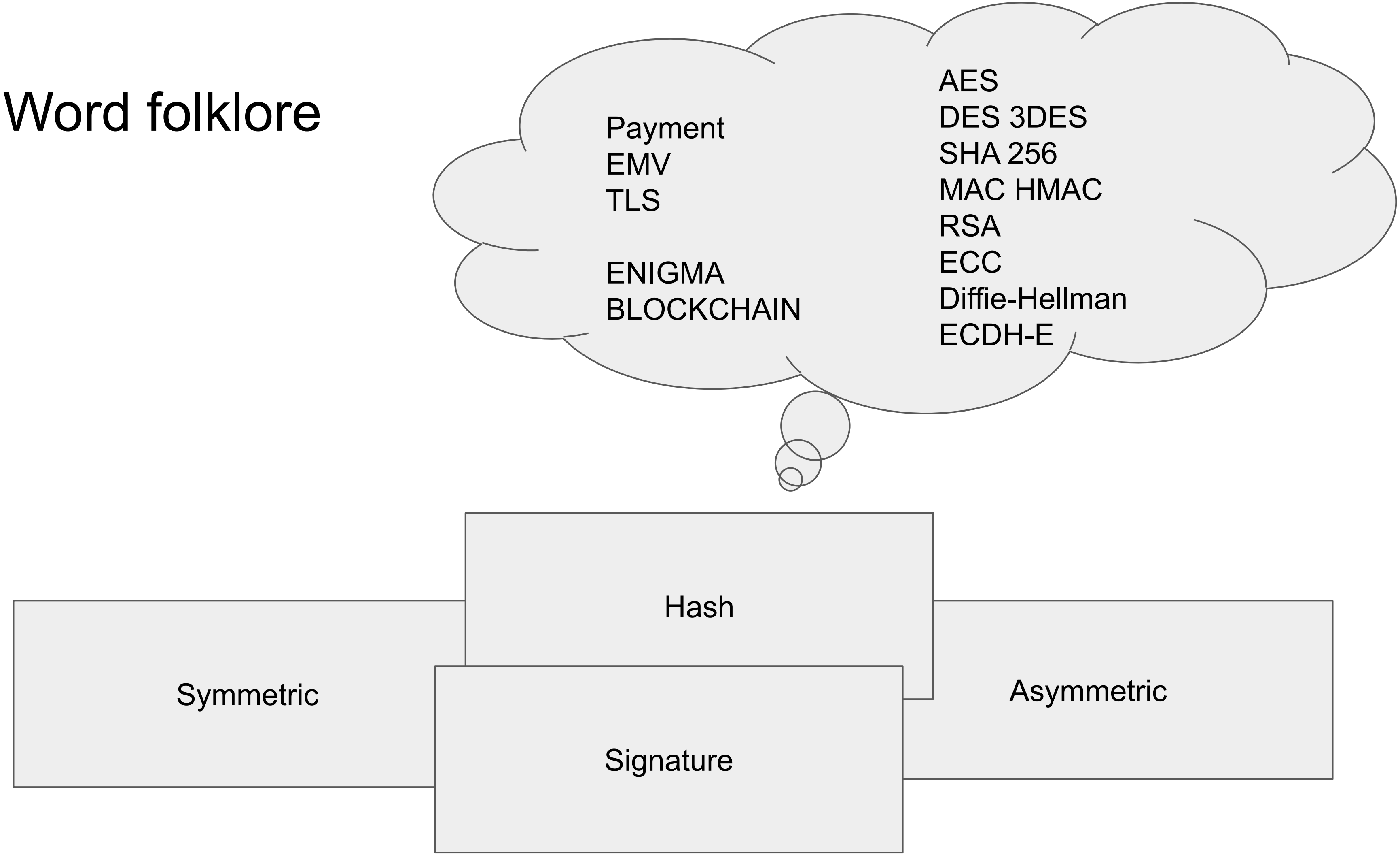

MY CRYPTOSYSTEM IS LIKE ANY FEISTEL CIPHER, EXCEPT IN THE S-BOXES WE SIMPLY TAKE THE BITSTRING DOWN, FLIP IT, AND REVERSE IT.
DECRYPTION
01101010
00110101
11001010
01010011
I'VE BEEN BARRED FROM SPEAKING AT ANY MAJOR CRYPTOGRAPHY CONFERENCES EVER SINCE IT BECAME CLEAR THAT ALL MY ALGORITHMS WERE JUST THINLY DISGUISED MISSY ELLIOTT SONGS.

Menu

- Symmetric cryptography
- Asymmetric cryptography
- Hashing
- Signature
- PKI

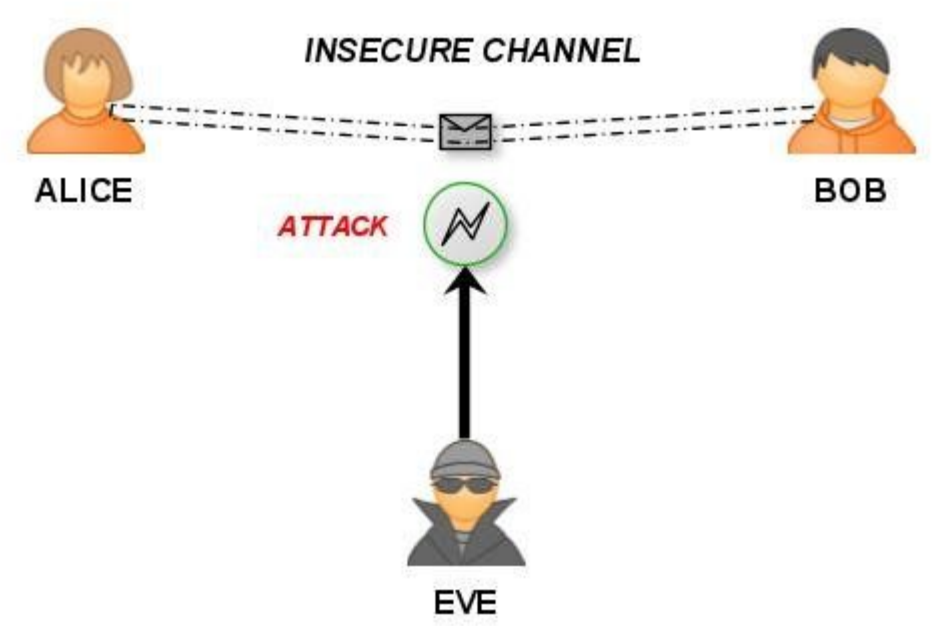
and along the lines talk about the goals of cryptography, key management, cryptographic protocols

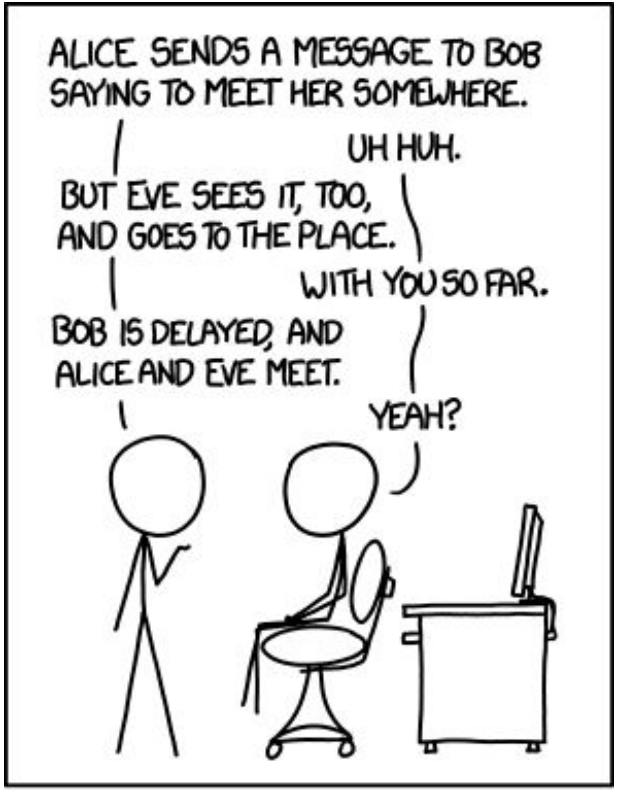
Word folklore



Old example

Who





I'VE DISCOVERED A WAY TO GET COMPUTER SCIENTISTS TO LISTEN TO ANY BORING STORY.

What

Data

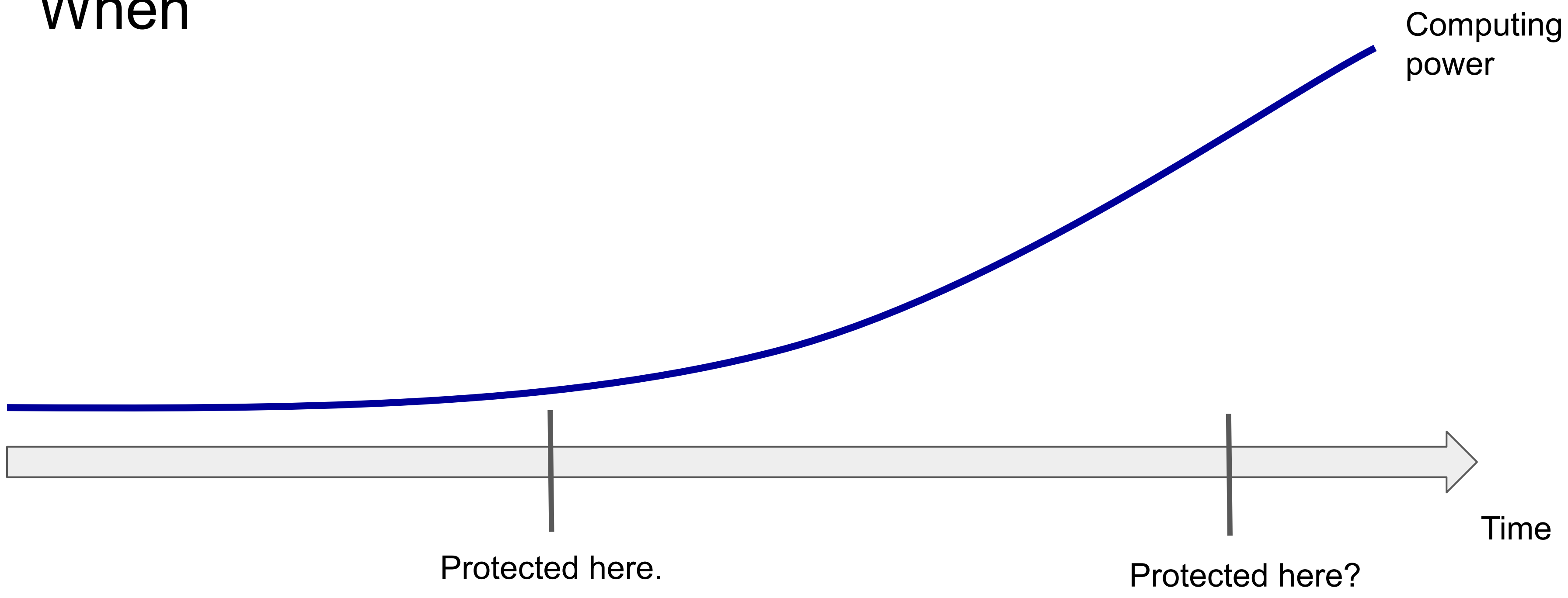
- At rest
- In transit

Security Properties

- Confidentiality
- Integrity
- Authenticity
- Non-Repudiation

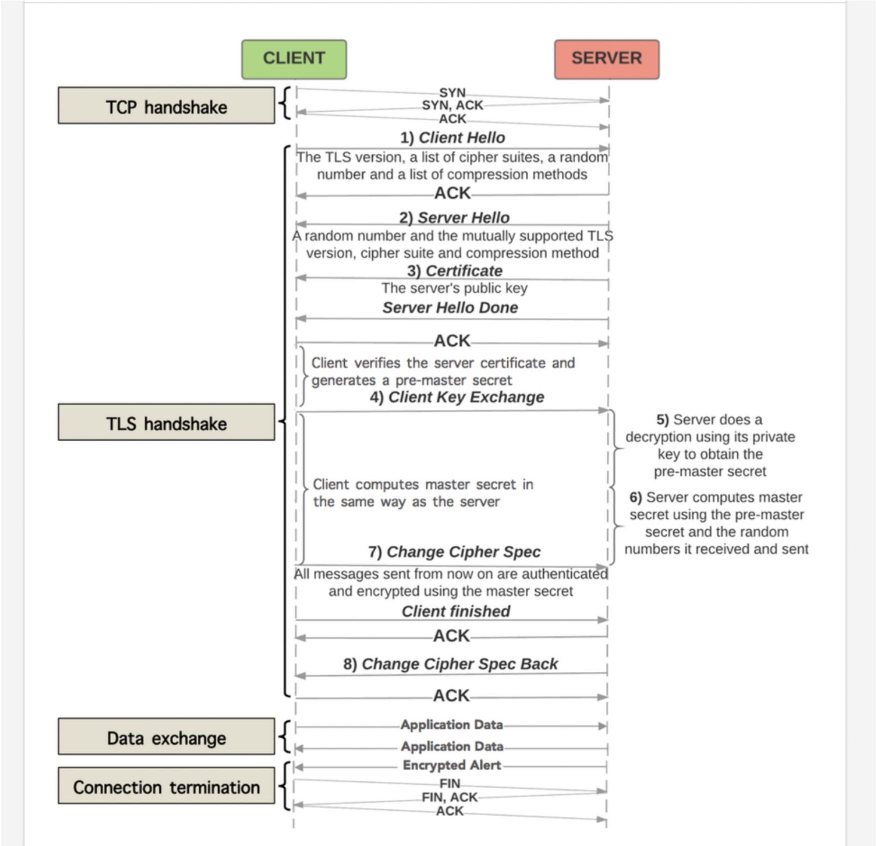
⇨ **Security guarantees**

When



Use case: secret pasta recipe

Real World Example: TLS & ciphersuites

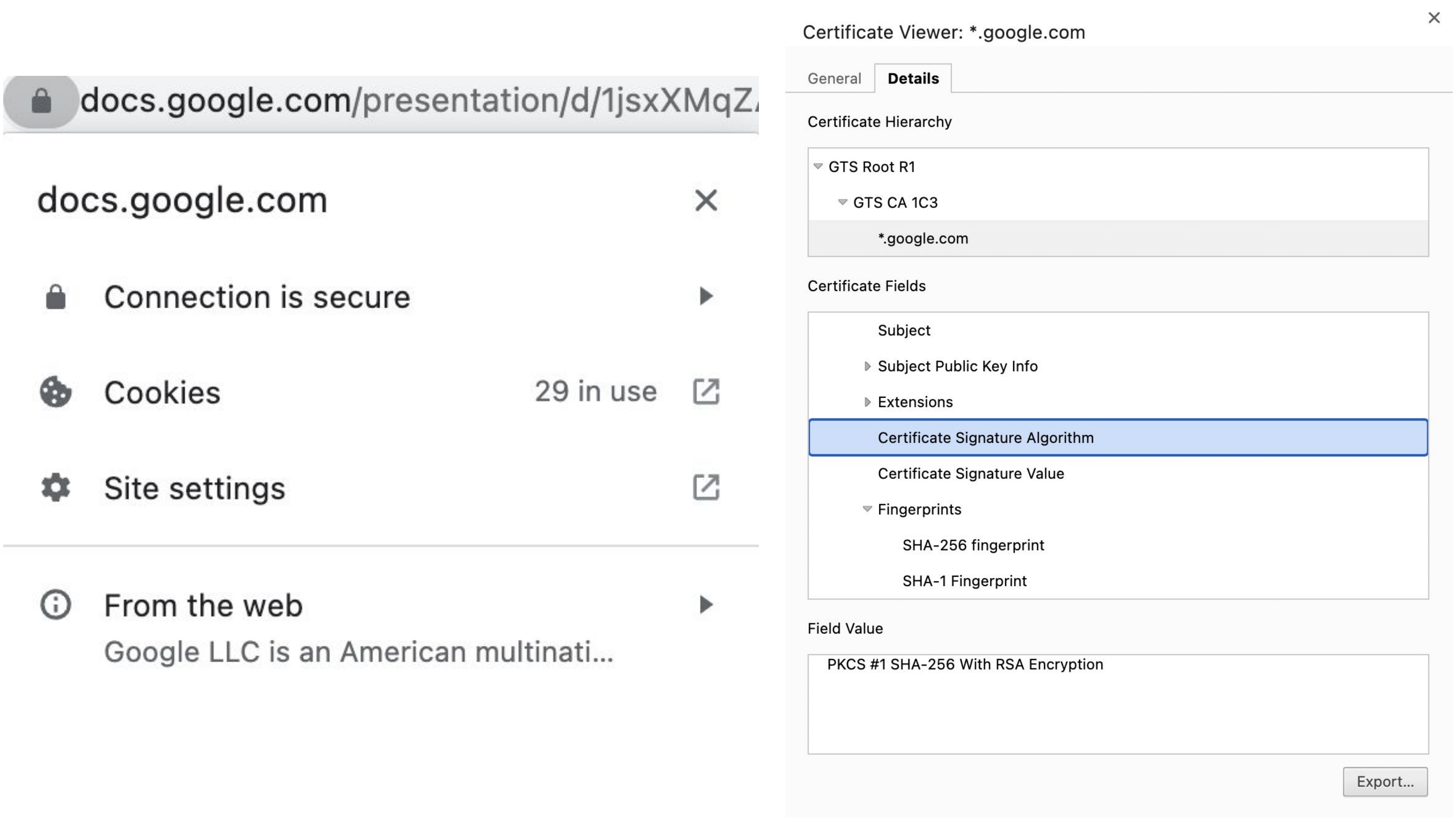


TLSv1.3:

- 0x13,0x01 TLS_AES_128_GCM_SHA256
- 0x13,0x02 TLS_AES_256_GCM_SHA384
- 0x13,0x03 TLS_CHACHA20_POLY1305_SHA256

TLSv1.2:

- 0xC0,0x2B ECDHE-ECDSA-AES128-GCM-SHA256
- 0xC0,0x2F ECDHE-RSA-AES128-GCM-SHA256
- 0xC0,0x2C ECDHE-ECDSA-AES256-GCM-SHA384
- 0xC0,0x30 ECDHE-RSA-AES256-GCM-SHA384
- 0xCC,0xA9 ECDHE-ECDSA-CHACHA20-POLY1305
- 0xCC,0xA8 ECDHE-RSA-CHACHA20-POLY1305
- 0x00,0x9E DHE-RSA-AES128-GCM-SHA256


docs.google.com/presentation/d/1jsxXMqZ
docs.google.com
Connection is secure
Cookies
29 in use
Site settings
From the web
Google LLC is an American multinati...
Certificate Viewer: *.google.com
General
Details
Certificate Hierarchy
GTS Root R1
GTS CA 1C3
*.google.com
Certificate Fields
Subject
Subject Public Key Info
Extensions
Certificate Signature Algorithm
Certificate Signature Value
Fingerprints
SHA-256 fingerprint
SHA-1 Fingerprint
Field Value
PKCS #1 SHA-256 With RSA Encryption
Export...

Cryptographic primitives

- One-way hash function, sometimes also called as one-way compression function—compute a reduced hash value for a message (e.g., SHA-256)
- Symmetric key cryptography—compute a ciphertext decodable with the same key used to encode (e.g., AES)
- Public-key cryptography—compute a ciphertext decodable with a different key used to encode (e.g., RSA)
- Digital signatures—confirm the author of a message
- Mix network—pool communications from many users to anonymize what came from whom
- Private information retrieval—get database information without server knowing which item was requested
- Commitment scheme—allows one to commit to a chosen value while keeping it hidden to others, with the ability to reveal it later
- Cryptographically secure pseudorandom number generator

Cryptography principles

1. Kerckhoffs's principle: The secrecy of your message should always depend on the secrecy of the key, and not on the secrecy of the encryption system.

2. Don’t roll your own crypto

What are we up against?

TABLE 2. Minimum Key Lengths for Symmetric Ciphers.

Type of Attacker	Budget	Tool	Time and Cost Per Key Recovered: 40 bits	Time and Cost Per Key Recovered: 56 bits	Key Length Needed For Protection In Late-1995
Pedestrian Hacker	Tiny	Scavenged computer time	1 week	Infeasible	45
	$400	FPGA	5 hours ($0.08)	38 years ($5,000)	50
Small Business	$10,000	FPGA	12 minutes ($0.08)	18 months ($5,000)	55
Corporate Department	$300K	FPGA	24 seconds ($0.08)	19 days ($5,000)	60
		ASIC	0.18 seconds ($0.001)	3 hours ($38)	
Big Company	$10M	FPGA	7 seconds ($0.08)	13 hours ($5,000)	70
		ASIC	0.005 seconds ($0.001)	6 minutes ($38)	
Intelligence Agency	$300M	ASIC	0.0002 seconds ($0.001)	12 seconds ($38)	75

Evolution of cryptographic algorithms

Schemes	Strength	Direction
RC4, MD2, MD5, SHA-1	Broken	Shall be forbidden
< 2048-bit RSA, FFDH	< 112	Shall be forbidden
< 255-bit ECC	< 128	Shall be forbidden
Block ciphers with < 128-bit block length (3DES, Blowfish, etc.)	< 112	Shall be forbidden
SSLv3, TLS 1.0, TLS 1.1, DTLS 1.0	Broken	Shall be forbidden
Key exchange without PFS (psk_ke, etc.)	Weak	Shall be phased out
IND-CPA encryption (encryption without integrity)	Broken	Shall be phased out
CBC-mode, RSAES-PKCS1-v1_5	Broken	Shall be phased out
HMAC-MD5	Nearly Broken	Shall be phased out
DSA, Binary ECC curves	Deprecated	Shall be phased out
SHA-224, SHA3-224	112	Shall be phased out
HMAC-SHA1 for MAC and PRF	Legacy	Shall be phased out
RSASSA-PKCS-v1_5	Legacy	Avoid in new systems
2048-bit RSA, FFDH	Legacy	Avoid in new systems
TLS 1.2	Obsolete	Avoid in new systems

Figure 3: Broken and legacy cryptographic algorithms and security protocols

Evolution of cryptographic algorithms

Algorithm	Generate key pair	Sign	Verify	Encapsulate	Decapsulate	Public key size	Signature/ciphertext size
LMS/XMSS/HSS/MXSS^MT (hash-based PQC)	43 - 165000 ms	0.51 - 6.0 ms	0.096 - 5.3 ms			50 - 100 B	700 - 8000 B
Picnic picnic3-L1 (symmetric primitives)	0.001 ms	7.272 ms	5.508 ms			35 B	12492 B
Rainbow 1a (multivariate PQC)	401.505 ms	0.020 ms	0.014 ms			152097 B	64 B
Dilithium-1024x768-AES (lattice-based PQC)	0.018 ms	0.098 ms	0.025 ms			1184 B	2044 B
EdDSA Ed25519	0.018 ms	0.019 ms	0.066 ms			64 B	64 B
ECDSA P-256	0.041 ms	0.053 ms	0.113 ms			64 B	64 B
RSA-3072	232.704 ms	3.542 ms	0.033 ms	0.038 ms	3.530 ms	384 B	384 B
NIST P-256	0.094 ms			0.231 ms	0.231 ms	32 B	32 B
Curve25519	0.050 ms			0.054 ms	0.054 ms	32 B	32 B
Kyber-512-90s (lattice-based PQC)	0.006 ms			0.010 ms	0.008 ms	800 B	736 B
Classic McEliece 348864 (code-based PQC)	18.494 ms			0.015 ms	0.048 ms	261120 B	128 B
SIKE p503 (isogeny-based PQC)	4.292 ms			7.041 ms	7.511 ms	378 B	402 B

Figure 4: Some candidates (post-quantum security level 1) in the third and final round of NIST PQC Standardization. The performance measurements are single-core on Skylake 2.5 GHz https://bench.cr.yp.to/ebats.html (lower is better)

Minimal key sizes for PCI-PTS

	Algorithm				
	TDES	IFC (RSA)	ECC (ECDSA, EdDSA, ECDH, ECMQV)	FFC (DSA, DH, MQV)	AES
Minimum key size in number of bits:	112	2048	224	2048/224	128

Equivalent key sizes

	Algorithm				
	TDES	IFC (RSA)	ECC (ECDSA, EdDSA, ECDH, ECMQV)	FFC (DSA, DH, MQV)	AES
Key size in number of bits:	112	1024	160	1024/160	–
	168	2048	224	2048/224	–
	–	3072	256	3072/256	128
	–	7680	384	7680/384	192
	–	15360	512	15360/512	256

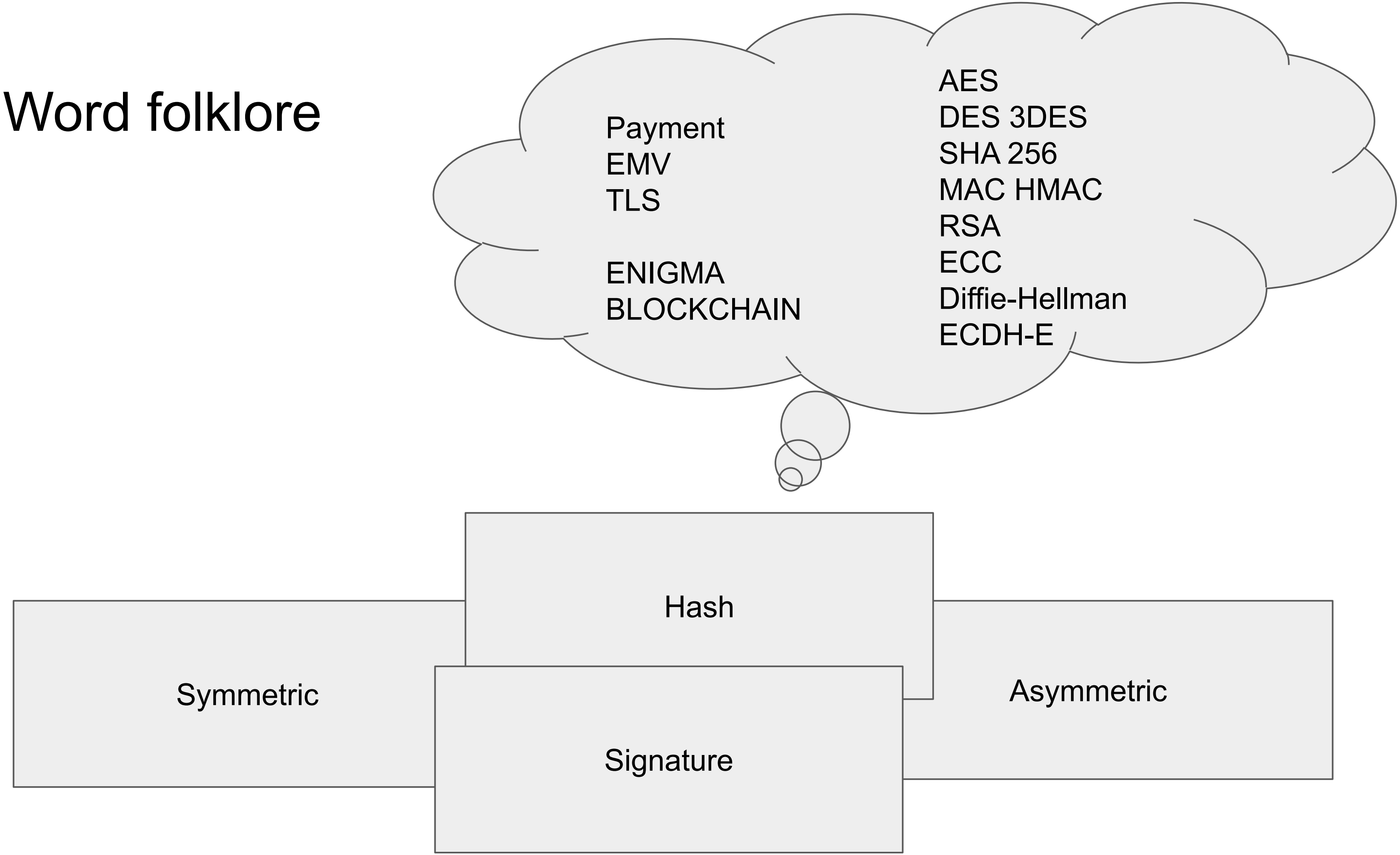
Strength in usage vs Computing power

Also Speed of execution

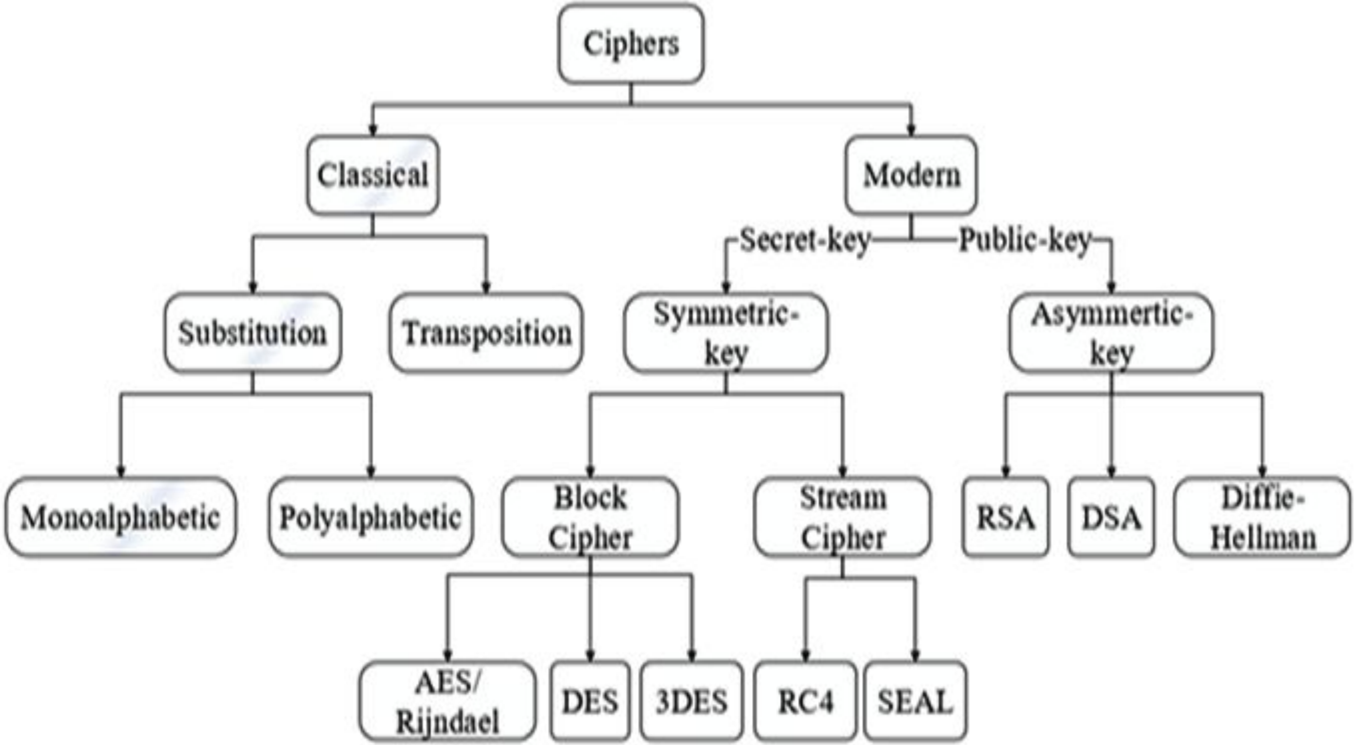
https://sumupteam.atlassian.net/wiki/spaces/HAR/pages/1297580216/Crypto+HW+accelerator+support
https://sumupteam.atlassian.net/wiki/spaces/HAR/pages/21929854104/Solo+UL+Crypto+HW+accelerator+support

Operation	1024			2048			3072			4096		
	SW	HW	Speed up	SW	HW	Speed up	SW	HW	Speed up	SW	HW	Speed up
RSA_ES_PKCS1_V1_5 (E)	31 ms	1 ms	**x31**	33 ms	4 ms	**x8**	30 ms	7 ms	**x4**	28 ms	11 ms	**x2.5**
RSA_ES_PKCS1_V1_5 (D)	1 s 58 ms	13 ms	**x81**	1 s 977 ms	92 ms	**x21**	2 s 919 ms	304 ms	**x9**	3 s 848 ms	704 ms	**x5.5**
RSAES_PKCS1_OAEP_MGF1_SHA1 (E)	31 ms	1 ms	**x31**	33 ms	4 ms	**x8**	30 ms	7 ms	**x4**	28 ms	11 ms	**x2.5**
RSAES_PKCS1_OAEP_MGF1_SHA1 (D)	1 s 196	13 ms	**x92**	2 s 21 ms	92 ms	**x22**	2 s 877 ms	304 ms	**x9**	3 s 894 ms	704 ms	**x5.5**
RSA_NOPAD (E)	31 ms	1ms	**x31**	33 ms	4 ms	**x8**	30 ms	7 ms	**x4**	27 ms	11 ms	**x2.5**
RSA_NOPAD (D)	1 s 59 ms	13 ms	**x81**	1 s 976 ms	92 ms	**x21**	2 s 906 ms		**x9**	3 s 819 ms	704 ms	**x5.5**
RSASSA_PKCS1_V1_5_SHA1 (S)	1 s 64 ms	13 ms	**x82**	2 s 29 ms	92 ms	**x22**	2 s 884 ms	304 ms	**x9**	3 s 852 ms	704 ms	**x5.5**
RSASSA_PKCS1_V1_5_SHA1 (V)	31 ms	1 ms	**x31**	33 ms	4 ms	**x8**	30 ms	7 ms	**x4**	27 ms	11 ms	**x2.5**
RSASSA_PKCS1_PSS_SHA1 (S)	1 s 62 ms	13 ms	**x82**	2 s 190 ms	92 ms	**x23**	2 s 956 ms	304 ms	**x9**	3 s 909 ms	704 ms	**x5.5**
RSASSA_PKCS1_PSS_SHA1 (V)	31 ms	1 ms	**x31**	33 ms	4 ms	**x8**	30 ms	7 ms	**x4**	27 ms	11 ms	**x2.5**

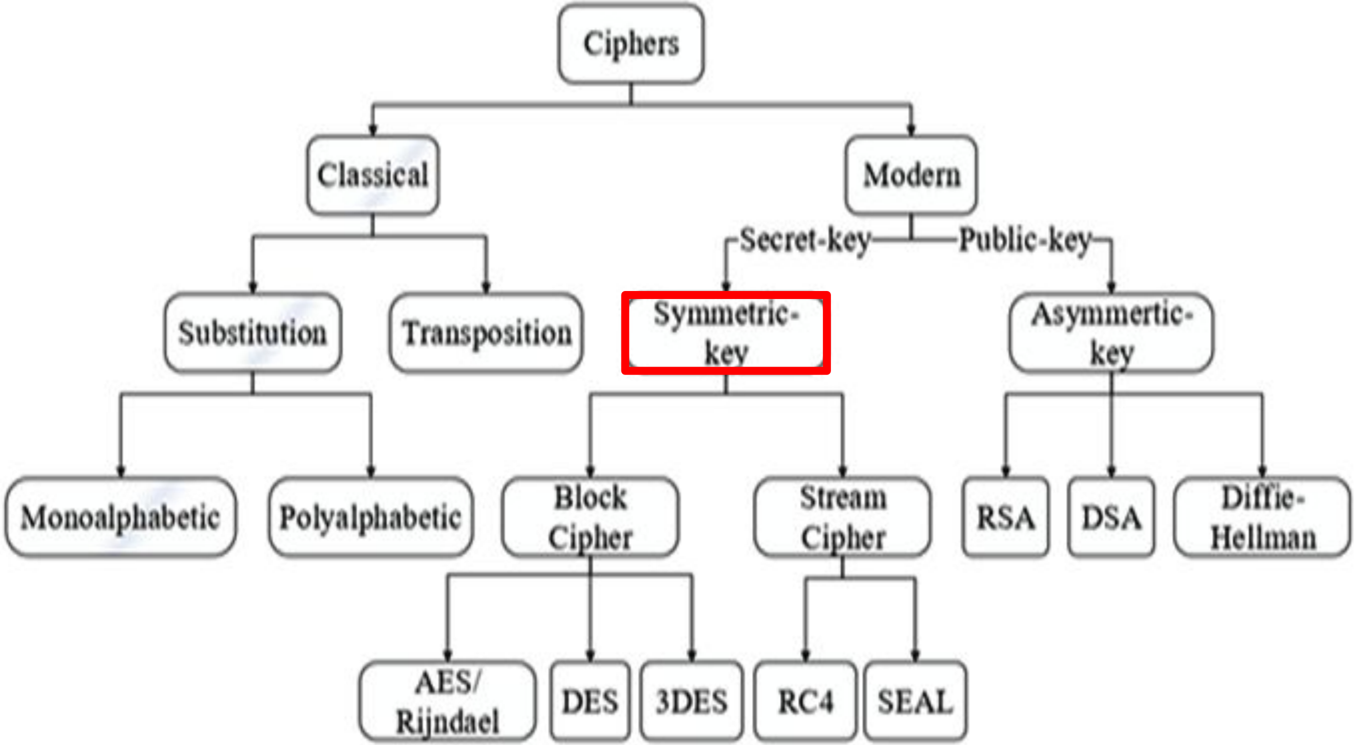
Word folklore



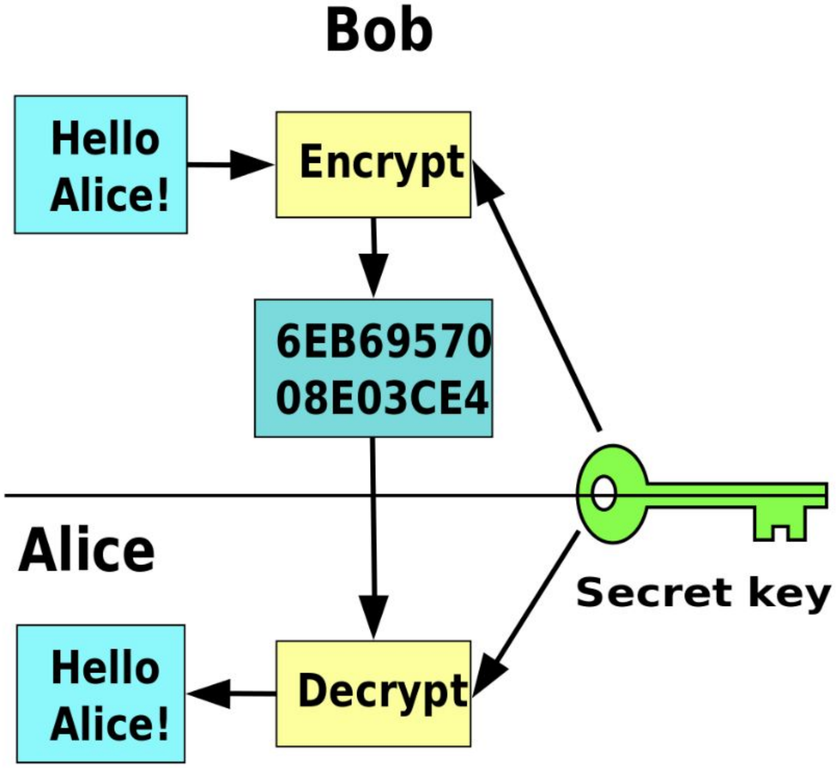
Cryptography ~~algorithms~~ jungle overview



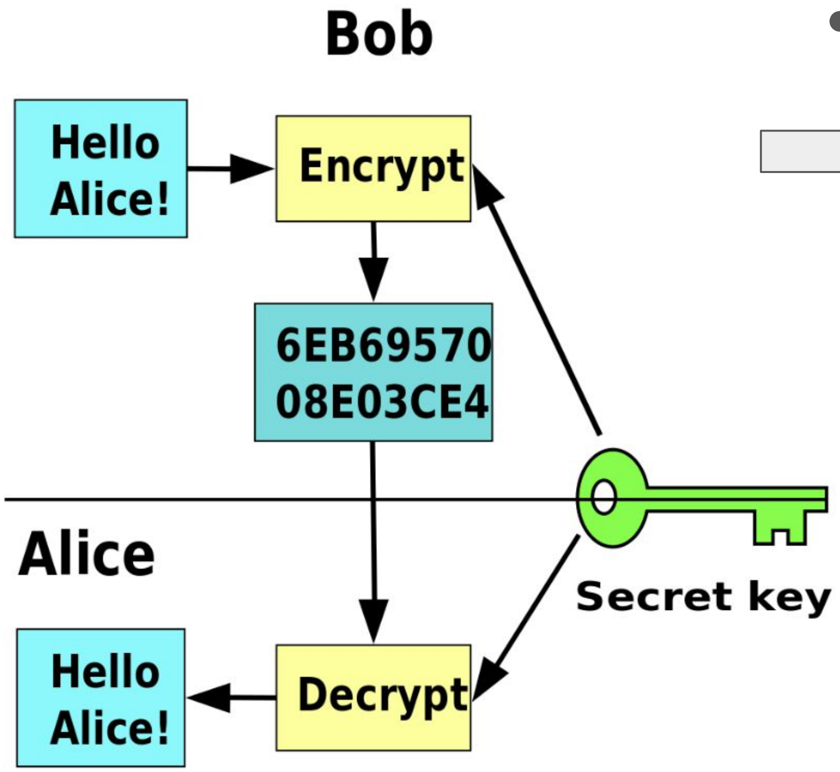
Cryptography algorithms overview



Symmetric Cryptography



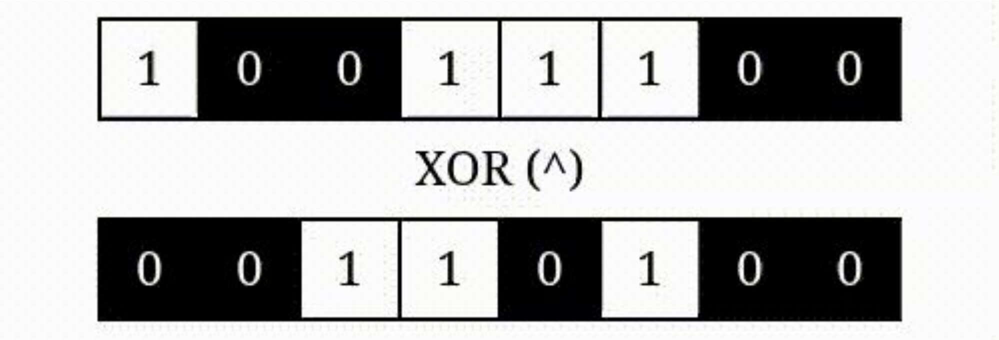
Symmetric Cryptography



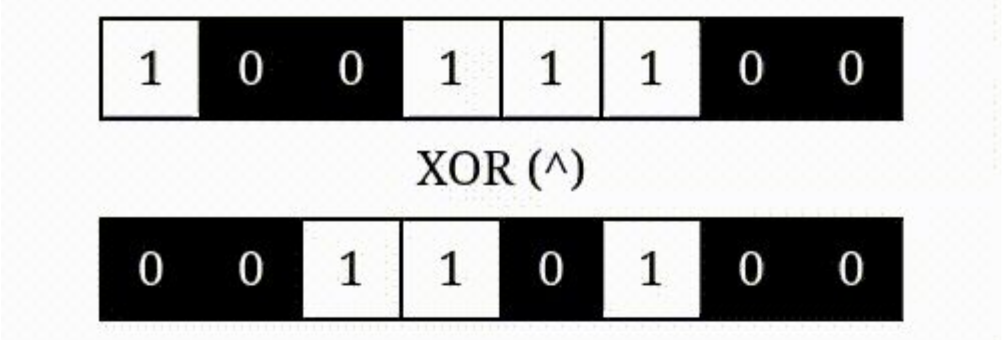
- Confidentiality

Non-linearity but Symmetry

Introducing XOR!



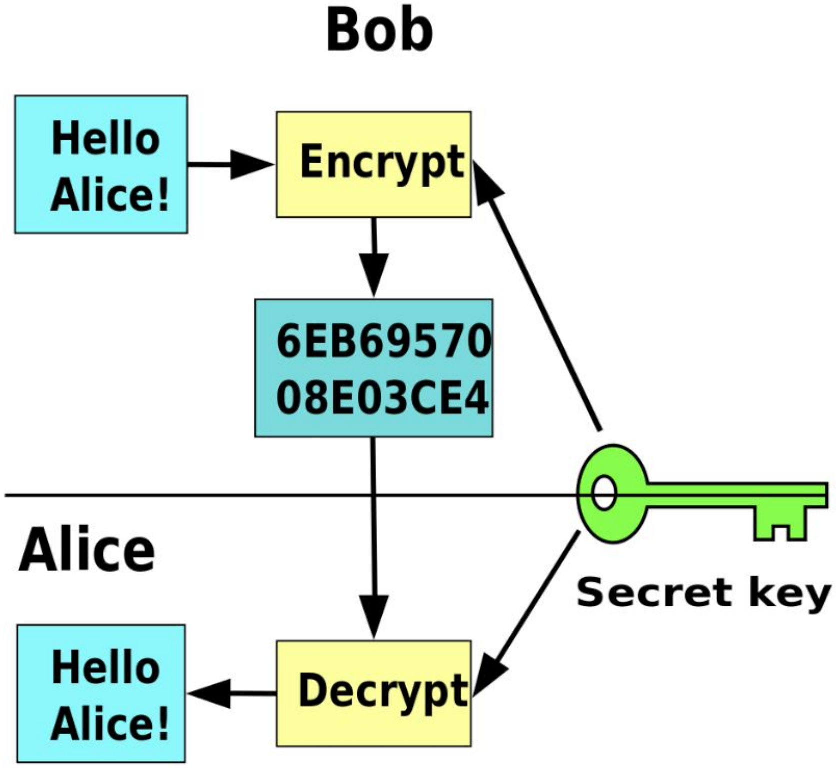
Introducing XOR!




```
# xor in
M = 00000111110111111100
K = 11010101010101010101
C = 11010010100010101001
# xor out
C = 11010010100010101001
K = 11010101010101010101
D = 00000111110111111100
```

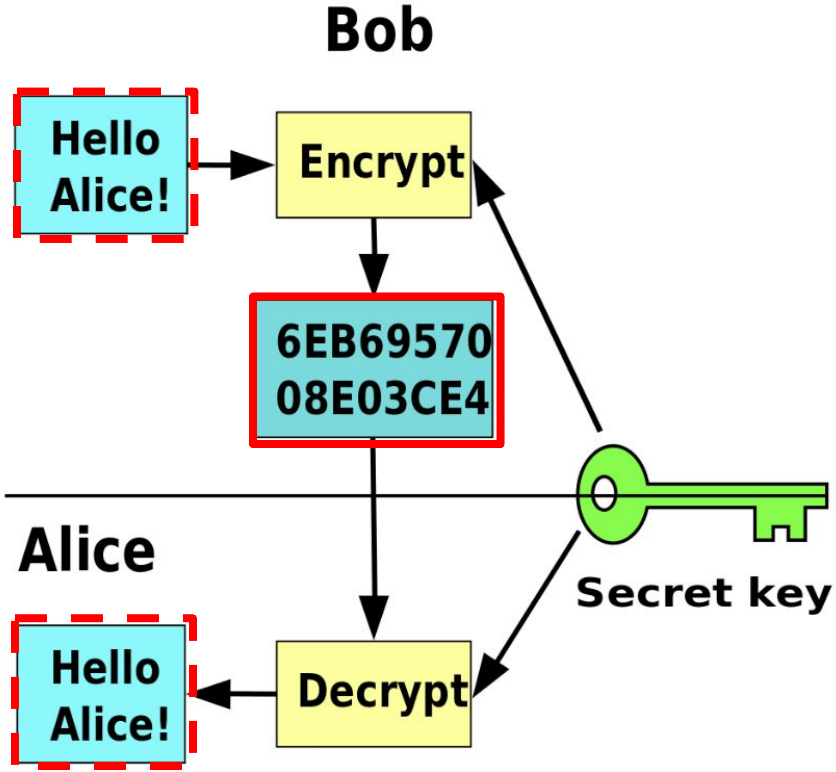
What moves & what stays still?

```
# xor in
M = 00000111110111111100
K = 11010101010101010101
C = 11010010100010101001
# xor out
C = 11010010100010101001
K = 11010101010101010101
D = 00000111110111111100
```



What can an attacker control?

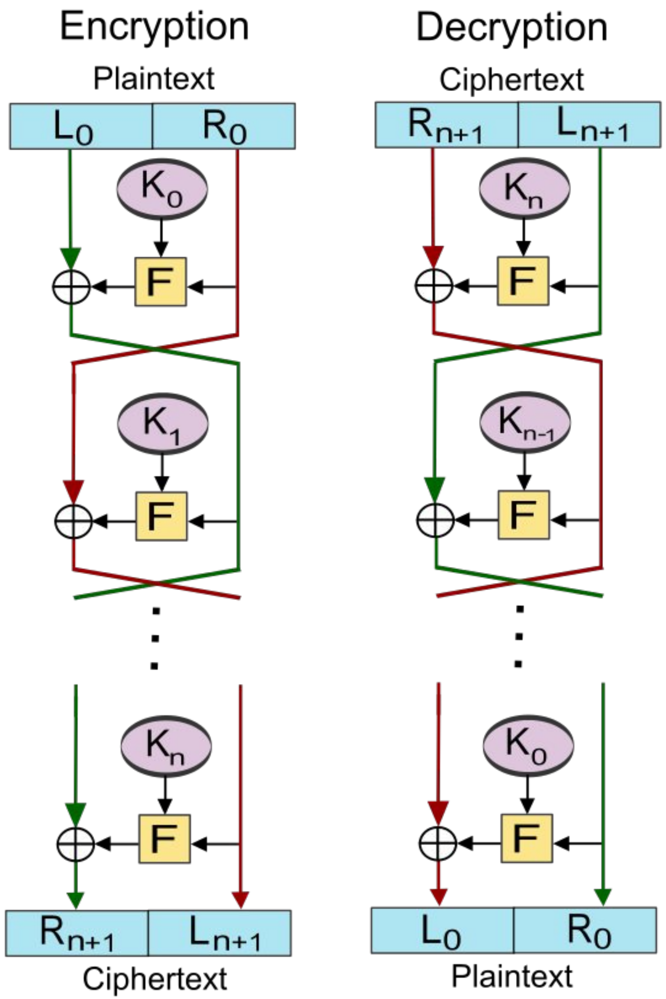
```
# xor in
M = 00000111110111111100
K = 11010101010101010101
C = 11010010100010101001
# xor out
C = 11010010100010101001
K = 11010101010101010101
D = 00000111110111111100
```



Where is my Non-Linearity?

- Confusion and Diffusion

⇨ Feistel Cipher

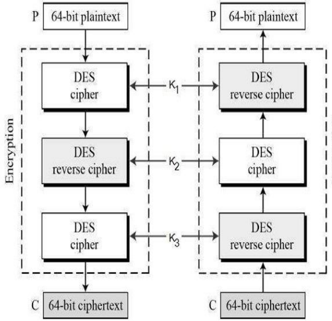


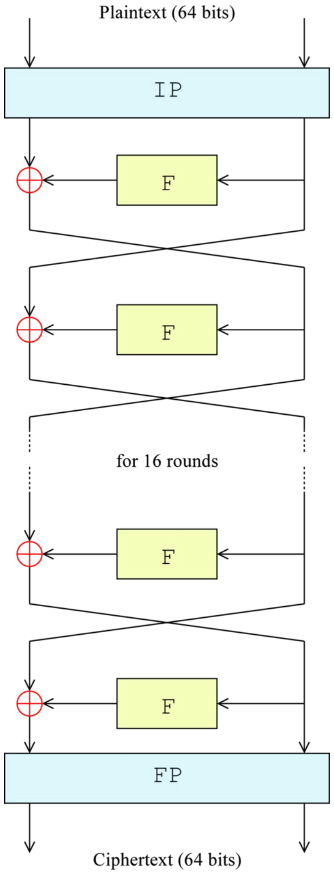
3DES

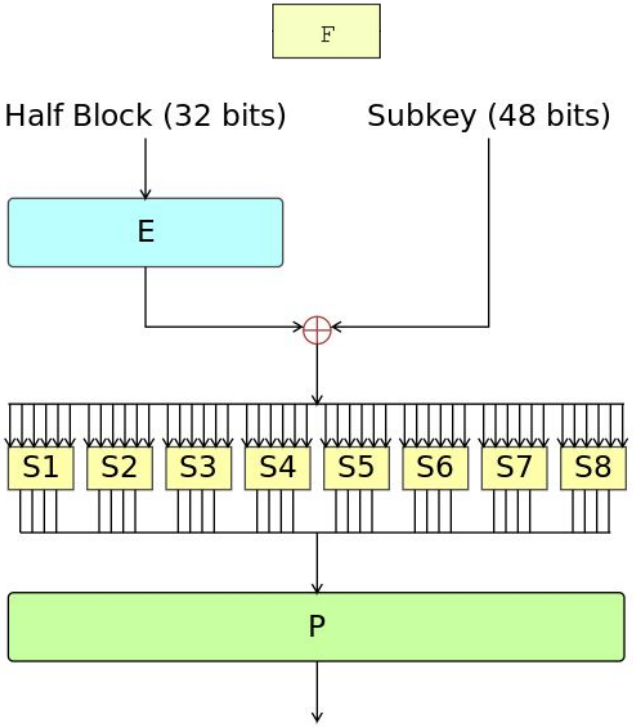
112 bits key

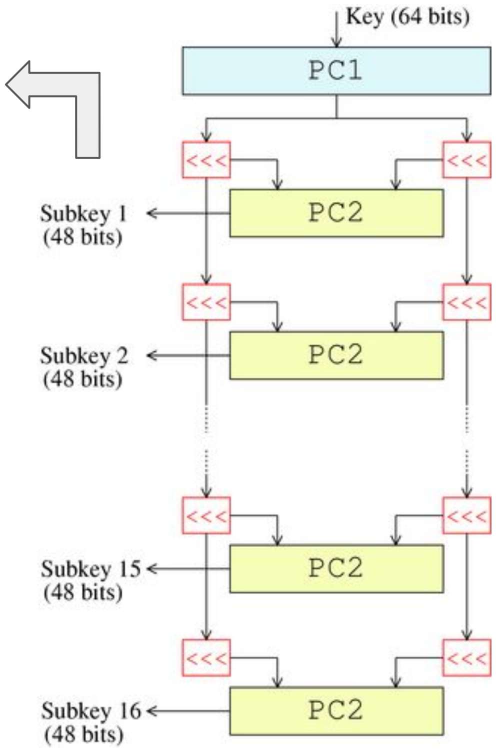
64 bits blocs

16 rounds

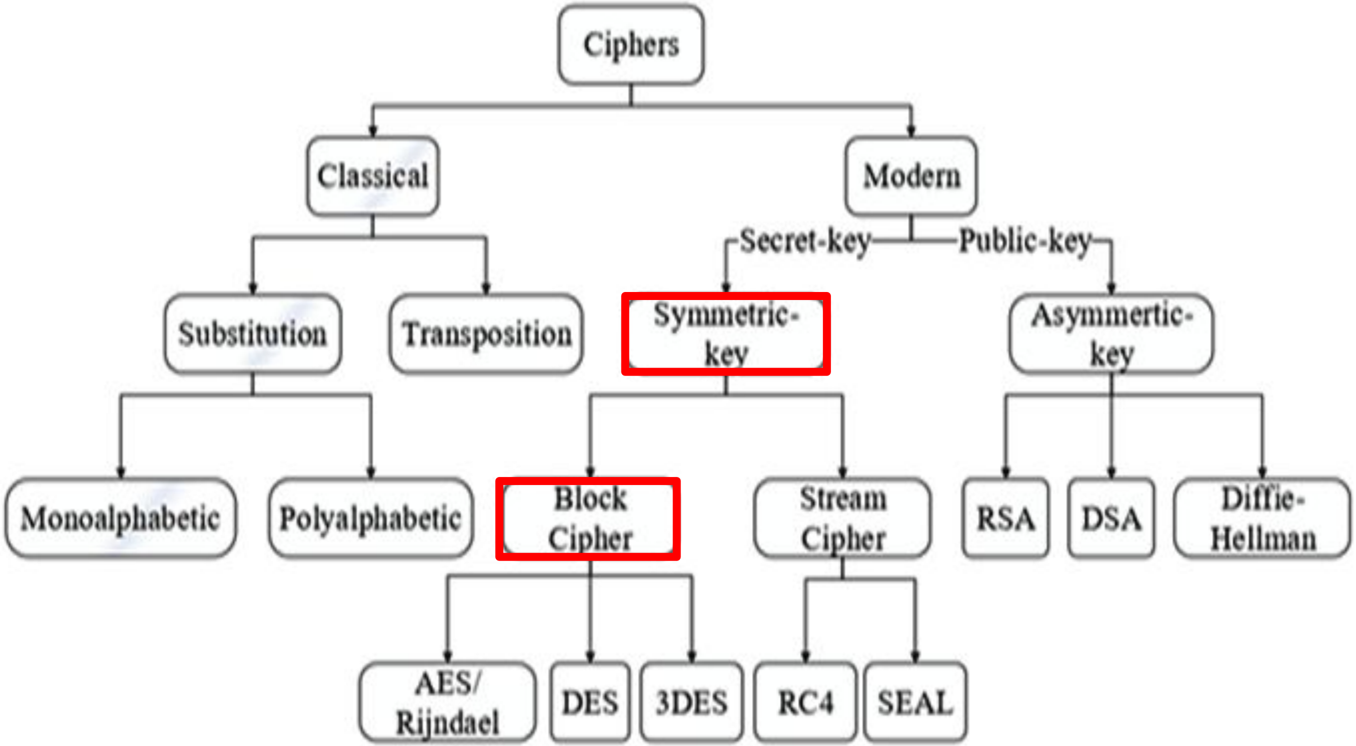




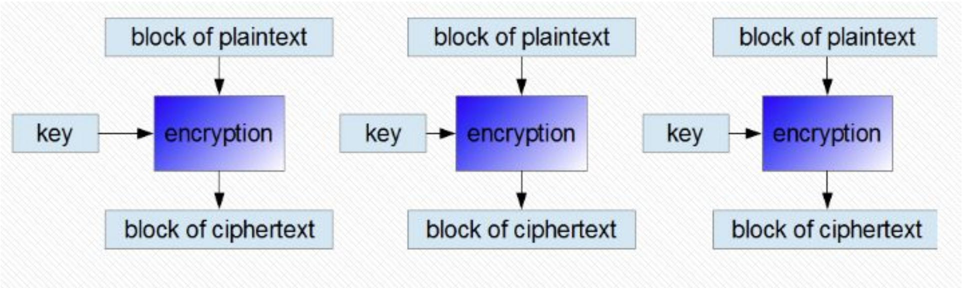




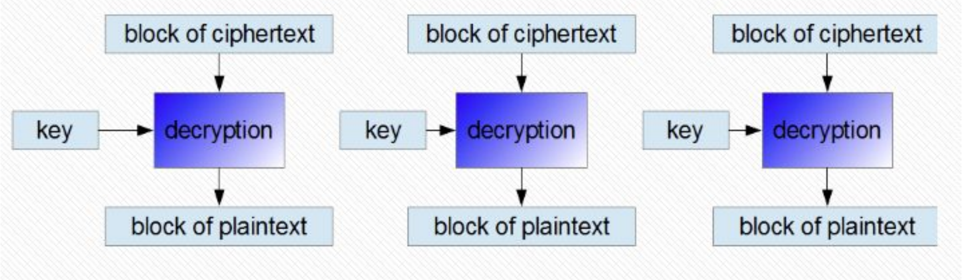
Cryptography algorithms overview



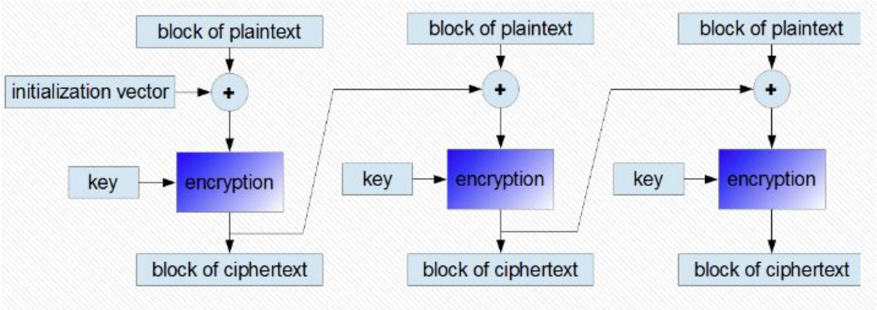
Slice and Dice – to account for message size



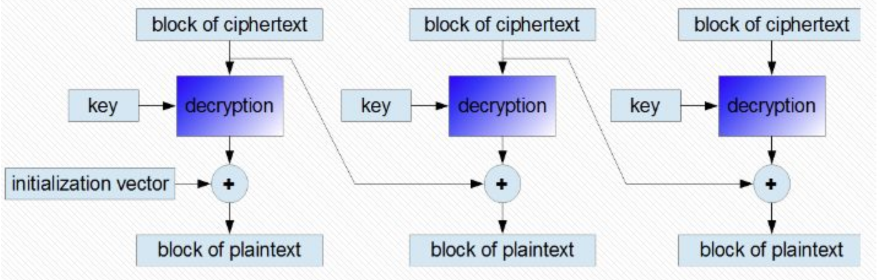
Encryption in the ECB mode



Decryption in the ECB mode



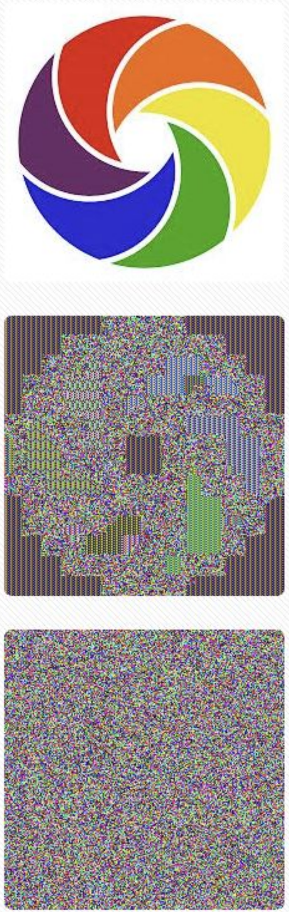
Encryption in the CBC mode



Decryption in the CBC mode

Encryption Modes

Even a strong encryption algorithm used in ECB mode cannot blur efficiently the plaintext.



The bitmap image encrypted using DES and the same secret key. The ECB mode was used for the middle image and the more complicated CBC mode was used for the bottom image.

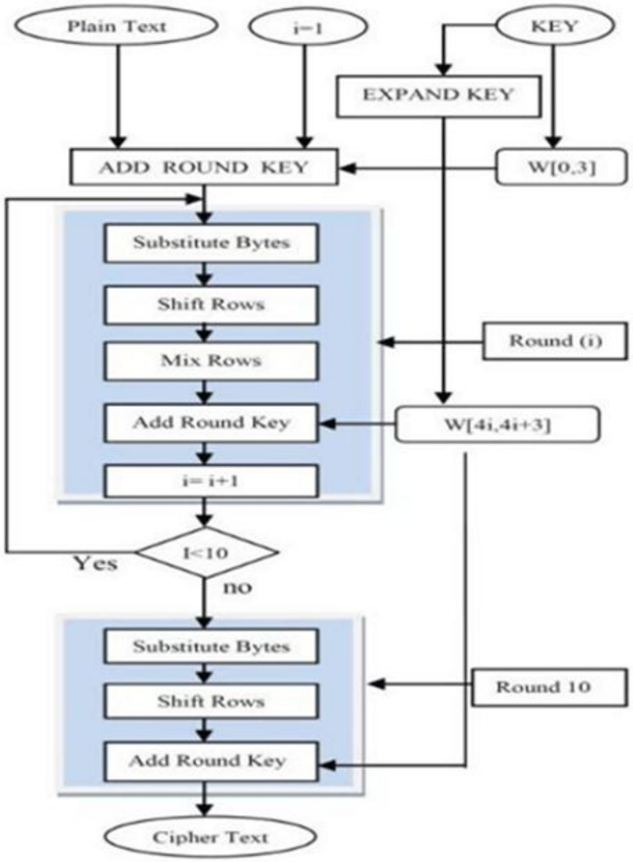
AES

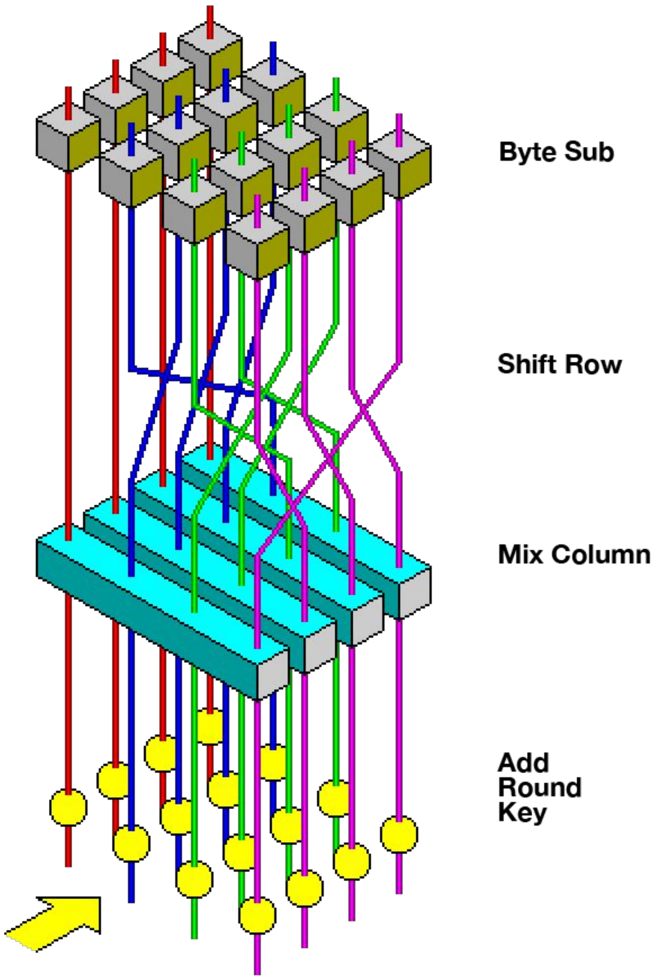
10 rounds for 128-bit keys,

12 rounds for 192-bit keys,

14 rounds for 256-bit keys.

128 bits block size

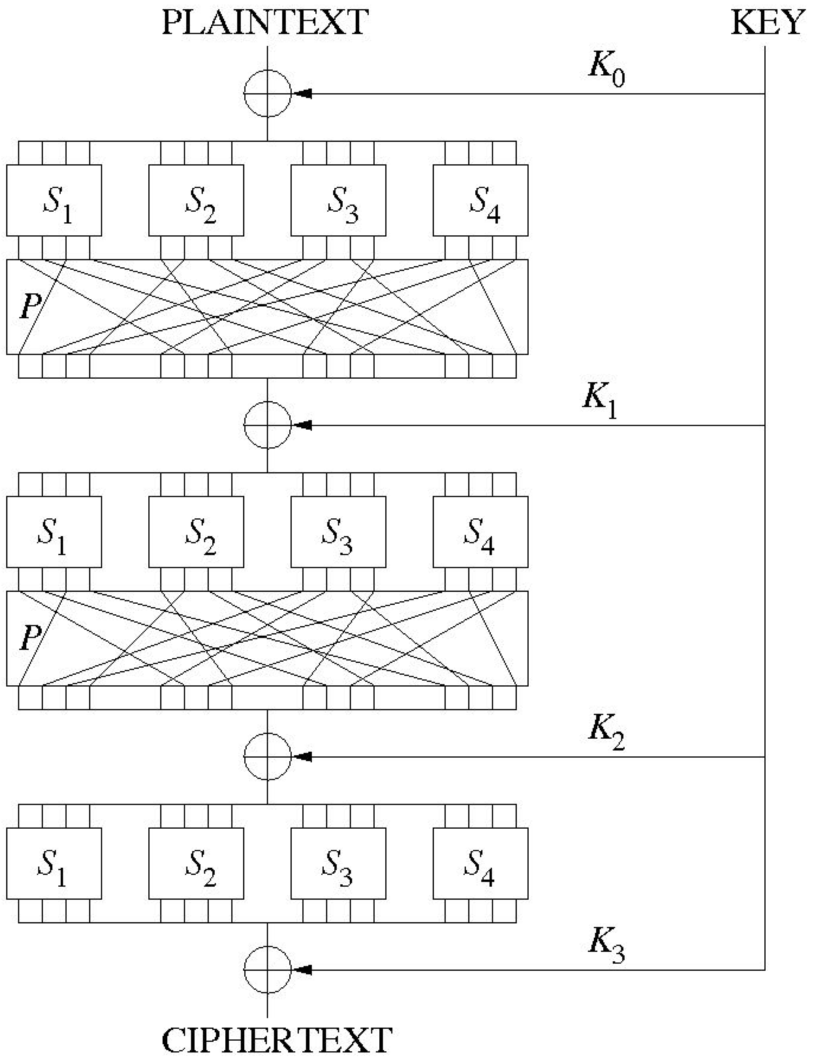




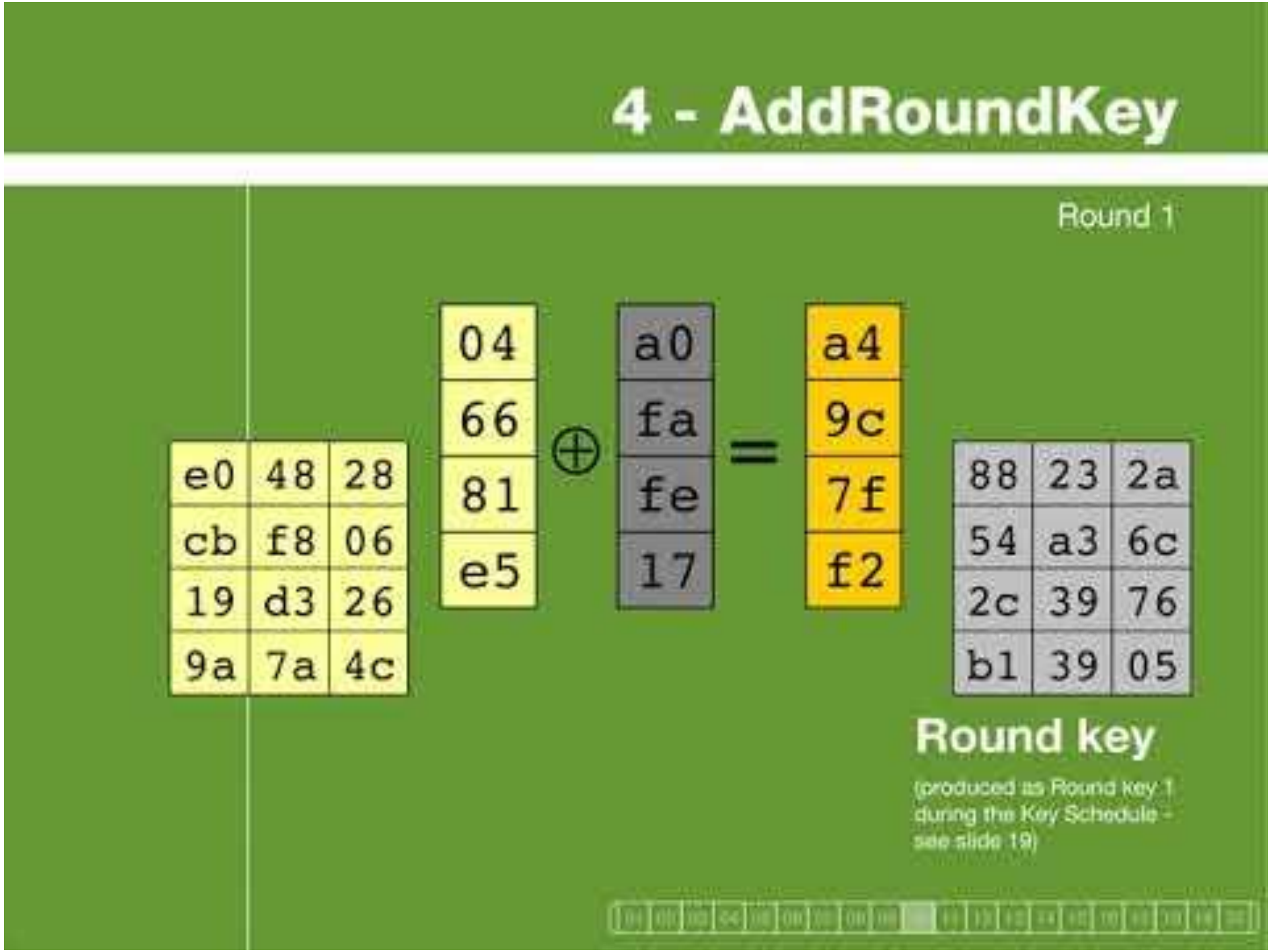
Where is my Non-Linearity?

- Confusion and Diffusion

⇒ Substitution & Permutation Network



AES


4 - AddRoundKey
Round 1
e0 48 28
cb f8 06
19 d3 26
9a 7a 4c
04
66
81
e5
⊕
a0
fa
fe
17
=
a4
9c
7f
f2
88 23 2a
54 a3 6c
2c 39 76
b1 39 05
Round key
(produced as Round key 1
during the Key Schedule -
see slide 19)

Principles behind symmetric cryptography

- Non-Linearity / Confusion and Diffusion

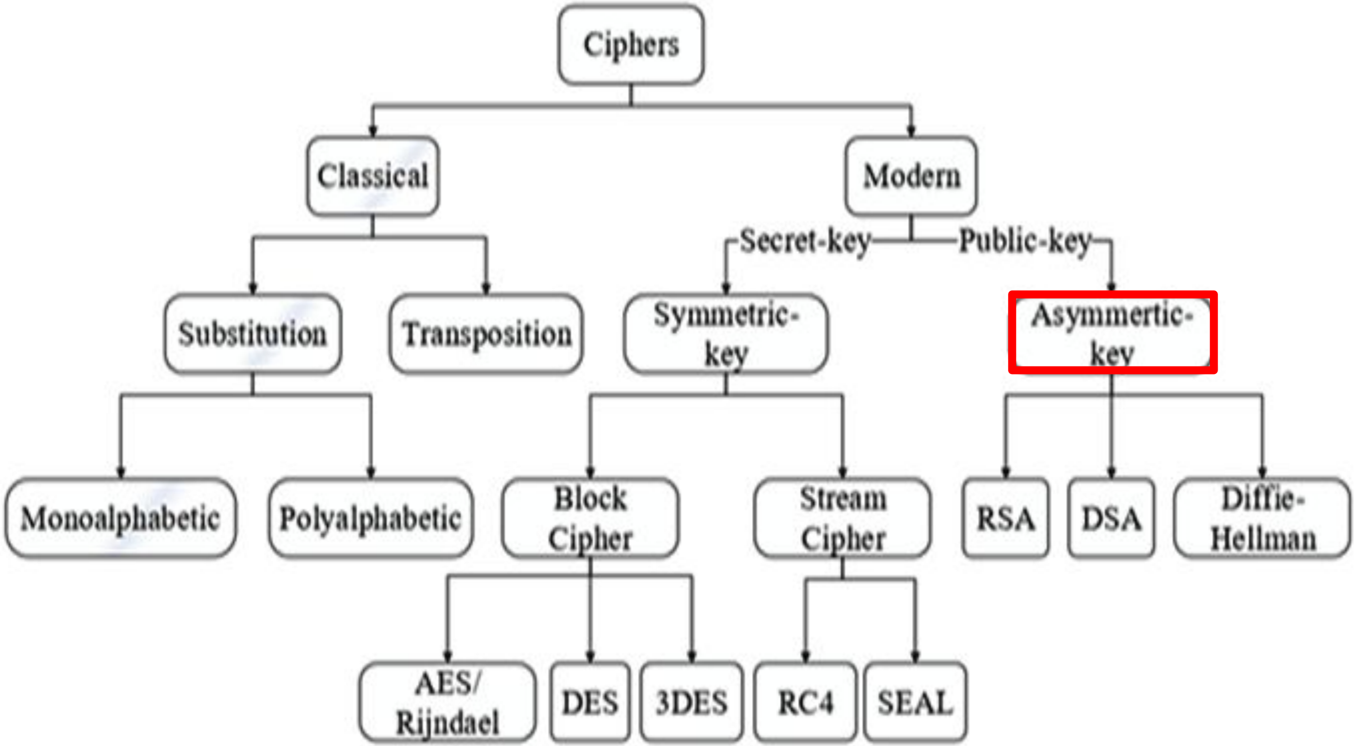
- Feistel cipher — DES
- Substitution-Permutation network — AES

- Guarantees the Confidentiality security property

I can use a secret to hide information

What next?

Cryptography algorithms overview

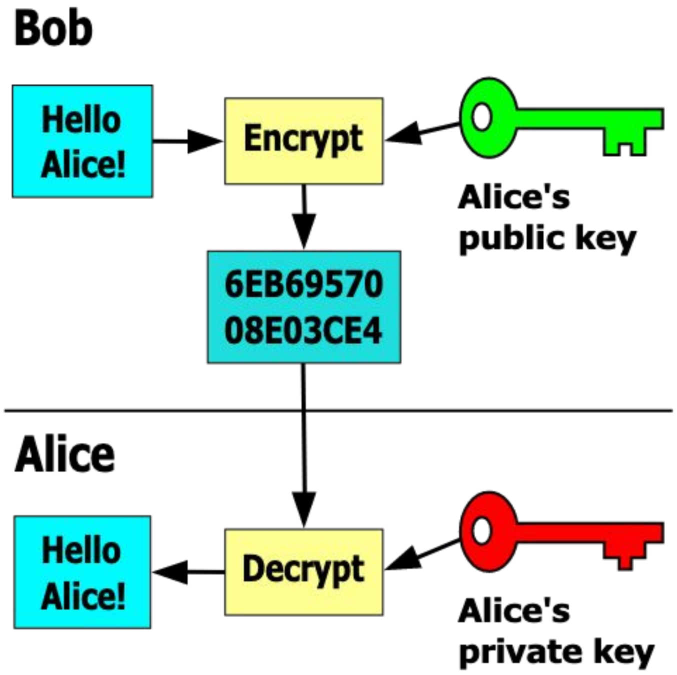


Asymmetric cryptography

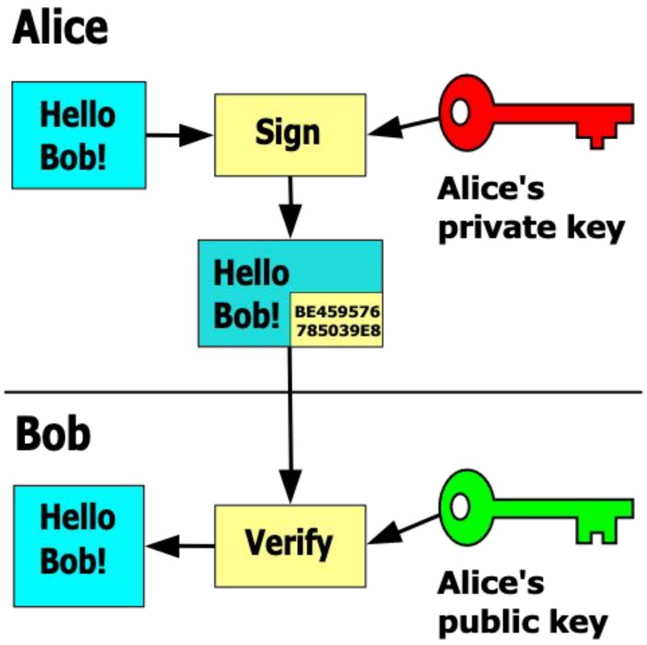
Into the real of public key cryptography …

Public key cryptography

Encrypt

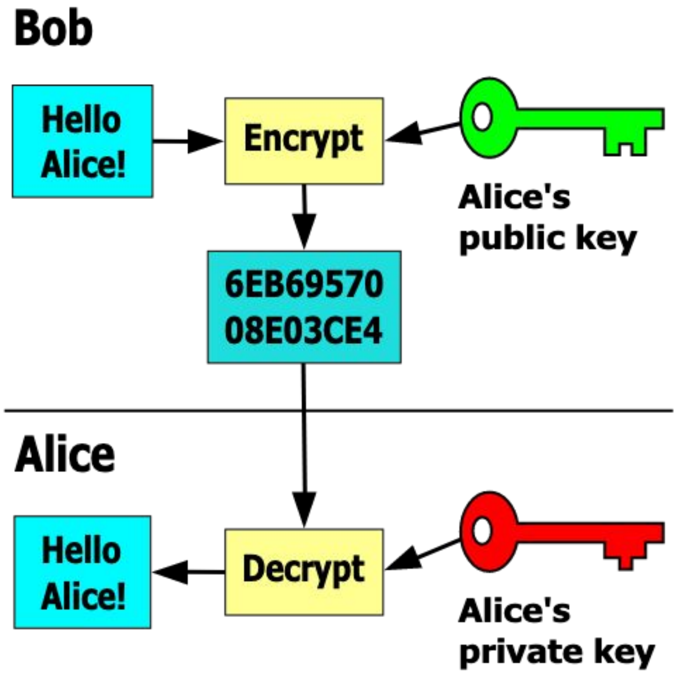


Sign

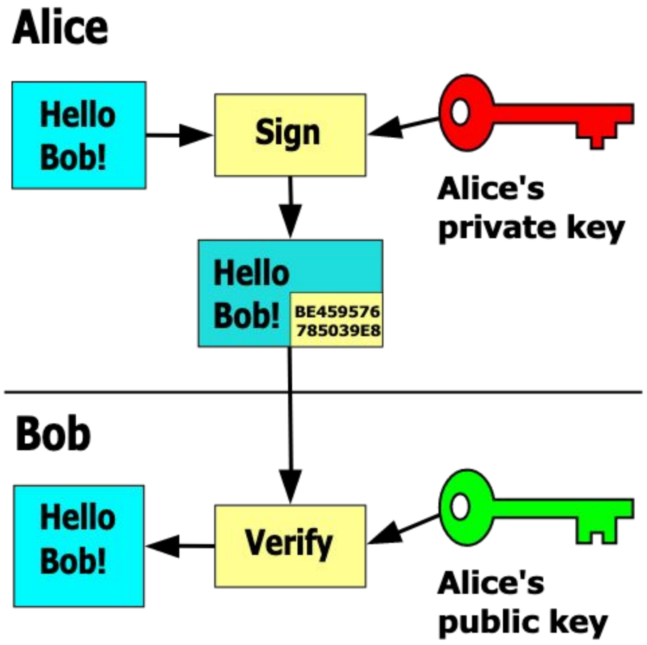


Public key cryptography – introducing *Eve*

Encrypt

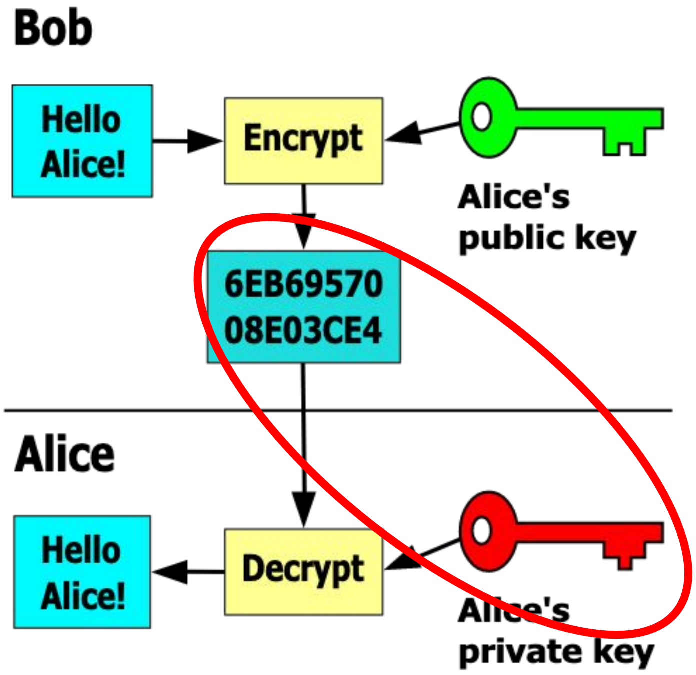


Sign

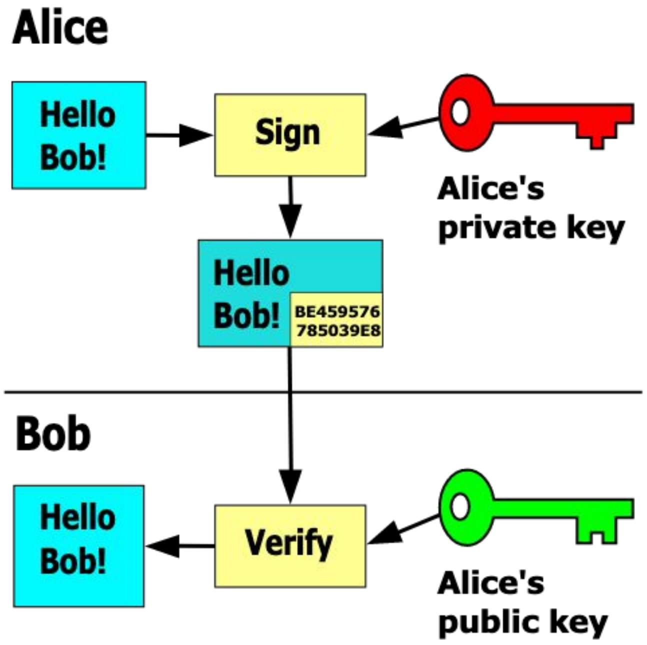


Public key cryptography – with *Eve*

Encrypt

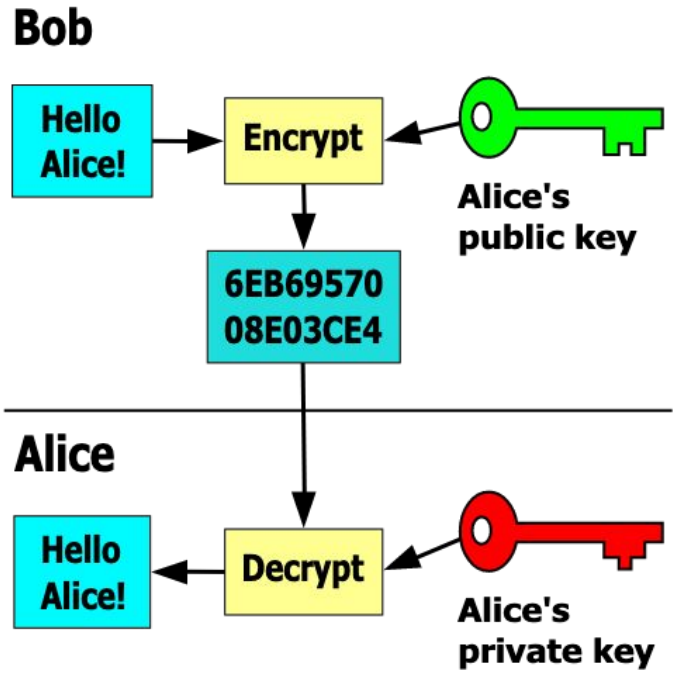


Sign

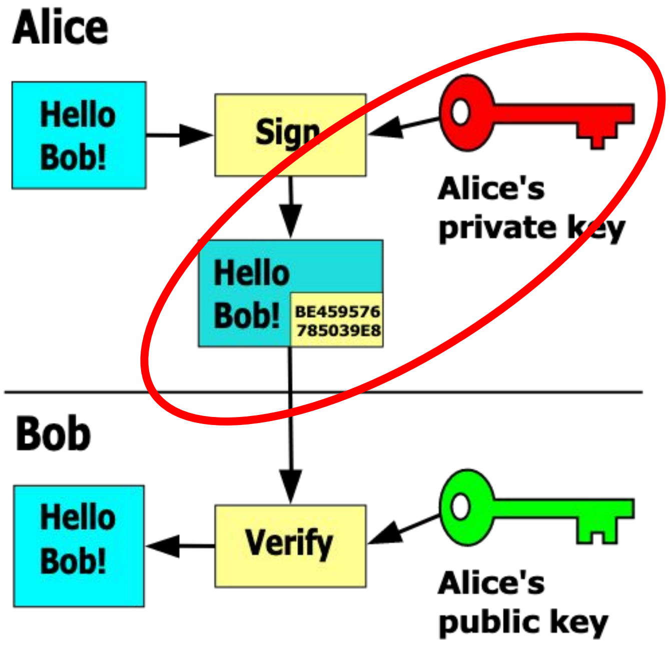


Public key cryptography – with *Eve*

Encrypt

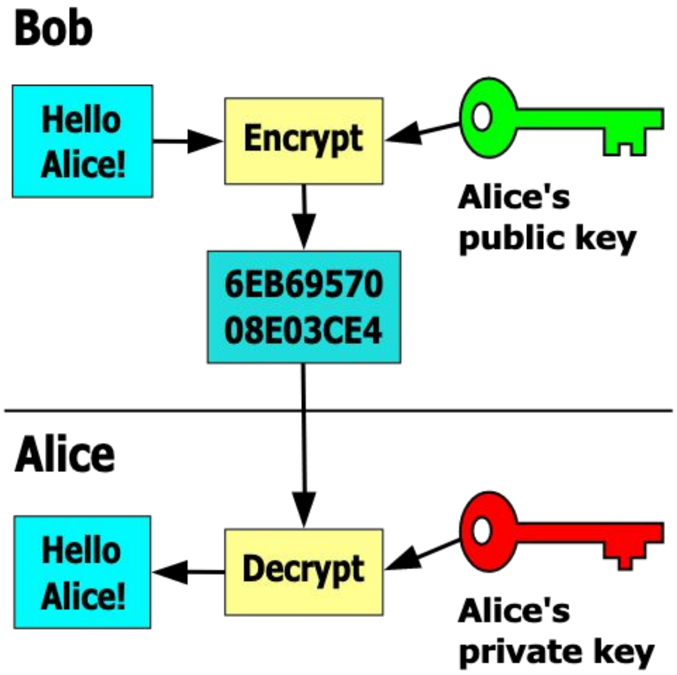


Sign

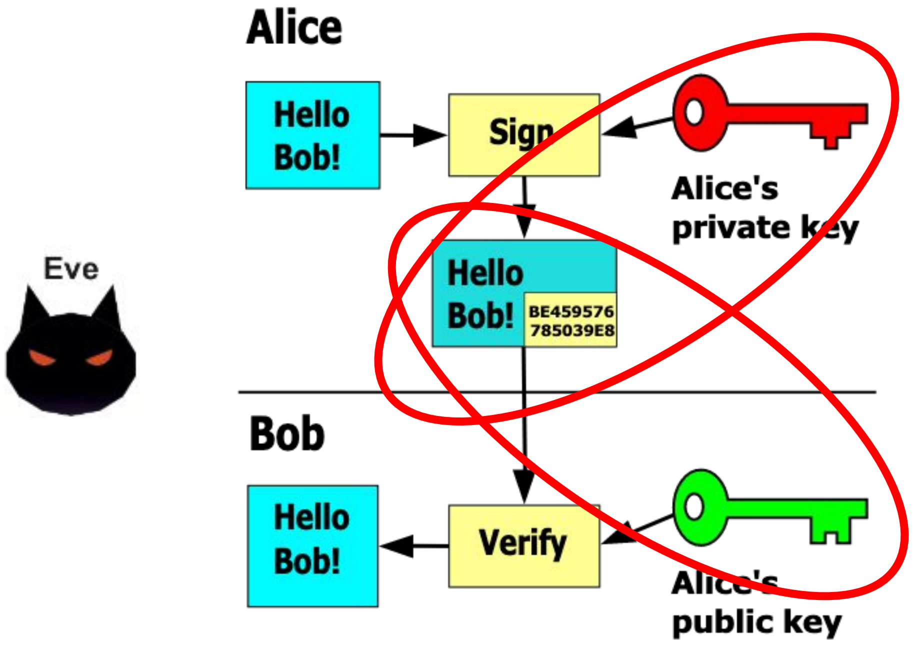


Public key cryptography – with *Eve*

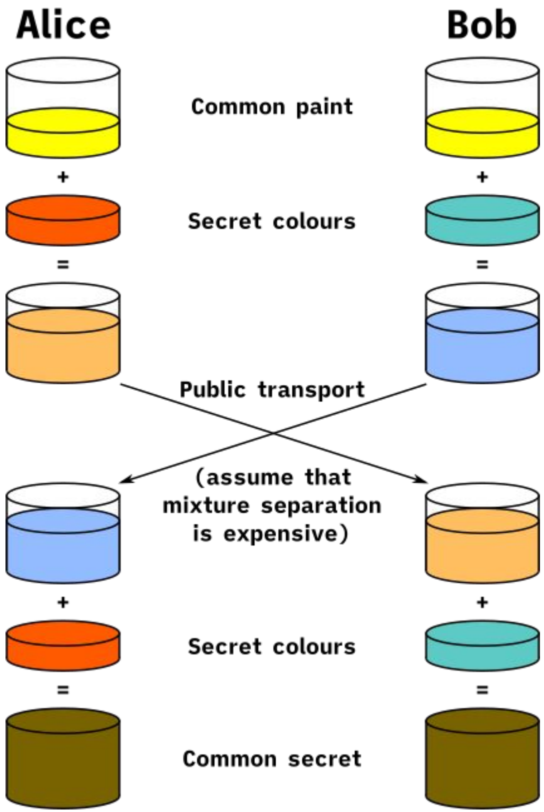
Encrypt

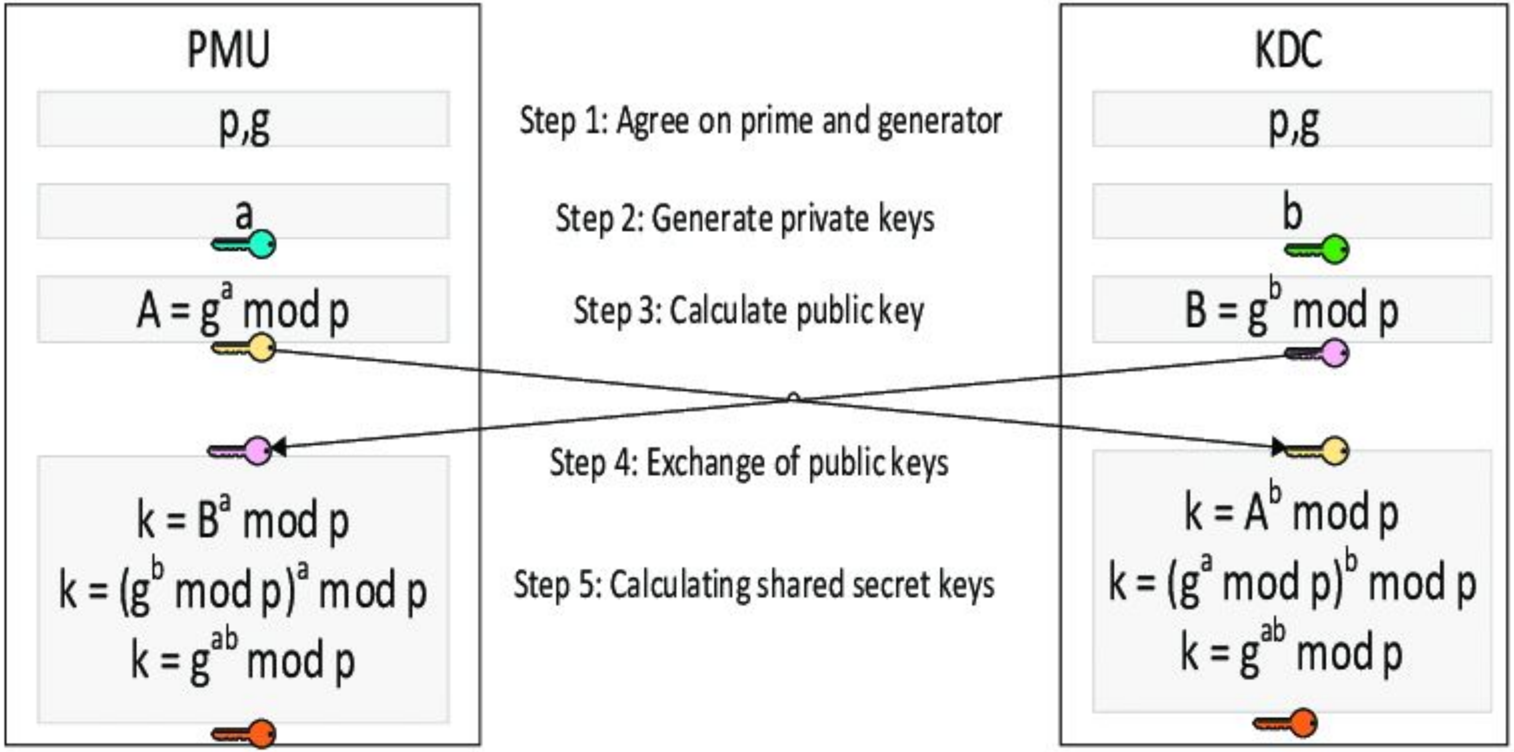


Sign



Diffie-Hellman





PMU
KDC
p,g
p,g
Step 1: Agree on prime and generator
a
b
Step 2: Generate private keys
A = g^a mod p
B = g^b mod p
Step 3: Calculate public key
Step 4: Exchange of public keys
k = B^a mod p
k = (g^b mod p)^a mod p
k = g^ab mod p
k = A^b mod p
k = (g^a mod p)^b mod p
k = g^ab mod p
Step 5: Calculating shared secret keys

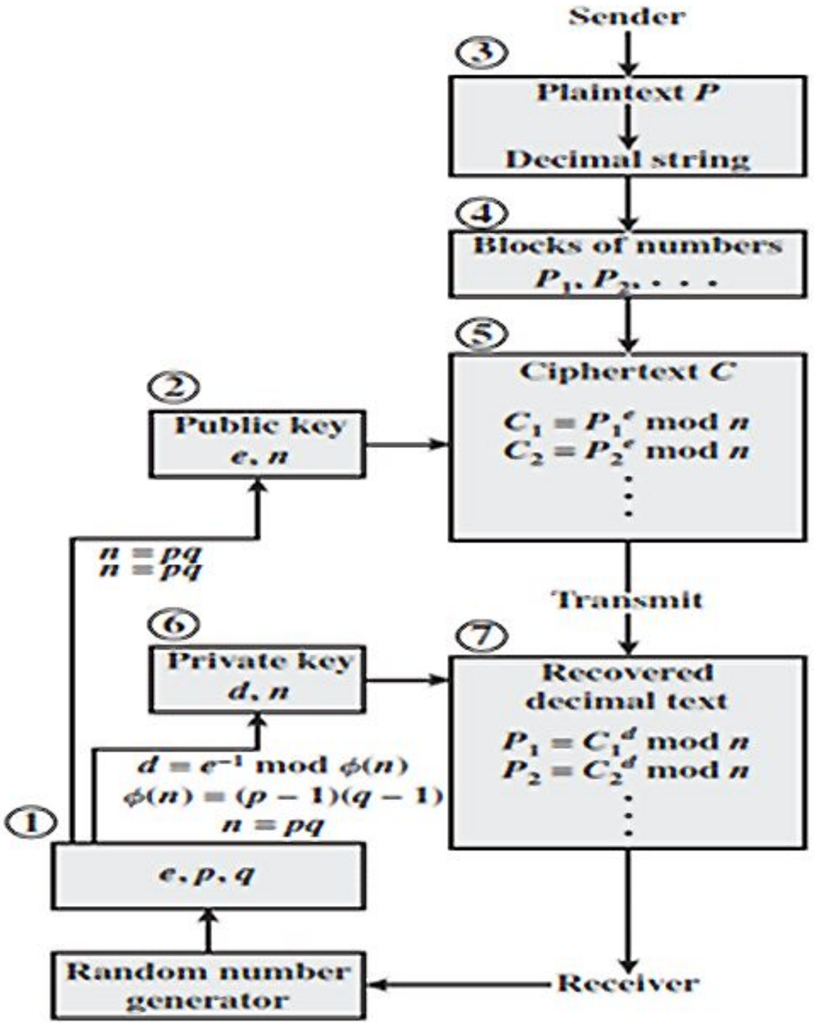
1. Alice and Bob agree to use a prime number p = 23 and base g = 5.
2. Alice chooses a secret integer **a = 6**, then sends Bob $A = g^a \bmod p$
   - $A = 5^6 \bmod 23$
   - A = **15,625** mod 23
   - $A = 8$
3. Bob chooses a secret integer **b = 15**, then sends Alice $B = g^b \bmod p$
   - $B = 5^{15} \bmod 23$
   - B = **30,517,578,125** mod 23
   - $B = 19$
4. Alice computes $s = B^a \bmod p$
   - $s = 19^6 \bmod 23$
   - s = **47,045,881** mod 23
   - **s = 2**
5. Bob computes $s = A^b \bmod p$
   - $s = 8^{15} \bmod 23$
   - s = **35,184,372,088,832** mod 23
   - **s = 2**
6. Alice and Bob now share a secret (the number **2**).

Diffie Hellman problem

$$\text{Given } q^a \text{ and } q^b \text{ produce } q^{ab}$$

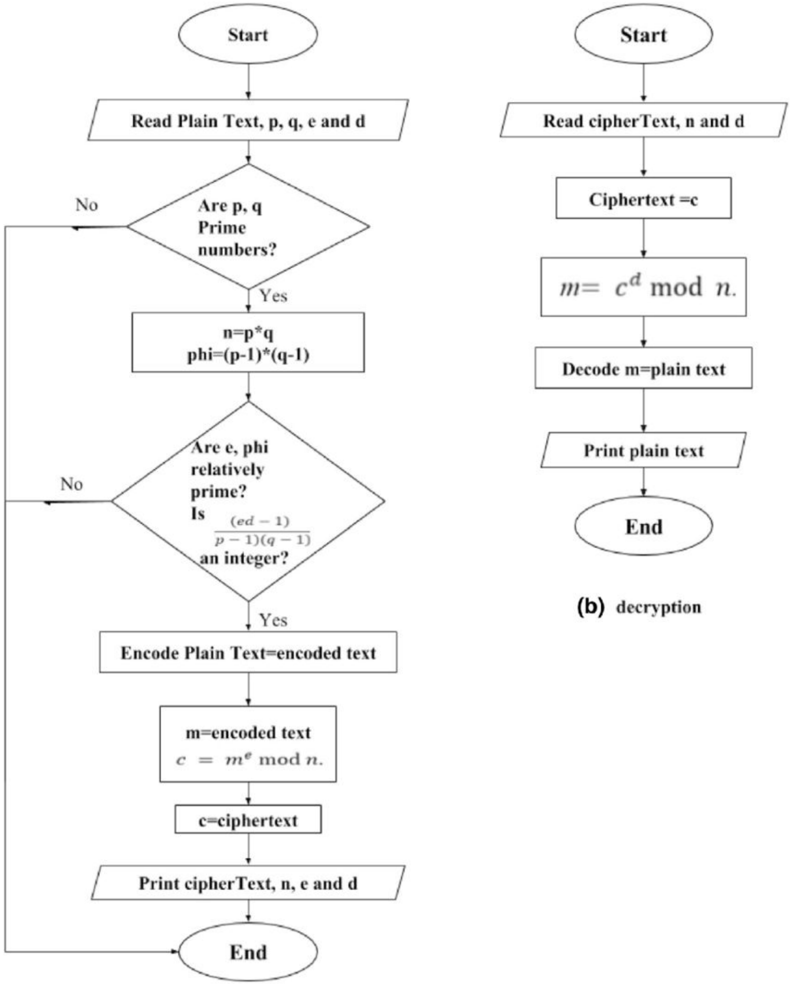
Hard without knowing ***a*** and ***b*** in advance

RSA



Sender
3
Plaintext P
Decimal string
4
Blocks of numbers
$P_1, P_2, \ldots$
5
Ciphertext C
$C_1 = P_1^e \bmod n$
$C_2 = P_2^e \bmod n$
2
Public key
e, n
$n = pq$
$n = pq$
Transmit
6
Private key
d, n
7
Recovered decimal text
$P_1 = C_1^d \bmod n$
$P_2 = C_2^d \bmod n$
$d = e^{-1} \bmod \phi(n)$
$\phi(n) = (p-1)(q-1)$
$n = pq$
1
e, p, q
Random number generator
Receiver

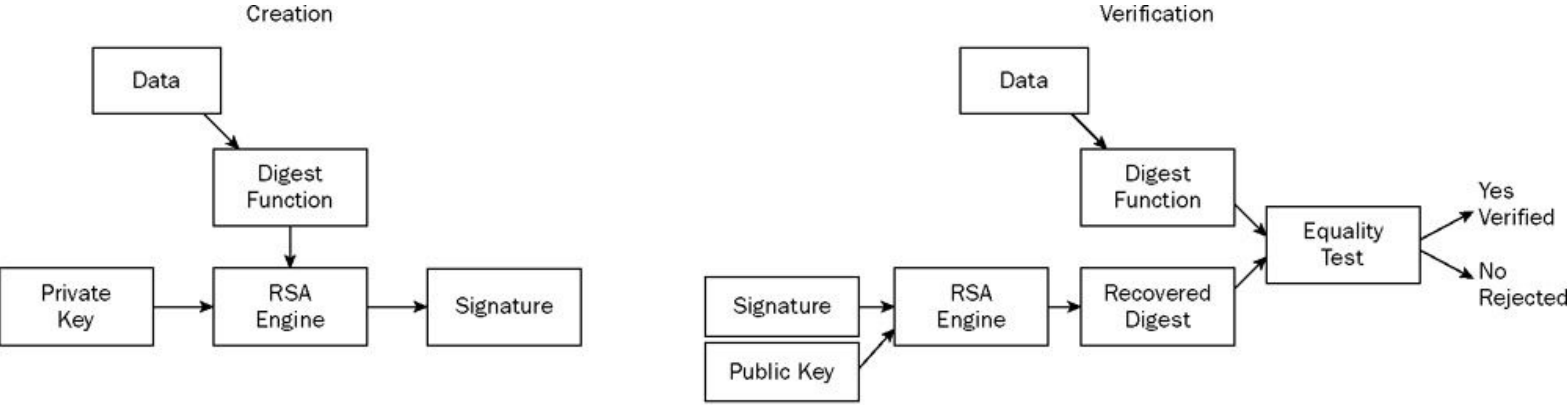
RSA



(a) encryption

(b) decryption

RSA



RSA Signature Processes

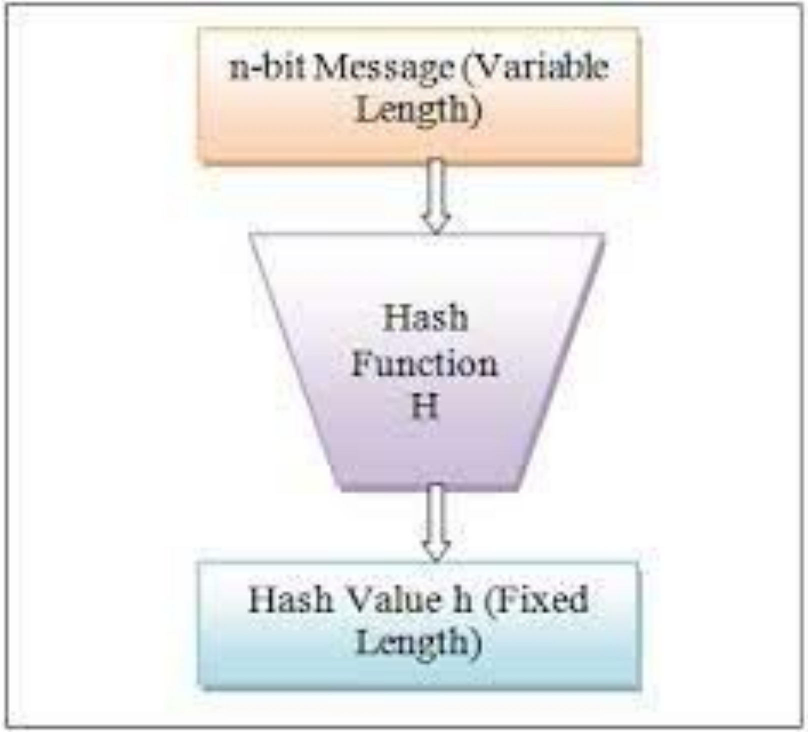
Discrete root problem / prime factorisation

$$x \mapsto q^x \pmod{p}$$

⇨ is easy, but given q^x finding x is hard

Little Hash digression

- One way function
- Used for Integrity



Hash properties

- Determinism
- Pre-Image Resistance
- Collision Resistance
- Avalanche Effect
- Hash Speed

Recipe

Hard problems

Trust

How do I transmit my first root secret?

Usage

- Asymmetric to exchange symmetric secret
- Solution to the chicken and egg problem

- Base for Public Key Infrastructure system

Aber, how do I make a secure protocol?

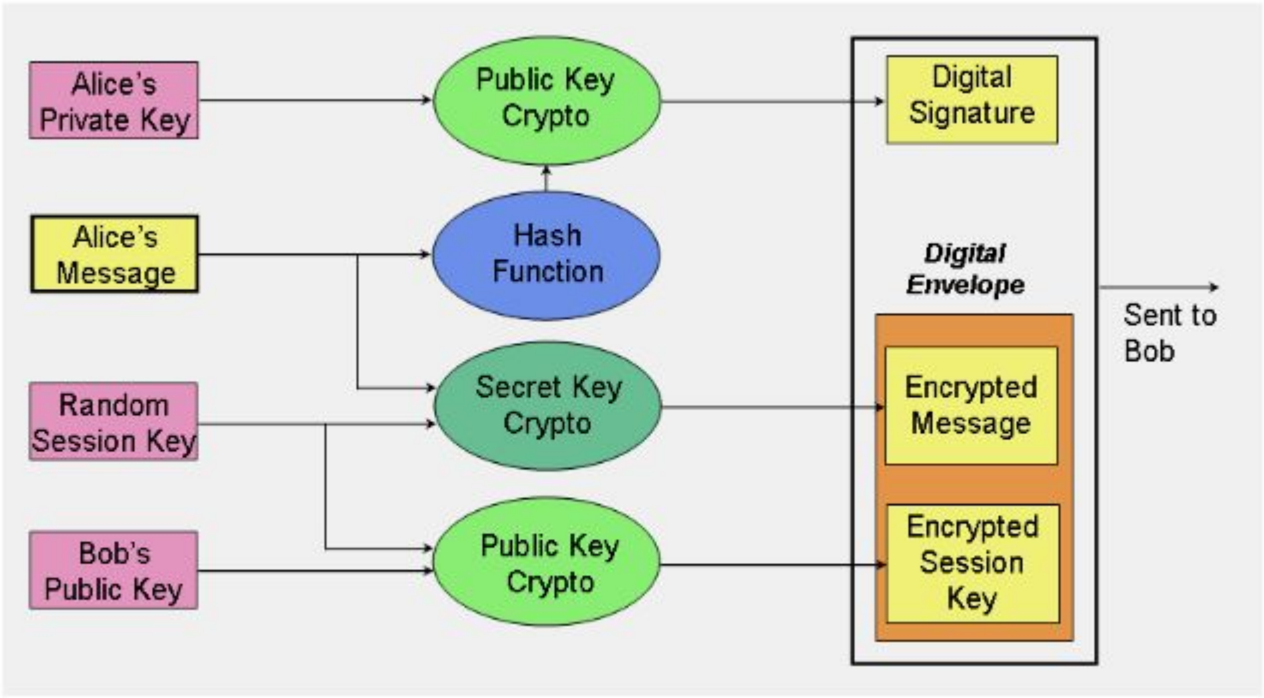


FIGURE 4: Use of the three cryptographic techniques for secure communication.

Aber, how do I make a secure protocol?

Authenticity of Alice

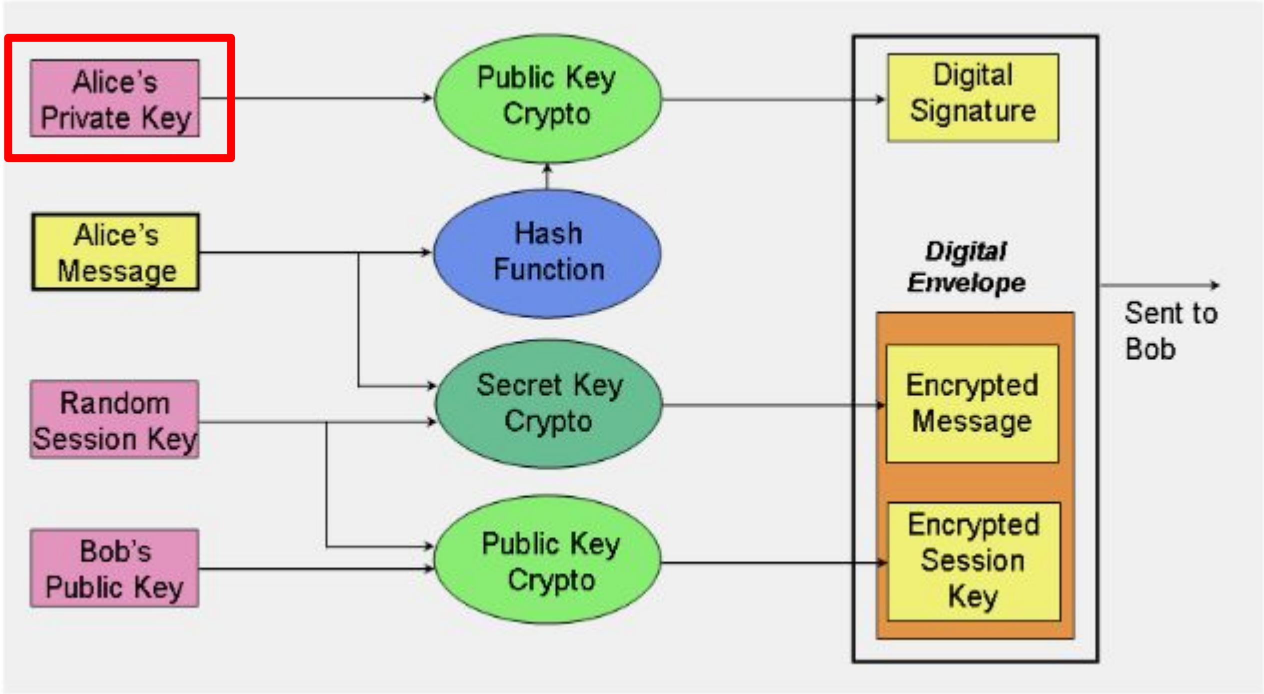


FIGURE 4: Use of the three cryptographic techniques for secure communication.

Aber, how do I make a secure protocol?

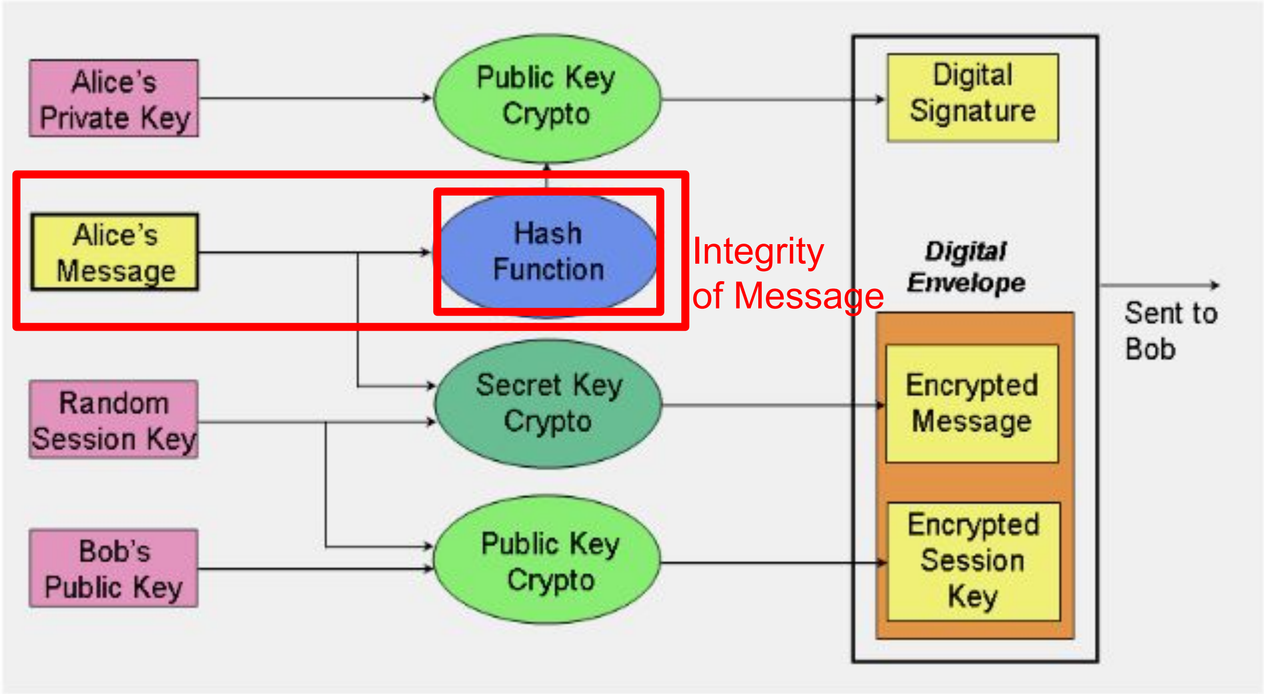


FIGURE 4: Use of the three cryptographic techniques for secure communication.

Aber, how do I make a secure protocol?

Confidentiality of message

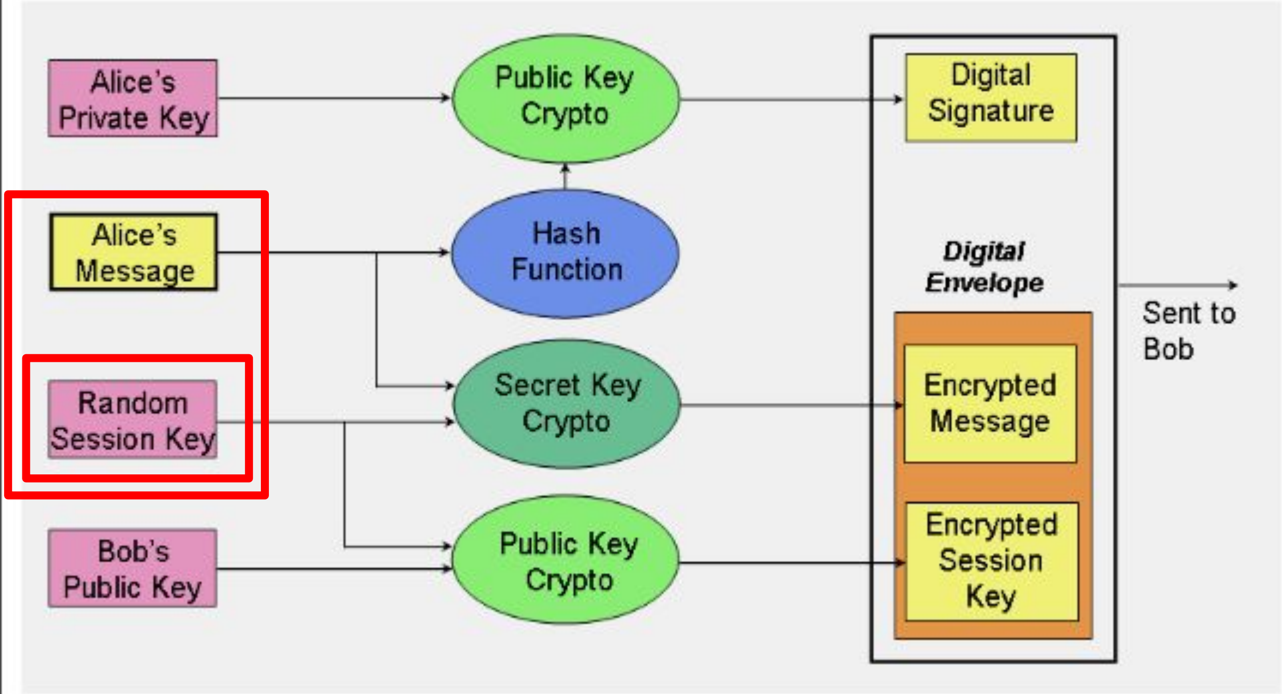


FIGURE 4: Use of the three cryptographic techniques for secure communication.

Aber, how do I make a secure protocol?

Confidentiality of session key

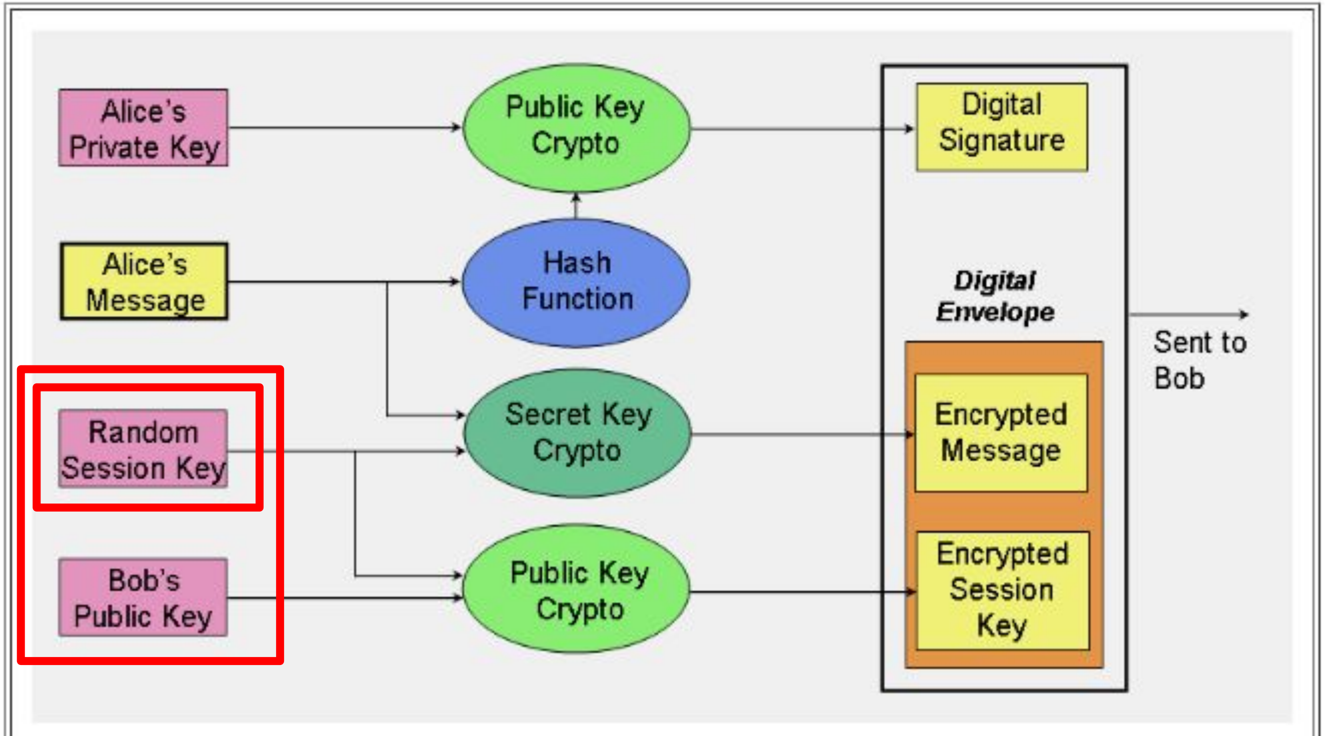
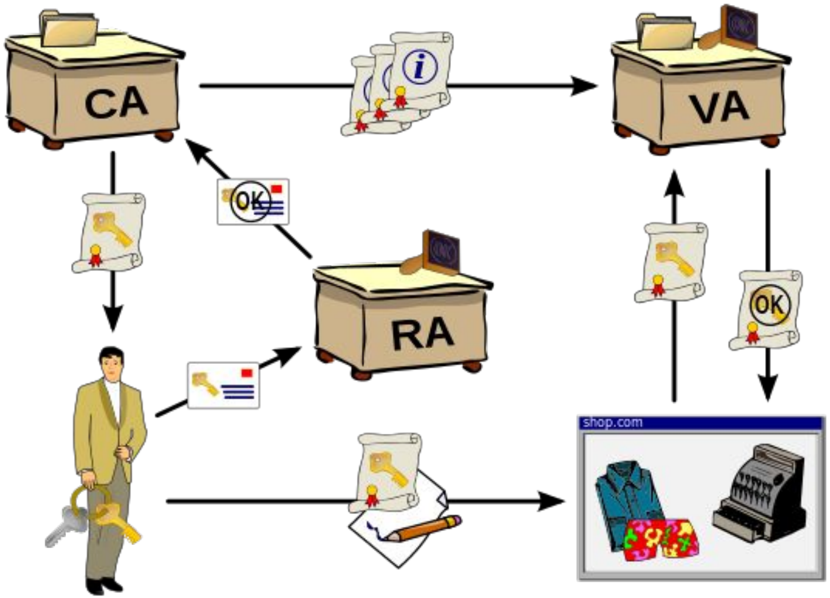


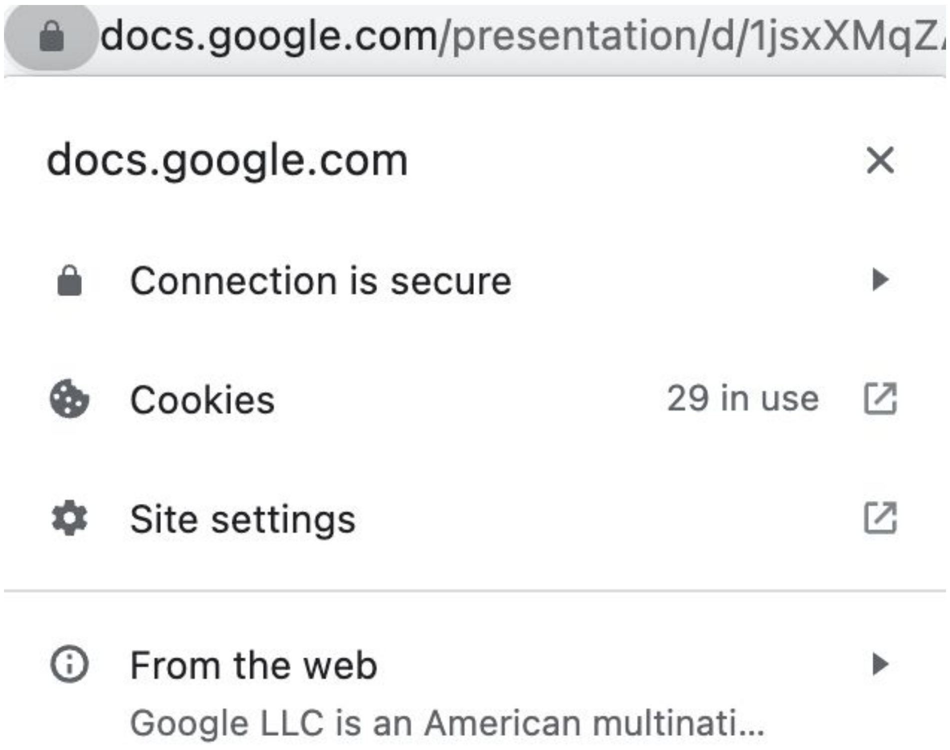
FIGURE 4: Use of the three cryptographic techniques for secure communication.

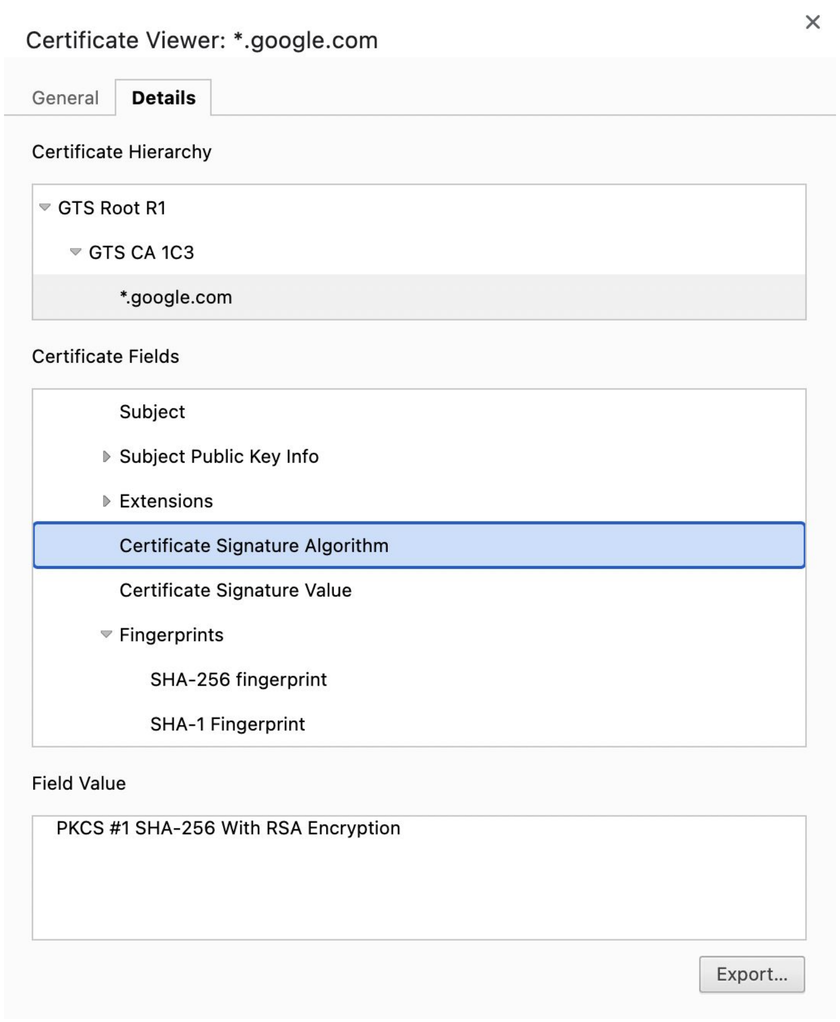
Certificates and PKI

- Certificate is a Public Key signed by a Trusted party

- Public Key Infrastructure
  - Provides a key hierarchy
  - Manages certificates




docs.google.com/presentation/d/1jsxXMqZ
docs.google.com
Connection is secure
Cookies
29 in use
Site settings
From the web
Google LLC is an American multinati...


Certificate Viewer: *.google.com
General
Details
Certificate Hierarchy
GTS Root R1
GTS CA 1C3
*.google.com
Certificate Fields
Subject
Subject Public Key Info
Extensions
Certificate Signature Algorithm
Certificate Signature Value
Fingerprints
SHA-256 fingerprint
SHA-1 Fingerprint
Field Value
PKCS #1 SHA-256 With RSA Encryption
Export...

We have reached TLS level!

Conclusion

- Security guarantees
- Hard problem to Trust to security property
- Secrecy of the key, not of the Algorithm
- Chicken and egg symmetry

Resources

- Introduction to Modern Cryptography
- BSI recommendations and glossary
- Boneh-Shoup cryptobook

Also see slides comments throughout the presentation

Math & Learn by doing

CONTENTS

CATEGORY	CHALLENGE
General - Mathematics	Greatest Common Divisor
General - Mathematics	Extended GCD
General - Mathematics	Modular Arithmetic 1
General - Mathematics	Modular Arithmetic 2
General - Mathematics	Modular Inverting
Mathematics - Modular Math	Quadratic Residues
Mathematics - Modular Math	Legendre Symbol
Mathematics - Modular Math	Modular Square Root
Mathematics - Modular Math	Chinese Remainder Theorem

Questions?