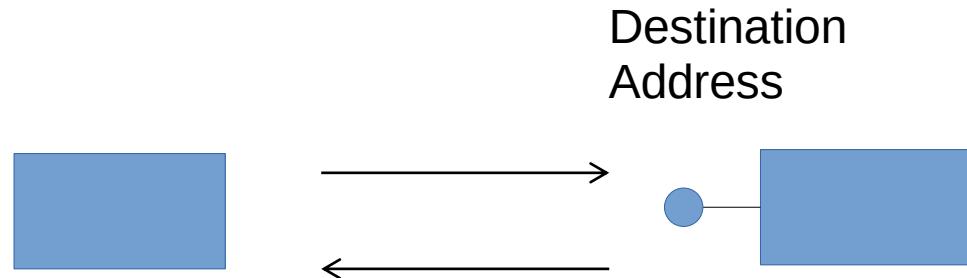


Introduction to Cryptography

arnaud.nauwynck@gmail.com

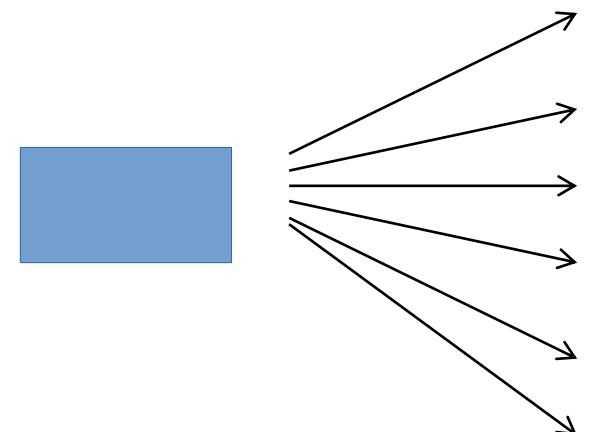
Communications

Point - to - Point



Dual Channel Communication
...example for Request - Response

Broadcast



Who are Alice, Bob, Eve, Malory, & Trent .. ?

Article Talk

Read Edit View history

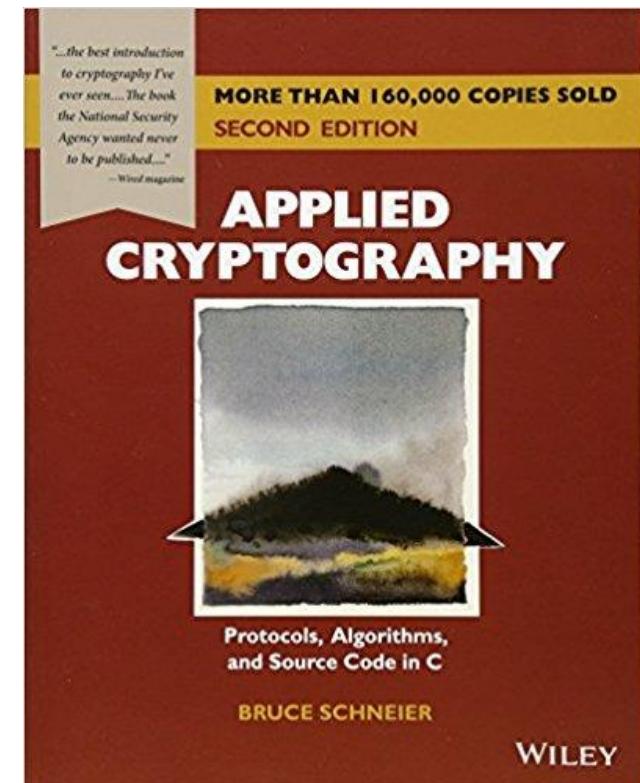
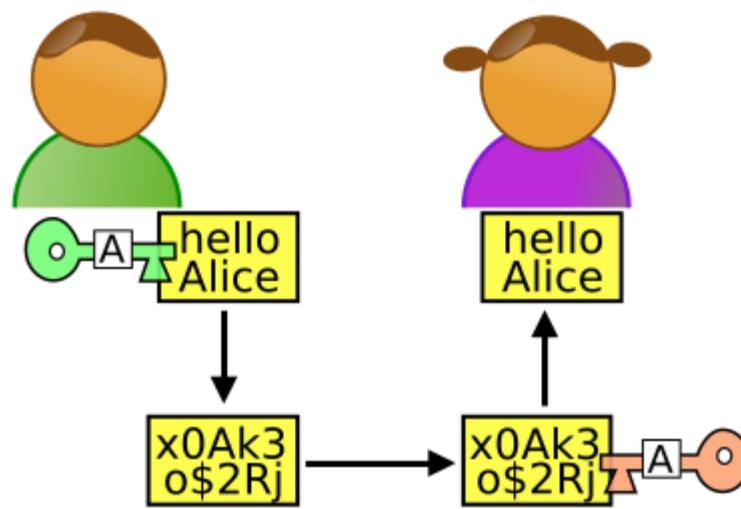
Search Wikipedia



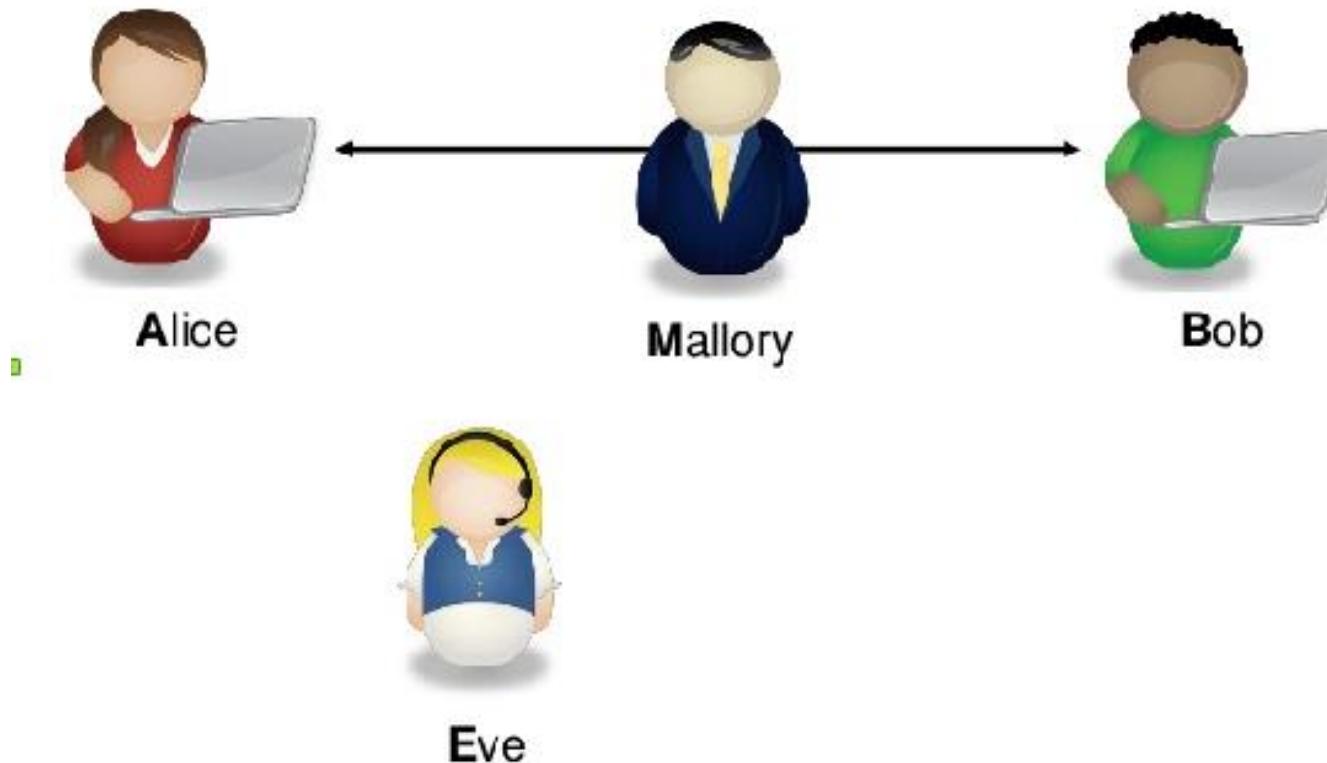
Alice and Bob

From Wikipedia, the free encyclopedia

Alice and Bob are [fictional characters](#) commonly used in [cryptology](#), as well as science and engineering literature. The Alice and Bob characters were invented by [Ron Rivest](#), [Adi Shamir](#), and [Leonard Adleman](#) in their 1978 paper "A method for obtaining digital signatures and public-key cryptosystems."^[1] Subsequently, they have become

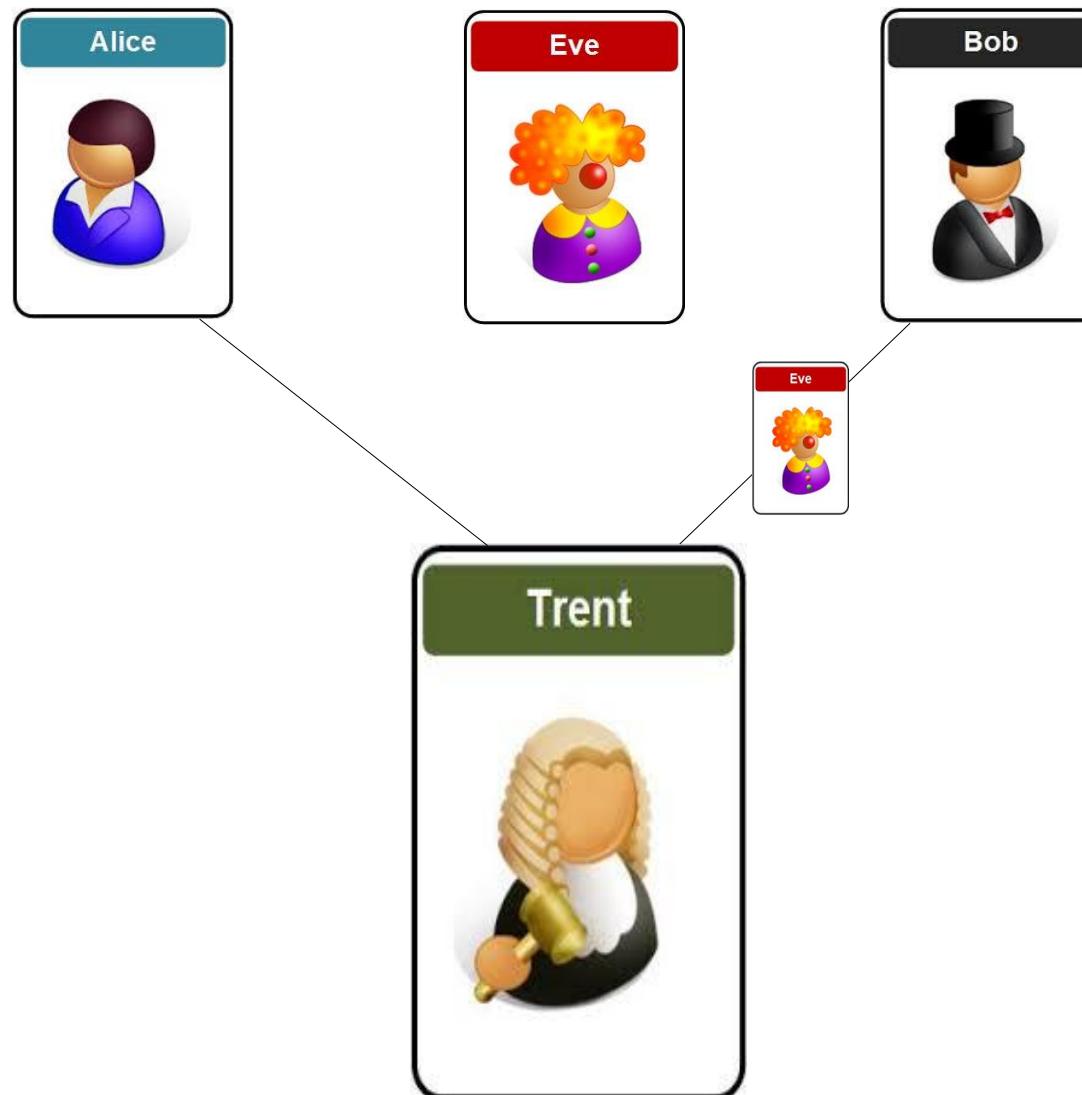


Eve (E=Environment ... or Evil) and Malory (M=Malicious)



Mallory intercept + modify data
... Eve does't not

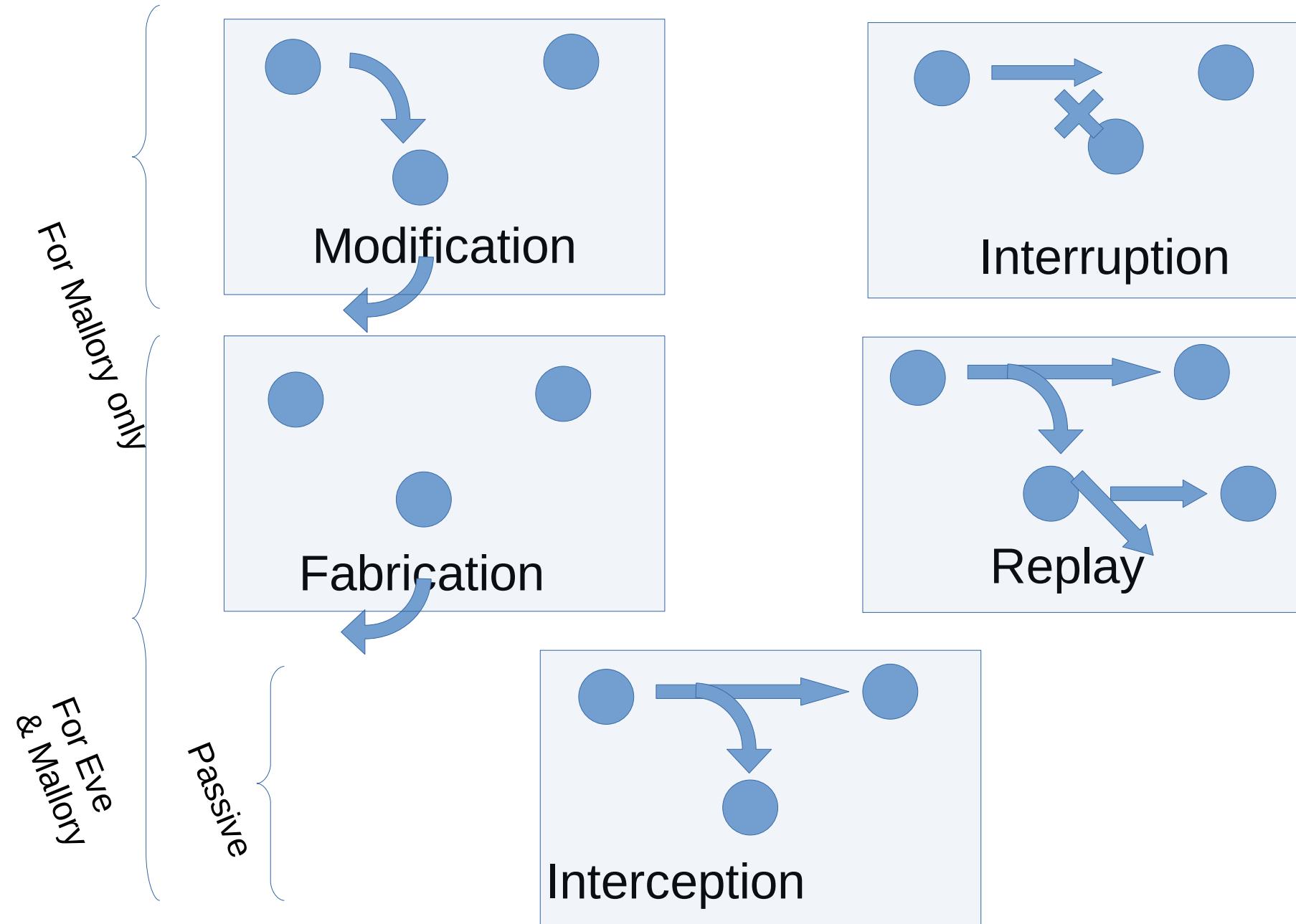
In Trent you (may) Trust



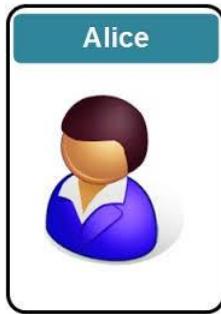
M = Malory = Malicious User
Man-In-The-Middle



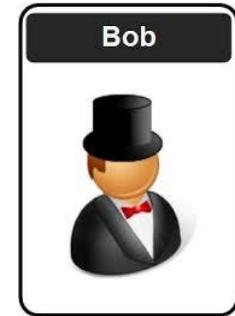
Man-In-The-Middle Weapons



A (Secured?) Chat



Allo Bob ? Alice speaking



Hi Alice

I have a secret to tell you
I will tell you only if ...

Ok, I will do/pay/.. it

Secret = bla bla

What's (Possibly) Wrong ?

Be Paranoiac !!

By Default, Everything is wrong

Proofs next ...

Possibly Wrong

Authentication

Data Leak

Confidentiality

Data Replay

Integrity

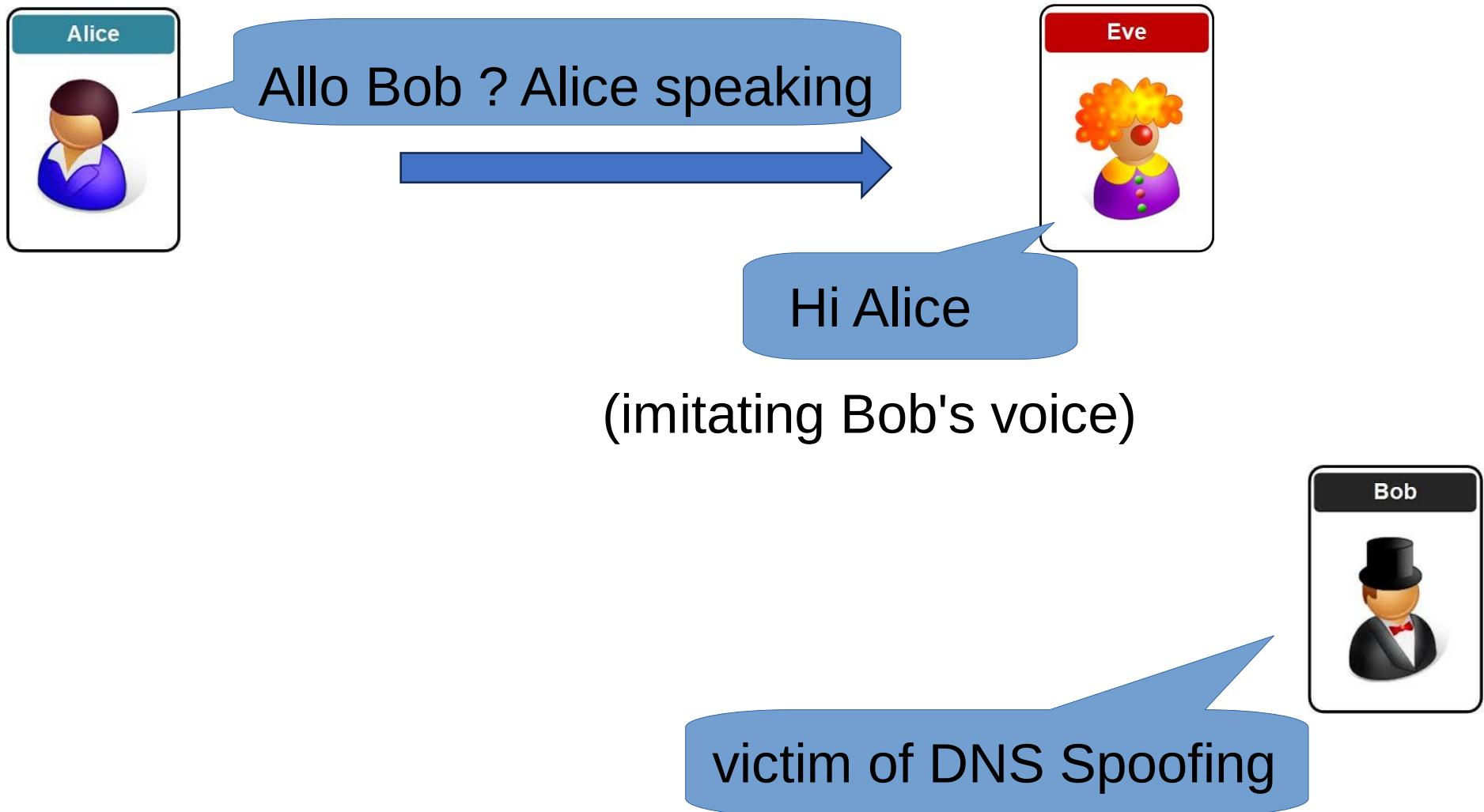
Predictable guess

Authorisation

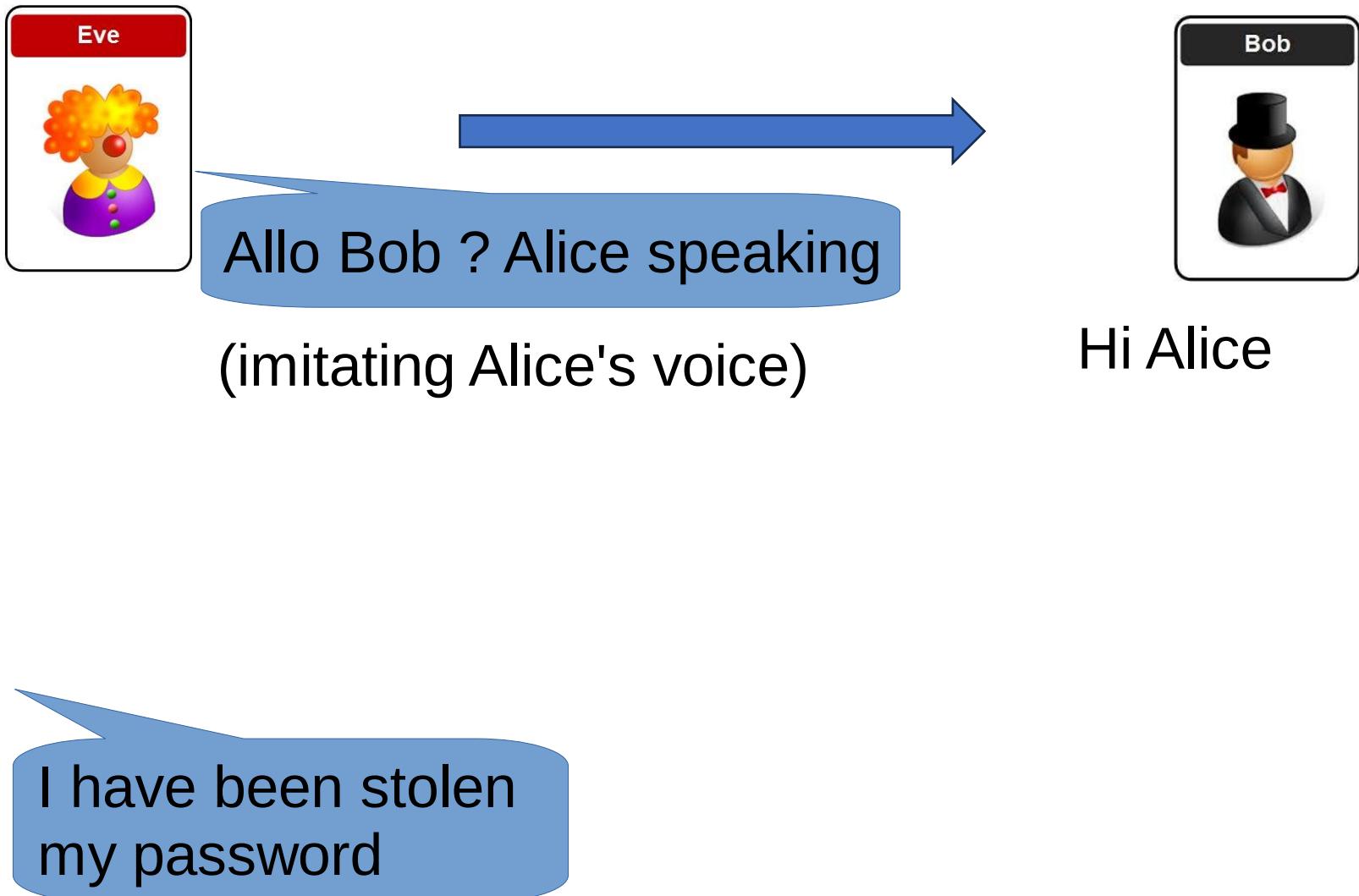
...

Password Crack

Authentication Server (or substituted server?)



Client Authentication



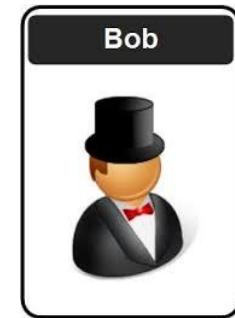
... Social Engineering



Eve

Allo Bob ? I am a friend of Alice

She asked me to tell you..



Bob

Hi

Sure, go ahead



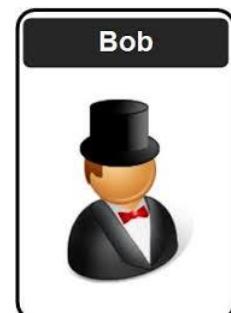
Alice

Bob is so naive

Authentication denied



Allo Bob ? Alice speaking



Your voice is strange
I don't recognize Alice's voice
Bye



Basic Authentication



Allo Bob ?
Alice speaking
proof I am Alice:
my password is ..

Hi Alice

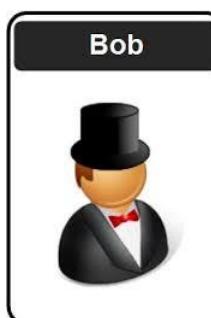
(imitating Bob's voice)



Your password is OK

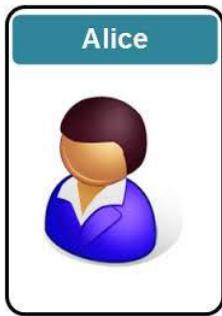


Allo Bob ?
Alice speaking
proof I am Alice:
my password is ..



Hi Alice
Your password is OK

Password Challenge



Allo Bob ?
Alice speaking

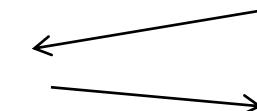


Really ?
Proove me your identity

I don't give my password
I can give you only a clue
you can check with Trent

OK, Send me you pass hashed by “X”

(.. Compute $Y = \text{hash}(P, X)$)
It is “Y”



Is it True that ?
“Y” ?= Alice pass hashed by “X”

OK Alice

Trusted Thirdparty...



Allo Bob ?
Alice speaking



Really ?
Proove me your identity

Please Attest I am Alice
(here is my password)



I ("Trent") certify this token paper
was delivered to Alice on jun 2017:
XADS4FSQ3RTXF

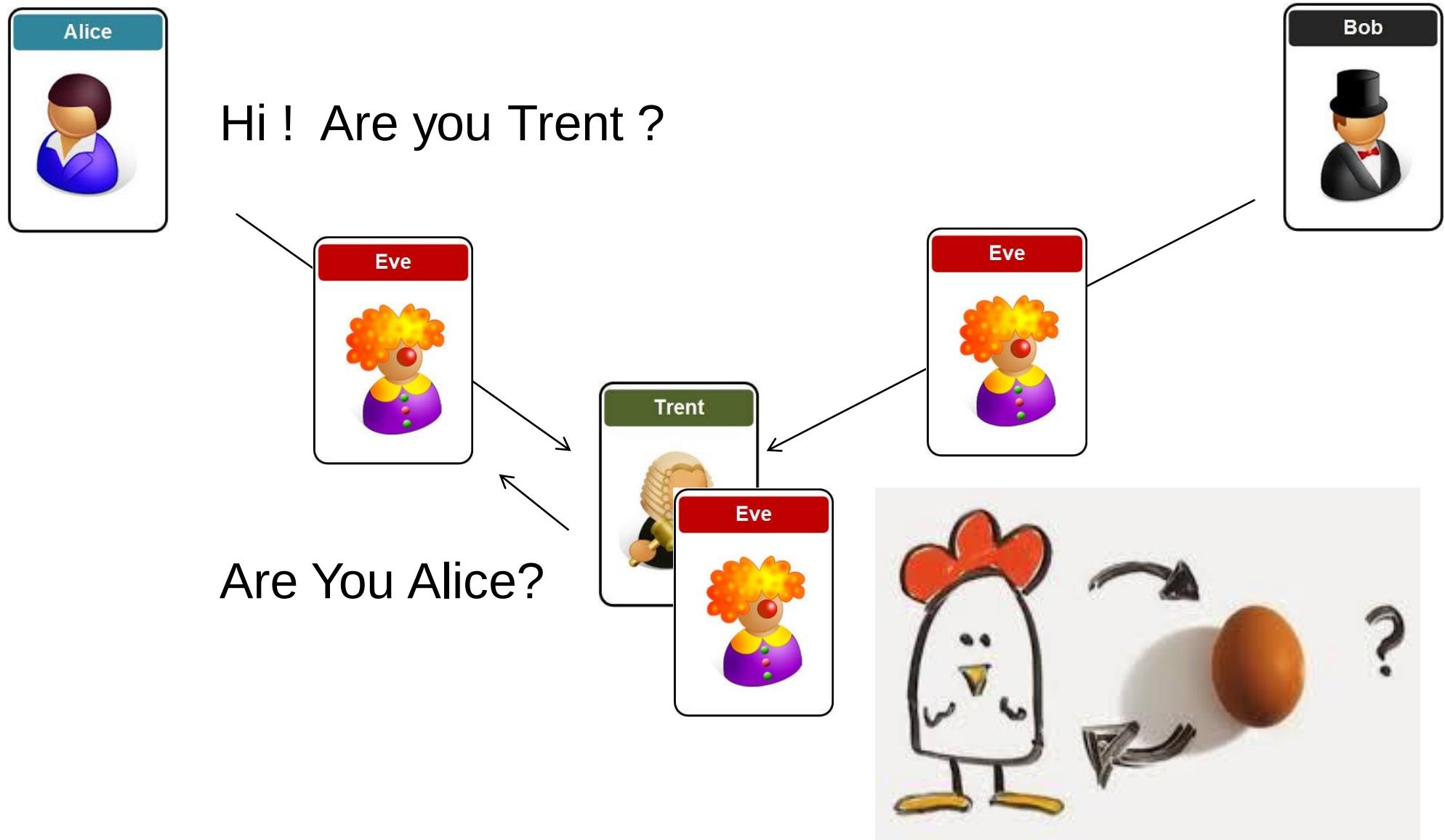
Bob: Ask "Trent" ... he prooved my identity by token XADS4FSQ3RTXF

Is it true this token was given to Alice:
XADS4FSQ3RTXF ?

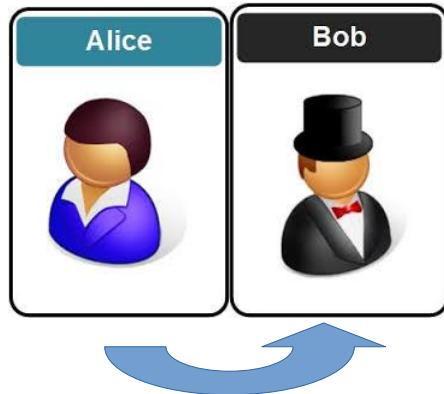
OK Alice

Yes (And Alice did not complain yet being stolen)

Problems With Thirdparties...



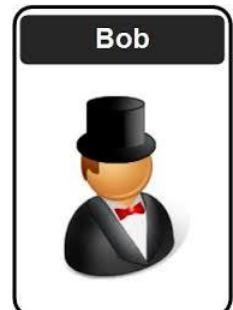
Physical Hand-Shake ... Exchange Shared Key



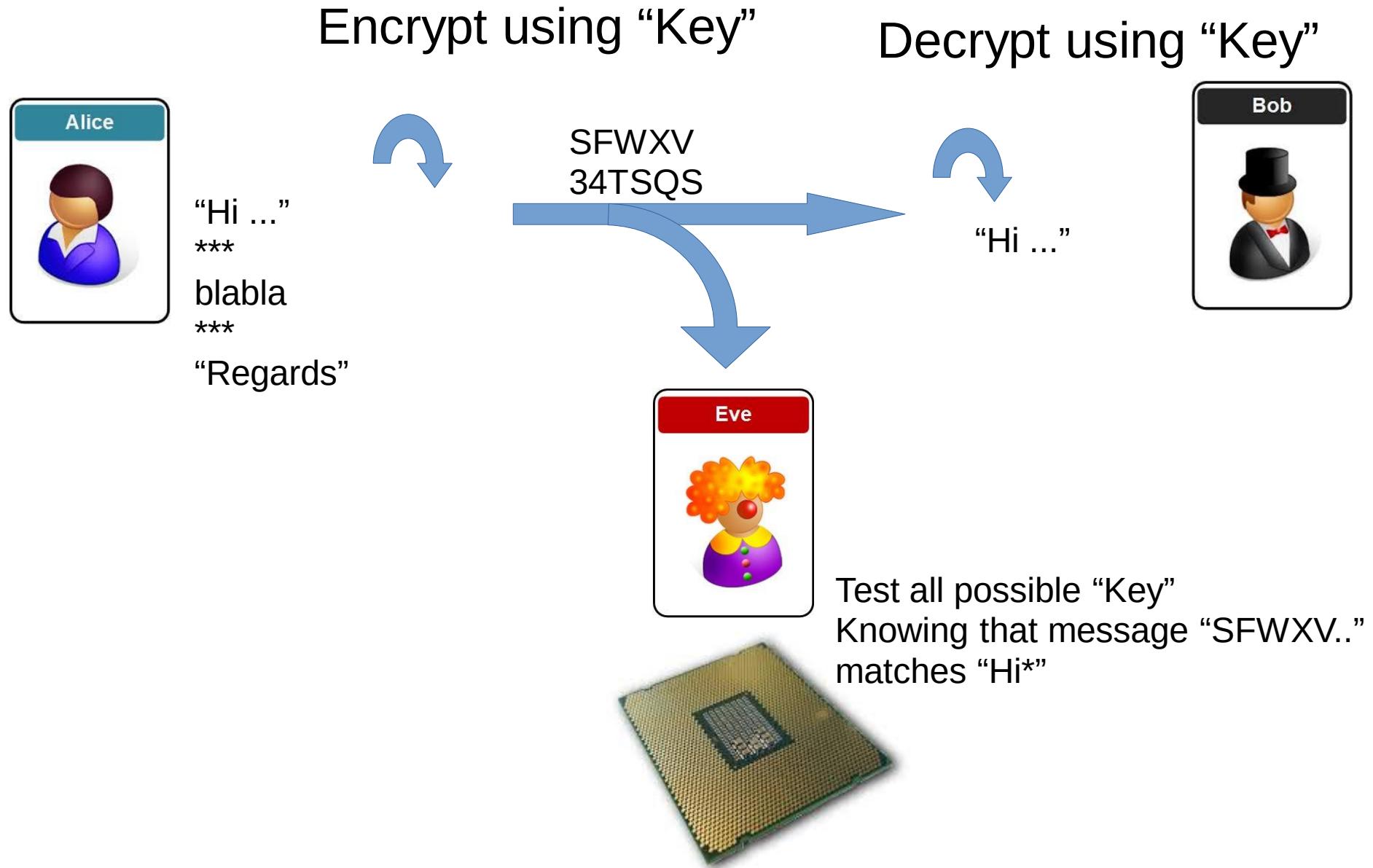
Bob, here is a secret “key” for talking to you in 6 monthes



Hi Bob
Let us use our private shared key
(remember I gave you last 6 monthes)
..
** Switch Encrypted **
SDFG23456RGQEST43



“Key” recomputed from Data ...



“Key” inversion from Predicatable Data

WARN : do NOT use TOO many data
 too easily predictable data

otherwise “Key” could be recomputed by Eve...

Throw “Key” at end of communication

Famous during World War 2 :
Alan Turing decrypted German Submarine “Enigma” Messages
... partly because all messages ended with “Hi Hitler”

Encrypted ... & Still Weak

Authentication

Data Leak

Confidentiality

Data Replay

Integrity

Predictable guess

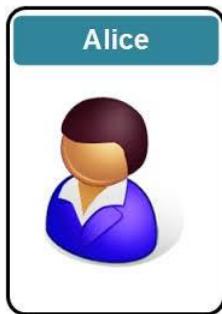
Authorisation

...

Password Crack

(Partial) Data Leak

Example : Modulo - Hashes



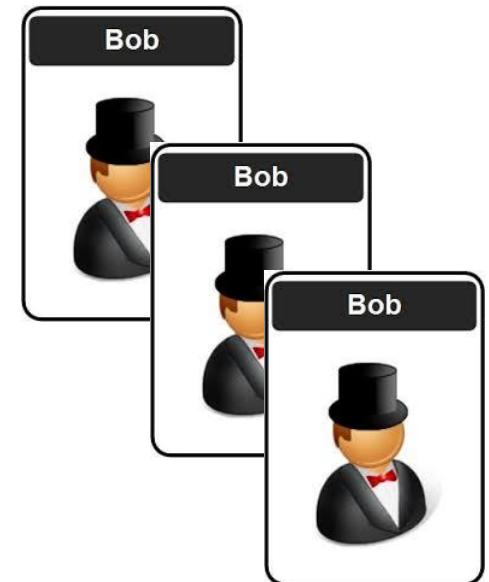
Challenge response (1)
My password modulo 1000 is 4



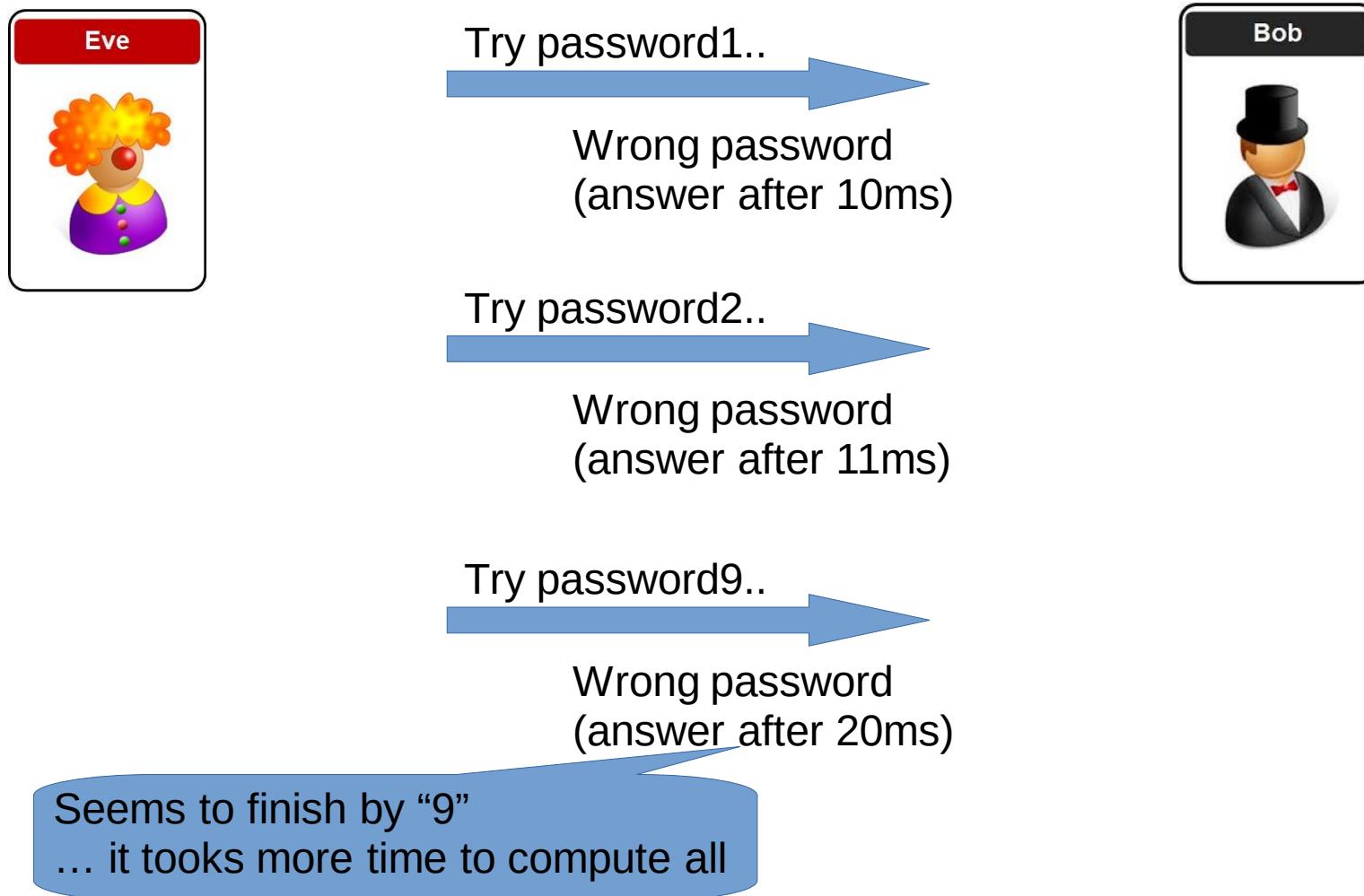
Challenge response (2)
My password modulo 6421 is 2



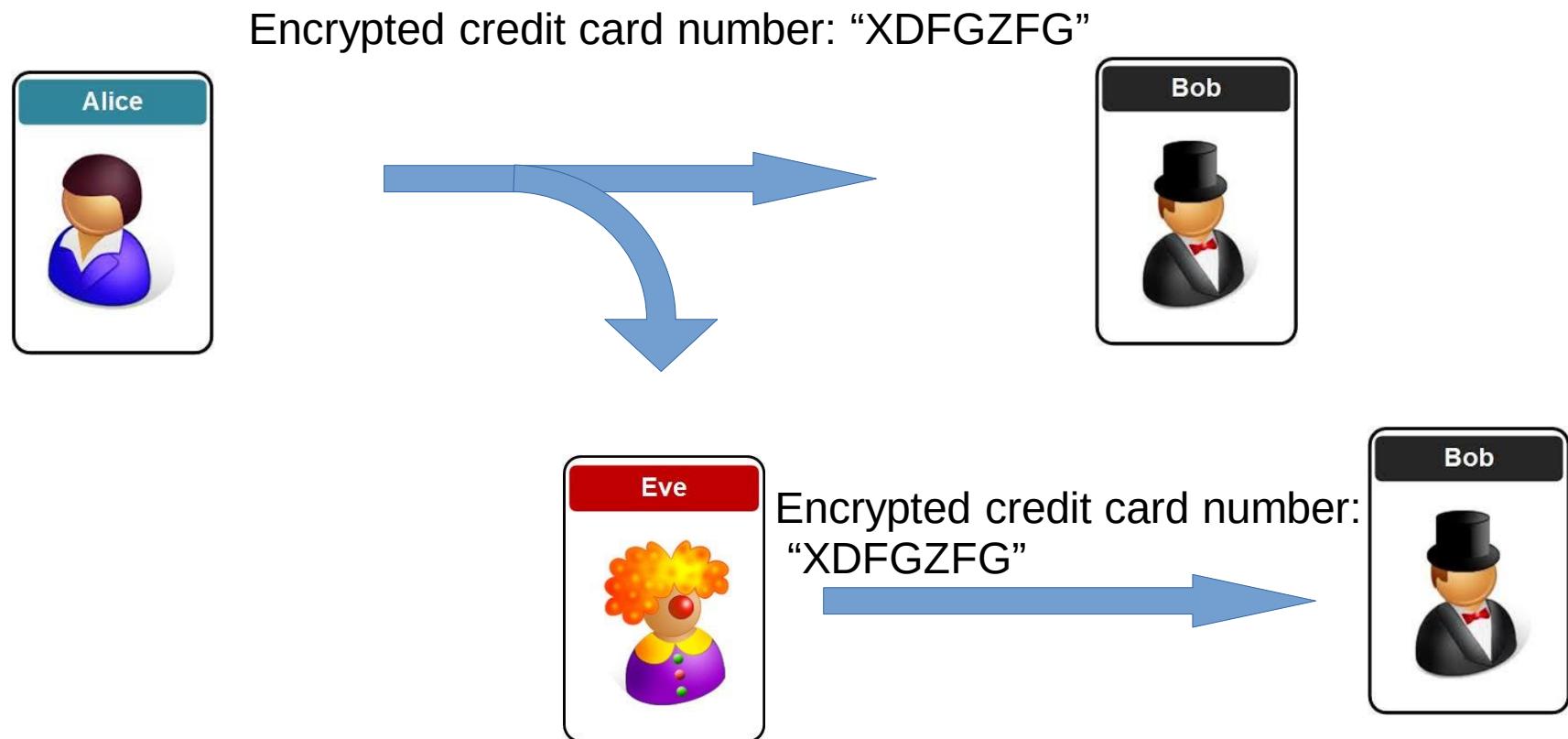
Challenge response (3)
My password modulo 9482 is 7



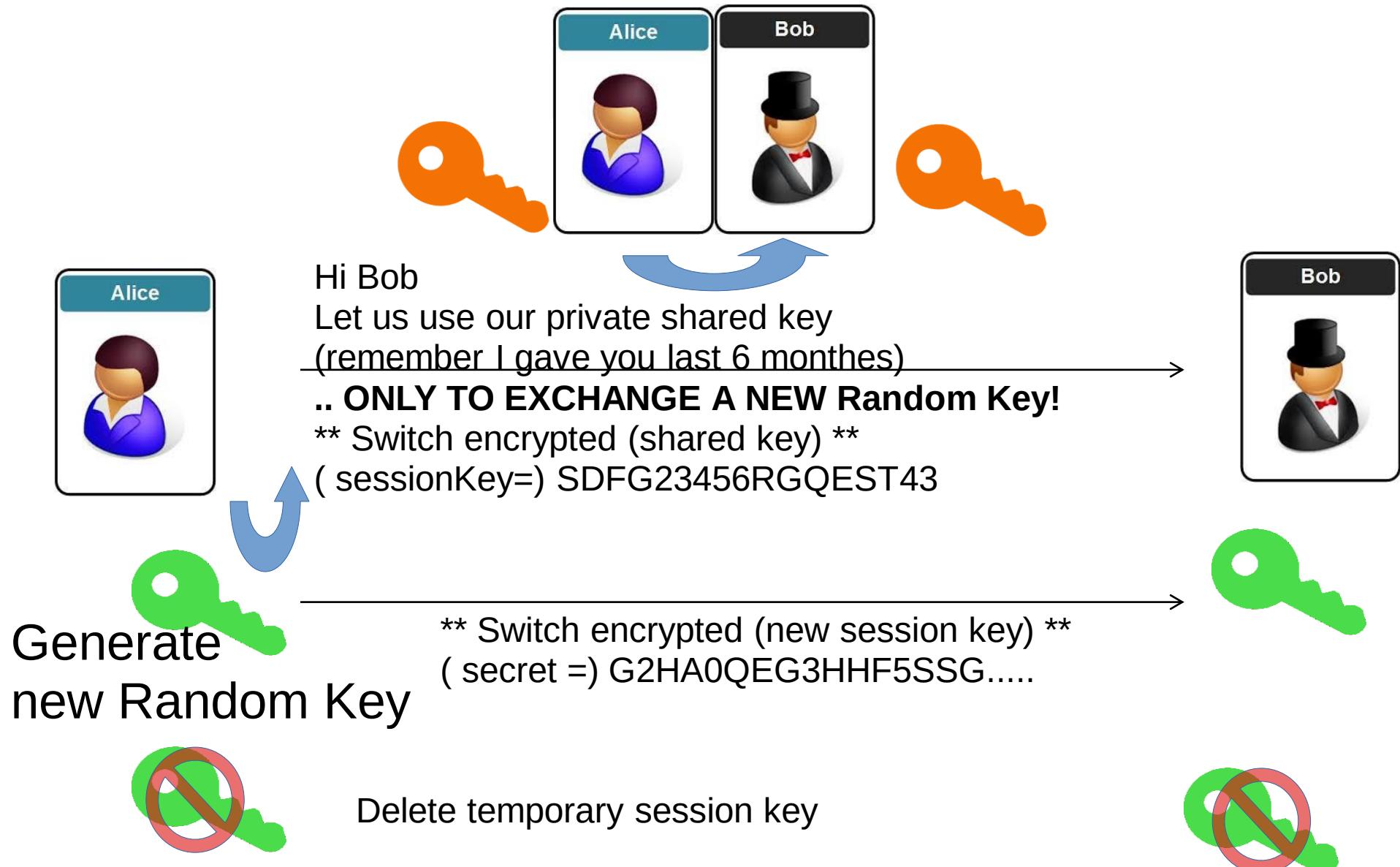
Partial Data Leak Padding – Delay...



Encrypted ... Data Replay



Using Shared Key for Exchanging TMP SessionKey ONLY



Fast Crypt using for example XOR + Session Key Random Generator (modern app would use AES-256)



Generate
new Seed Random Generator

(seed)
..... → 1 0 1 0

(encrypt seed =) G2HA0QEG3

(seed)
..... → 1 0 1 0

** Switch encrypted (XOR seed generator mode) **

...

Secret message blabla

Seed gen
.. 0 1 0 1 1 1 0 1 1 ..
1 0 1 0 0 1 0 1 0

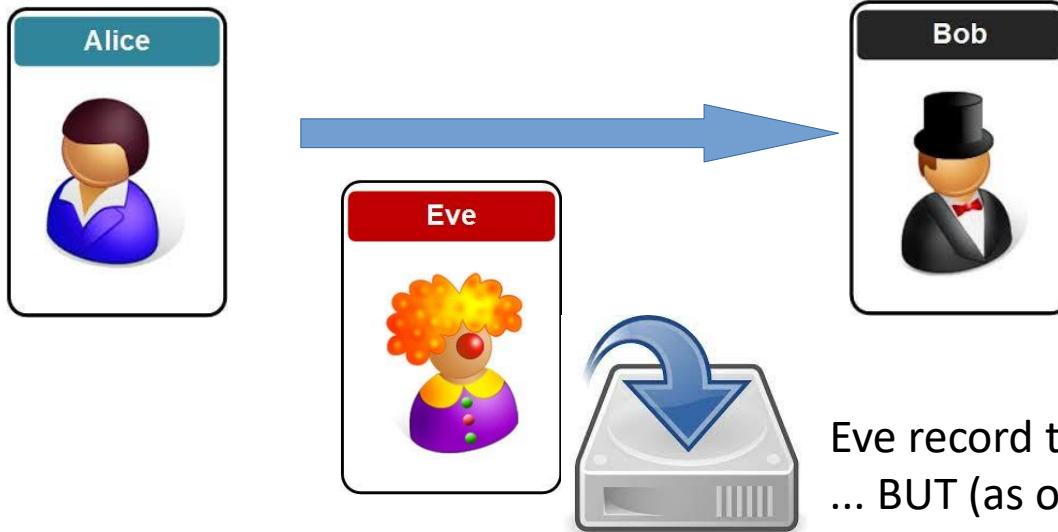


Seed gen
.. 0 1 0 1 1 1 0 1 1 ..
1 0 1 0 0 1 0 1 0



Secret message blabla

Forward Secrecy Property ?



Eve record the full conversation
... BUT (as of 2024) is NOT able to decrypt (yet)



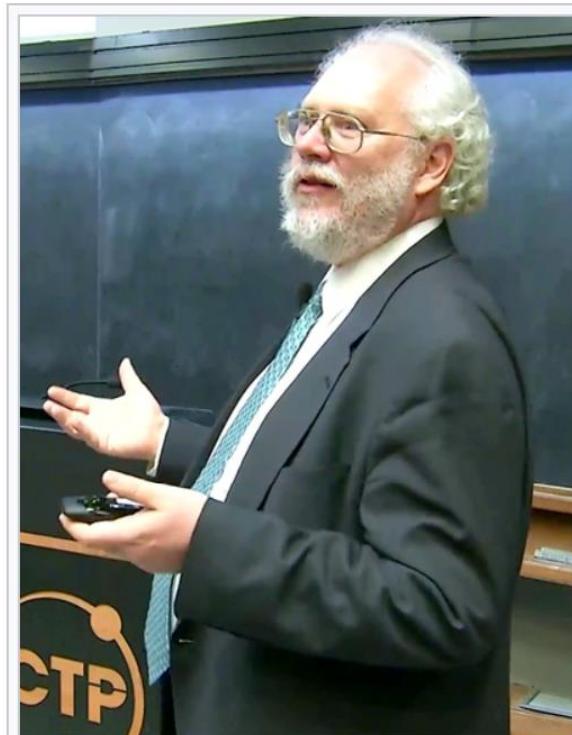
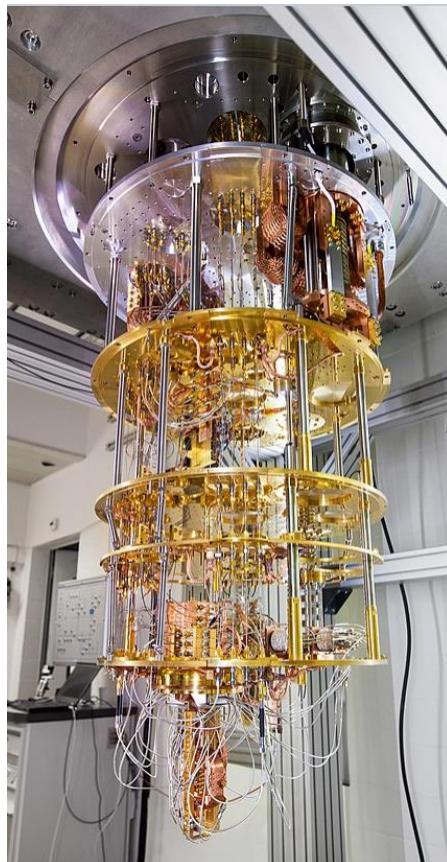
Quantum computing

Article [Talk](#)

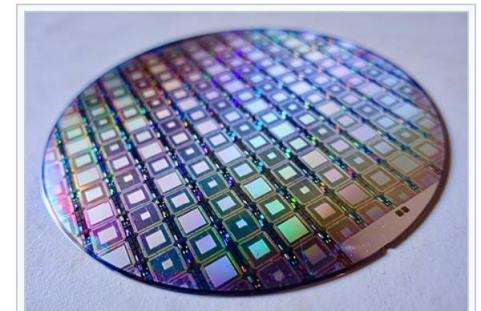
From Wikipedia, the free encyclopedia

A **quantum computer** is a [computer](#) that takes advantage of [quantum mechanical](#) phenomena.

At small scales, physical matter exhibits properties of [both particles and waves](#), and quantum computing leverages this behavior, specifically quantum [superposition](#) and [entanglement](#), using specialized hardware that supports the preparation and manipulation of [quantum states](#).

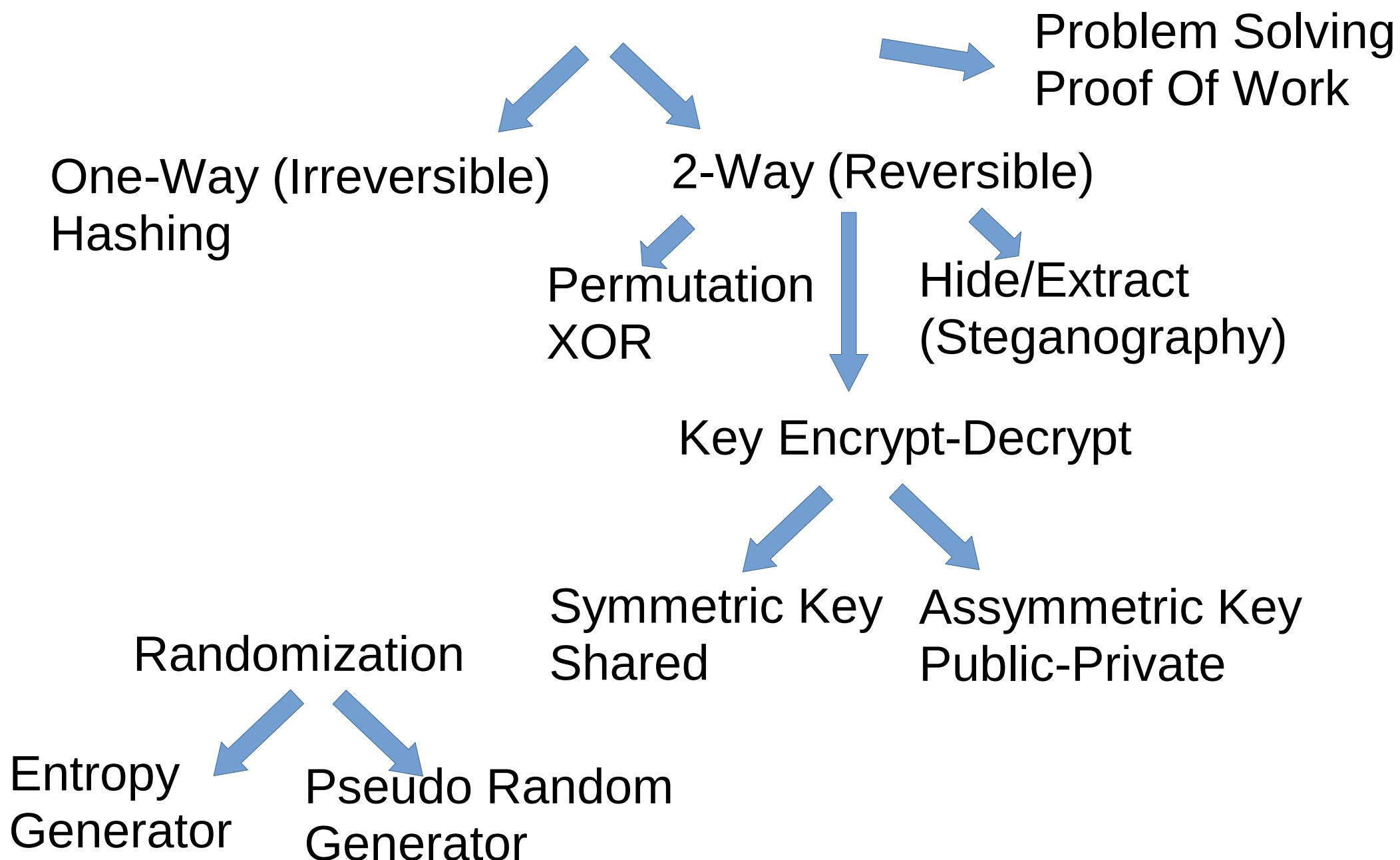


Peter Shor (pictured here in 2017) showed in 1994 that a scalable quantum computer would be able to break [RSA encryption](#). 🔗

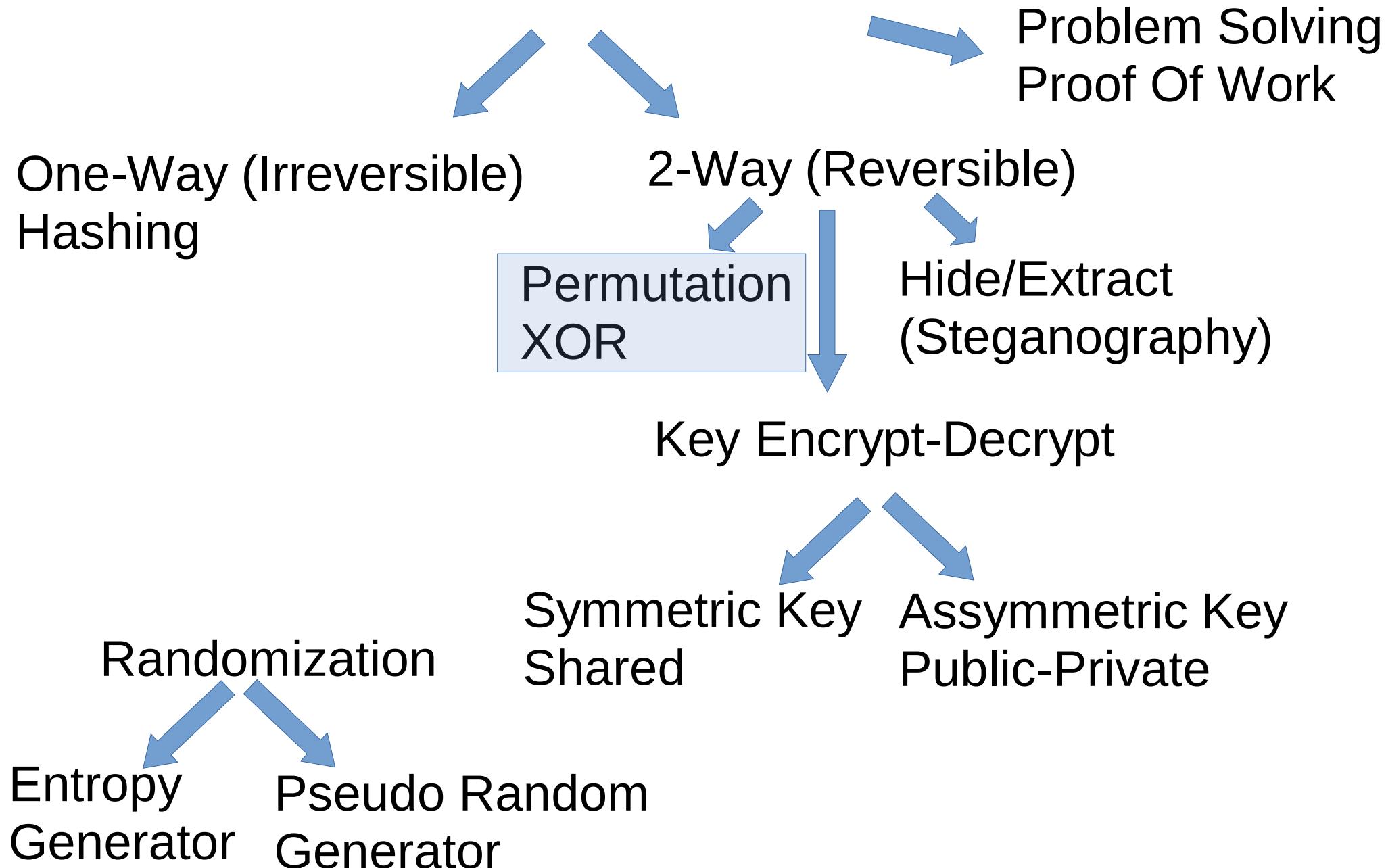


A [wafer of adiabatic quantum computers](#) 🔗

Family of Cryptographic Functions



Family of Cryptographic Functions



History Cypher

Antiquity [edit]

The earliest known use of cryptography is found in non-standard [hieroglyphs](#) carved into the wall of a tomb from the [Old Kingdom of Egypt](#) circa 1900 BC.^[1] These are not thought to be serious attempts at secret communications, however, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers.^[1] [failed verification]

Some [clay tablets](#) from Mesopotamia somewhat later are clearly meant to protect information—one dated near 1500 BC was found to encrypt a craftsman's recipe for pottery glaze, presumably commercially valuable.^[2] [better source needed]^[3] Furthermore, [Hebrew](#) scholars made use of simple monoalphabetic [substitution ciphers](#) (such as the [Atbash cipher](#)) beginning perhaps around 600 to 500 BC.^{[4][5]}

In India around 400 BC to 200 AD, [Mlecchita vikalpa](#) or "the art of understanding writing in cypher, and the writing of words in a peculiar way" was documented in the [Kama Sutra](#) for the purpose of communication between lovers. This was also likely a simple substitution cipher.^{[6][7]} Parts of the Egyptian [demotic Greek Magical Papyri](#) were written in a [cypher](#) script.^[8]



Scytale

文 A 24 languages ▾

Article Talk

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia

For other uses, see [Scytale \(disambiguation\)](#).

In [cryptography](#), a **scytale** (/ˈskɪtəli:/; also transliterated **skytale**, Ancient Greek: σκυτάλη *skutálē* "baton, [cylinder](#)", also σκύταλον *skútalon*) is a [tool](#) used to perform a [transposition cipher](#), consisting of a cylinder with a strip of parchment wound around it on which is written a message. The [ancient Greeks](#), and the [Spartans](#) in particular, are said to have used this cipher to communicate during [military](#) campaigns.

The recipient uses a rod of the same [diameter](#) on which the parchment is wrapped to read the message.



Julius Caesar Cypher



Caesar cipher

文 A 55 languages ▾

Article Talk

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia



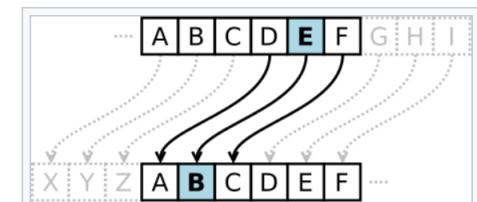
In [cryptography](#), a **Caesar cipher**, also known as **Caesar's cipher**, the **shift cipher**, **Caesar's code**, or **Caesar shift**, is one of the simplest and most widely known [encryption](#) techniques. It is a type of [substitution cipher](#) in which each letter in the [plaintext](#) is replaced by a letter some fixed number of positions down the [alphabet](#). For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after [Julius Caesar](#), who used it in his private correspondence.^[1]

The encryption step performed by a Caesar cipher is often incorporated as part of more complex schemes, such as the [Vigenère cipher](#), and still has modern application in the [ROT13](#) system. As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in modern practice offers essentially no [communications security](#).

Example [edit]

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places, equivalent to a right shift of 23 (the shift parameter is used as the [key](#)):

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



The action of a Caesar cipher is to replace each plaintext letter with a different one a fixed number of places down the alphabet. The cipher illustrated here uses a left shift of three, so that (for example) each occurrence of E in the plaintext becomes B in the ciphertext.

Caesar Java Code

```
public static String caesarShift(String data, int shift) {
    int len = data.length();
    StringBuilder res = new StringBuilder(len);
    for(int i = 0; i < len; i++) {
        char ch = data.charAt(i);
        if (ch == ' ') {
            res.append(' ');
            continue;
        }
        int chIndex = ch - 'a';
        int newIndex = (chIndex+shift) % 26;
        char newChar = (char) ('a' + newIndex);
        res.append(newChar);
    }
    return res.toString();
}
```

Caesar Cypher... why it is so weak ?

cfdbvtf pg gsfrvfodz bobmztjt gps dpnnpo mfuufs mjlf f

Frequency Analysis

```
int[] freq = new int[26];
for(int i = 0, len = cypherStr.length(); i < len;
i++) {
    char ch = cypherStr.charAt(i);
    if (ch == ' ') {
        continue;
    }
    int chIndex = ch - 'a';
    if (0 <= chIndex && chIndex < 26) {
        freq[chIndex]++;
    }
}
for(int i = 0; i < 26; i++) {
    System.out.print(i + ":" + freq[i] + " ");
}
```

most occurrence found = 8



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



this letter is probably "e"



it is the 6th position (index 5 starting at 0)
while "e" is the 5th letter of alphabet



suspecting shift by "1"

Too Few Possibilities (Brute Force), Leaking information (space not encrypted) Frequency Analysis, ...

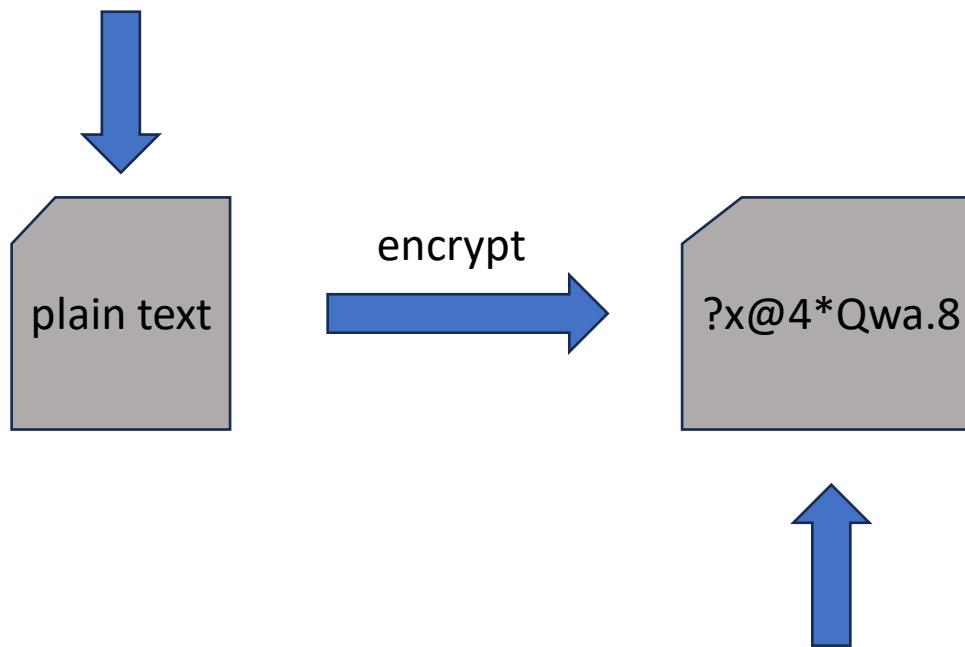
dgecwug qh htgswgpea cpcnauku hqt eqooqp ngvvgt nkmg g
ehfdxvh ri iuhtxhqfb dqdobv lv iru frpprq ohwwhu olnh h
figeywi sj jviuyirgc erepcwmw js v gsqqsr pixxiv pmoi i
gjhfzxj tk kwjvzjshd fsfqdxnx ktw htrrts qjyyjw qnpj j
hkigayk ul lxkwaktie gtgreyoy lux iussut rkzzkx roqk k
iljhbzl vm mylxblujf huhsfpz mv yjttvu slaaly sprl l
jm kicam wn nzmycmvkg ivitgaqa nwz kwuuuwv tmbbmz tqsm m
knljdbn xo oanzdnwlh jwjuhbrb oxa lxvxw unccna urtn n
lomkeco yp pboaeoxmi kxkvicsc pyb mywwyx voddob vsuo o
mpnlfdp zq qcpbfpynj lylwjtdt qzc nzxxzy wpeepc wtvp p
nqomgeq ar rdqcgqzok mzmxkeue rad oayyaz xqffqd xuwq q
orpnhfr bs serdhrapl nanylrvf sbe pbzzba yrggre yvrx r
psqoigs ct tfseisbqm obozmgwg tcf qcaacb zshhsf zwys s
qtrpjht du ugtfjtcrn pcpanhxh udg rd bbdct atiitg axzt t
rusqkiu ev vhugkudso qdqboiyi veh seced bujuh byau u
svtrljv fw wivhlvetp rercpjzj wfi tfddfe cvkkvi czbv v
twusmkw gx xjwimwfuq sfsdqkak xgj ugeegf dwllwj dacw w
uxvtlnx hy ykxjnxgvr tgterlbl yhk vhffhg exmmxk eb dx x
vywuomy iz zlykoyhws uhufsmcm zil wiggih fynnyl fc ey y
wzxvpnz ja amzlpzixt vivgtndn ajm xjhhji gzoозm gdfz z
xaywqoa kb bnamqajyu wjwhuoeo bkn ykiikj happen hega a
ybzxrbp lc cobnrbkzv xkxivpf clo zljjlk ibqqbo ifhb b
zcaysqc md dpcosclaw ylyjwqgq dmp amkkml jcrrcp jgic c
adbzrd ne eqdptdmbx zmzkrhr enq bnllnm kdssdq khjd d
because of frequency analysis for common letter like e

```
for (int shift = 1; shift < 26; shift++) {  
    System.out.println(caesarShift(cypherStr, shift));  
}
```

Signal Theory ...

Information / Entropy / Randomness

This signal has many redundant information
(constrained by grammar, known dictionary words, logic..)

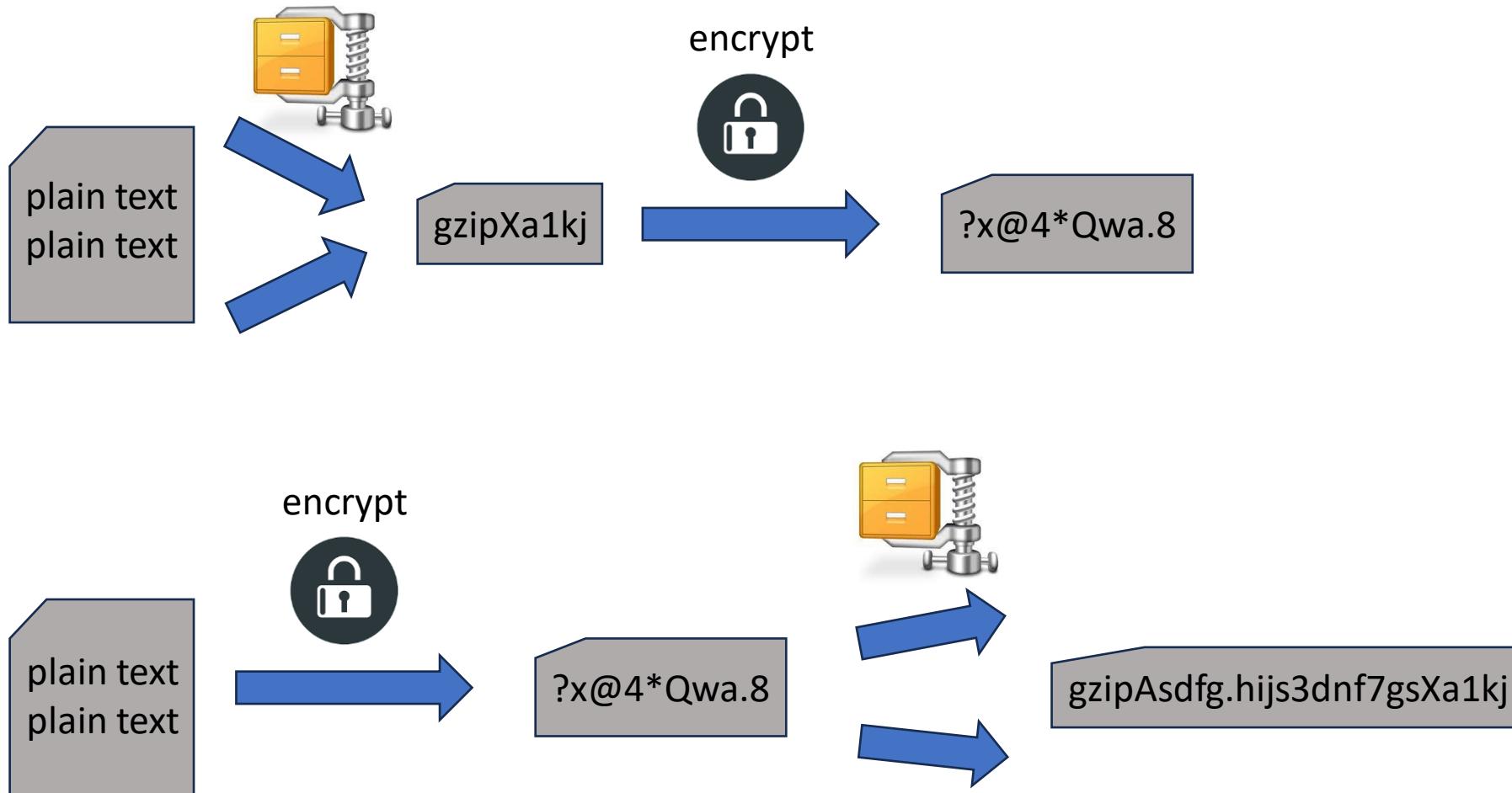


This signal should appear completely random
with no repetition, no "information"

Compress Then Encrypt

... not

Encrypt Then Compress



XOR ...

VERY VERY Fast

(might) works with “Good” Pseudo Random Generators



A

You

what about using xor and speudo random generator for cryptographic encrypting ?



ChatGPT

Using XOR with a pseudo-random generator (PRG) for cryptographic encryption can be a part of a cryptographic scheme, but it's not typically used on its own due to security concerns.

XOR is a bitwise operation that can be reversible, which makes it suitable for encryption purposes. However, its security relies heavily on the quality of the pseudo-random generator. If the PRG is not truly random or if its output can be predicted or easily guessed, then the encryption scheme becomes vulnerable to attacks.

PRGs are designed to generate sequences of numbers that appear random. However, they are deterministic algorithms, meaning that if you know the initial state (the seed), you can predict all subsequent outputs. Therefore, using a secure and properly designed PRG is crucial for the security of the encryption scheme.

In modern cryptography, XOR may be used as a component within more complex cryptographic algorithms, such as stream ciphers or block ciphers. These algorithms typically incorporate additional security features and undergo rigorous analysis to ensure their resistance against known cryptographic attacks.

In summary, while XOR with a PRG can be a component of a cryptographic encryption scheme, it's important to use it in conjunction with other cryptographic techniques and ensure that the PRG used is secure and properly implemented.



Problem

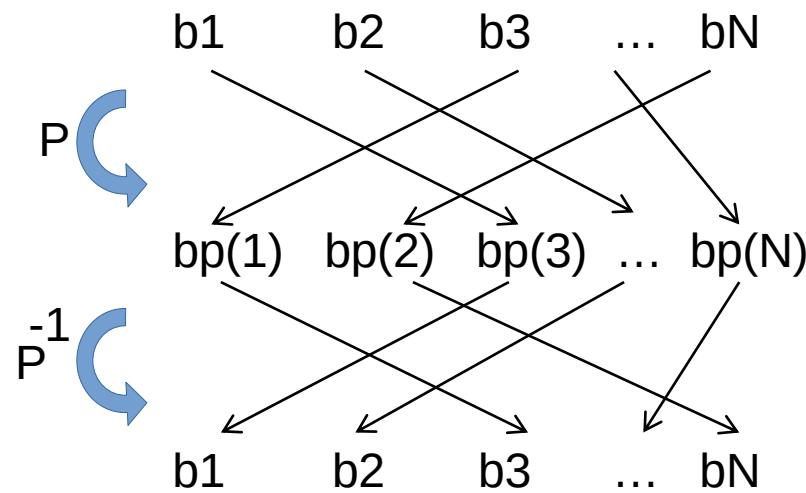
writing BIG predictable fragments of 000000...000

=> output leaks internal information of the state (=Key)

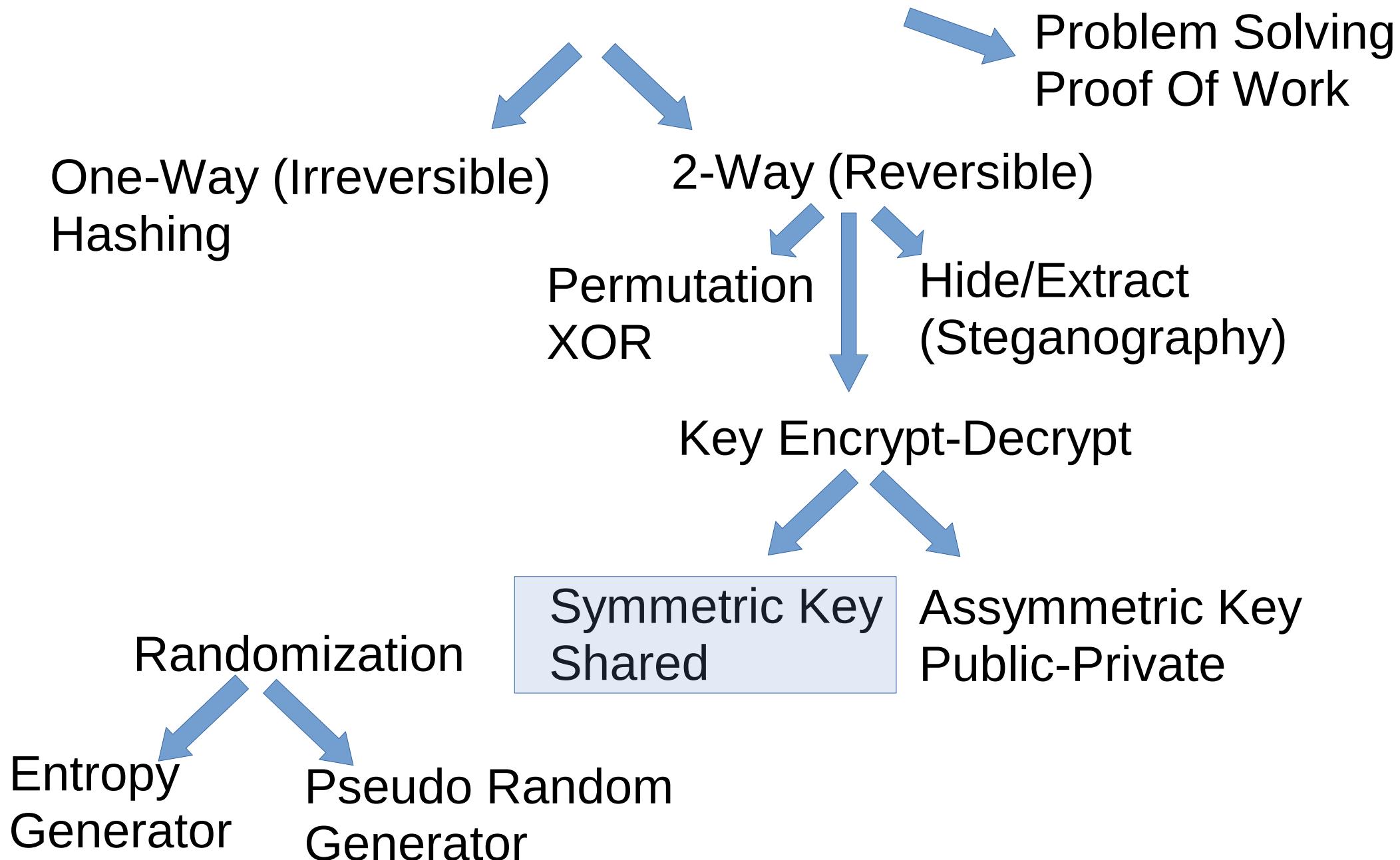
Permutations

$P: [1..N] \rightarrow [1..N]$

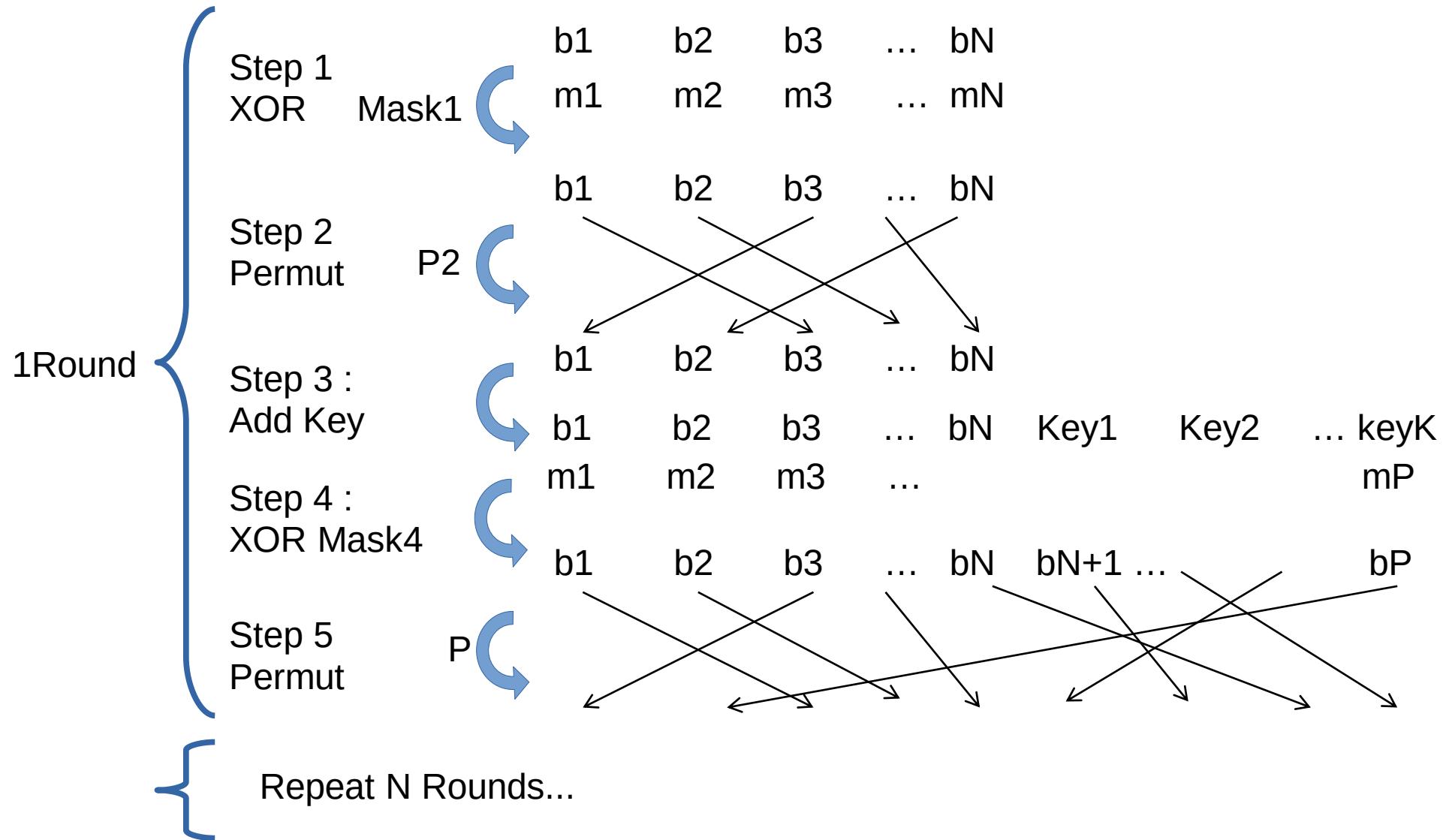
P is a permutation if it is invertible (all elts are reached once)



Family of Cryptographic Functions



Combining Several Permutations (and XOR & Non-Linear & ...)



Example : 3-DES

Wikipedia article as of 2017

Not So LONG Ago...

Article Talk

Read Edit View histo

Triple DES

From Wikipedia, the free encyclopedia
(Redirected from [3-DES](#))

In [cryptography](#), **Triple DES (3DES)**, officially the **Triple Data Encryption Algorithm (TDEA or Triple DEA)**, is a [symmetric-key block cipher](#), which applies the [Data Encryption Standard \(DES\)](#) cipher algorithm three times to each data block.

The original DES cipher's [key size](#) of 56 bits was generally sufficient when that algorithm was designed, but the availability of increasing computational power made [brute-force attacks](#) feasible. Triple DES provides a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm.

3-DES

Wikipedia page updated 2024

Data Encryption Standard

⋮ A 50 languages ▾

Article Talk

Read Edit View history Tools ▾

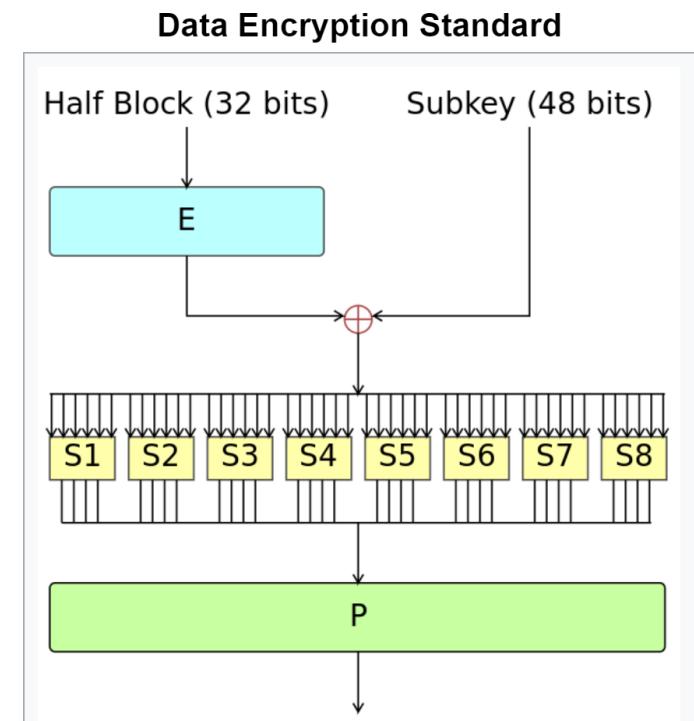
From Wikipedia, the free encyclopedia

The Data Encryption Standard (DES /di:^ɛi:es, dez/) is a symmetric-key algorithm for the encryption of digital data. Although its short key length of 56 bits makes it **too insecure** for modern applications, it has been highly influential in the advancement of [cryptography](#).

Developed in the early 1970s at [IBM](#) and based on an earlier design by [Horst Feistel](#), the algorithm was submitted to the [National Bureau of Standards](#) (NBS) following the agency's invitation to propose a candidate for the protection of sensitive, unclassified electronic government data. In 1976, after consultation with the [National Security Agency](#) (NSA), the NBS selected a slightly modified version (strengthened against [differential cryptanalysis](#), but weakened against [brute-force attacks](#)), which was published as an official [Federal Information Processing Standard](#) (FIPS) for the United States in 1977.^[2]

The publication of an NSA-approved encryption standard **led to its quick international adoption** and widespread academic scrutiny. Controversies arose from [classified](#) design elements, a relatively short [key length](#) of the [symmetric-key block cipher](#) design, and the involvement of the NSA, raising **suspicions about a backdoor**. The [S-boxes](#) that had prompted those suspicions were designed by the NSA to remove a backdoor they secretly knew ([differential cryptanalysis](#)). However, the NSA also ensured that the key size was drastically reduced **so that they could break the cipher** by brute force attack.^[2] The intense academic scrutiny the algorithm received over time led to the modern understanding of block ciphers and their [cryptanalysis](#).

DES is insecure due to the relatively short [56-bit key size](#). In January 1999, [distributed.net](#) and the [Electronic Frontier Foundation](#) collaborated to publicly break a DES key in 22 hours and 15 minutes



General	
Designers	IBM
First published	1975 (Federal Register) (standardized in January 1977)
Derived from	Lucifer

as of 2024 ... AES-256 = established standard

Advanced Encryption Standard

文 A 49 languages ▾

Article Talk

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia

(Redirected from [Aes256](#))

The **Advanced Encryption Standard (AES)**, also known by its original name **Rijndael** (Dutch pronunciation: [rɛində:l]),^[5] is a specification for the [encryption](#) of electronic data established by the U.S. [National Institute of Standards and Technology](#) (NIST) in 2001.^[6]

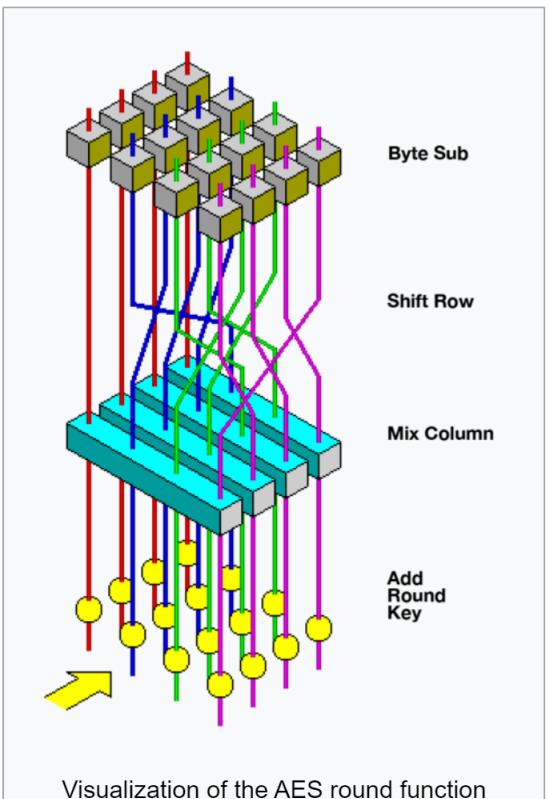
AES is a variant of the Rijndael [block cipher](#)^[5] developed by two [Belgian](#) cryptographers, [Joan Daemen](#) and [Vincent Rijmen](#), who submitted a proposal^[7] to NIST during the [AES selection process](#).^[8] Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

AES has been adopted by the [U.S. government](#). It supersedes the [Data Encryption Standard \(DES\)](#),^[9] which was published in 1977. The algorithm described by AES is a [symmetric-key algorithm](#), meaning the same key is used for both encrypting and decrypting the data.

In the United States, AES was announced by the NIST as U.S. [FIPS PUB 197](#) (FIPS 197) on November 26, 2001.^[6] This announcement followed a five-year standardization process in which fifteen competing designs were presented and evaluated, before the Rijndael cipher was selected as the most suitable.^[note 3]

AES is included in the [ISO/IEC 18033-3](#) standard. AES became effective as a U.S. federal government standard on May 26, 2002, after approval by U.S. [Secretary of Commerce Donald Evans](#). AES is available in many different encryption packages, and is the first (and only) publicly accessible [cipher](#) approved by the U.S. [National Security Agency](#) (NSA) for [top secret](#) information when used in an NSA approved cryptographic module.^[note 4]

Advanced Encryption Standard (Rijndael)

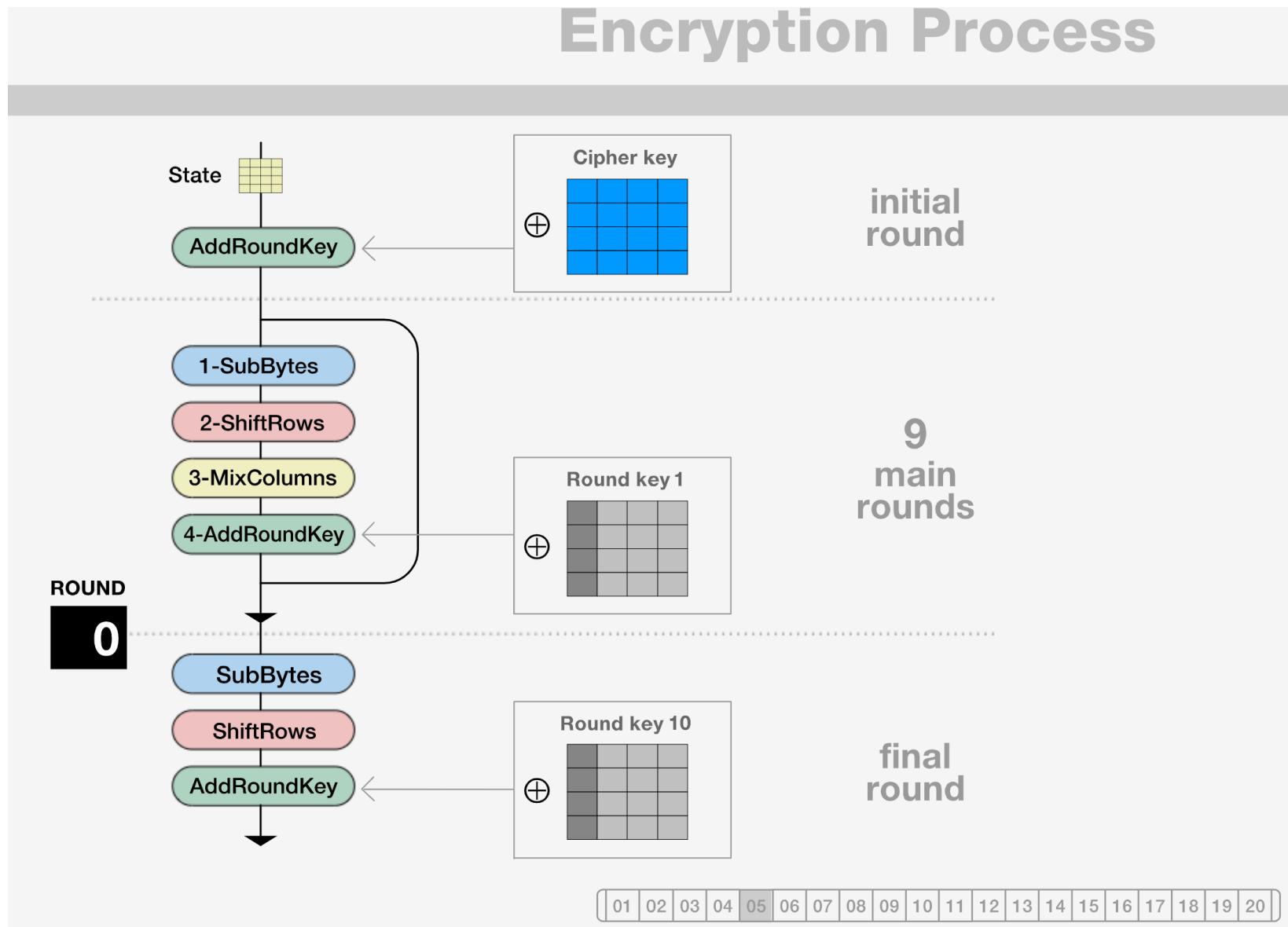


Visualization of the AES round function

AES - Animation Video

- [Animation of Rijndael](#) – AES deeply explained and animated using Flash (by Enrique Vidal) – this animation (in English, Spanish, and German) is also part of [CrypTool 1](#) (menu Indiv.).
- [HTML5 Animation of Rijndael](#) – Same Animation as above made in HTML5.

https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html



AES : S-BOX [step 7/20]

1 - SubBytes

Round 1

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

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

S-BOX byte substitution table

AES : Rotate ShiftRows [step 8/20]

2 - ShiftRows

Round 1

d4	e0	b8	1e
bf	f1	1	27
5d	f1	1	98
e5	30	ae	



..... rotate over 3 bytes

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

AES : MixColumn [step 9/20]

3 - MixColumns

e0	b8	1e
b4	41	27
52	11	98
ae	f1	e5

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ e5 \end{bmatrix}$$

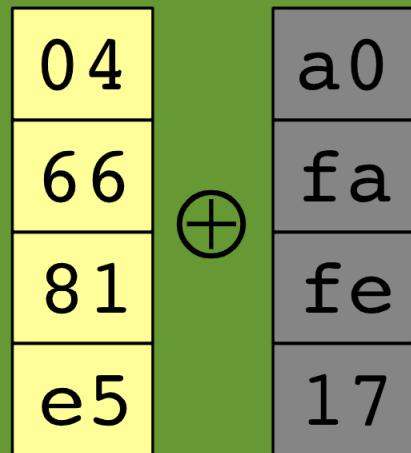
The four numbers of one column
are modulo multiplied in Rijndael's
Galois Field by a given matrix.

Round 1

AES : XOR AddRoundKey [step 10/20]

4 - AddRoundKey

e0	48	28
cb	f8	06
19	d3	26
9a	7a	4c



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)

AES optimized instructions in Intel,Amd processors

x86 architecture processors [edit]

AES-NI (or the Intel **Advanced Encryption Standard New Instructions**; **AES-NI**) was the first major implementation instruction set architecture for [microprocessors](#) from [Intel](#) and [AMD](#) proposed by Intel in March 2008.^[1]

A wider version of AES-NI, AVX-512 *Vector AES instructions* (VAES), is found in [AVX-512](#).^[2]

Instructions [edit]

Instruction	Description ^[3]
AESENCLAST	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation ^[note 1]
AESIMC	Assist in AES decryption round key generation. Applies Inverse Mix Columns to round keys.

AES performances

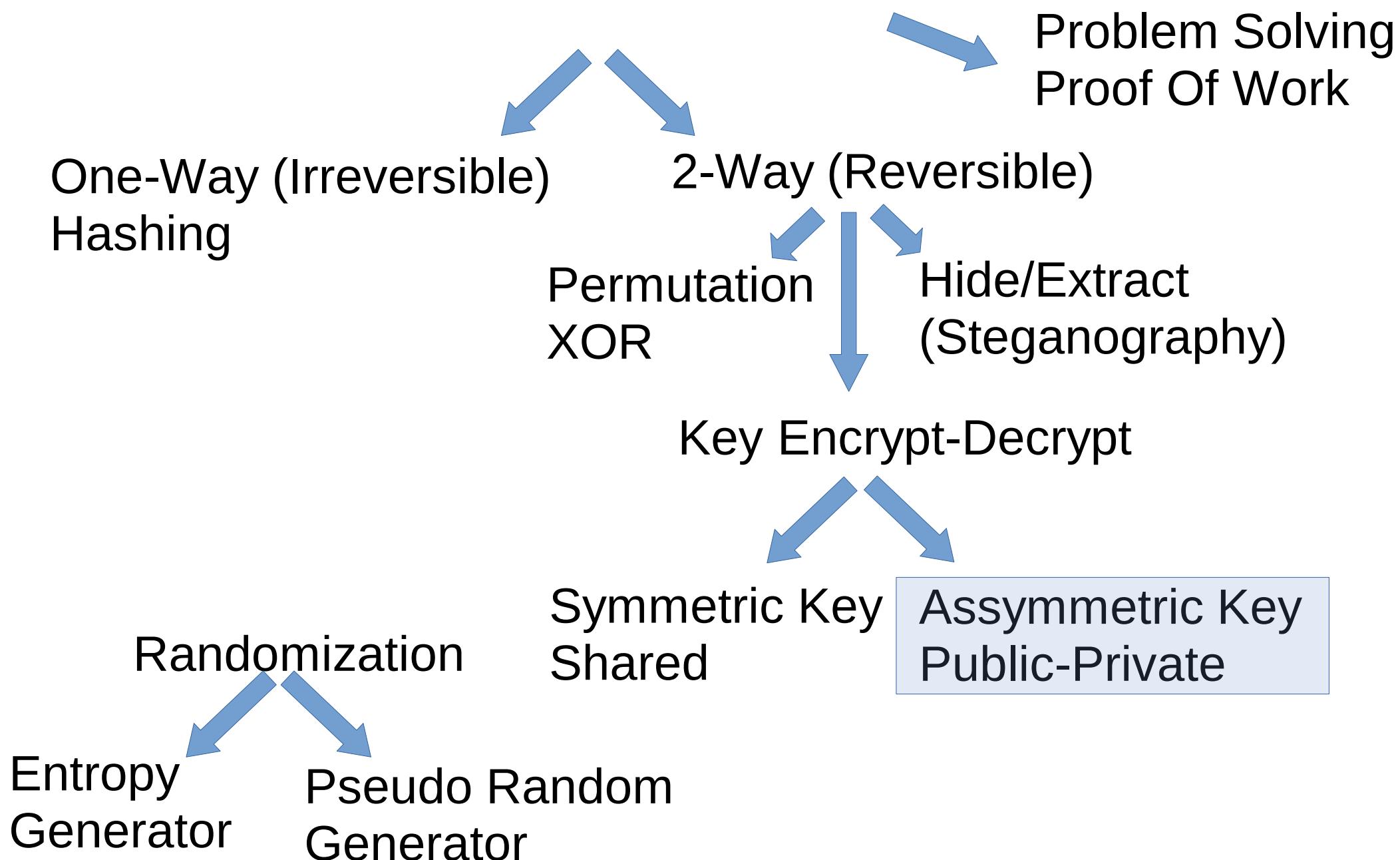
Performance [edit]

In *AES-NI Performance Analyzed*, Patrick Schmid and Achim Roos found "impressive results from a handful of applications already optimized to take advantage of Intel's AES-NI capability".^[33] A performance analysis using the [Crypto++ security library](#) showed an increase in throughput from approximately 28.0 cycles per byte to 3.5 cycles per byte with [AES/GCM](#) versus a [Pentium 4](#) with no acceleration.^{[34][35][failed verification][better source needed]}

Software => 28 cycles per byte

Hardware => 3.5 cycles per byte (x9 faster)

Family of Cryptographic Functions



Assymmetric Keys

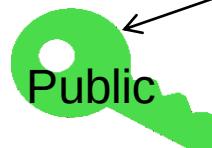


Only “Priv” can encode
 Private

msg



encrypted

 Public

Everybody can decode (= c)

msg

Every body can encode
 Public

msg



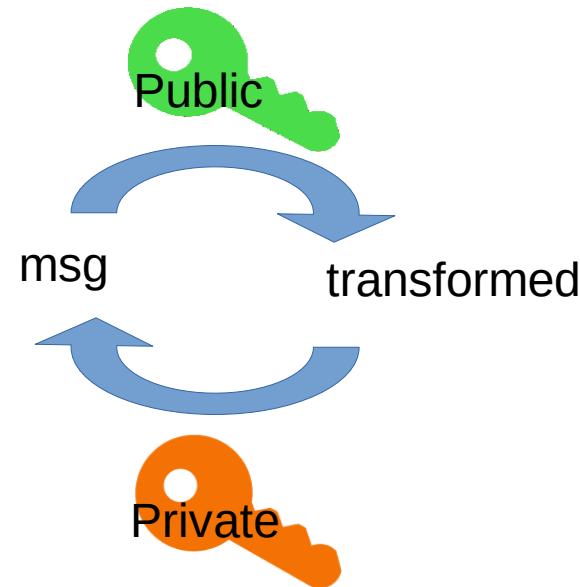
encrypted

 Private

Only “Priv”
can decode

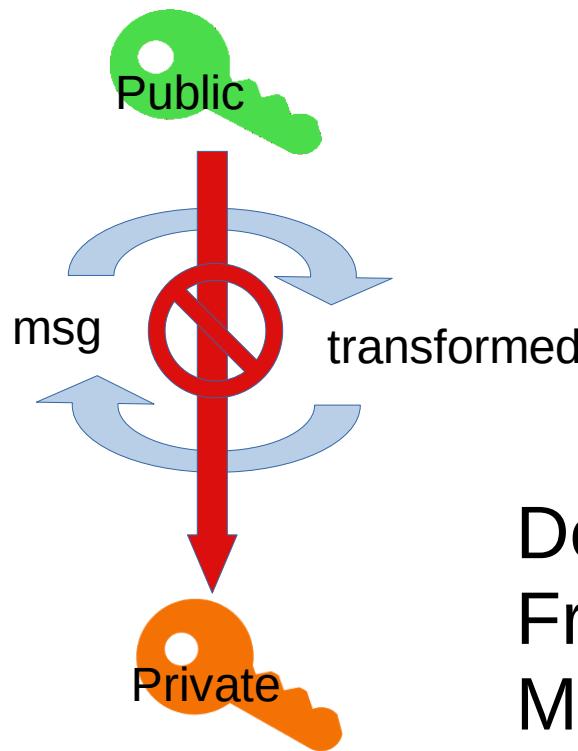
msg

Public Keys – Private Keys...



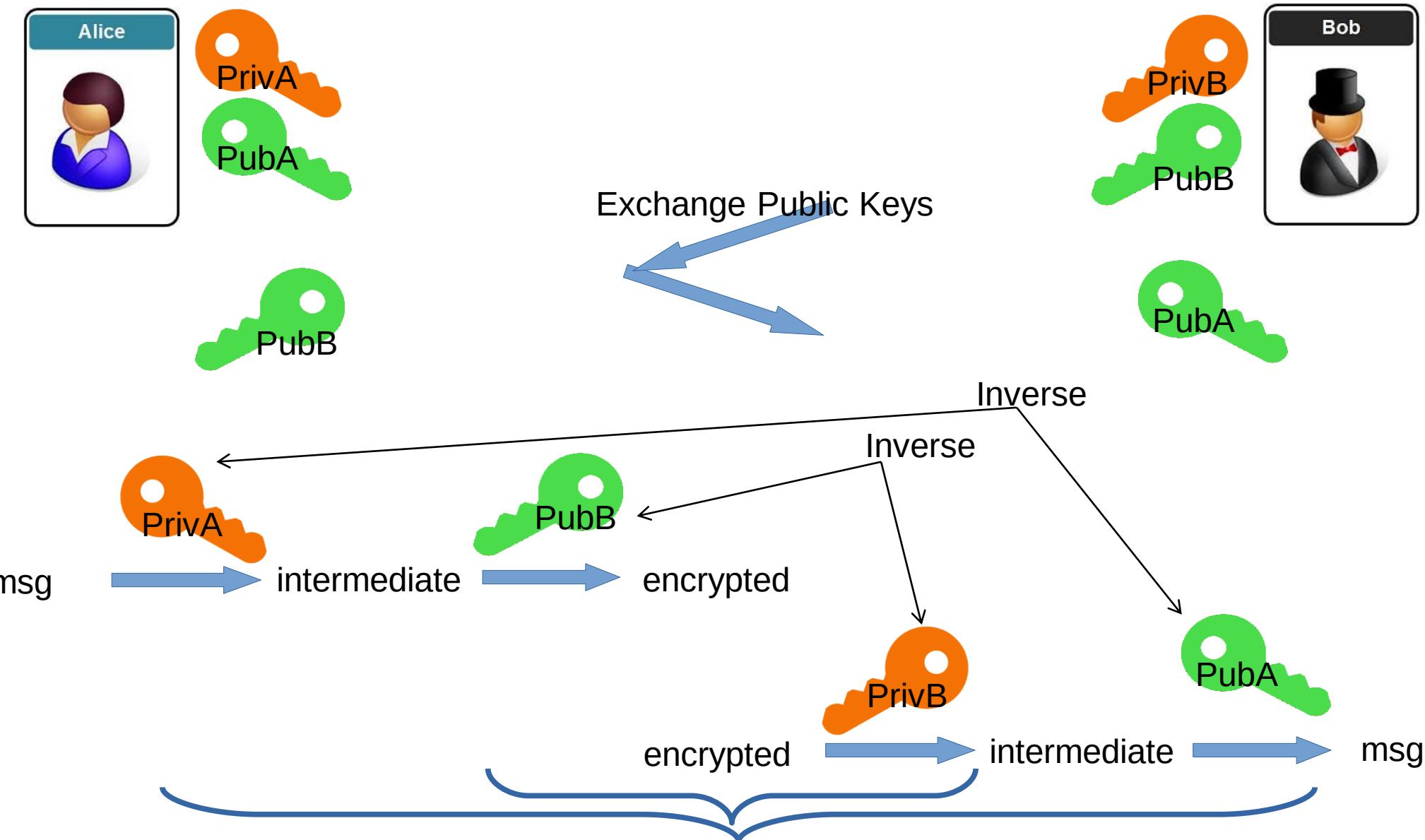
Keys define 2 operations : inverse of each others
Like $+X \rightarrow -X$
 $*X \rightarrow /X$
 $^N \rightarrow \sqrt{^N}$

But Deducing Keys MUST Be Impossible/Difficult

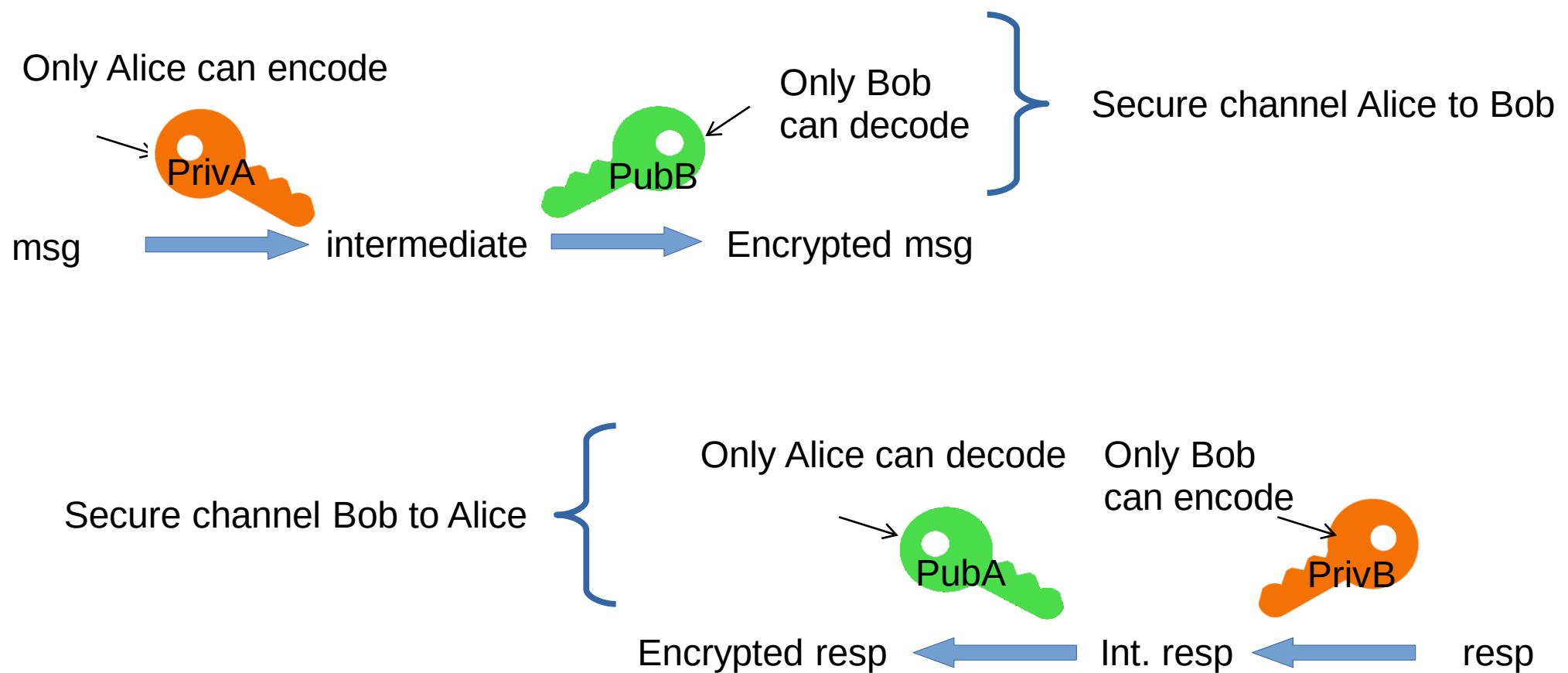


Deducing Private Key
From Public Key
MUST Be Impossible

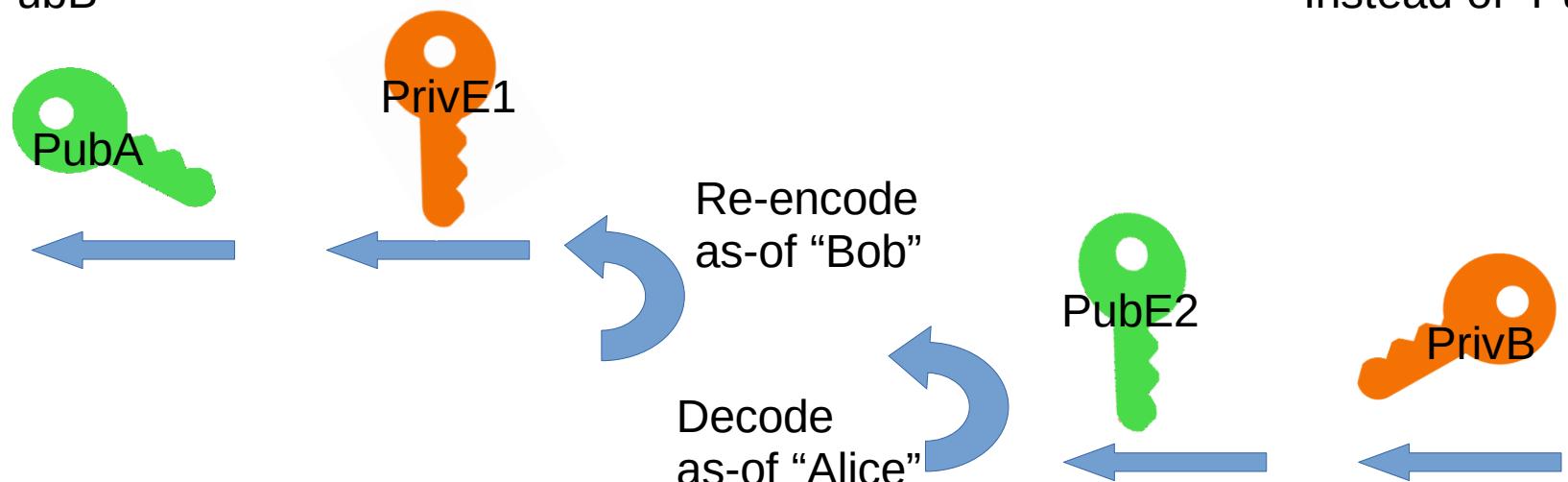
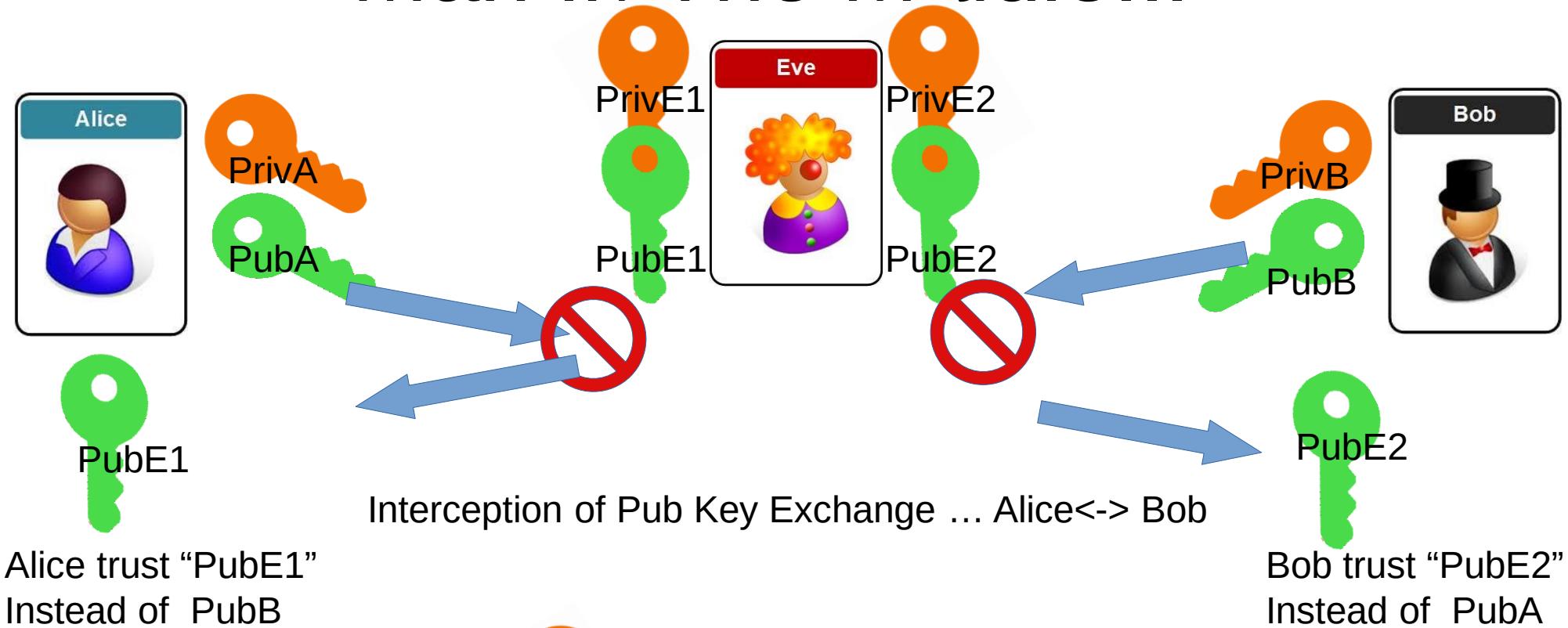
Combining 2 Pairs of Pub-Priv Keys



Assymmetric Keys For Secured 2-Way Channel

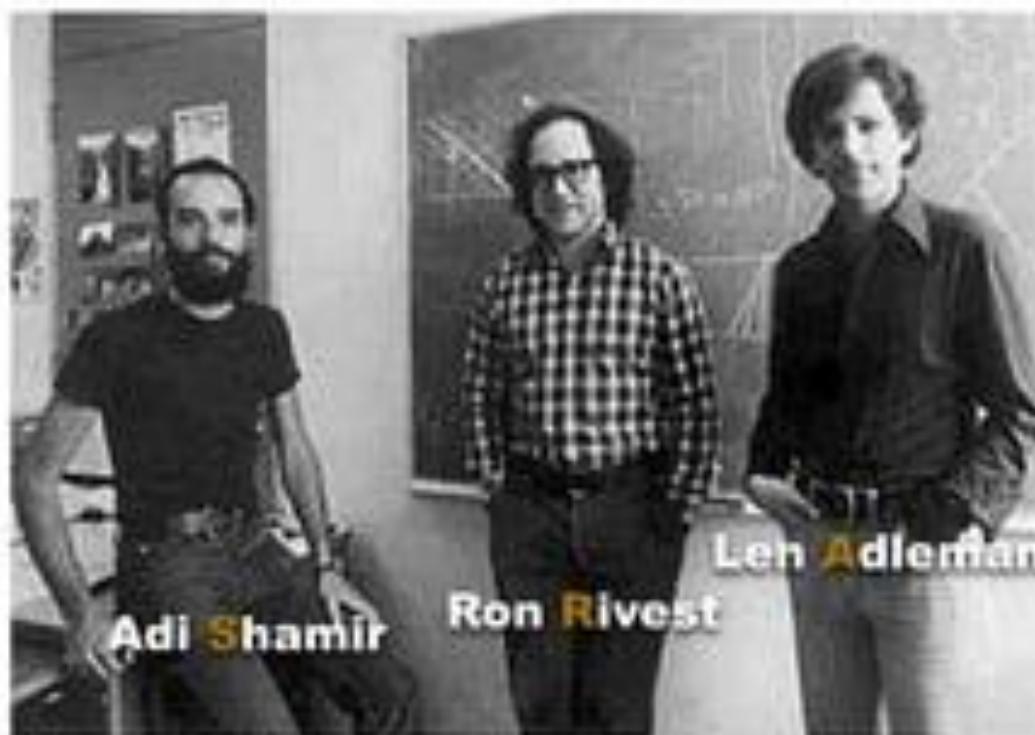


Man In The Middle...



RSA : Rivest Shamir Adleman

The Founders



RSA :first practical, widely used

Article [Talk](#)

Read [Edit](#) [Vi](#)

RSA (cryptosystem)

From Wikipedia, the free encyclopedia

For other uses, see [RSA \(disambiguation\)](#).

RSA is one of the first practical [public-key cryptosystems](#) and is widely used for secure data transmission. In such a [cryptosystem](#), the [encryption key](#) is public and differs from the [decryption key](#) which is kept secret. In RSA, this asymmetry is based on the practical difficulty of [factoring](#) the product of two large [prime numbers](#), the [factoring problem](#). RSA is made of the initial letters of the surnames of [Ron Rivest, Adi Shamir, and Leonard Adleman](#), who first publicly described the algorithm in 1978. [Clifford Cocks](#), an English mathematician working

Operation [edit]

The RSA algorithm involves four steps: [key](#) generation, key distribution, encryption, and decryption.

A basic principle behind RSA is the observation that it is practical to find three very large positive integers e , d , and n , such that with [modular exponentiation](#) for all integers m (with $0 \leq m < n$):

$$(m^e)^d \equiv m \pmod{n}$$

and that knowing e and n , it can be extremely difficult to find d . Here the symbol \equiv denotes [modular congruence](#): i.e. both $(m^e)^d$ and m have the same [remainder](#) when divided by n .

In addition, for some operations it is convenient that the order of the two exponentiations can be changed: the previous relation also implies

$$(m^d)^e \equiv m \pmod{n}.$$

Basic Math: Modulo definition

Remember Euclidian Division

Let N a number > 0

Every number divided by N
has a quotient and a remainder – also called modulo

x / n : for all $x \Rightarrow$ there exist(uniq) q & r with $0 \leq r < n$
 $x = q n + r$

Modulo N function: $x \rightarrow x \bmod[n] = r$

Algebra Operation on Modulos

$$(x + y) \bmod [n] = (x \bmod [n] + y \bmod [n]) \bmod [n]$$

$$(x * y) \bmod [n] = (x \bmod [n] * y \bmod [n]) \bmod [n]$$

Reasonning on equivalence classes, for equivalence relation “R”: $a R b \Leftrightarrow a - b \bmod [n] = 0$

$$\mathbb{Z}/n\mathbb{Z} = \{\dot{0}, \dot{1}, \dot{2}, \dots, \dot{N-1}\}$$

$$\overline{\dot{x} + \dot{y}} = \dot{x} + \dot{y}$$

$$\overline{\dot{x} \cdot \dot{y}} = \dot{x} \cdot \dot{y}$$

$$\overline{\dot{x}^y} = \dot{x}^y$$

Example on modulo 9

Doing modulo 9 as a 9-year old boy at school
... sum digits to check operations

$$\begin{aligned}123 \times 456 \bmod[9] &= 56088 \bmod[9] = (5 + 6 + 0 + 8 + 8) \bmod[9] = 0 \\&= (1 + 2 + 3) \times (4 + 5 + 6) \bmod[9] = 6 \times 6 \bmod[9] = 0\end{aligned}$$

This works because $10 \bmod[9]=1$.. $100 \bmod[9]=1$, $1000 \bmod[9]=1$
so whatever X written in decimal format:

$$(a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_2 10^2 + a_1 10 + a_0) \bmod[9] = (a_n + \dots + a_1 + a_0) \bmod[9]$$

When p is prime, $\mathbb{Z}/p\mathbb{Z}^*$ is a multiplicative Group

Remember Group definition ?

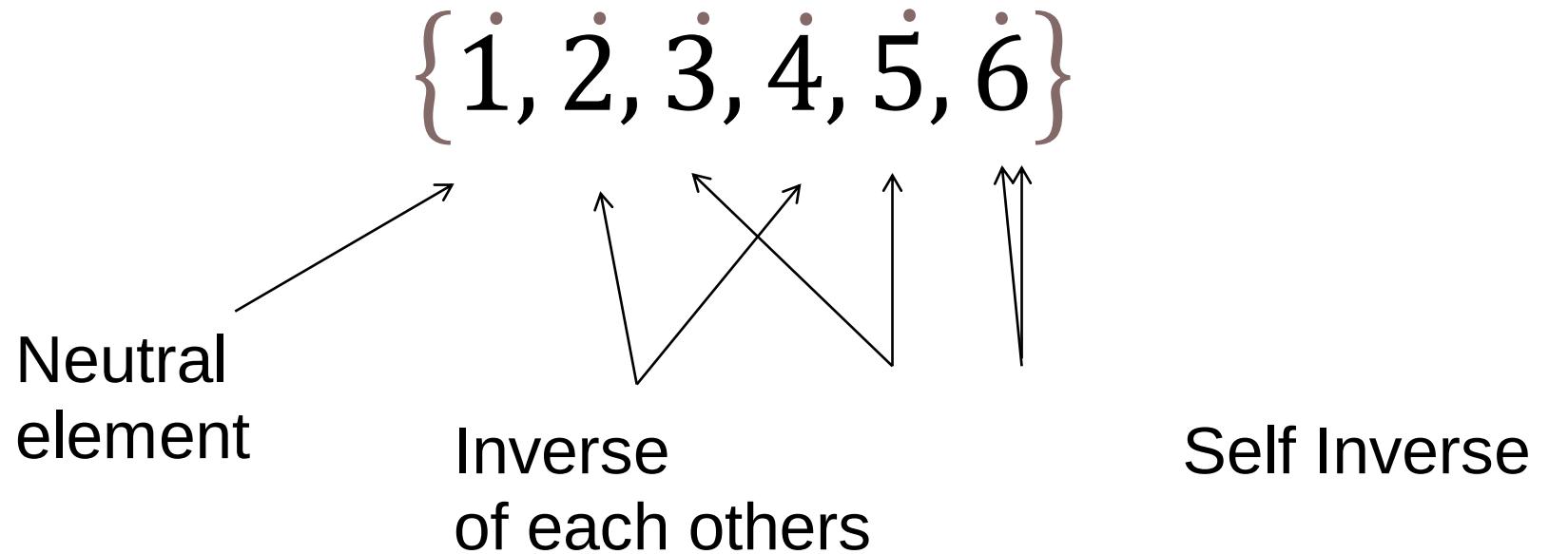
$G = \{ g_0, g_1, \dots, g_N \}$ a Set of elements
with an operation “.” (inside G)
Is a “group over .”

If it exists a neutral element “ e ” $\forall a, e \cdot a = a \cdot e = e$

every element has an inverse: $\forall a, \exists b \mid a \cdot b = b \cdot a = e$

(b is unique and is the inverse of a : $b = a^{-1}$)

Example $\mathbb{Z}/7\mathbb{Z}^*$



$$\dot{2} \cdot \dot{4} = \dot{8} = \dot{1}$$

$$\dot{3} \cdot \dot{5} = \dot{15} = \dot{1}$$

$$\dot{6} \cdot \dot{6} = \dot{36} = \dot{1}$$

Consecutive Powers ... (=Orbits)

$$a \in \mathbb{Z}/p\mathbb{Z}^*, p \text{ prime}$$

$\{a, a^2, a^3, a^4, \dots, a^{p-1}, \dots\}$ Is inside finite set..
so is cyclic ...

let k2 index be the first repetition of already seen k1 elt:

$$a^{k2} = a^{k1}$$

Taking inverse $a^{k2-k1} = e$

then proof next that $k2-k1=p-1$

Orbits are all similar... all equals

whatever a, the orbit contains e... so orbits contains

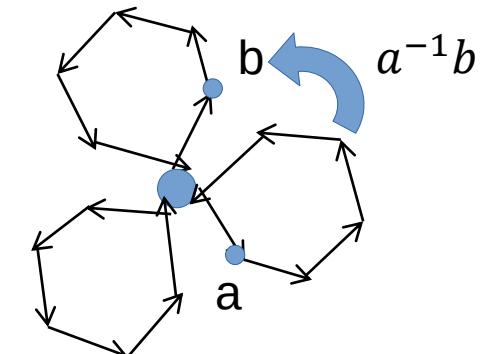
$$\{a, a^2, \dots, a^{p-2}, e\}$$

For 2 elements a & b ... orbits of a and b are identical, “scaled”
By a factor

$$a^{-1}b$$

So For any orbit, card Orbit = cst

If they are N orbits, they all cover the set $\mathbb{Z}/p\mathbb{Z}^*$



If p is a prime number ... (sub-orbit N can not divide P prime)

... There is only 1 orbit, containing ALL elements !

Example $\mathbb{Z}/7\mathbb{Z}^*$

Start for example with $a = 3$

Compute $a, a^2 \dots$

$$a = \dot{3}$$

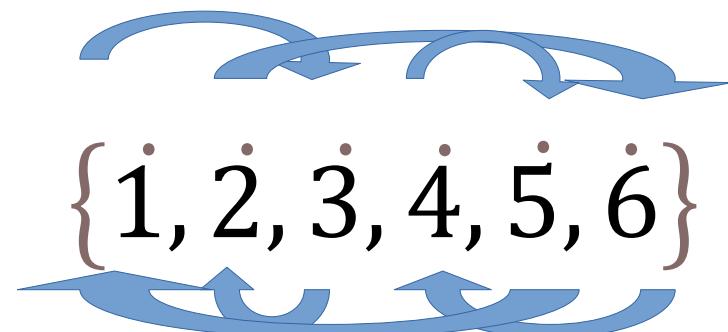
$$a^2 = \dot{3}\dot{3} = \dot{9} = \dot{2}$$

$$a^3 = \overline{\dot{3}\cdot\dot{2}} = \dot{6}$$

$$a^4 = \overline{\dot{3}\cdot\dot{6}} = \dot{4}$$

$$a^5 = \overline{\dot{3}\cdot\dot{4}} = \dot{5}$$

$$a^6 = \overline{\dot{3}\cdot\dot{5}} = \dot{1}$$



“Small Theorem” of Fermat

$\forall a \in \mathbb{Z}/p\mathbb{Z}^*, p \text{ prime}$

$$a^{p-1} = 1 \bmod[p]$$

$$a^p = a \bmod[p]$$

Relationship with Crypto? ...

choose 2 primes

$$k < n \quad n = k + (n-k)$$

$$a \mid\rightarrow a^k = b \qquad \qquad b \Rightarrow b^{n-k} = a$$



encrypt



decrypt

$$= (a^k)^{n-k} = a^{k+(n-k)} = a \bmod p$$

Making Cryptography out of 2 primes ?

For p_1, p_2 2 prime numbers...

$$a^{p_1} = a \bmod [p_1]$$

$$a^{p_2} = a \bmod [p_2]$$

$$a^{p_1.p_2} = (a^{p_1})^{p_2} = a \bmod [p_1p_2]$$

Intuitively ... Try decompose $p_1.p_2$ as a different product of 2 key parts “ $e.d \bmod[...]$ ” ... but not as $p_1.p_2$ (too easy) !

RSA ...

Technically.. solve e and d...

$$e \cdot d = 1 \bmod [\phi(p_1 p_2)]$$

Then

$$(a^e)^d = a \bmod [p_1 p_2]$$

And also

$$(a^d)^e = a \bmod [p_1 p_2]$$

Inverting Key

.. need Decomposing N in p1.p2

Choose 2 HUGE primes p_1, p_2

Multiply $N = p_1 \cdot p_2$

... give N to someone

and reward 1M\$ IF he finds back p_1 & p_2

Naive Search...

Given N

Try decompose :

```
for ( BigInteger i = N; ; i = i-2 ) {  
    if (divides(i, N)) {  
        p1 = i; break;  
    }  
}
```

How HUGE is HUGE enough?

Example: choose $p_1 > 2^{1024}$

= 2.2.2. (1024 times)
~ 1000 . 1000 (1014 times)
~ 1 000 000 000 ... (3000 zeros)

In Theory : Decomposition Program finishes

In Practise :

- More than Atoms in Universe
- Slow... 1CPU => 1000...00000 centuries
- Energy : 1000...000 CPUS => requires more power
than 1000...000 solars

Possible to Find so HUGE Primes?

In Theory : infinite numbers of Primes

Storing all primes from $[0, 2^{1024}]$??
... is even harder than counting them

Primes become rare : Count primes in $[0, N] \sim \log N$

There always exist a prime in $[k, 2k]$

Find One Random (!) Huge Prime Efficiently (?)

Given **start** = random number $> 2^N-1$... with $N \sim 1024$

```
for ( BigInteger i = start; ; i = i+2 ) {  
    if (isPrime(i)) {  
        p = i; break;  
    }  
}
```

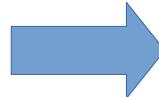
Need Efficient Primality Test

$p \text{ is prime} \Leftrightarrow$

$$\forall a \in \mathbb{Z}/p\mathbb{Z}^*$$

$$a^{p-1} = 1 \bmod[p]$$

Choose any a , for example $a=27$ Compute $a^{p-1}[p]$

If $\neq 1$  p is NOT prime

If $= 1$  Probability $1/p$ that p is anyway not prime

Reapeat for $b=43$  Confirm .. or proba $1/p^2$

Efficient Compute Powers..

Write in basis-2 (binary representation)

$$p - 1 = b_0 2^0 + b_1 2^1 + \dots b_n 2^n \quad \forall i, b_i \in \{0,1\}$$

Then

$$a^{p-1} = a^{b_0} \cdot a^{2b_1} \cdot \dots \cdot a^{2^n b_n}$$

$$= \begin{cases} a^1 & \text{if } b_1 = 1 \\ 1 & \text{if } b_1 = 0 \end{cases} \cdot \begin{cases} a^2 & \text{if } b_2 = 1 \\ 1 & \text{if } b_2 = 0 \end{cases} \cdot \dots \cdot \begin{cases} a^{2^n} & \text{if } b_n = 1 \\ 1 & \text{if } b_n = 0 \end{cases}$$



Efficient (Enough?) Powers Compute

$$P \sim 2^{1024} \quad N = \log_2(p) = 1024$$

a^{p-1}

Computable in $O(2 \cdot N)$ multiplications
BUT Would be a HUGE number ...

$a^{p-1}[p]$

Computable in
 $O(2 \cdot N)$ multiplications + $1 \cdot N$ modulo
... quite FAST, with small memory need

Not Efficient Enough For Video & High Traffic Network !!!

But **OK for small DATA**

~ 1ms on a CPU

Embeddable in RDA SecurID card ship



OK For Exchanging a symmetric Key
... then using AES-256 cypher

Application : **TLS** protocol (previous SSL), for example



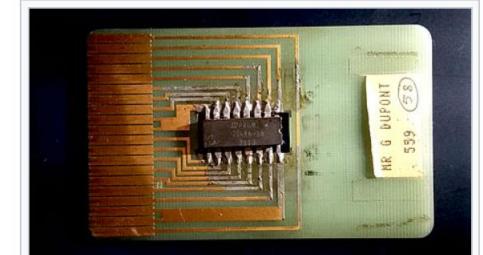
SmartCard ... Not only "memory", but also RSA "Compute" (auto-block after 3 PIN attempts)

Smart card

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

A **smart card (SC)**, **chip card**, or **integrated circuit card (ICC or IC card)** , used to control access to a resource. It is typically a plastic credit card-sized card with an embedded integrated circuit (IC) chip.^[1] Many



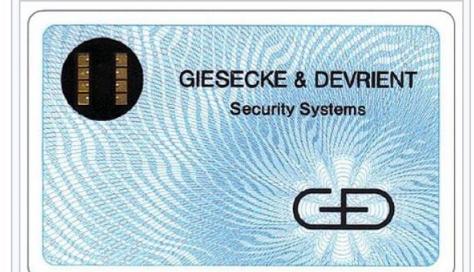
One of the first smart card prototypes, created by its inventor [Roland Moreno](#) around 1975. The chip has not yet been miniaturized. On this prototype, one can see how each pin of the microchip (center) is connected to the exterior world by a copper connector.

Functionalities [\[edit\]](#)

Complex Cards support all communication protocols present on regular smart cards: contactless following the [ISO/IEC 14443](#) standard, and magstripe.

Developers of Complex Cards target several needs when developing them:

- One Time Password,
- Provide account information,
- Provide computation capabilities,
- Provide a means of transaction security,
- Provide a means of user authentication.



First smart card manufactured by [Giesecke & Devrient](#) in 1979, already with the finally standardized dimension (ID-1) and a contact area with eight pads (initially on the upper left corner)

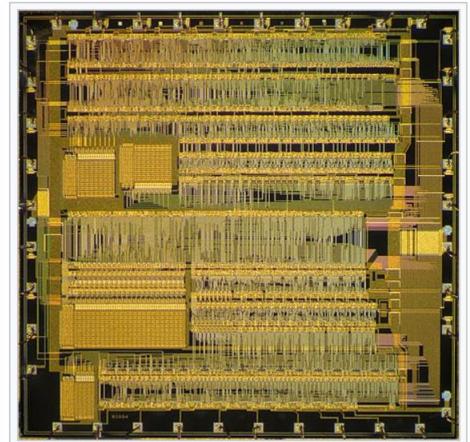
Securing Hardware ... not only Network/Software !

A **secure cryptoprocessor** is a dedicated [computer-on-a-chip](#) or [microprocessor](#) for carrying out [cryptographic](#) operations, embedded in a packaging with multiple [physical security](#) measures, which give it a degree of [tamper resistance](#). Unlike cryptographic processors that output decrypted data onto a bus in a secure environment, a secure cryptoprocessor does not output decrypted data or decrypted program instructions in an environment where security cannot always be maintained.

The purpose of a secure cryptoprocessor is to act as the keystone of a security subsystem, eliminating the need to protect the rest of the subsystem with physical security measures.^[1]

Examples [edit]

A [hardware security module](#) (HSM) contains one or more secure cryptoprocessor [chips](#).^{[2][3][4]} These devices are high grade secure cryptoprocessors used with enterprise servers. A hardware security module can have multiple levels of physical security with a single-chip cryptoprocessor as its most secure component. The cryptoprocessor



Western Electric 229G
cryptoprocessor.

RSA = "big" Compute Resource ... maybe not suitable => Alternatives .. ECC

Elliptic-curve cryptography

⋮ A 21 languages ▾

Article Talk

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia



This article **may be too technical for most readers to understand**. Please [help improve it](#) to make it understandable to non-experts, without removing the technical details. (May 2023) ([Learn how and when to remove this template message](#))

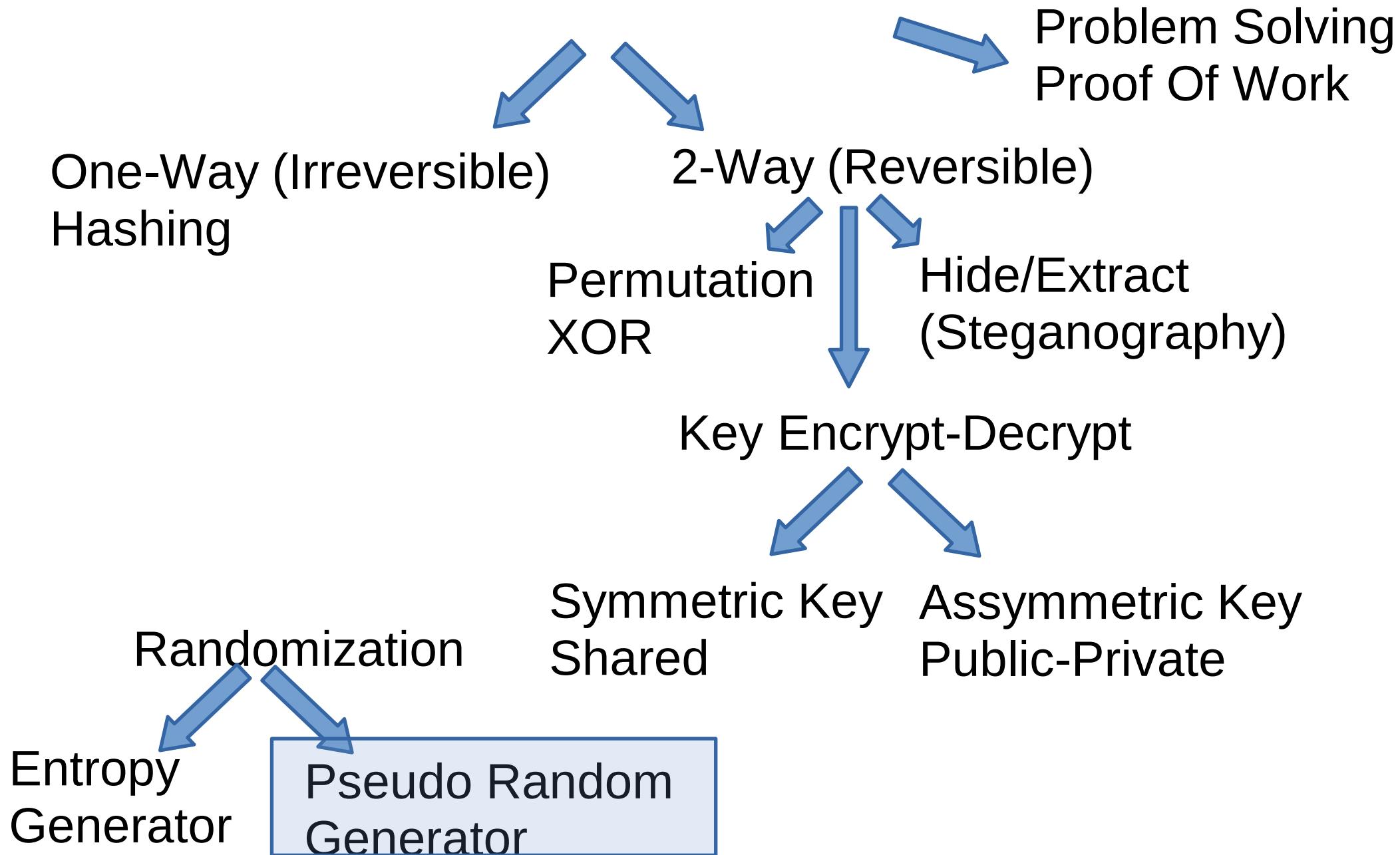
Elliptic-curve cryptography (ECC) is an approach to [public-key cryptography](#) based on the [algebraic structure](#) of [elliptic curves](#) over [finite fields](#). ECC allows smaller keys compared to non-EC cryptography (based on plain [Galois fields](#))^[vague] to provide equivalent security.^[1]

Elliptic curves are applicable for [key agreement](#), [digital signatures](#), [pseudo-random generators](#) and other tasks. Indirectly, they can be used for [encryption](#) by combining the key agreement with a [symmetric encryption](#) scheme. They are also used in several [integer factorization algorithms](#) that have applications in cryptography, such as [Lenstra elliptic-curve factorization](#).

History [edit]

The use of elliptic curves in cryptography was suggested independently by [Neal Koblitz](#)^[2] and [Victor S. Miller](#)^[3] in 1985. Elliptic curve cryptography algorithms entered wide use in 2004 to 2005.

Family of Cryptographic Functions



Fast Pseudo-Random Generators

Back to Math : Doing Polynomials

$$P[X] = a_0 + a_1X + a_2X^2 + \dots a_nX^n$$

... Polynomials over $\mathbb{Z}/2\mathbb{Z}$ modulus, truncated to Rank N

$$X \in \mathbb{Z}/2\mathbb{Z} = \{0, 1\} \quad \forall i, a_i X \in \mathbb{Z}/2\mathbb{Z} \quad \Rightarrow P[X] \in \mathbb{Z}/2\mathbb{Z}$$

... we are interested not in X ... put in P (in coefficients)
algebra of P : P1, P2, ... P1.P2, P1+P2

Consecutive Powers of P

Consecutive (truncated) powers of P :

$$\{P[X], P^2[X], P^3[X] \dots P^{2^N}[X] \dots\}$$

If “P” is chosen irreducible in $Z/2Z[n]$
... then orbit contains all polynomials!
(all coefficients of all polynomials)

$$card(\{a_1 a_2 \dots a_n\}) = 2^{2^n}$$

Suppose $N = 64$... then $2^{2^{64}} \simeq 2^{10^9}$.. a Pseudo-Random generator HUGE cycle

Multiply (and Truncate) Polynomials

$$A[X] = a_0 + a_1X + a_2X^2 + \dots a_nX^n$$

$$B[X] = b_0 + b_1X + b_2X^2 + \dots b_nX^n$$

$$A \cdot B[X] = A[X] \cdot B[X] = a_0 \cdot b_0 + (a_0b_1 + a_1b_0)X$$

$$+ (a_0b_2 + a_1b_1 + a_2b_0)X^2 + \dots (a_0b_n + \dots a_nb_0)X^n$$

+ truncated $\dots X^{n+1} \dots X^{2n}$

Efficient for P=1+X using Bit Shift, Bit And..

$$A[X] = a_0 + a_1X + a_2X^2 + \dots a_nX^n$$

$$B[X] = 1 + X$$

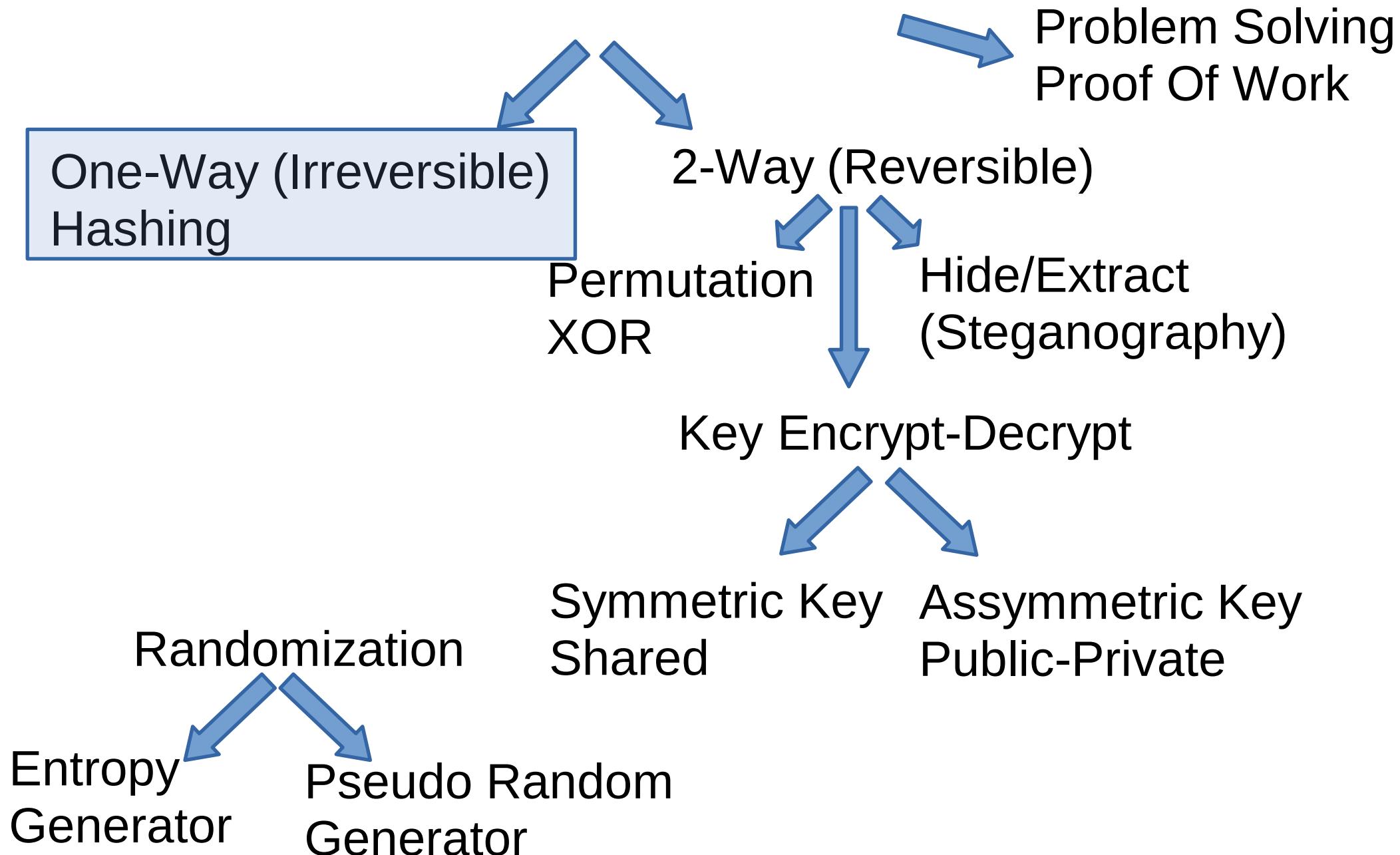
$$A \cdot B[X] = a_0 + (a_0 + a_1)X + (a_1 + a_2)X^2 + \dots$$

$a_1 a_2 a_3 \dots a_{n-1} a_n$

$$\begin{array}{r} + \quad \quad \quad a_2 a_3 a_4 \dots a_n 0 \\ \hline (a_1 + a_2)(a_2 + a_3) \dots (a_{n-1} + a_n)a_n \end{array}$$

← Bit Shift

Family of Cryptographic Functions



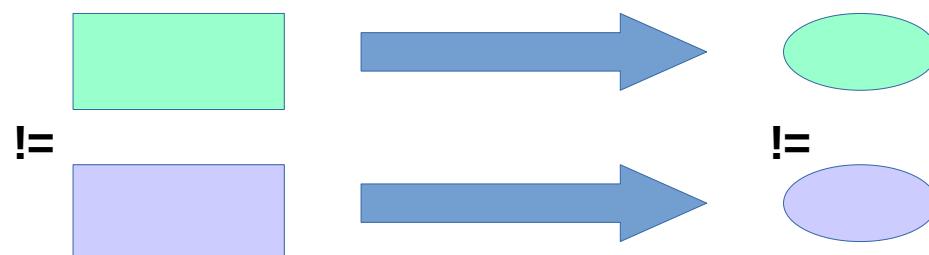
Hashing ...



Hashing is One-Way transformation



Hashing different product **usually** give different result



Hashing ... must be equi-distributed minimise “Collisions”

A Hashing Collision



Of course it can happen ... example in java:

“public int hashCode()”

by default `Object.hashCode = System.identityHashCode(x)`

= ... 64bits pointer (when first hashed) → hashed to 32 bits

= `(hashPtr64 ^ (hashPtr64 >>>32))`

Hashing : ~~MD5, SHA-1~~, SHA-256, SHA-512 ..

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

MD5

From Wikipedia, the free

(Redirected from [Md5](#))

The **MD5 algorithm** is a widely used hash function producing a 128-bit hash value. Although it was once considered secure against collisions, it has since been demonstrated to be vulnerable to collisions. It can be used to verify the integrity of files, but only against unstructured data.

Like most hash functions, MD5 has been cracked by brute-force attacks. The details of these attacks are detailed in the security analysis section below.

MD5 was designed by Ronald L. Rivest as an improvement over his previous function [MD4](#).^[3] The MD5 algorithm was first published in 1992 and became part of the RSA license. The abbreviation "MD" stands for "Message Digest."

The security of the MD5 has been severely compromised, with its weaknesses having been exploited in the field, most infamously by the Flame malware in 2012. The CMU Software Engineering Institute considers

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

SHA-1

From Wikipedia, the free encyclopedia

In [cryptography](#), **SHA-1 (Secure Hash Algorithm 1)** is a [cryptographic hash function](#) designed by the United States [National Security Agency](#) and is a U.S. [Federal Information Processing Standard](#) published by the United States [NIST](#).^[3]

SHA-1 produces a [160-bit](#) ([20-byte](#)) hash value known as a [message digest](#). A SHA-1 hash value is typically rendered as a [hexadecimal](#) number, 40 digits long.

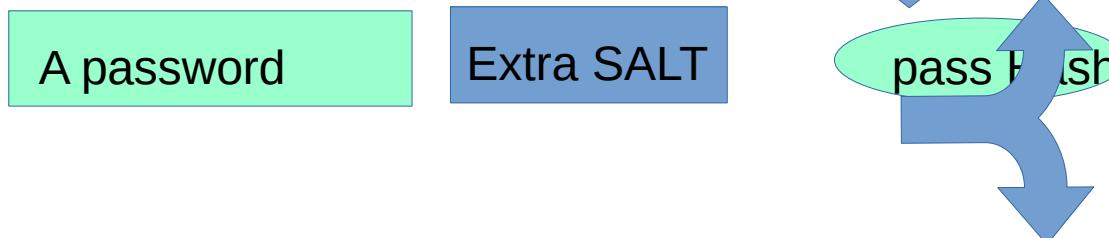
SHA-1 is no longer considered secure against well-funded opponents. In 2005, cryptanalysts found attacks on SHA-1 suggesting that the algorithm might not be secure enough for ongoing use,^[4] and since 2010 many organizations have

Hashing For Storing “Passwords”

Create Login Account

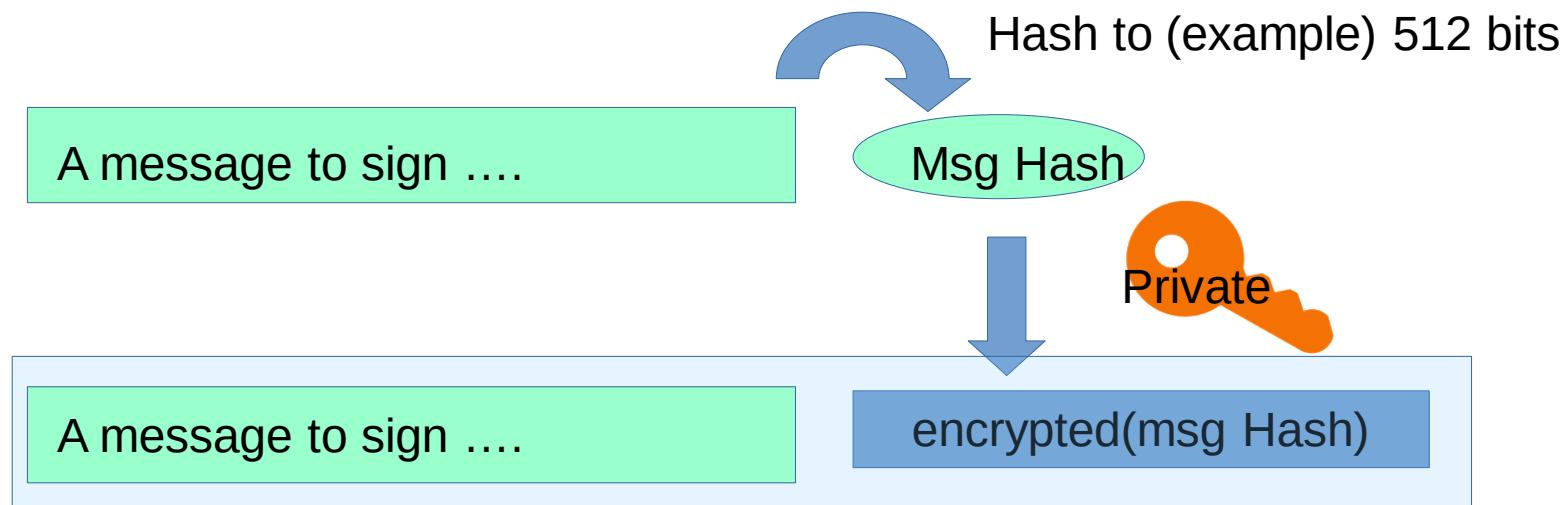


Check Password



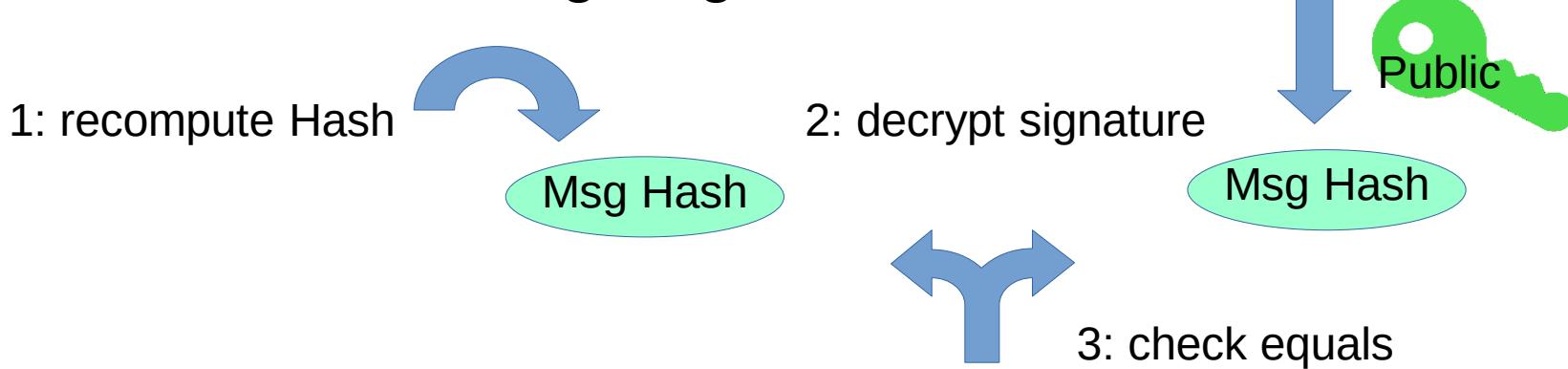
Same HASH ?
(assume pass OK)

Hashing : Good for Signing

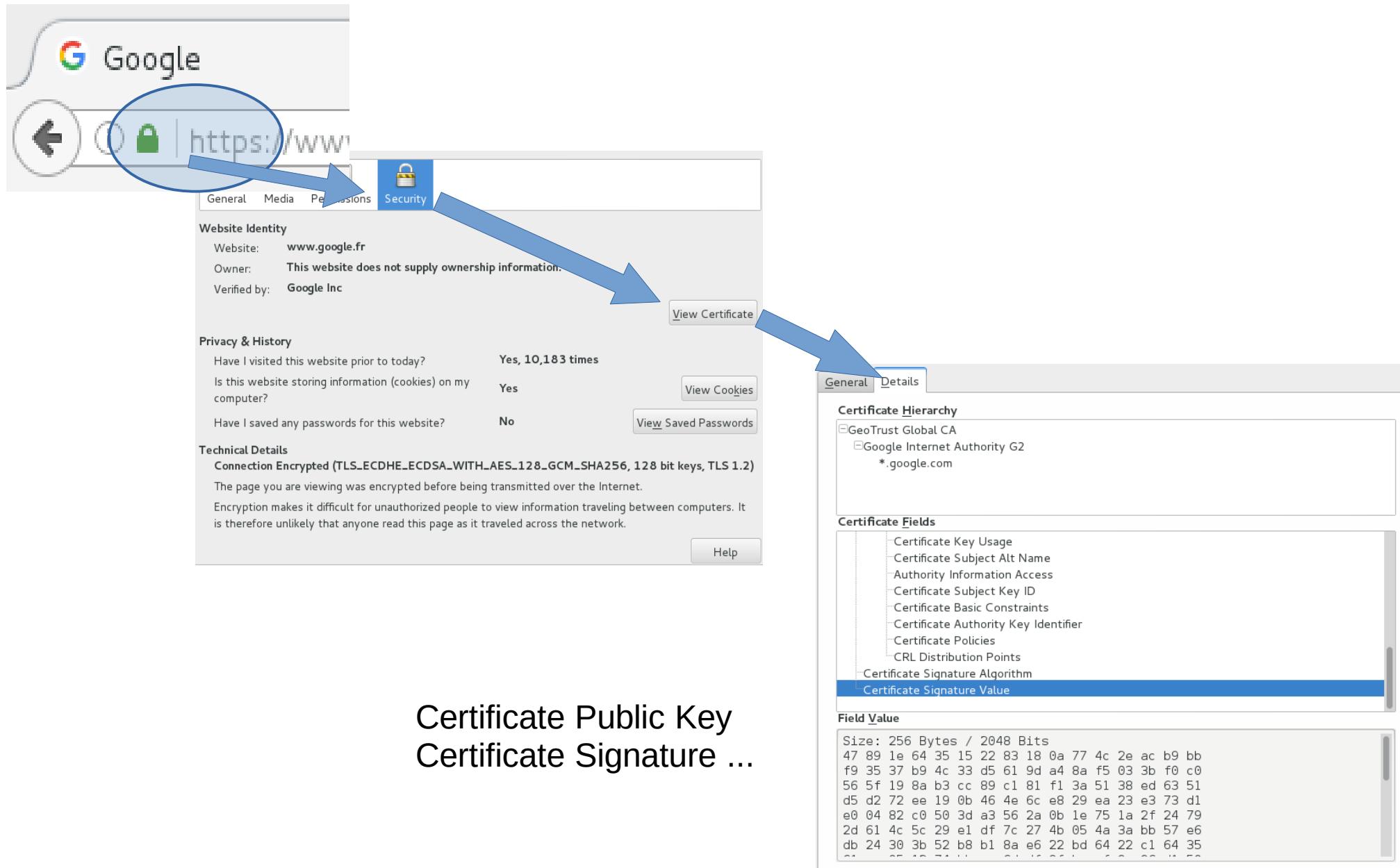


Signed Message : clear text + extra signature (at bottom of page)

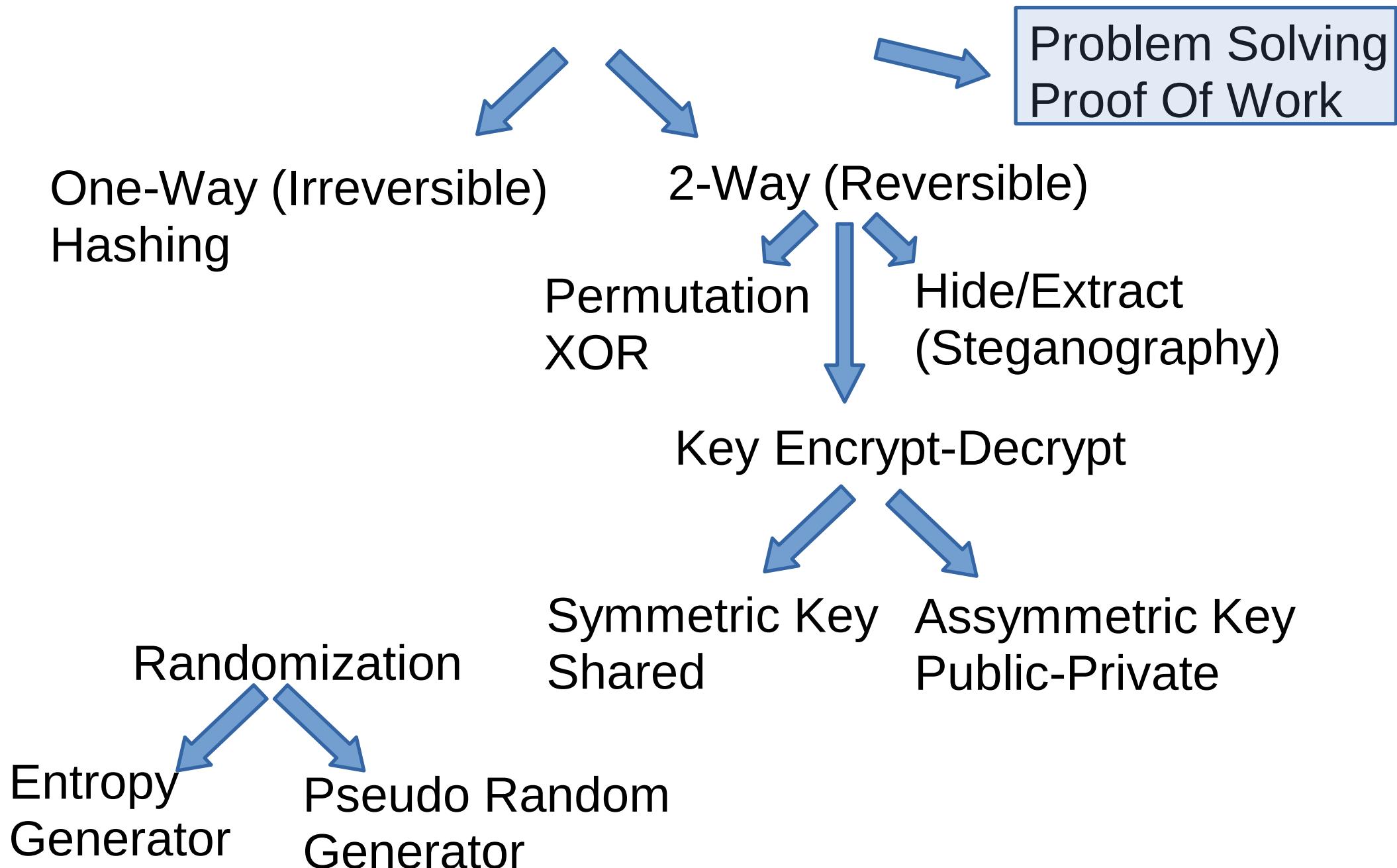
Check that a message signature is VALID :



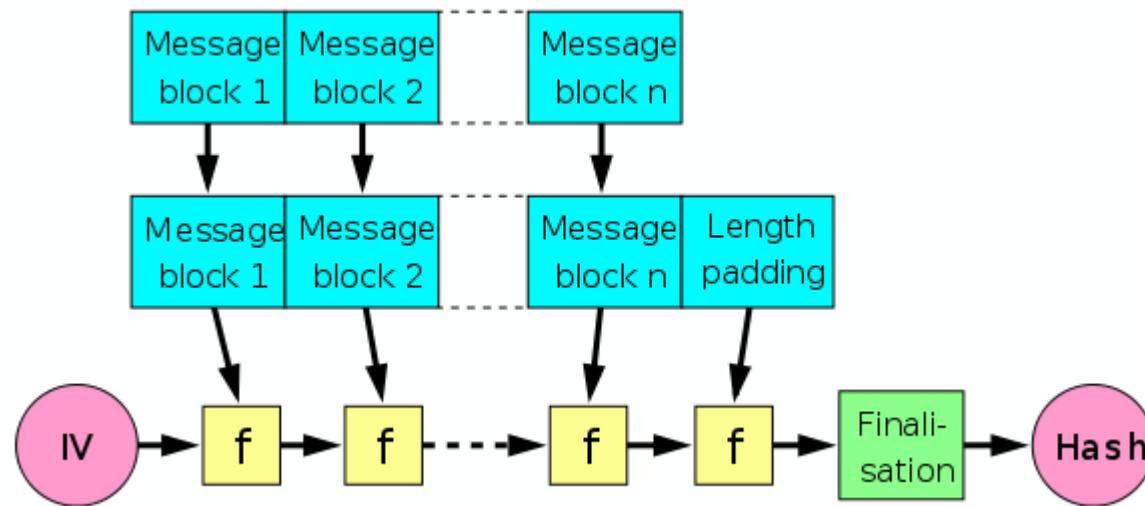
Certificates = Signed Public Keys



Family of Cryptographic Functions



Hash a BitCoin Transactions Chain BlockChain



Merkle Tree

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Search Wikipedia



Merkle tree

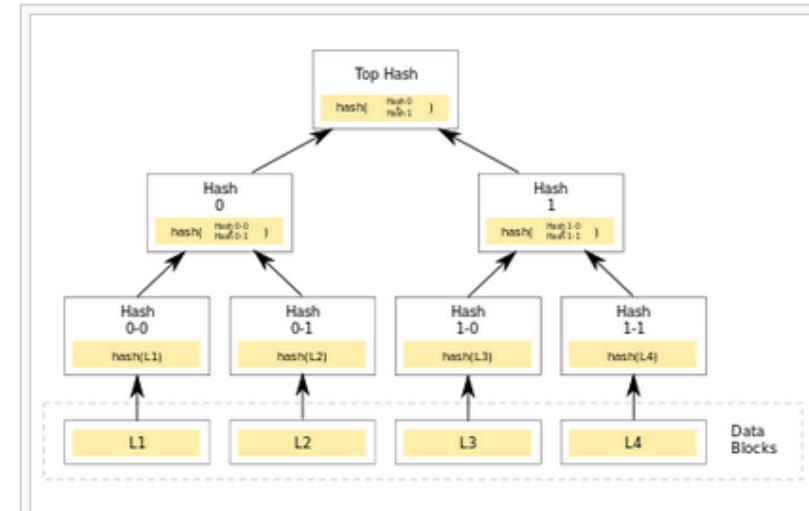
From Wikipedia, the free encyclopedia

In [cryptography](#) and [computer science](#), a **hash tree** or **Merkle tree** is a [tree](#) in which every non-leaf node is labelled with the [hash](#) of the labels or values (in case of leaves) of its child nodes.

Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of [hash lists](#) and [hash chains](#).

Demonstrating that a leaf node is a part of the given hash tree requires processing an amount of data proportional to the [logarithm](#) of the number of nodes of the tree;^[1] this contrasts with hash lists, where the amount is proportional to the number of nodes.

The concept of hash trees is named after [Ralph Merkle](#) who



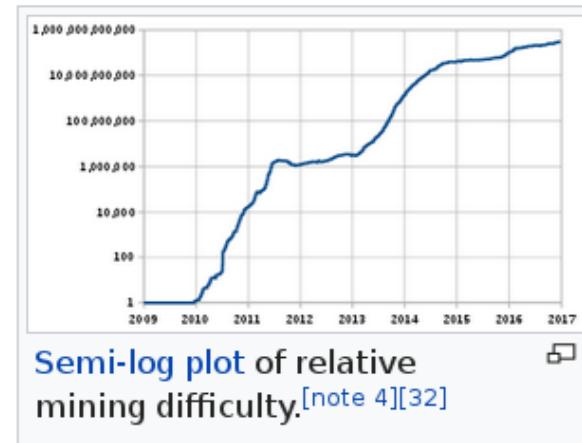
An example of a binary hash tree. Hashes 0-0 and 0-1 are the hash values of data blocks L1 and L2, respectively, and hash 0 is the hash of the concatenation of hashes 0-0 and 0-1.

Proof Of Work, Mining

Mining [edit]

Mining is a record-keeping service.^[note 5] Miners keep the blockchain consistent, complete, and unalterable by repeatedly verifying and collecting newly broadcast transactions into a new group of transactions called a *block*.^[29] Each block contains a [cryptographic hash](#) of the previous block,^[29] using the [SHA-256](#) hashing algorithm,^{[8]:ch. 7} which links it to the previous block,^[29] thus giving the blockchain its name.

In order to be accepted by the rest of the network, a new block must contain a so-called [proof-of-work](#).^[29] The proof-of-work requires miners to find a number called a [nonce](#), such that when the block content is hashed along with the nonce, the result is numerically smaller than the network's *difficulty target*.^{[8]:ch. 8} This proof is easy for any node in the network to verify, but extremely time-consuming to generate, as for a secure cryptographic hash, miners must try many different nonce values (usually the sequence of tested values is 0, 1, 2, 3, ...^{[8]:ch. 8}) before meeting the difficulty target.



Other BlockChains (for contracts, legal ownership, assurance, ..)

Article [Talk](#)

[Read](#) [Edit](#) [View h](#)

Ethereum

From Wikipedia, the free encyclopedia

Ethereum is an [open-source](#), public, blockchain-based [distributed computing](#) platform featuring [smart contract](#) (scripting) functionality, which facilitates online contractual agreements.^[2] It provides a decentralized [Turing-complete virtual machine](#), the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. Ethereum also provides a [cryptocurrency](#) token called "ether", which can be transferred between accounts and used to compensate participant nodes for computations performed. Gas, an internal transaction pricing mechanism, is used to mitigate [spam](#) and allocate resources on the network.^{[2][3]}

Conclusion

