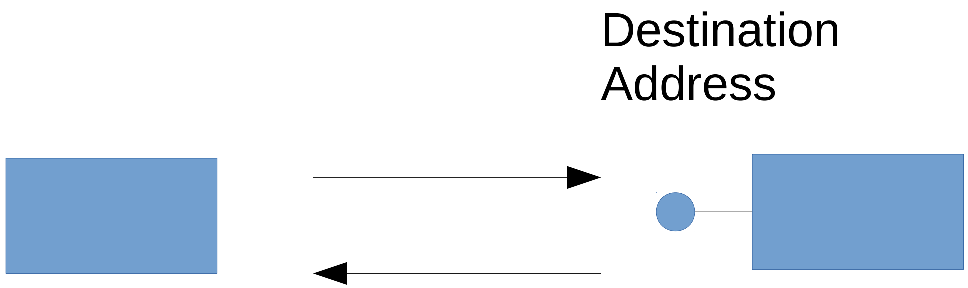
Introduction to Cryptography

arnaud.nauwynck@gmail.com

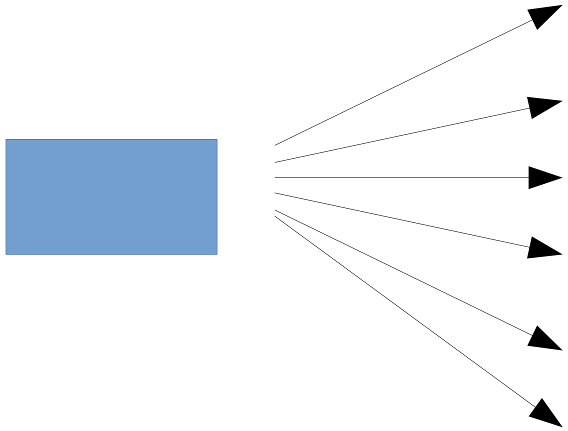
Communications

Point - to - Point

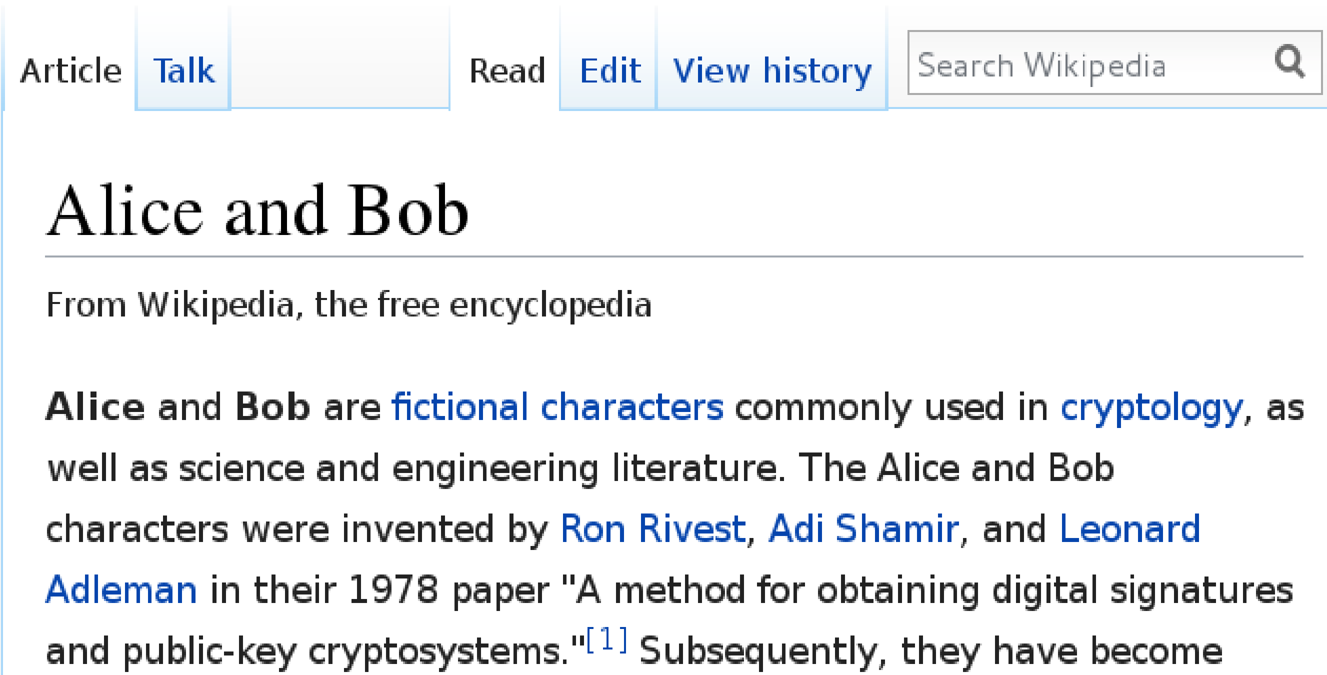


Dual Channel Communication
...example for Request - Response

Broadcast



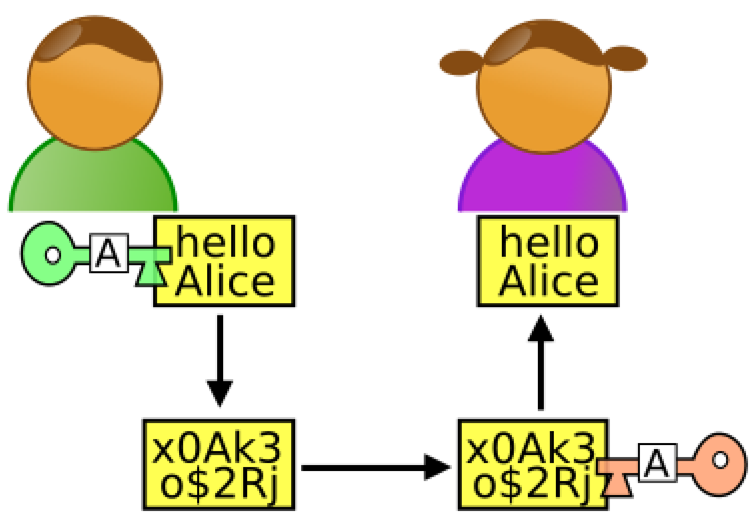
Who are Alice, Bob, Eve, Malory, & Trent .. ?

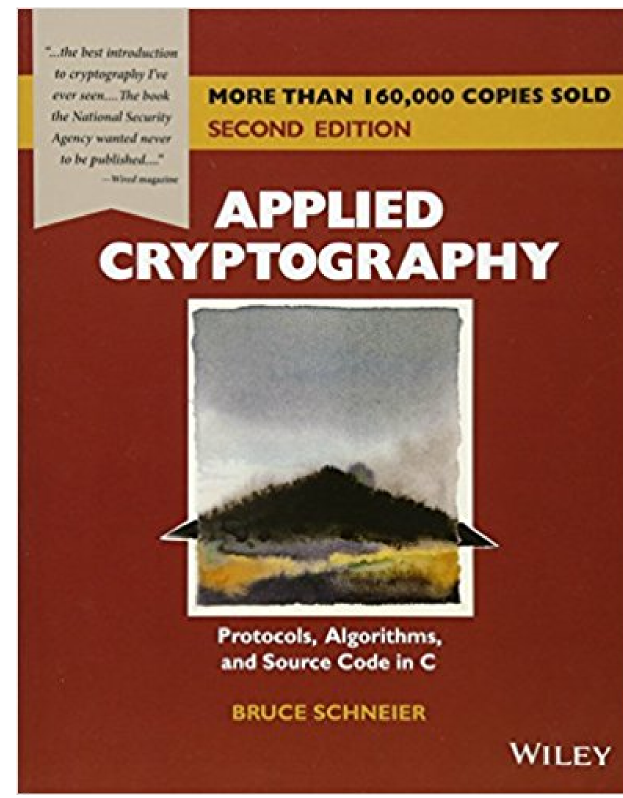

Article Talk Read Edit View history Search Wikipedia

Alice and Bob

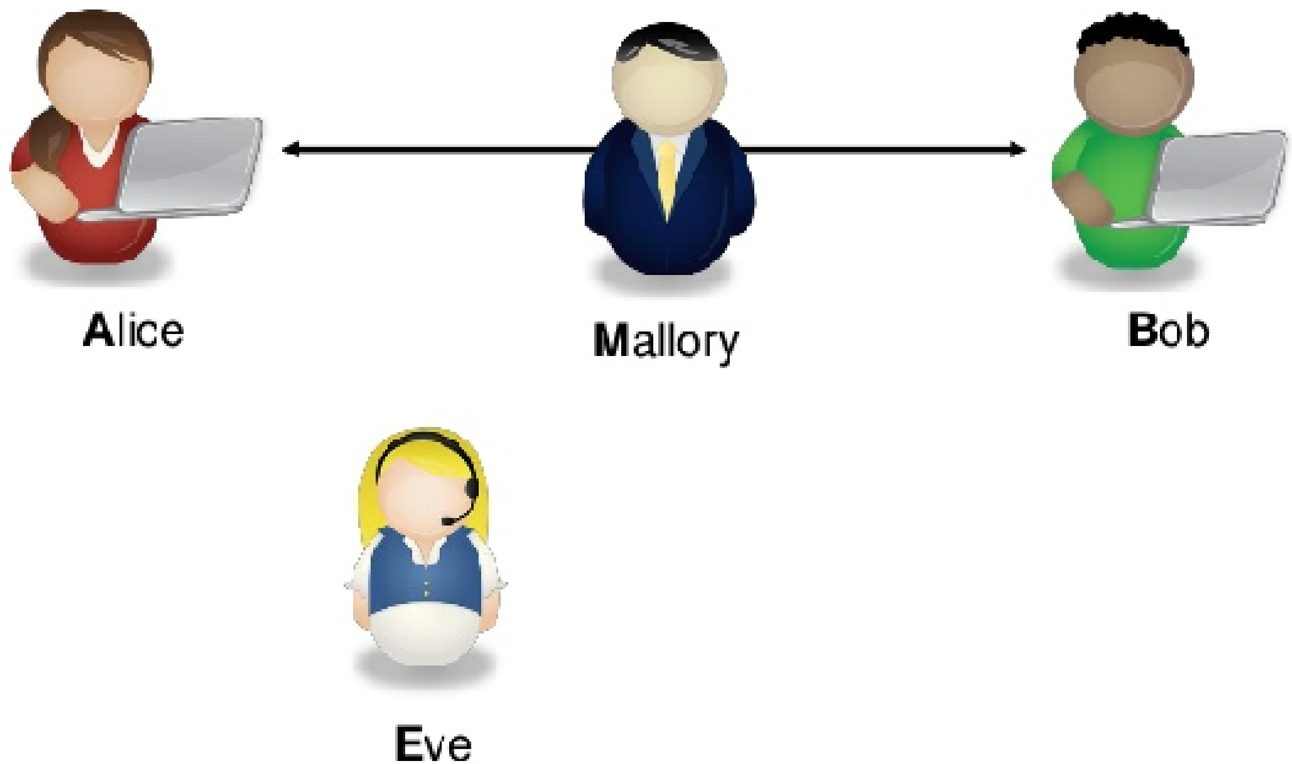
From Wikipedia, the free encyclopedia

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



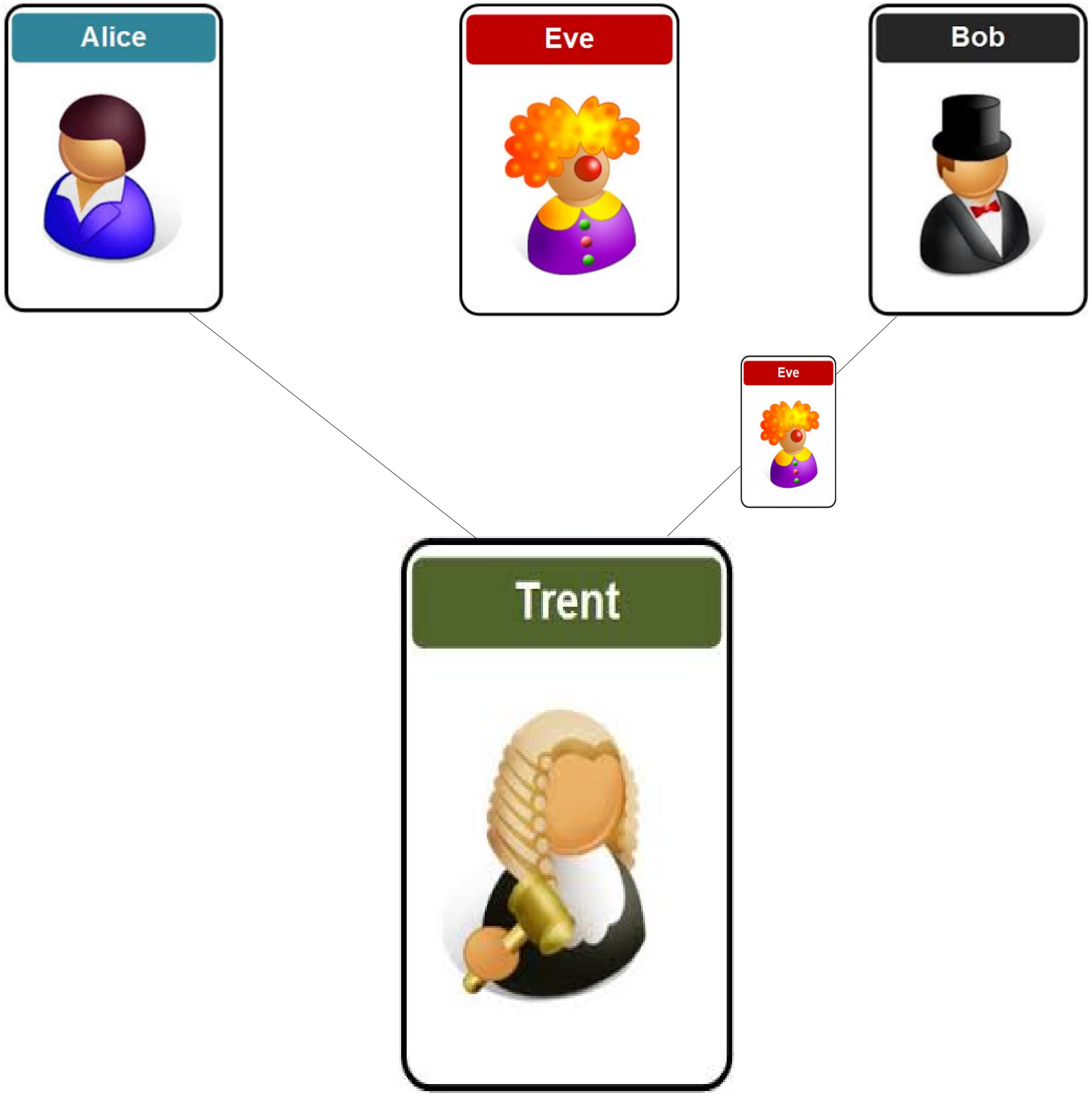


Eve (E=Environment … or Evil) and Malory (M=Malicious)



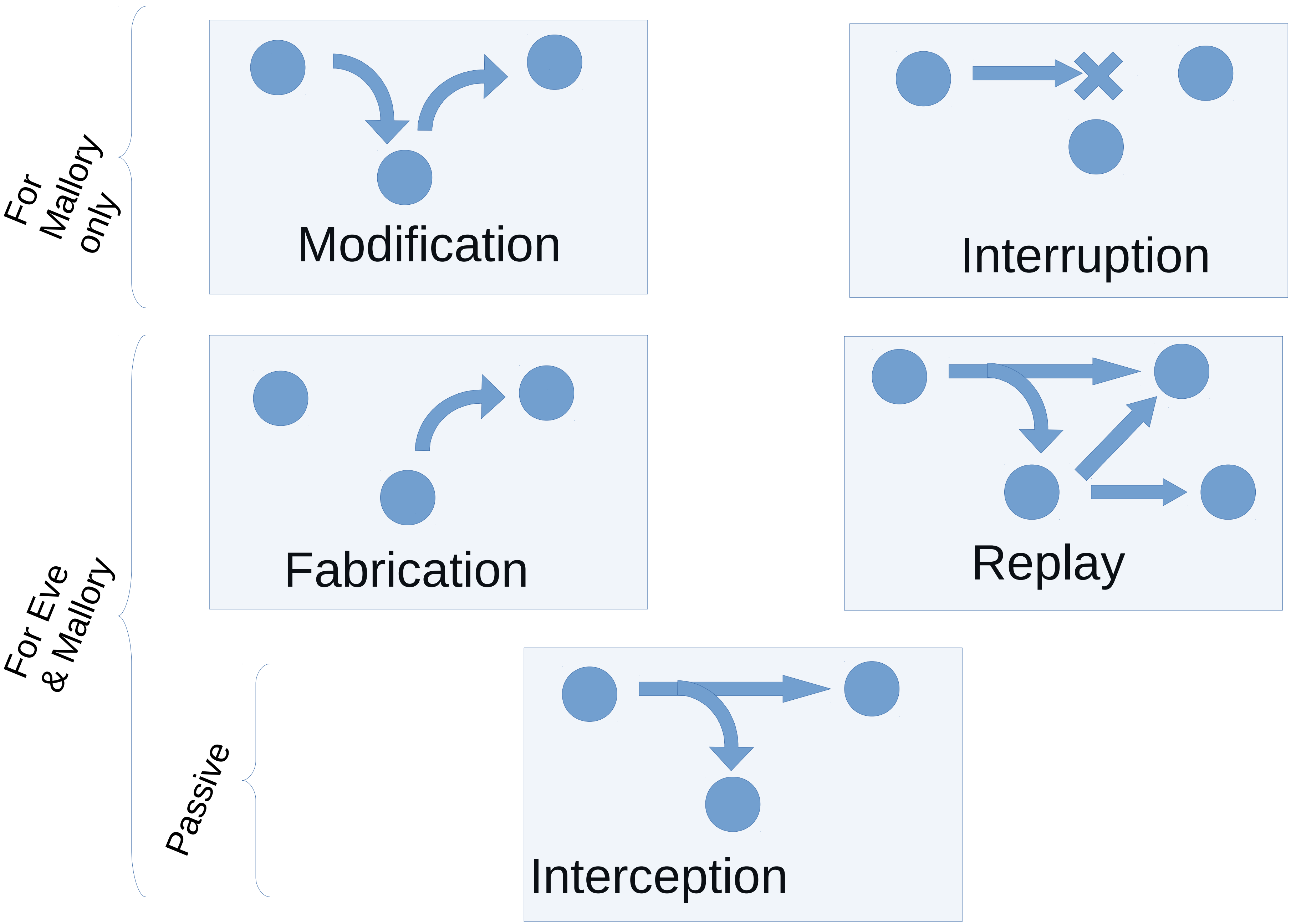
Mallory intercept + modify data
… Eve does't not

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

Confidentiality

Integrity

Authorisation

Password Crack

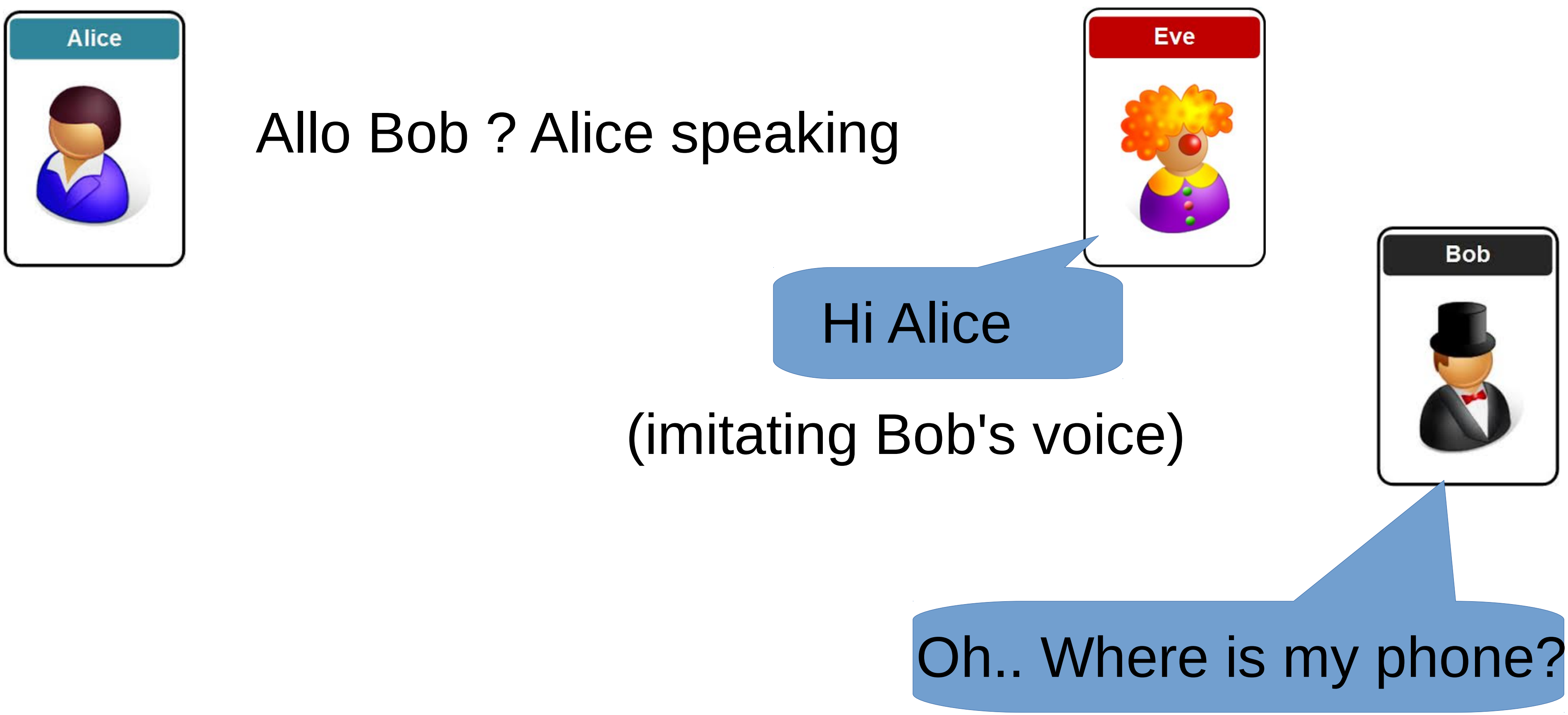
Data Leak

Data Replay

Predicatable guess

...

Authentication (Receiver)

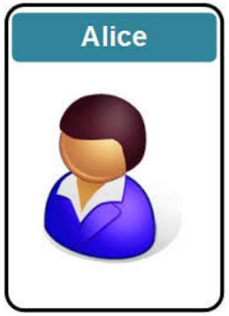


Authentication (Sender)

Allo Bob ? Alice speaking

(imitating Bob's voice)

Hi Alice



Oh.. Where is my phone?

… Social Engineering

Allo Bob ? I am a friend of Alice

She asked me to tell you..

Hi

Sure, go ahead

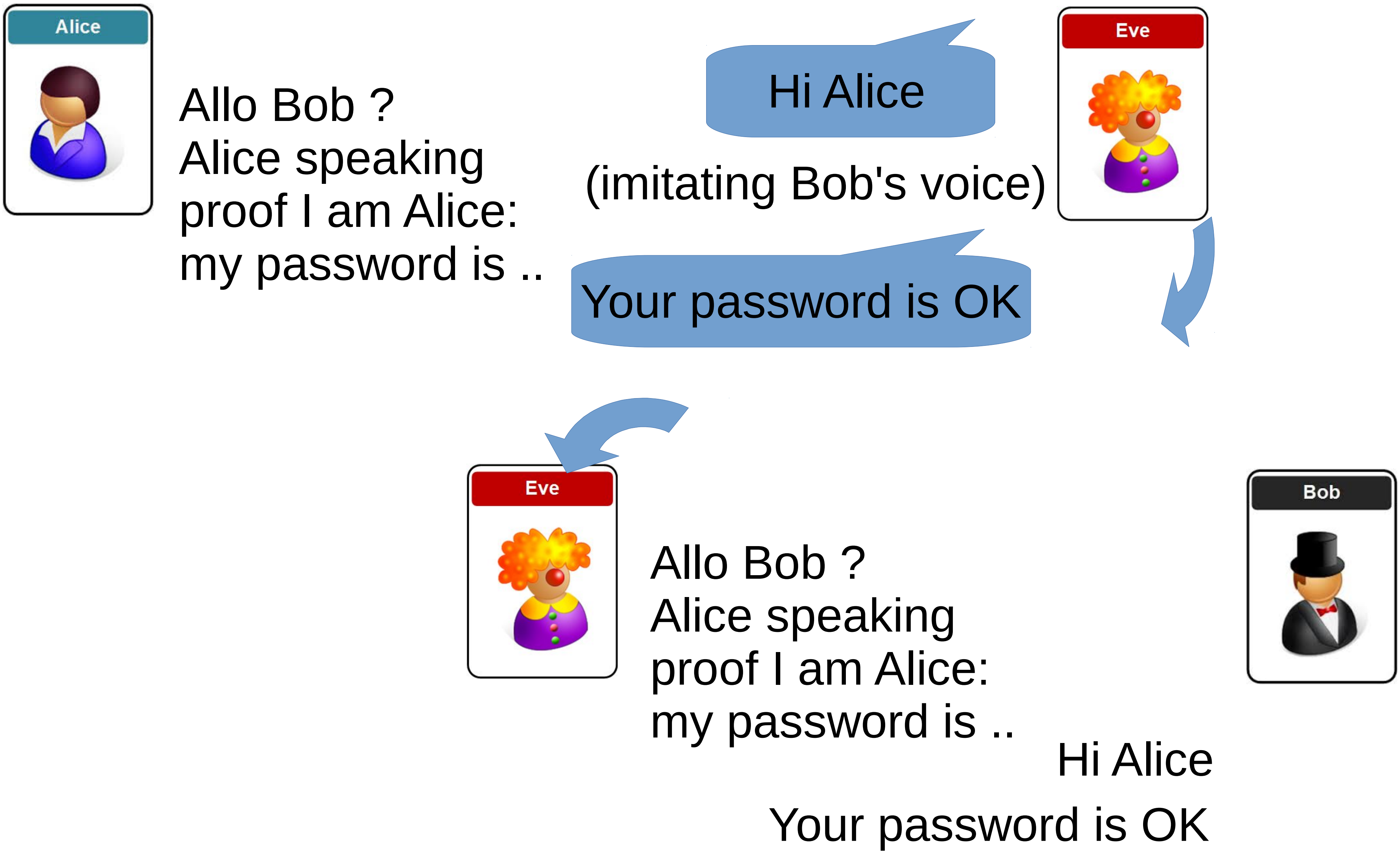
Bob is so naive

Authentication denied

Allo Bob ? Alice speaking

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

Basic Authentication



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

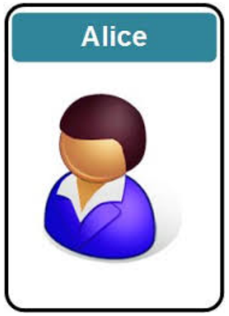
OK, Send me you pass hashed by “X”

(.. Compute Y=hash(P,X))
It is “Y”

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

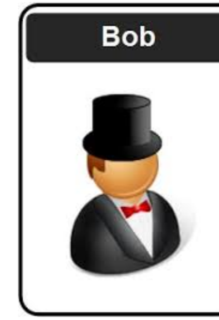
OK Alice

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

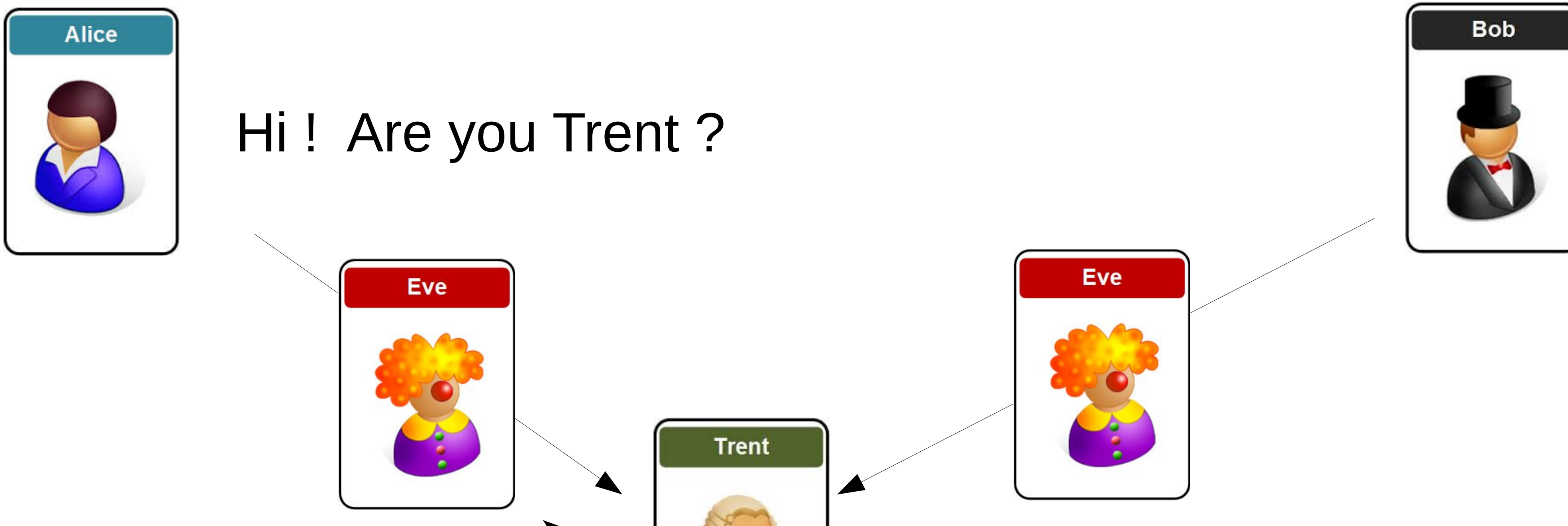
Bob: Ask “Trent” … he prooved my identity by token XADS4FSQ3RTXF

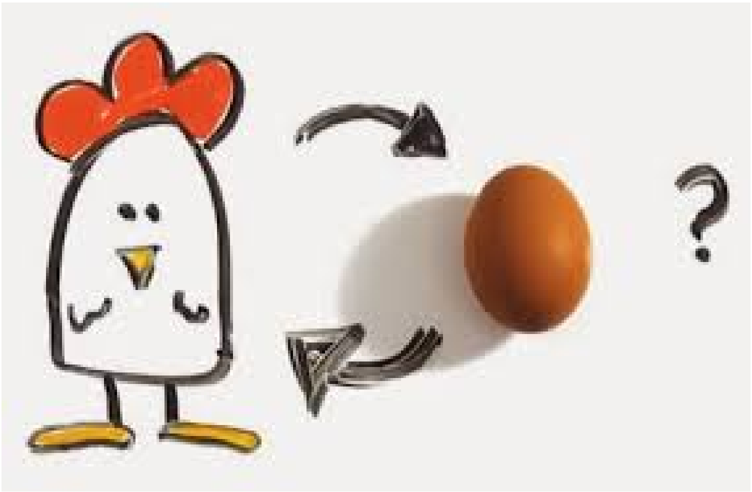
Is it true this token was given to Alice:
XADS4FSQ3RTXF ?

Yes (And Alice did not complain yet being stolen)

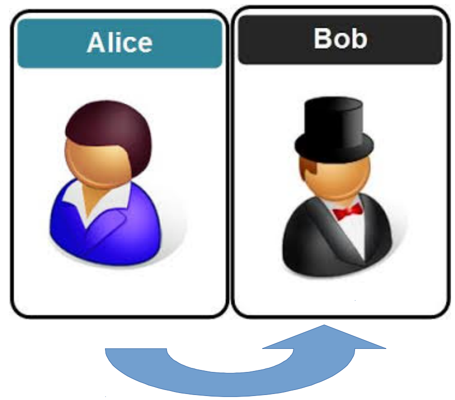
OK Alice

Problems With Thirdparties...





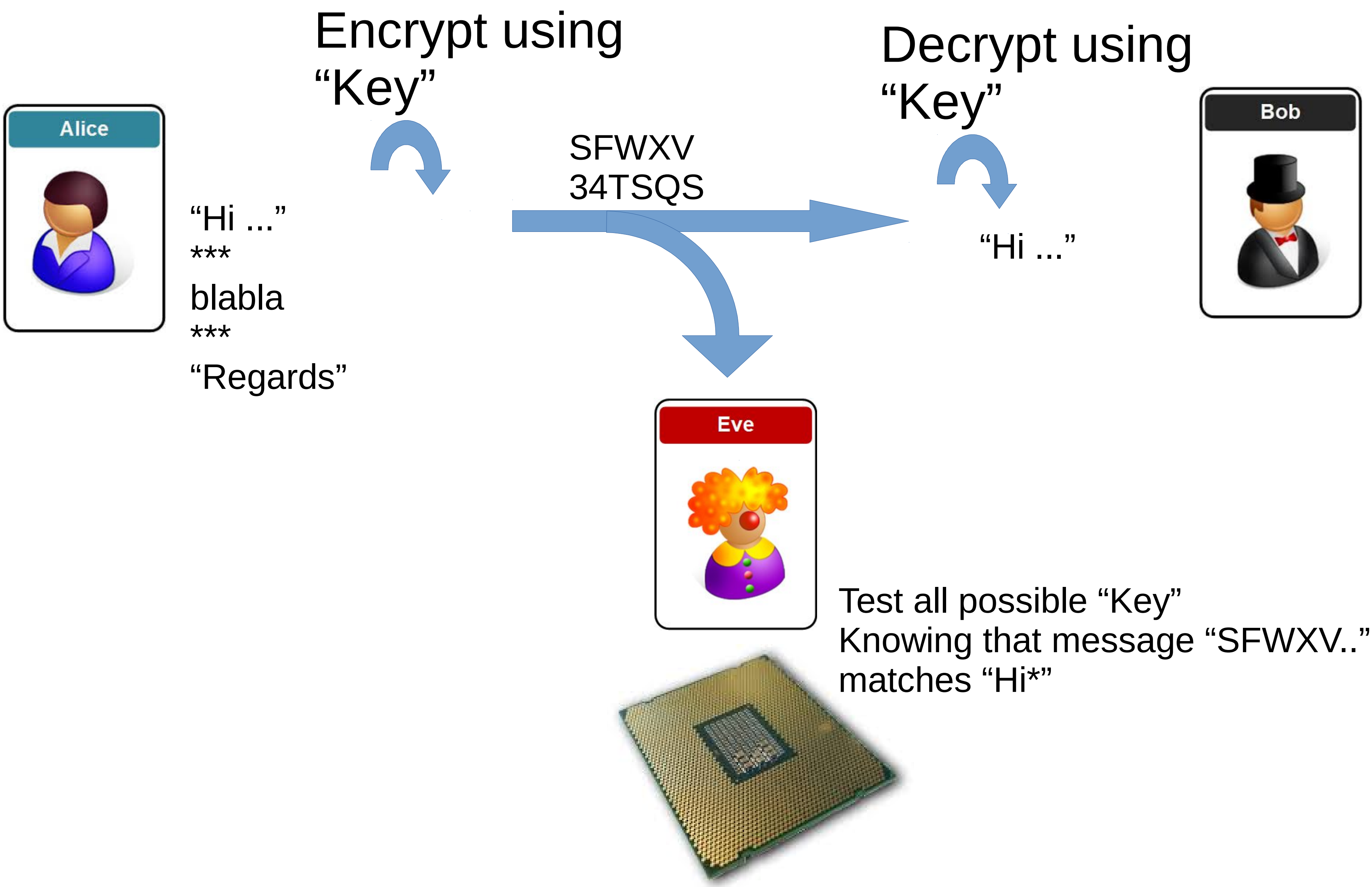
Physical Hand-Shake
... Exchange Shared Key



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 …
AND NO predicatable data …

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

Thow “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

Predicatable guess

Authorisation

...

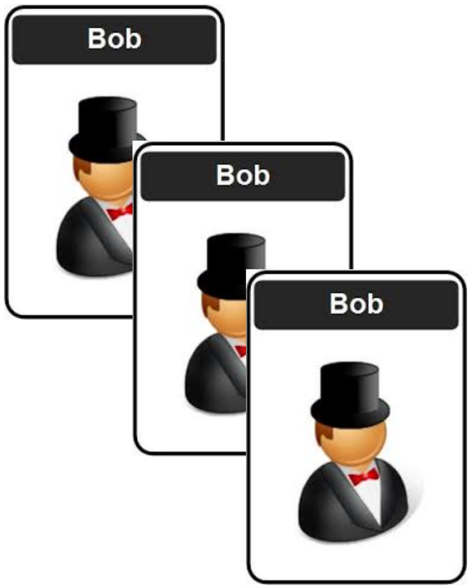
Password Crack

(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...

Try password1..

Wrong password
(answer after 10ms)

Try password2..

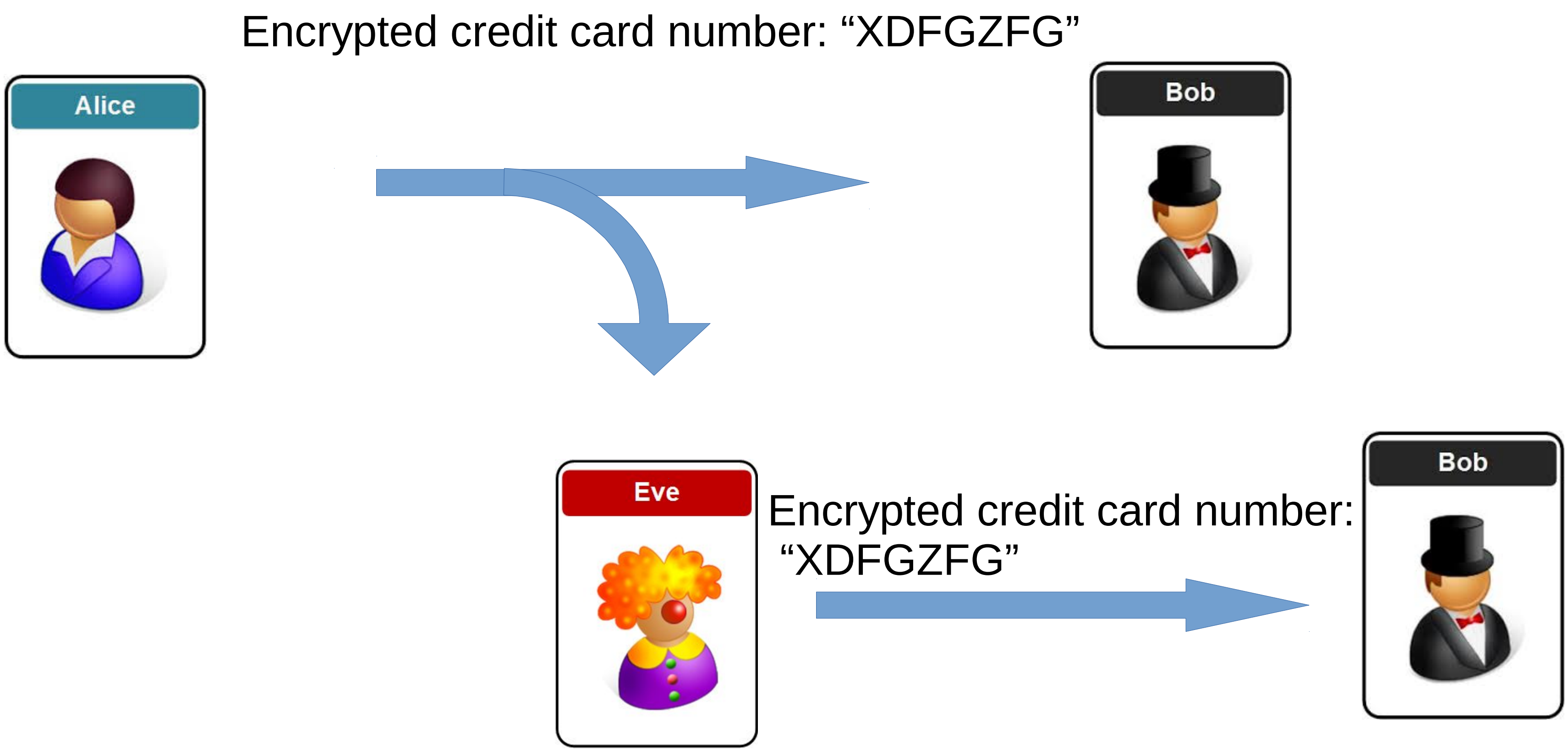
Wrong password
(answer after 11ms)

Try password9..

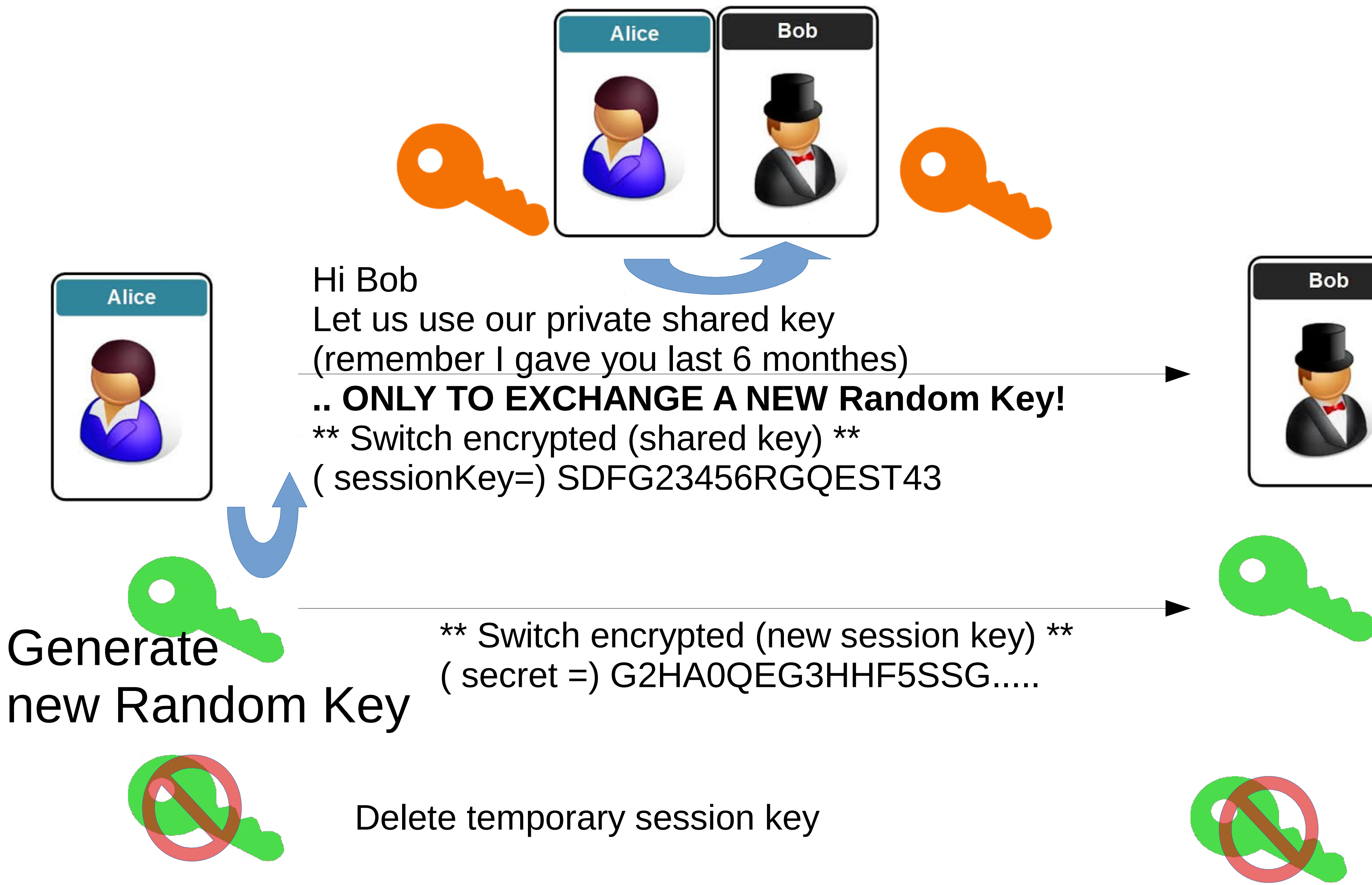
Wrong password
(answer after 20ms)

Seems to finish by “9”
… it tooks more time to compute all

Encrypted … Data Replay



Using Shared Key for Exchanging TMP SessionKey ONLY



Fast Crypt using XOR
+ Session Key Random Generator

** Switch encrypted (shared key) **
(seed =) G2HA0QEG3HHF5SSG.....

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

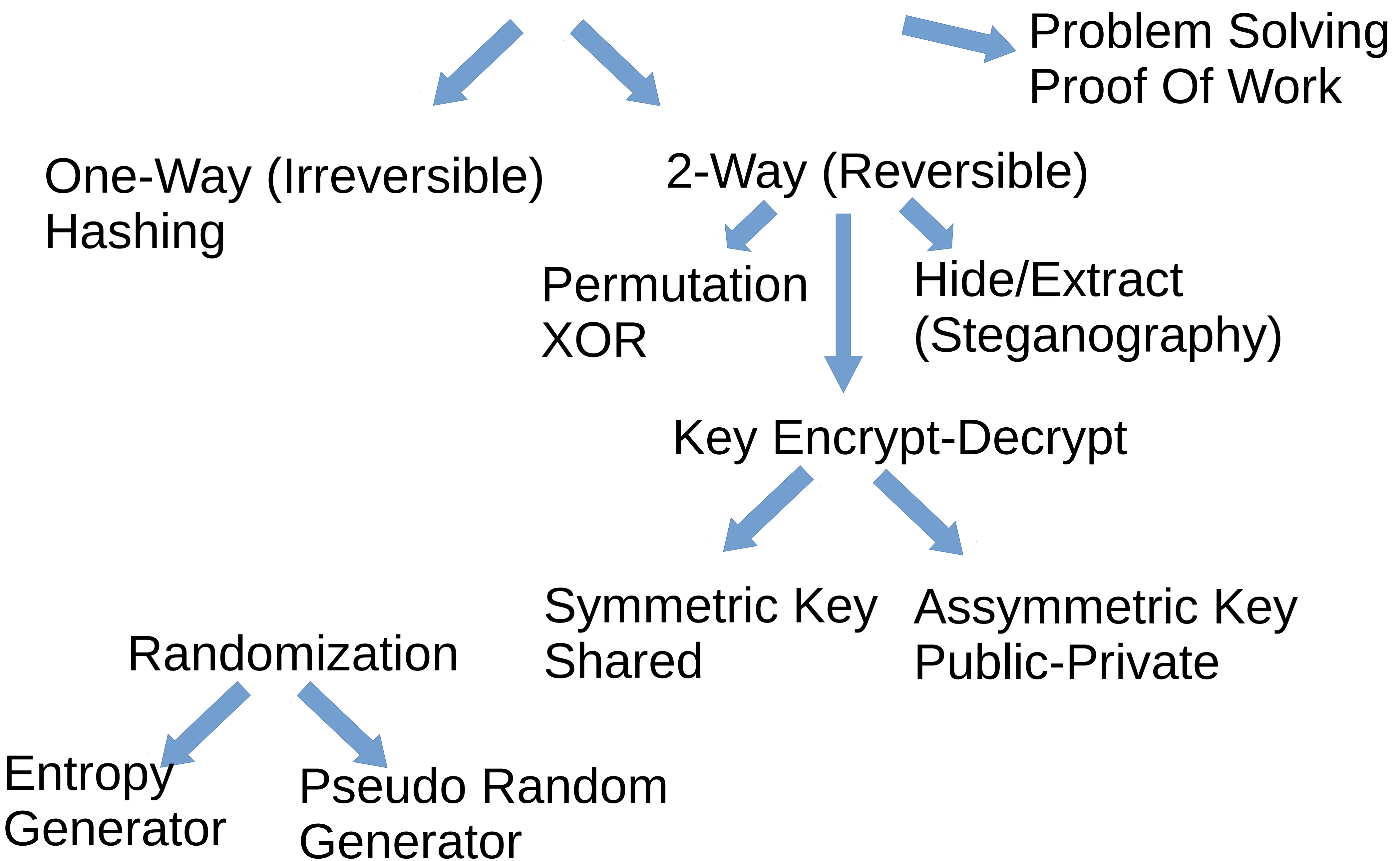
Seed gen

.. 0 1 0 1 1 1 0 1 1 ..

1 0 1 0 0 1 0 1 0

Secret message blabla ….

Family of Cryptographic Functions



Family of Cryptographic Functions

Problem Solving
Proof Of Work

One-Way (Irreversible)
Hashing

2-Way (Reversible)

Permutation
XOR

Hide/Extract
(Steganography)

Key Encrypt-Decrypt

Symmetric Key
Shared

Assymmetric Key
Public-Private

Randomization

Entropy
Generator

Pseudo Random
Generator

XOR ...

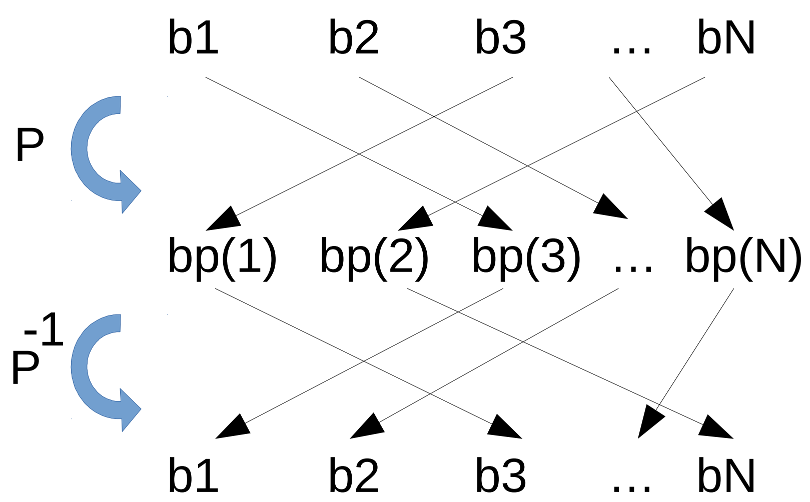
VERY VERY Fast

Works Great with “Good” Pseudo Random Generators

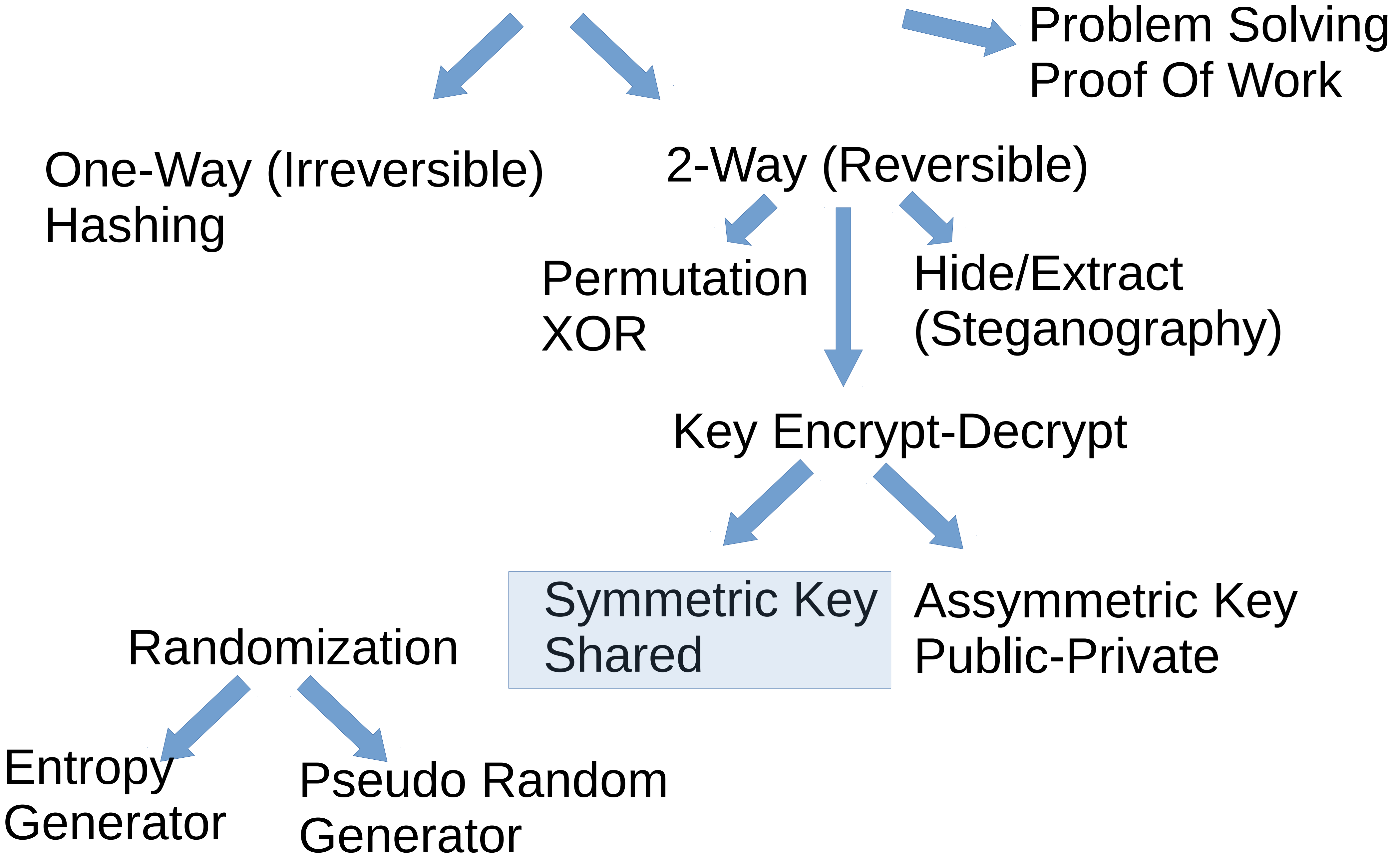
Permutations

P: [1..N] → [1..N]

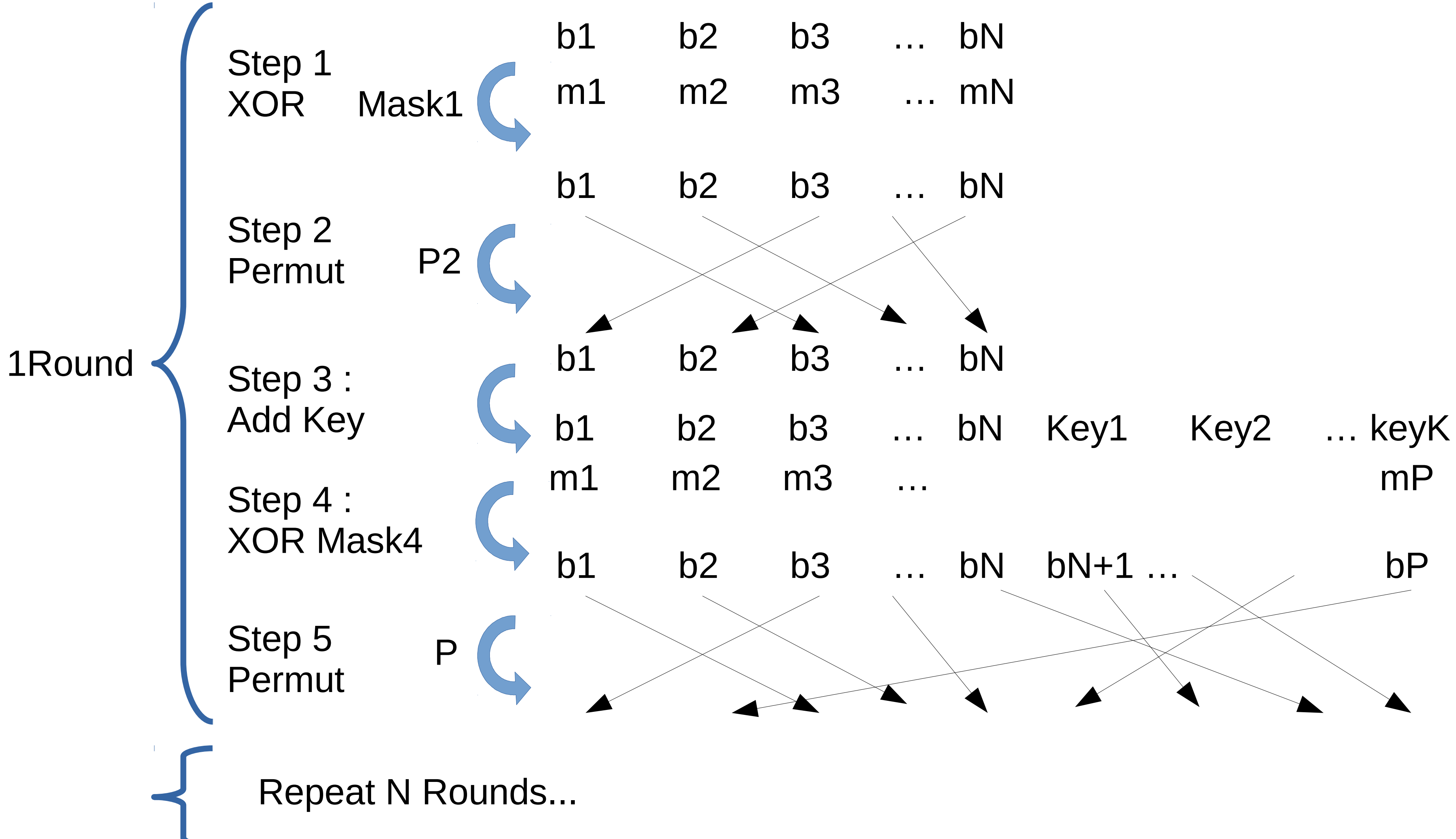
P is a permutation if it is inversible (all elts are reached once)



Family of Cryptographic Functions



Combining XOR & Permutations & Add



Example : AES, 3-DES, ...

Article Talk Read Edit View histo

Triple DES

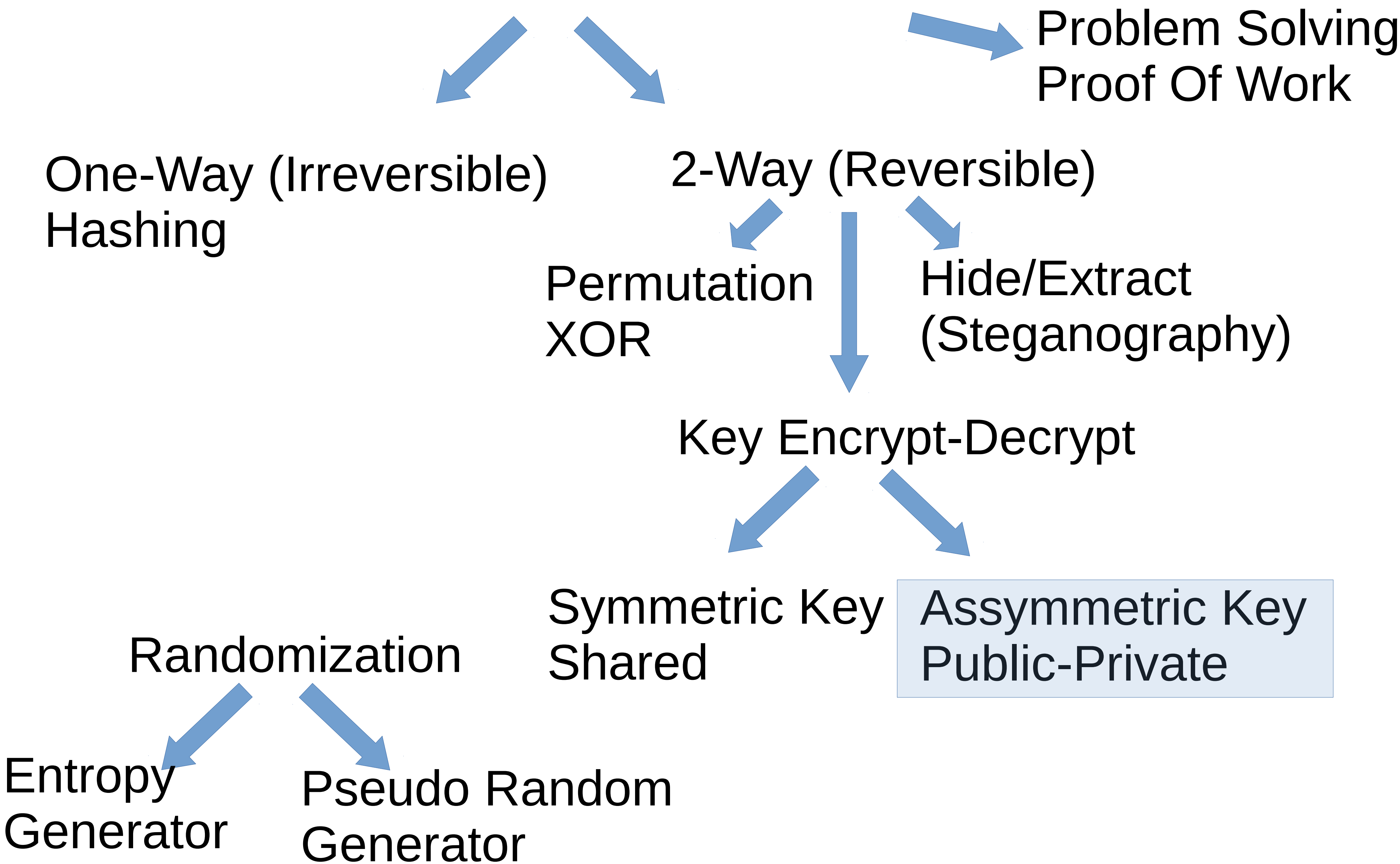
From Wikipedia, the free encyclopedia

(Redirected from 3-DES)

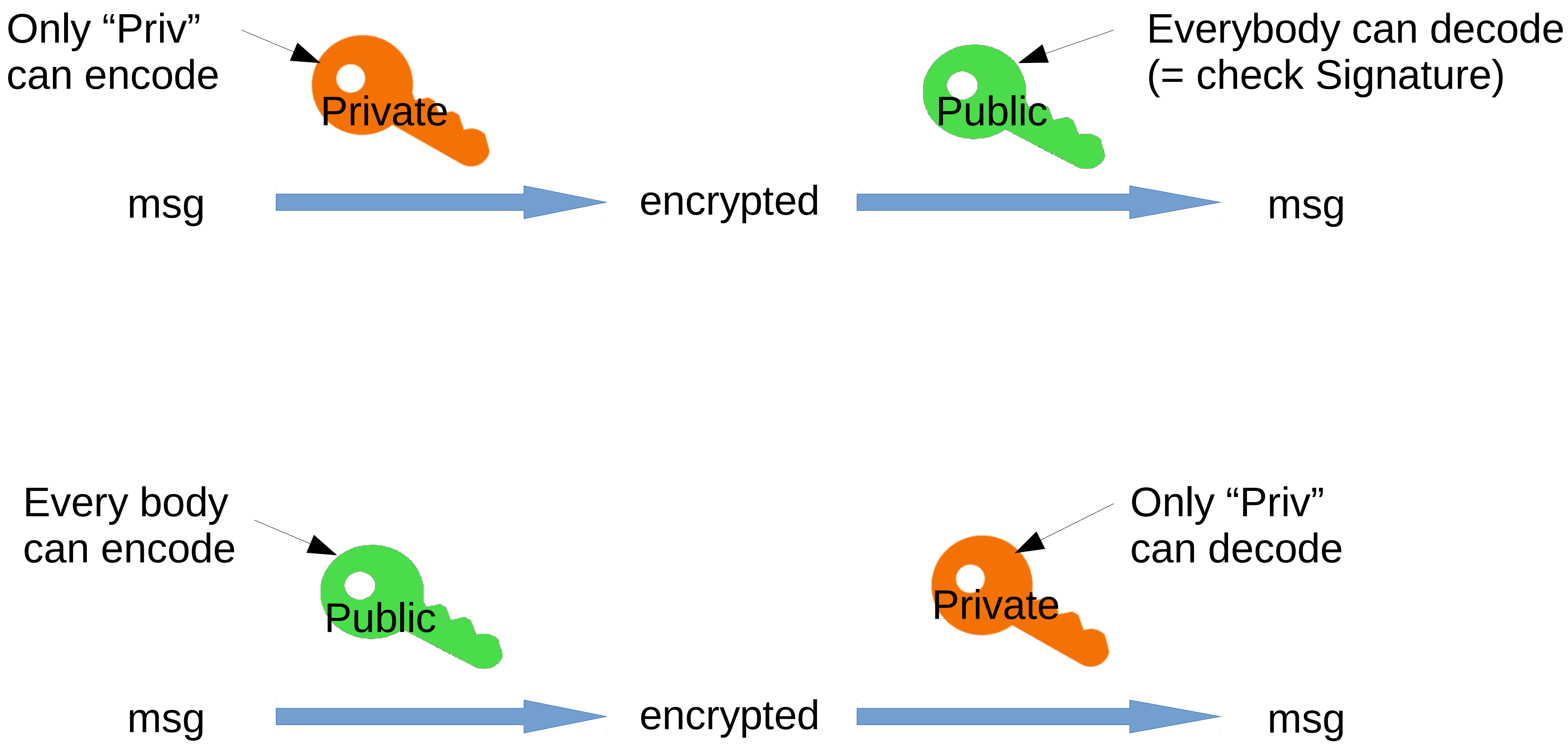
In cryptography, **Triple DES** (**3DES**), officially the **Triple Data Encryption Algorithm** (**TDEA** or **Triple DEA**), is a symmetric-key block cipher, which applies the Data Encryption Standard (DES) cipher algorithm three times to each data block.

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.

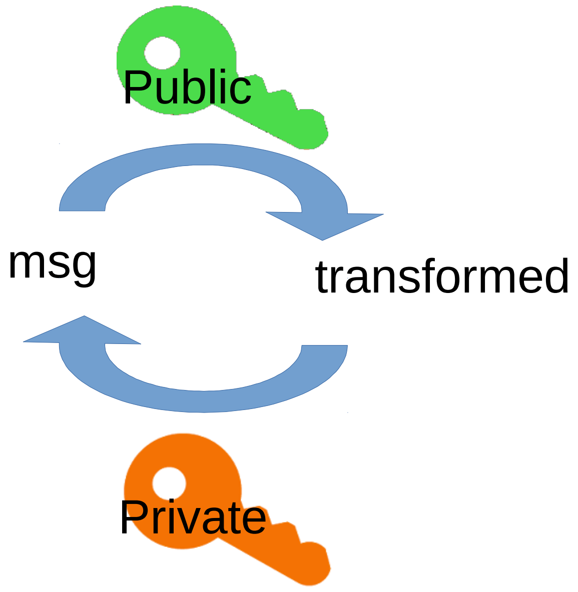
Family of Cryptographic Functions



Assymetric Keys



Public Keys – Private Keys...



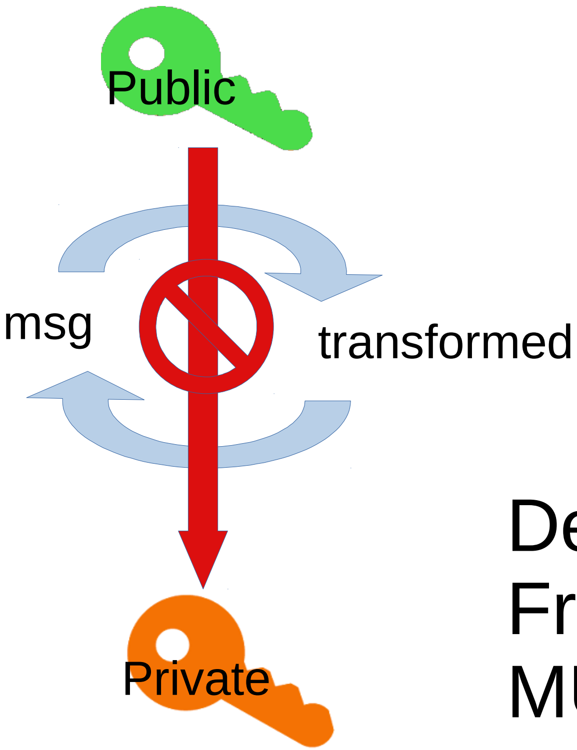
Keys define 2 operations : inverse of each others

Like +X → -X

*X → /X

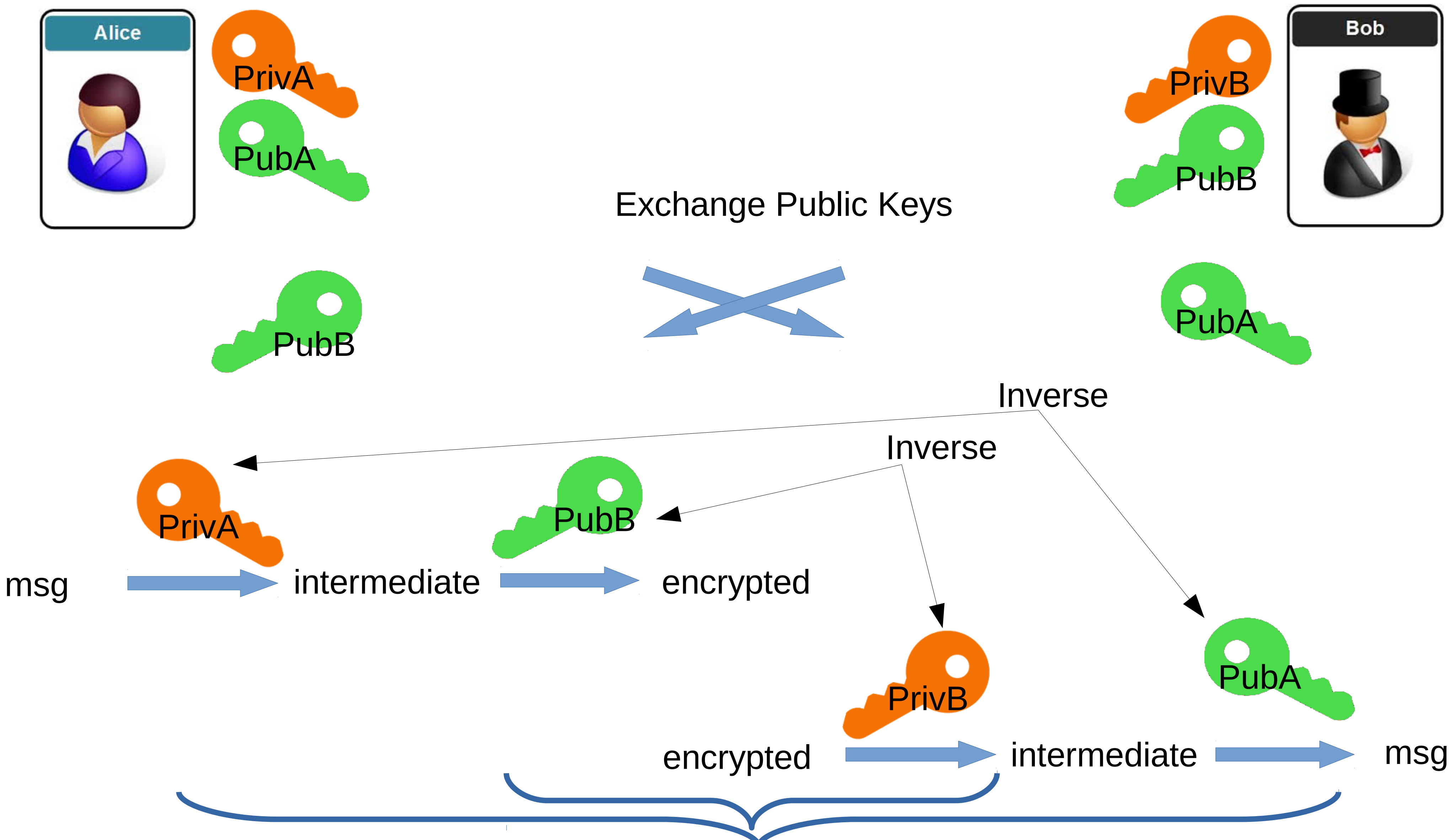
^N → sqrt^N

But Deducing Keys MUST Be Impossible/Difficult

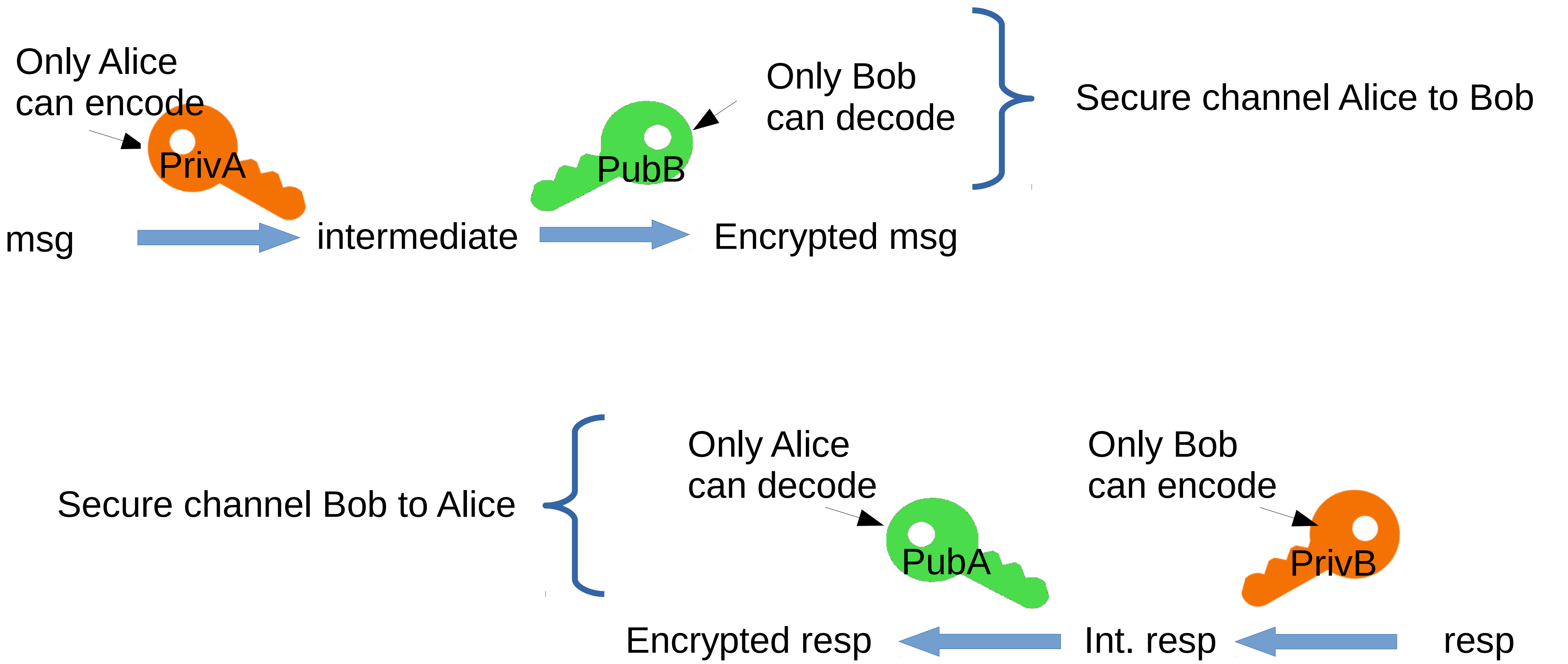


Deducing Private Key
From Public Key
MUST Be Impossible

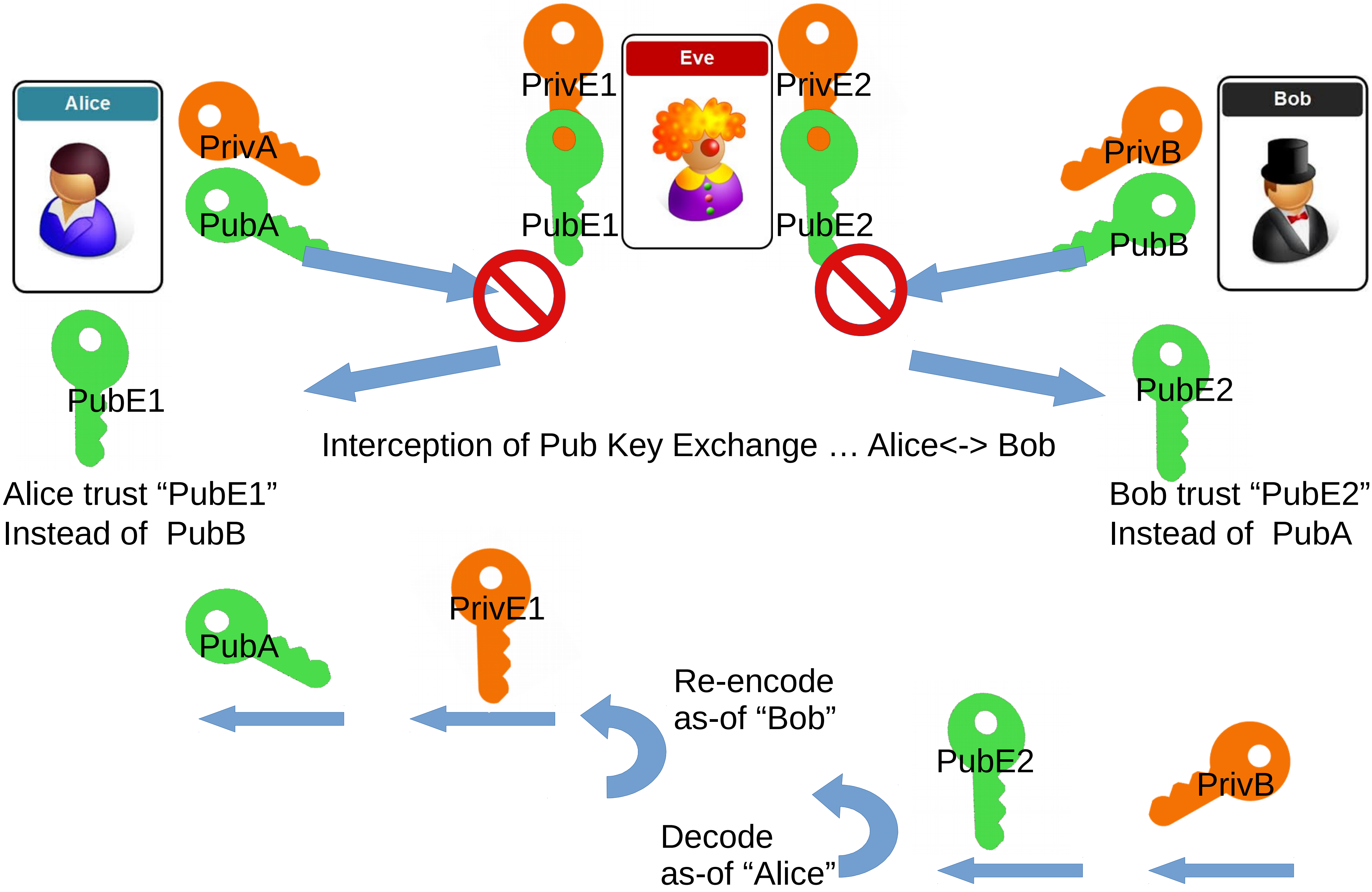
Combining 2 Pairs of Pub-Priv Keys



Assymetric Keys For Secured 2-Way Channel



Man In The Middle...



RSA : Rivest Shamir Adleman

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

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 => there exist(uniq) q & r with 0<=r<n
x = q n + r

Modulo N function: x → x mod[n] = r

Algebra Operation on Modulos

(x + y) mod [n] = (x mod [n] + y mod [n]) mod[n]

(x * y) mod [n] = (x mod [n] * y mod [n]) mod[n]

Reasonning on equivalence classes, for equivalence relation “R”: a R b <=> a-b mod[n]=0

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

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

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

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

Example on modulo 9

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

$$123.456\,mod[9]=56088\,mod[9]=(5+6+0+8+8)\,mod[9]=0$$

$$=(1+2+3).(4+5+6)\,mod[9]=6.6\,mod[9]=0$$

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

$$\left(a_n 10^n + a_{n-1} 10^{n-1} + .. a_2 10^2 + a_1 10 + a_0\right) mod[9] = \left(a_n + .. + a_1 + a_0\right) mod[9]$$

When p is prime
Z/pZ* is a multiplicative Group

Remember Group definition ?

G = { g0, g1, ...gN } a Set of elements
with an operation “.” (inside G)
Is a “group over .”

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

every element has an inverse: $\forall a, \exists b \,|\, a.b = b.a = e$

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

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

$$\left\{\dot{1}, \dot{2}, \dot{3}, \dot{4}, \dot{5}, \dot{6}\right\}$$

Neutral element

Inverse of each others

Self Inverse

$$\dot{2}.\dot{4}=\dot{\overline{8}}=\dot{1} \qquad \dot{3}.\dot{5}=\dot{\overline{15}}=\dot{1} \qquad \dot{6}.\dot{6}=\dot{\overline{36}}=\dot{1}$$

Consecutive Powers … (=Orbits)

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

$$\left\{a, a^2, a^3, a^4, \ldots a^{p-1}, \ldots\right\}$$

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

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

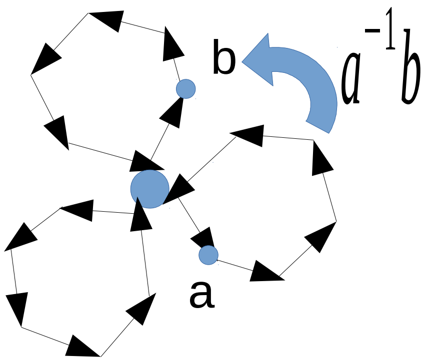
Proof next... that k2-k2=p-1

Orbits are all similar... all equals

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

$$\left\{a, a^{2}, .. a^{p-2}, e\right\}$$

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, a² ...

$$a = \dot{3}$$

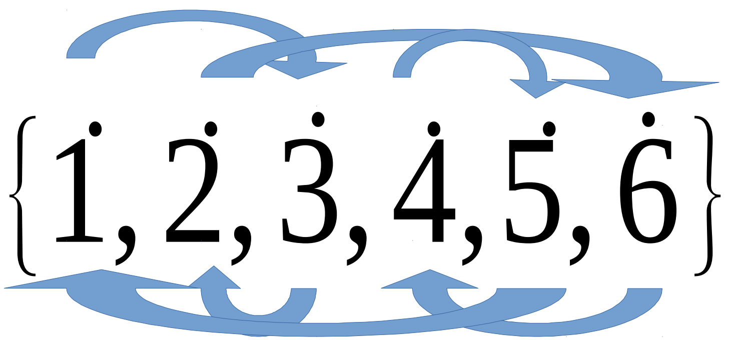
$$a^2 = \dot{3}\,\dot{3} = \dot{\overline{9}} = \dot{2}$$

$$a^3 = \dot{\overline{3.2}} = \dot{6}$$

$$a^4 = \dot{\overline{3.6}} = \dot{4}$$

$$a^5 = \dot{\overline{3.4}} = \dot{5}$$

$$a^6 = \dot{\overline{3.5}} = \dot{1}$$



$$\{\dot{1}, \dot{2}, \dot{3}, \dot{4}, \dot{5}, \dot{6}\}$$

“Small Theorem” of Fermat

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

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

$$a^{p} = a \, mod[p]$$

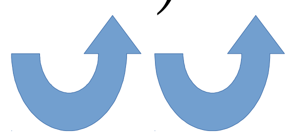
Relationship with Crypto? ...

For p1,p2 2 prime numbers...

$$a^{p1} = a \, mod[p1]$$

$$a^{p2} = a \, mod[p2]$$

$$a^{p1.p2} = (a^{p1})^{p2} = a \, mod[p1 \, p2]$$



encrypt   decrypt

Intuitively … Try decompose p1.p2 as a different product of 2 key parts “e.d mod[..]” … but not as p1.p2 (too easy) !

RSA ...

Technically.. solve e and d...

$$e.d = 1 \, mod[\varphi(p1p2)]$$

Then

$$(a^e)^d = a \, mod[p1p2]$$

And also

$$(a^d)^e = a \, mod[p1p2]$$

Inverting Key
.. need Decomposing N in p1.p2

Choose 2 HUGE primes p1,p2

Multiply N=p1.p2

… give N to someone

and reward 1M$ IF he finds back p1 & p2

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 $p1 > 2^{1024}$

=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] ~ 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~1024

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

Need Efficient Primality Test

$$p \text{ is prime} \Leftrightarrow$$

$$\forall a \in \mathbb{Z} / p\mathbb{Z}^* \qquad a^{p-1} = 1 \bmod [p]$$

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

If != 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)

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

Then

$$a^{p-1}=a^{b_0}.a^{2b_1}....a^{2^n b_n}$$

$$=\begin{Bmatrix} a^1 \mathit{if}\ b_1=1 \\ 1\ \mathit{if}\ b_1=0 \end{Bmatrix}.\begin{Bmatrix} a^2 \mathit{if}\ b_2=1 \\ 1\ \mathit{if}\ b_2=0 \end{Bmatrix}....\begin{Bmatrix} a^{2n} \mathit{if}\ b_n=1 \\ 1\ \mathit{if}\ b_n=0 \end{Bmatrix}$$

square square

Efficient (Enough?) Powers Compute

$P \sim 2^{1024}$ $N=\log2(p)=1024$

$$a^{p-1}$$

Computable in o(2.N multiplications)
BUT Would be a HUGE number …

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

Computable in
o(2.N multiplications + 1.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 Seed Pseudo Generator
… then using XOR encoding

Application : **TLS** protocol (ex SSL) , for example in

Family of Cryptographic Functions

Problem Solving
Proof Of Work

One-Way (Irreversible)
Hashing

2-Way (Reversible)

Permutation
XOR

Hide/Extract
(Steganography)

Key Encrypt-Decrypt

Symmetric Key
Shared

Assymmetric Key
Public-Private

Randomization

Entropy
Generator

Pseudo Random
Generator

Fast Pseudo-Random Generators

Back to Math : Doing Polynomials

$$P[X] = a_0 + a_1 X + a_2 X^2 + .. a_n X^n$$

… Polynomials over Z/2Z modulos, truncated to Rank N

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

… we are interrested 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 :

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

If “P” is chosen irreductible in Z/2Z[n]
… then orbit contains all polynoms!
(all coefficients of all polynoms)

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

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

Multiply (and Truncate) Polynoms

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

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

$$A.B[X]=A[X].B[X]=a_0.b_0+\left(a_0b_1+a_1b_0\right)X$$

$$+\left(a_0b_2+a_1b_1+a_2b_0\right)X^2+..\left(a_0b_n+..a_nb_0\right)X^n$$

$$+truncated..X^{n+1}..X^{2n}$$

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

$$A[X]=a_0+a_1 X+a_2 X^2+..a_n X^n$$

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

$$A.B[X]=a_0+(a_0+a_1)X+(a_1+a_2)X^2+...$$

$$
\begin{array}{rl}
 & a_1\ a_2\ a_3\ \ldots\ a_{n-1}\ a_n \\
+ & a_2\ a_3\ a_4\ \ldots\ a_n\ 0 \quad \Leftarrow \text{Bit Shift} \\
\hline
 & (a_1+a_2)(a_2+a_3)..(a_{n-1}+a_n)\ a_n
\end{array}
$$

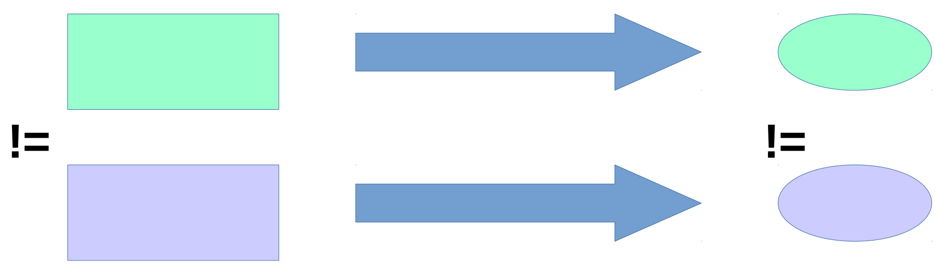
Family of Cryptographic Functions

- Problem Solving
  Proof Of Work
- One-Way (Irreversible)
  Hashing
- 2-Way (Reversible)
  - Permutation
    XOR
  - Hide/Extract
    (Steganography)
  - Key Encrypt-Decrypt
    - Symmetric Key
      Shared
    - Assymmetric Key
      Public-Private
- Randomization
  - Entropy
    Generator
  - Pseudo Random
    Generator

Hashing ...

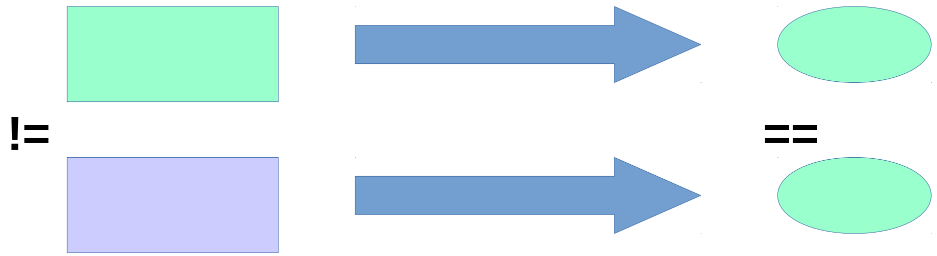
Hashing is One-Way transformation

Hashing different product **usually** give different result



Hashing … must be equi-distributed minimise “Collisions”

A Hashing Collision



Of course if 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 h

MD5

From Wikipedia, the fr

(Redirected from Md5

The **MD5 algorithn**

hash value. Althoug

cryptographic hash fu

vulnerabilities. It can

but only against unir

Like most hash funct

cracked by brute-forc

detailed in the securi

MD5 was designed b

function MD4.[3] 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

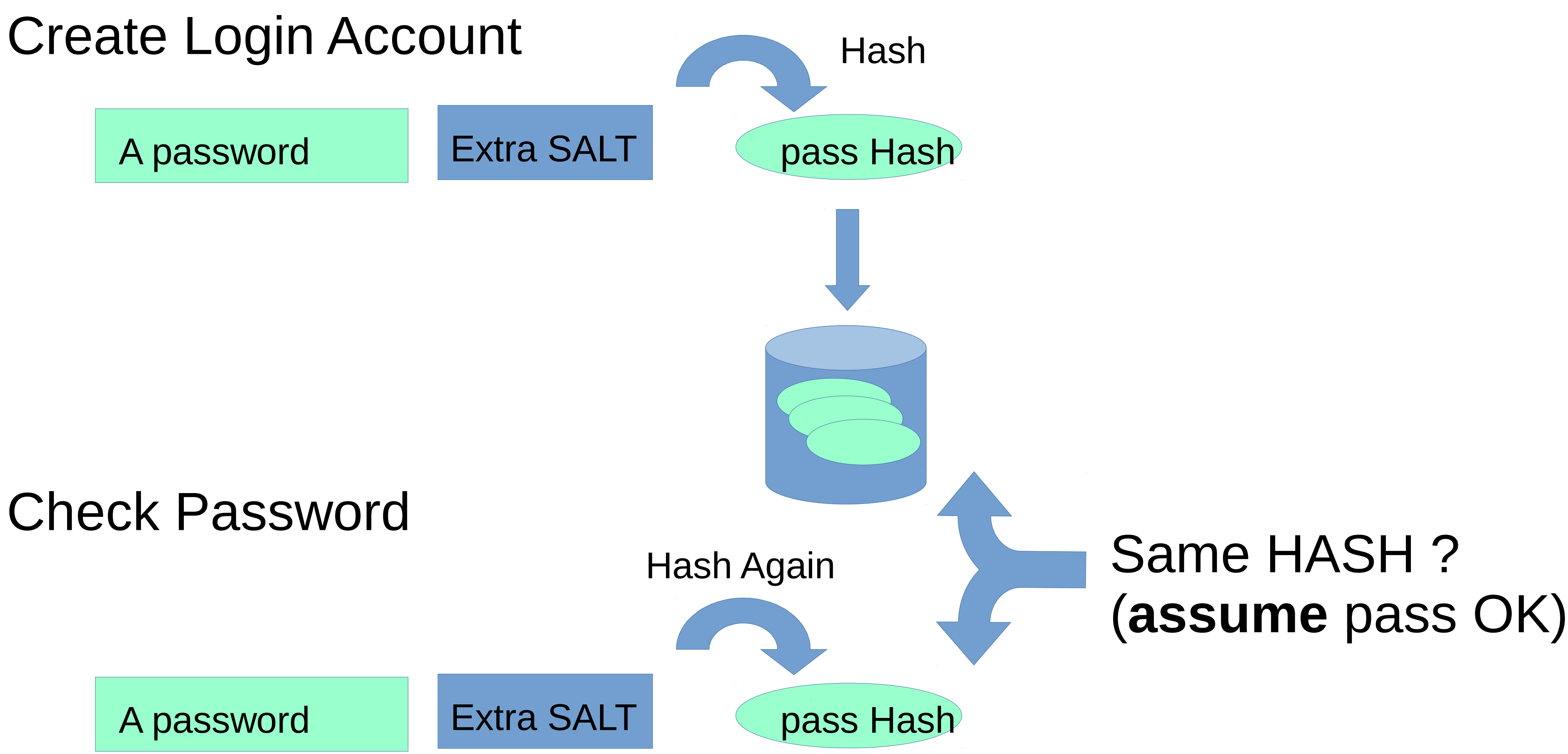
SHA-1

From Wikipedia, the free encyclopedia

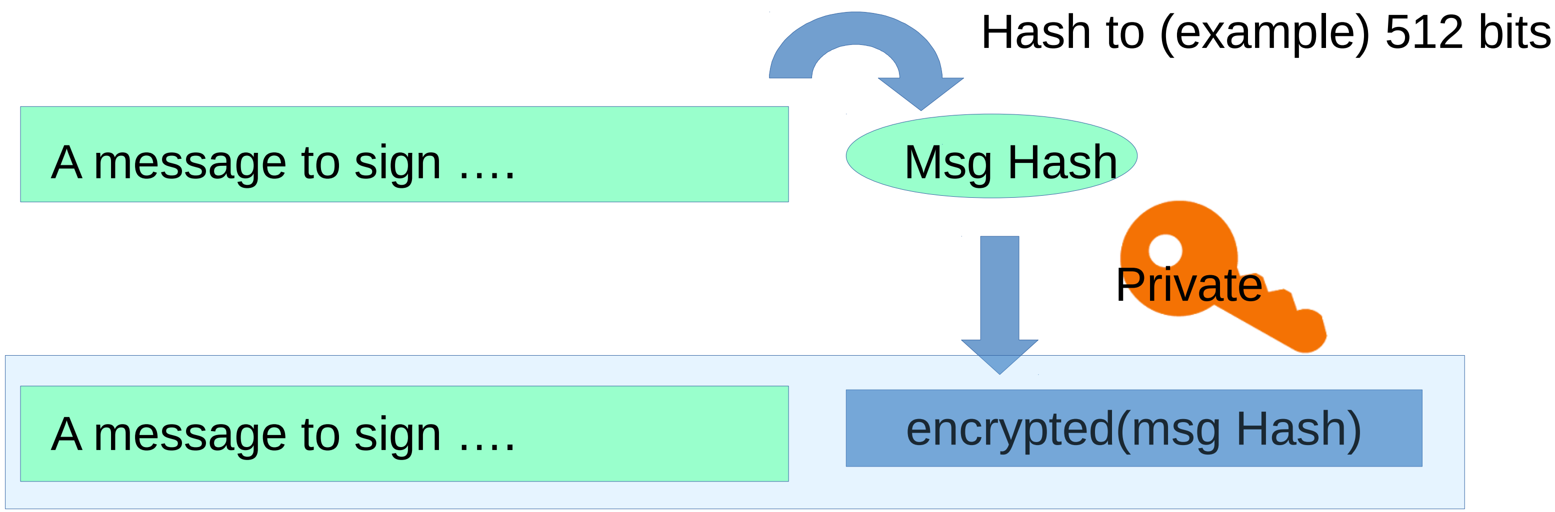
In cryptography, **SHA-1** (**Secure Hash Algorithm 1**) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST.[3] SHA-1 produces a 160-bit (20-byte) hash value known as a message 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”

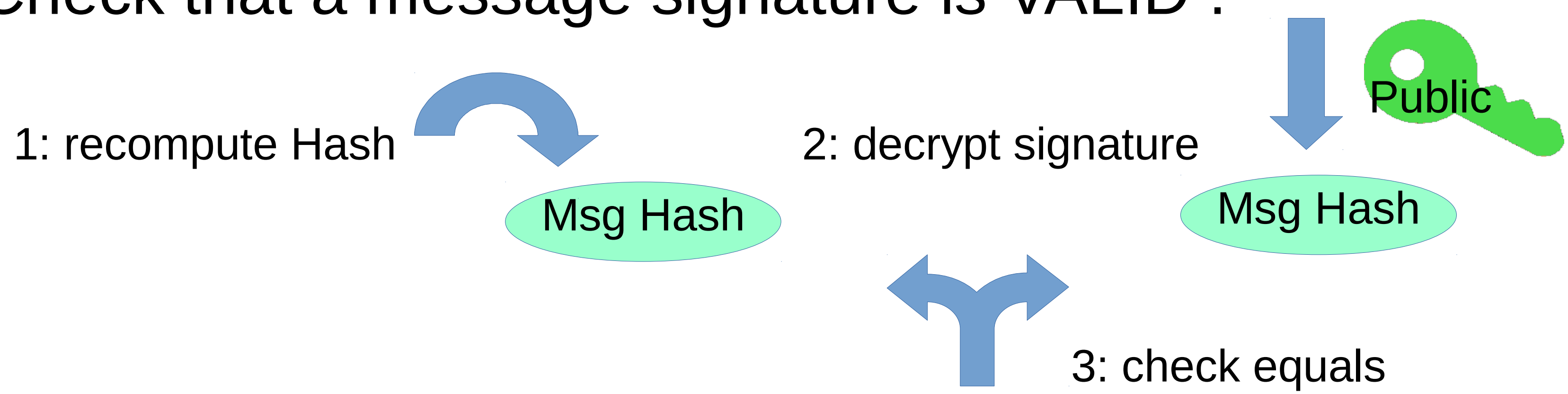


Hashing : Good for Signing

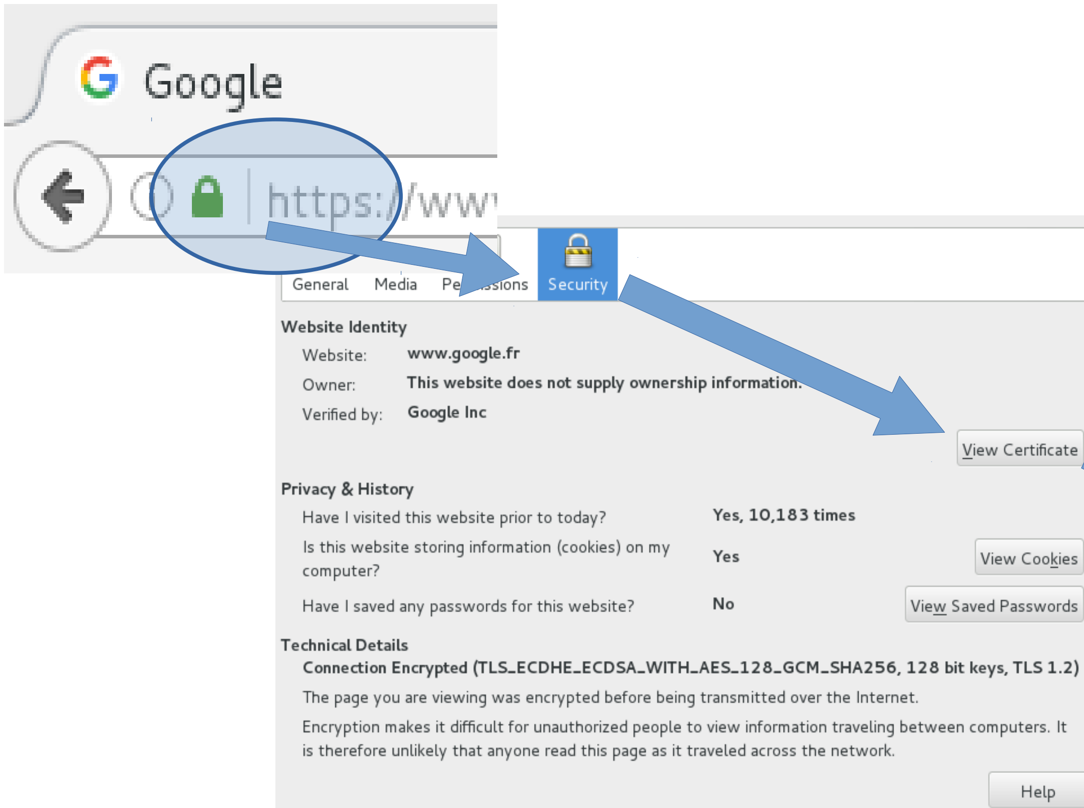


Signed Message : clear text + extra signature (at bottom of page)

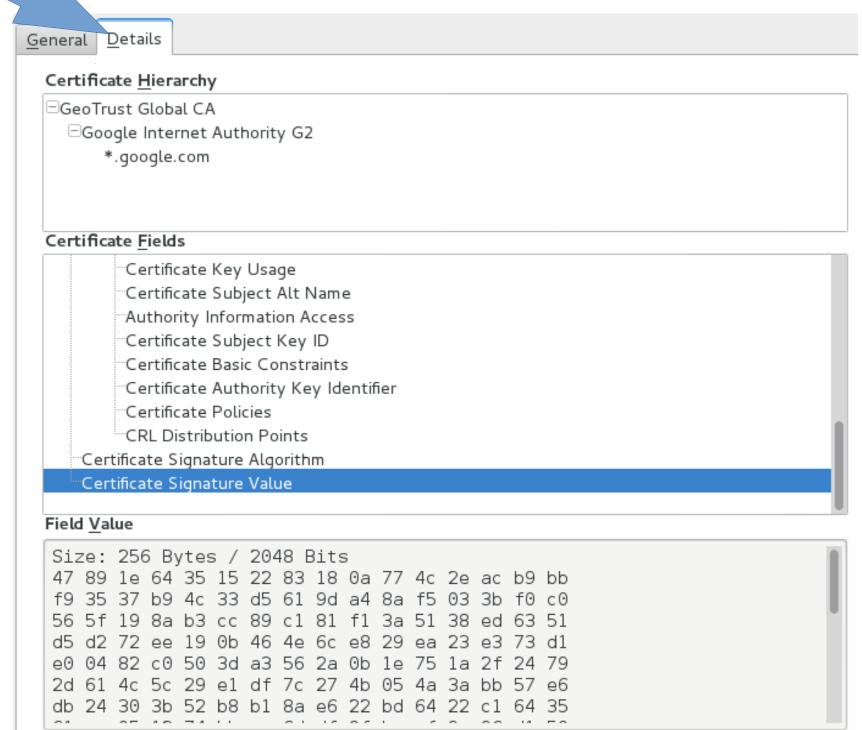
Check that a message signature is VALID :



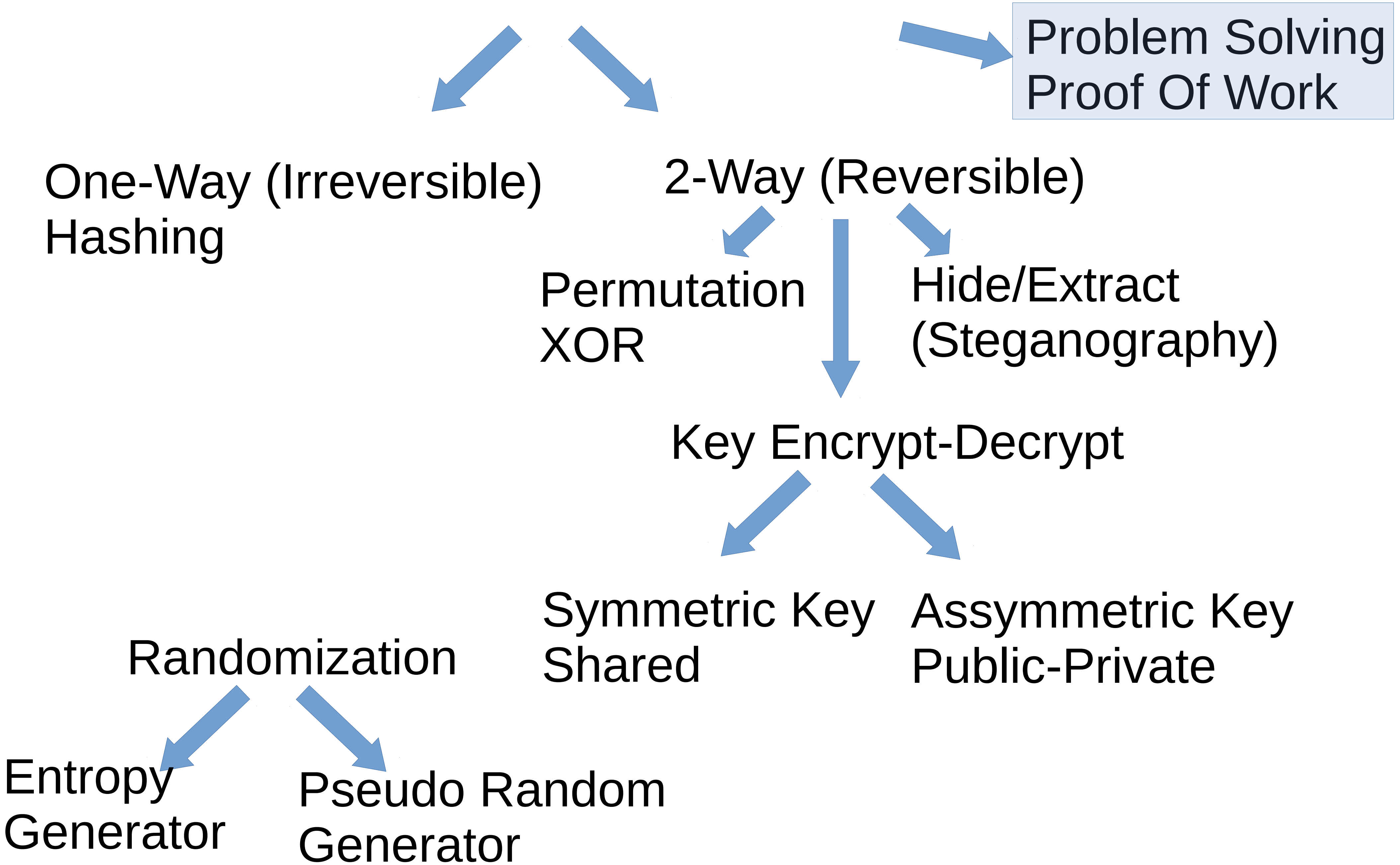
Certificates = Signed Public Keys



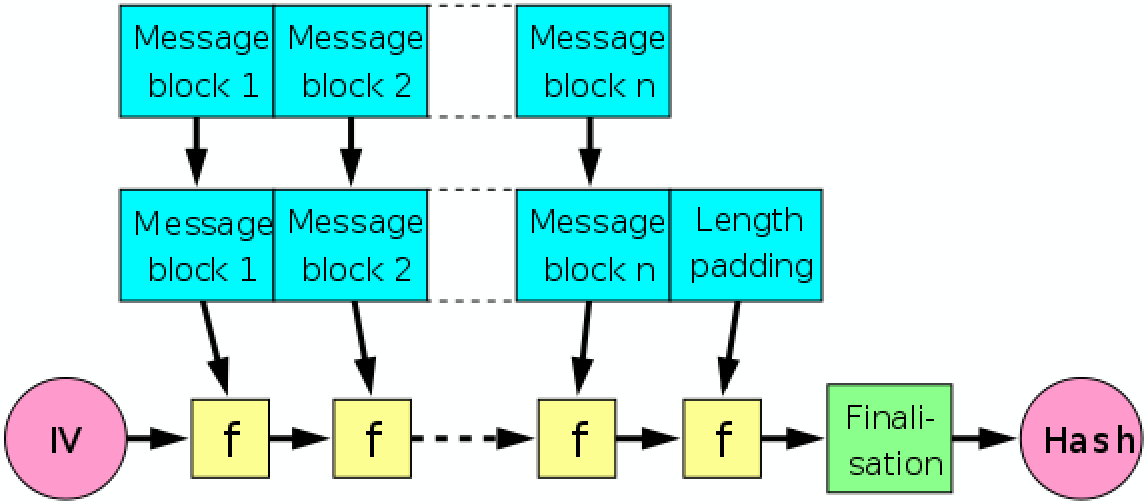
Certificate Public Key
Certificate Signature ...



Family of Cryptographic Functions



Hash a BitCoin Transactions Chain BlockChain



Merkle Tree

Article Talk | Read Edit View history | Search Wikipedia

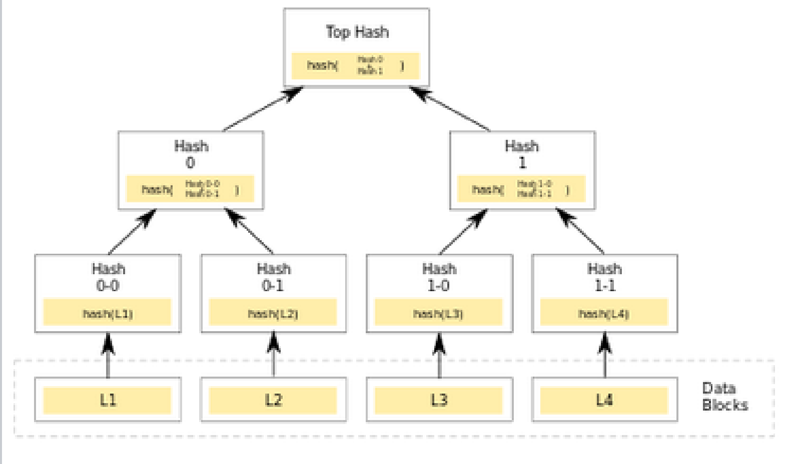
Merkle tree

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

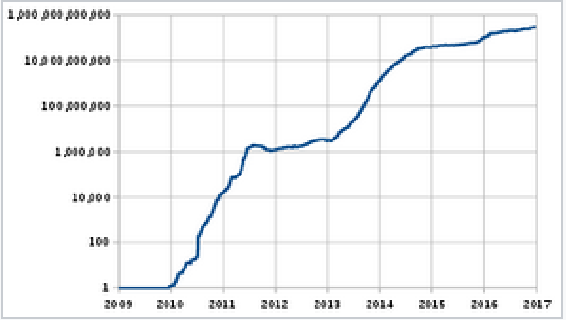


An example of a binary hash tree. Hashes 0-0 and 0-1 are the hash values of data blocks L1 and L2, respectively, and hash 0 is the hash of the concatenation of hashes 0-0 and 0-1.

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.



Semi-log plot of relative mining difficulty.[note 4][32]

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


Shared
Modulo
Des
Authentication
Blockchain
Permutations
Fermat
Bitcoin
Functions
Predicatable
Encrypted
Malicious
Shamir
Password
Bob
Naive
Powers
Hash
Secured
Session
Leak
Challenge
Eve
Math
Public
Efficient
Rsa
Rivest
Physical
Sha
Key
Compute
Alice
Exchange
Xor
Crypt
Cryptography
Prime
Middle
Trust
Aes
Denied
Random
Leak
Cryptographic
Inversion
Assymetric
Adleman
Generator
Partial
Weak
Introduction
Problems
Eve