Cryptography

Asymmetric cryptography

Motivation

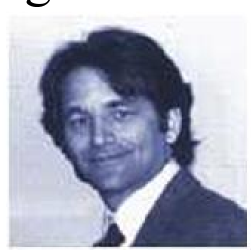
- Problem with symmetric crypto-system
  - Requires key exchange: difficult in practice
  - Authentication/non-repudiation: how to verify that a message comes intact from the claimed sender? Impossible to achieve

- Asymmetric crypto-system: a public / private key pair is used
  - public key known to everyone
  - private key known only by owner
  - Much slower to compute than secret key cryptography

History

- 1977: separates classical and modern eras
- Whitfield Diffie and Martin Hellman

  - Public-key encryption schemes
  - **Diffie-Hellman key agreement protocol**
  - Digital signature
  - Turing Award 2015: *For fundamental contributions to modern cryptography. Diffie and Hellman's groundbreaking 1976 paper, "New Directions in Cryptography," introduced the ideas of public-key cryptography and digital signatures, which are the foundation for most regularly-used security protocols on the internet today.*
- Ron Rivest, Adi Shamir, Len Adleman
  - **RSA algorithm:** Turing Award 2002



History*

- James Henry Ellis (1924-1997):
  - British engineer & cryptographer
  - Born in Australia, grown up in Britain, orphan
  - Worked at Government Communications Headquarters (GCHQ)
  - 1970: proposed the concept of public-key crypto
    - Same idea of RSA
    - Kept secret by GCHQ
  - 1973: with Clifford Cocks and Malcolm Williamson, developed a key distribution scheme
    - Similar as Diffie-Hellman key agreement protocol
    - Kept secret by GCHQ
  - 1976: denied publication by GCHQ again
  - Diffie travelled to Britain to see Ellis
  - Dec. 1997: public announcement of their work, Cocks delivered a talk
    - Nov. 1997: death of Ellis
  - 2016: GCHQ director emphasized contribution of Ellis, Cocks, Williamson

Misconception of asymmetric systems

- Asymmetric encryption is more secure than symmetric ones
- Public-key encryption is a general-purpose technique that has made symmetric encryption obsolete
- Key distribution is trivial in public-key encryption

Asymmetric system requirements

- Computationally
  - **easy** to generate a public / private key pair
  - **hard** to determine private key from public key

- Computationally
  - **easy** to encrypt using public key
  - **easy** to decrypt using private key
  - **hard** to recover plaintext from ciphertext and public key

Trapdoor one-way function

- Trapdoor one-way function
  - $Y=f_k(X)$: easy to compute if k and X are known
  - $X=f^{-1}_k(Y)$: easy to compute if k and Y are known
  - $X=f^{-1}_k(Y)$: hard if Y is known but k is unknown

- Goal of designing asymmetric algorithm is to find appropriate trapdoor one-way function

RSA

- The most popular public key method
  - Encryption and signature
- Mathematic basis:
  - **factorization of large numbers is hard**
- Variable key length (1024 bits or greater)

A mini-math detour: modular arithmic

- a mod n
  - the rest of a divided by n
  - E.g., 23 mod 7 = 2

- Some properties
  - a + b mod n = a mod n + b mod n
  - a * b mod n = (a mod n) * (b mod n)
  - 7^{25} mod 10 = ?

Greatest Common Divisor

- gcd(576, 135) = gcd(135, 36) = gcd(36, 27) = gcd(27, 9) = 9

- Euclidean algorithm
  - 576 = 4 * 135 + 36
  - 135 = 3 * 36 + 27
  - 36 = 1 * 27 + 9
  - 27 = 3 * 9 + 0

Extended Euclidean algorithm

- Theorem: given nonzero a and b, there exist x and y such that
  - ax + by = gcd(a, b)
  - Linear Diophantine equation in two variables

- Proof: extended Euclidean algorithm (576,135)

$$
\begin{aligned}
576 &= 4 \cdot 135 + 36 \\
135 &= 3 \cdot 36 + 27 \\
36 &= 1 \cdot 27 + 9 \\
27 &= 3 \cdot 9 + 0
\end{aligned}
$$

$$
\begin{aligned}
36 &= 576 - 4 \cdot 135 \\
27 &= 135 - 3 \cdot 36 \\
9 &= 36 - 1 \cdot 27
\end{aligned}
$$

$$
\begin{aligned}
9 &= 36 - 27 = 36 - (135 - 3 \cdot 36) = -135 + 4 \cdot 36 \\
&= -135 + 4 \cdot (576 - 4 \cdot 135) = 4 \cdot 576 - 17 \cdot 135
\end{aligned}
$$

Division mod n

- Corollary: if gcd(a, n) = 1, then there exists x
  - such that ax = 1 mod n
- x=1/a mod n: division mod n
  - Possible if gcd(a, n) = 1
- Proof
  - For any a, b, there exists x, y
    - ax + by = gcd(a, b)
  - Let b=n: ax+bn=1
    - ax = 1 mod n

Division mod n: example

- Solve 5x = −4 mod 11
  - Possible: gcd(5, 11) = 1

- Run extended Euclead algorithm
  - −2*5 + 1*11 = 1 implies −2 = 1/5 mod 11

  - 5x = −4 mod 11
  - -2*(5x)=8 mod 11
  - -11x+x=8 mod 11
  - x=8 mod 11

- Counterexample: solve 5x = −4 mod 10
  - Impossible: gcd(5, 10) = 5

Fermat's little theorem

- Given a prime p, an integer a non divisible by p, we have
  - $a^{p-1} \bmod p = 1$
  - Proof
    - $a*i \neq a*j \bmod p$, hence $(a*1)(a*2)...(a*(p-1)) = (p-1)! \bmod p$

- Exemple: p=3, a=2, $2^{3-1} \bmod 3 = 1$

Euler's totient function φ(n)

- φ(n) is the number of integers $1 \leq x \leq n$ such that $\gcd(x, n) = 1$
  - $\varphi(p) = p - 1$ if p is prime
  - $\varphi(pq) = (p - 1)(q - 1)$
  - Exemple: $\varphi(10) = 4$
- Euler's theorem: if $\gcd(a, n) = 1$, then $a^{\varphi(n)} = 1 \bmod n$
  - Proof: for x_i, x_j co-prime to n: $a*x_i \mathrel{!}= a*x_j \bmod n$,
  - Hence $(a*x_1)(a*x_2)...(a*x_{\varphi(n)}) = x_1*x_2*... *x_{\varphi(n)} \bmod n$
- Generalization of Fermat's little theorem
  - Given 2 primes p, q, and a mod (p-1)(q-1) = 1, we have
    - $x^a \bmod pq = x$, for any $x < pq$
    - Proof: ?
  - Example : a=9, p=3, q=5, pq=15, (p-1)(q-1)=8
  - x=1 : $1^9 \bmod 15 = 1$
  - x=2 : $2^9 \bmod 15 = 512 \bmod 15 = 2$

- End of math mini-detour

RSA algorithm

- M: plaintext;
- C: ciphertext

- Encryption
  - $C = M^e \bmod n$

- Decyption
  - $M = C^d \bmod n$

- Public key
  - (e,n)
- Private key
  - (d,n).

RSA: key setup

- Find large primes p and q
- Let n = p*q
  - Do not disclose p and q!
- Choose an e that is relatively prime to (p-1)(q-1)
  - public key = (e,n)
- Find d such that e*d mod (p-1)(q-1) = 1
  - $d = e^{-1} \bmod (p-1)(q-1)$
  - private key = (d,n)

- Encryption
  - $C = M^e \bmod n$
- Decryption
  - $M' = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n = M$
  - Following Fermat's theorem

RSA: example

- $p=3$, $q=11$
- $n=pq=3*11=33$
- $(p-1)(q-1)=2*10=20$
- $e=3$, $d=7$, $ed \bmod (p-1)(q-1) = 1$
  - $3*7 \bmod 20 = 1$

- $M = 29$
- $C=M^e \bmod n=29^3 \bmod 33=2$
- $M'=C^d=2^7 \bmod n=29=M$

- Test : $p=5$; $q=11$, $e=3$; $M=4$

RSA: Security

- Public key (e,n) is public information

- If one could factor n into p*q, then
  - could compute (p-1)(q-1)
  - could compute $d = e^{-1} \bmod (p-1)(q-1)$
  - would know private key (d,n)!

- But: factoring large integers is hard!
  - Classical problem worked on for centuries; no known fast method
  - Test: try to factor 2419 ? 373247 ?? 96171919154952919 ???

- Is RSA as strong as factorization?

RSA Factoring Challenge

- Launched by RSA Lab in 1991

RSA-232 [*]	232	768		February 17, 2020[9]	N. L. Zamarashkin, D. A. Zheltkov and S. A. Matveev.
RSA-768 [*]	232	768	US$50,000	December 12, 2009	Thorsten Kleinjung *et al.*
RSA-240 [*]	240	795		Dec 2, 2019[10]	F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann
RSA-250 [*]	250	829		Feb 28, 2020[11]	F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann
RSA-260	260	862			
RSA-270	270	895			
RSA-896	270	896	US$75,000		
RSA-280	280	928			
RSA-290	290	962			
RSA-300	300	995			
RSA-309	309	1024			
RSA-1024	309	1024	US$100,000		
RSA-310	310	1028			

RSA: Security

- At present, key sizes of 1024 bits are considered secure
  - but 2048 bits is better

- Tips for making n difficult to factor
  - p and q lengths should be similar (ex.: ~500 bits each if key is 1024 bits)
    - but not too close
      - Fermat factorization method: $n = x^2 - y^2 = (x + y)(x - y)$
  - both (p-1) and (q-1) should contain a "large" prime factor
    - Pollard $(p - 1)$ factorization
  - gcd(p-1, q-1) should be "small"
  - d should be larger than $n^{1/4}$

RSA implementation

- Select primes p and q
  - In practice, select random numbers, then test for primality
    - Prob. a randomly chosen number n being prime ≈ $1/\ln(n)$
  - Many implementations use the Rabin-Miller test

- Fermat test: theory
  - Fermat's little theorem: for prime p: $a^{p-1} = 1 \bmod p$

- Fermat test: algorithm
  - For i=1 to k
    - pick a randomly
    - if $a^{n-1} != 1 \bmod n$ output "n is composite"
  - EndFor
  - output "possible prime"

Carmichael numbers

- Carmichael numbers are composites that can pass Fermet test
  - Bad news: there are infinite many Carmichael numbers
    - First three: 561, 115, 1729

Carmichael numbers

- Bertrand postulate: there are infinite number of Carmichael numbers
  - Given a sufficiently large number x, there is a Carmichael number between x and 2x

NUMBER THEORY

Teenager Solves Stubborn Riddle About Prime Number Look-Alikes

In his senior year of high school, Daniel Larsen proved a key theorem about Carmichael numbers — strange entities that mimic the primes. "It would be a paper that any mathematician would be really proud to have written," said one mathematician.

RSA implementation

- Select e
  - e is usually chosen to be 3 or $2^{16} + 1 = 65537$
    - Why e cannot be 2?
  - Binary: 11 or 10000000000000001
  - In order to speed up the encryption
    - the smaller the number of 1 bits, the better

Square-and-Multiply Algorithm

- Directly computing $x^e \bmod n$ is very slow
- An efficient algorithm is called **square-and-multiply** algorithm
  - Let $e = e_k e_{k-1} \ldots e_0$ denote the binary expression of e
  - $e = ((((e_k \times 2 + e_{k-1}) \times 2 + e_{k-2}) \times 2 + e_{k-3}) \times 2 \ldots + e_1) \times 2 + e_0$

square-and-multiply algorithm

$z \leftarrow 1$

for $i \leftarrow k$ downto 0 do

$\quad z \leftarrow z^2 \bmod n$

$\quad\quad z \leftarrow z \times x^{e_i} \bmod n$

return z

Example: $11^{23} \bmod 187$

$23 = 10111_b$

$z \leftarrow 1$

$z \leftarrow z^2 \cdot 11 \bmod 187 = 11$ (square and multiply)

$z \leftarrow z^2 \bmod 187 = 121$ (square)

$z \leftarrow z^2 \cdot 11 \bmod 187 = 44$ (square and multiply)

$z \leftarrow z^2 \cdot 11 \bmod 187 = 165$ (square and multiply)

$z \leftarrow z^2 \cdot 11 \bmod 187 = 88$ (square and multiply)

Compute d

- ed mod (p-1)(q-1) = 1
  - Linear Diophantine equation in two variables
- Extended Euclidean algorithm

$$89 \times k \equiv 1 \bmod 197 \qquad 89k = 1 + 197l$$

$$\begin{aligned} 197 &= 89 \times 2 + 19 \\ 89 &= 19 \times 4 + 13 \\ 19 &= 13 \times 1 + 6 \\ 13 &= \ \ 6 \times 2 + 1 \end{aligned}$$

$$\begin{aligned} 1 &= 13 - 6 \times 2 \\ &= 13 - (19 - 13 \times 1) \times 2 && = 19 \times (-2) + 13 \times 3 \\ &= 19 \times (-2) + (89 - 19 \times 4) \times 3 && = 89 \times 3 + 19 \times (-14) \\ &= 89 \times 3 + (197 - 89 \times 2) \times (-14) && = 197 \times (-14) + 89 \times 31 \end{aligned}$$

Attacks against RSA: message guessing

- Alice and Bob are lovers.
  - When they are happy, the message between them is often “I love you”
  - Otherwise, “I hate you”

- Attacker can guess m and test each guess
  - because e is public

- How to prevent this attack?
  - Include a random number in the message

Attacks against RSA: timing attack

Algorithm: Square-and-multiply (x, n, $c = c_{k-1} c_{k-2} \ldots c_1 c_0$)

```
z=1
for i = k-1 downto 0 {
        z = z^2 mod n
        if c_i = 1 then z = (z * x) mod n
}
return z
```

Test:

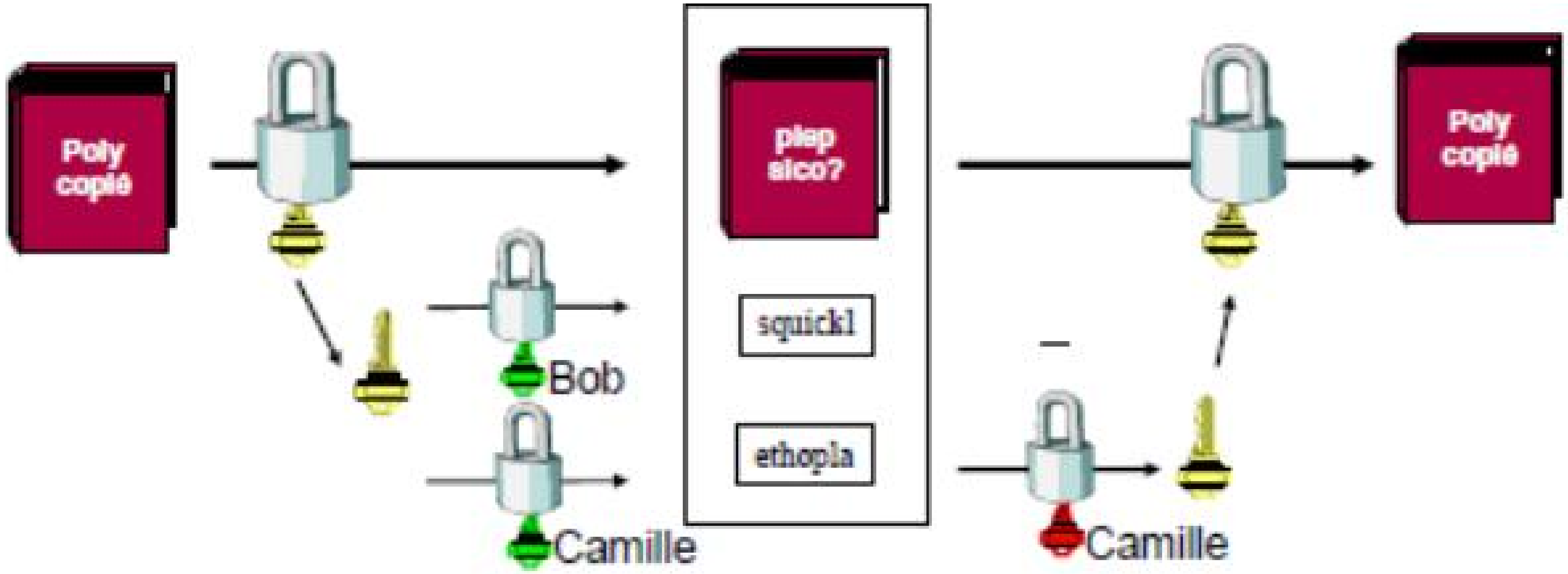
- Show that RSA is not resistant against chosen ciphertext attack
- Attacker disposes C, can access any plaintext M' corresponding to C' as long as C' != C
- Show that attacker can get M

Hybrid system

- Generate a session key
  - randomly-generated key for one communication session
- Use a public key algorithm to send the session key
- Use a symmetric algorithm to encrypt data with the session key

RSA application: mixed encryption

- Send efficiently an encrypted message to multiple destinations
  - Use a secret key K to encrypt M
  - Encrypt K with public keys of destinations



Using RSA for secret key negotiation

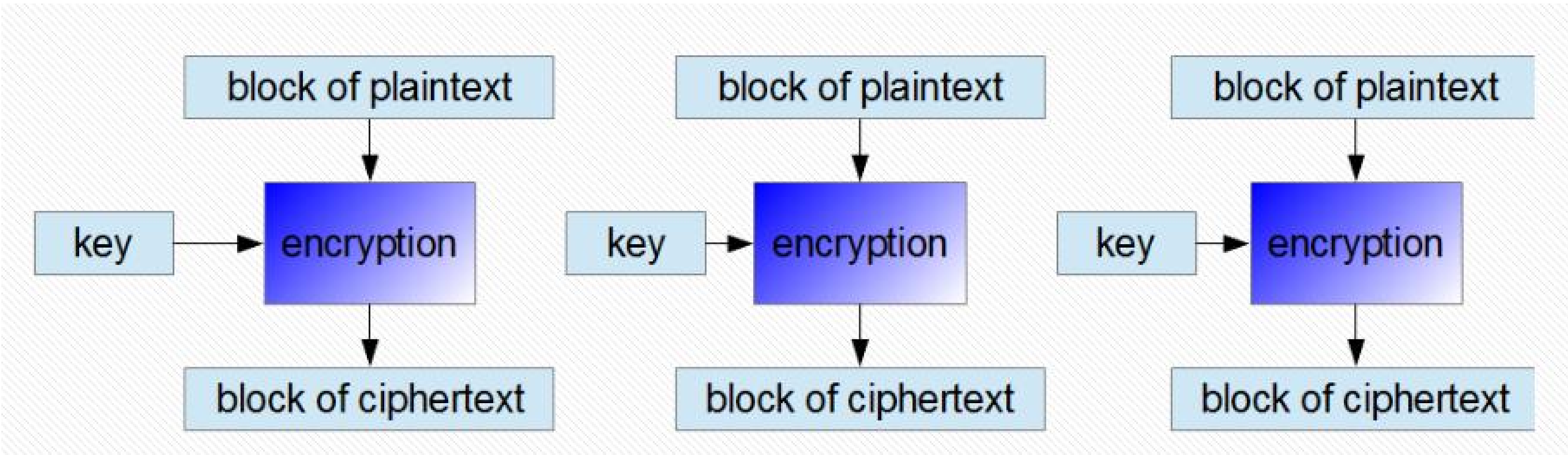
- A sends random number R_1 to B, encrypted with B's public key
- B sends random number R_2 to A, encrypted with A's public key
- A and B both decrypt received messages using their respective private keys
- A and B both compute $K = H(R_1||R_2)$, and use K as shared key

Block Ciphers Features

- Block size
- Key size
- Number of rounds
- Operation mode
  - How large messages are encrypted
  - Important for security

- NIST defines five operation modes
  - Electronic codebook mode (ECB)
  - Cipher block chaining mode (CBC) – most popular
  - Cipher feedback mode (CFB)
  - Output feedback mode (OFB)
  - Counter mode (CTR)

Electronic Code Book (ECB)



- Only strength
  - Decrypt any block independently
- Not secure
  - A file containing salairies
  - Using ECB

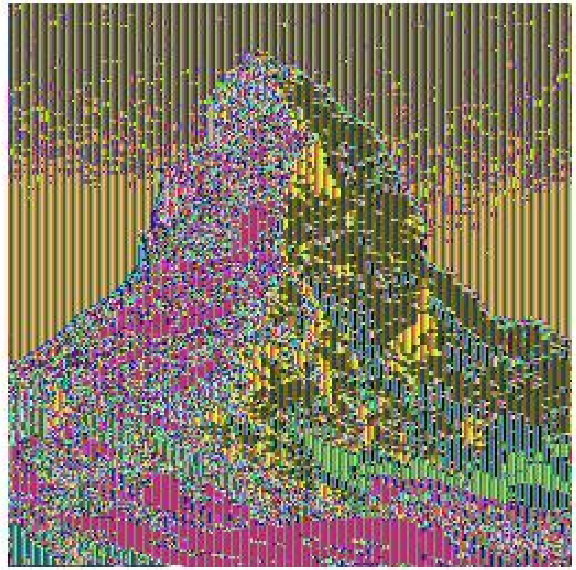
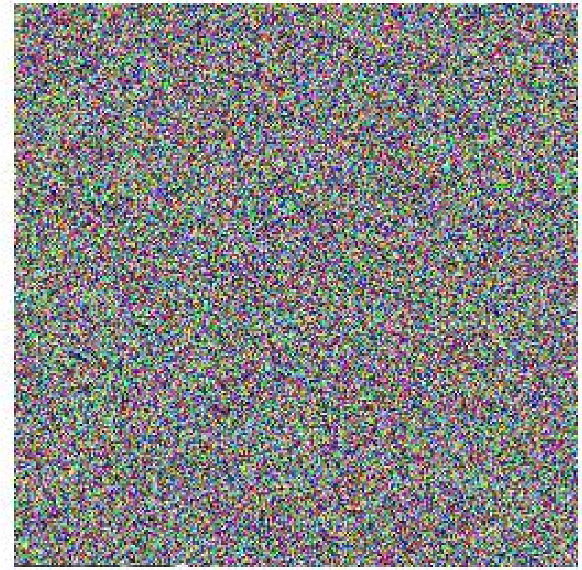
```
JOHN__105000
JACK__500000
```

```
JO|HN|__|10|50|00
Q9|2D|FP|VX|C9|IO
```

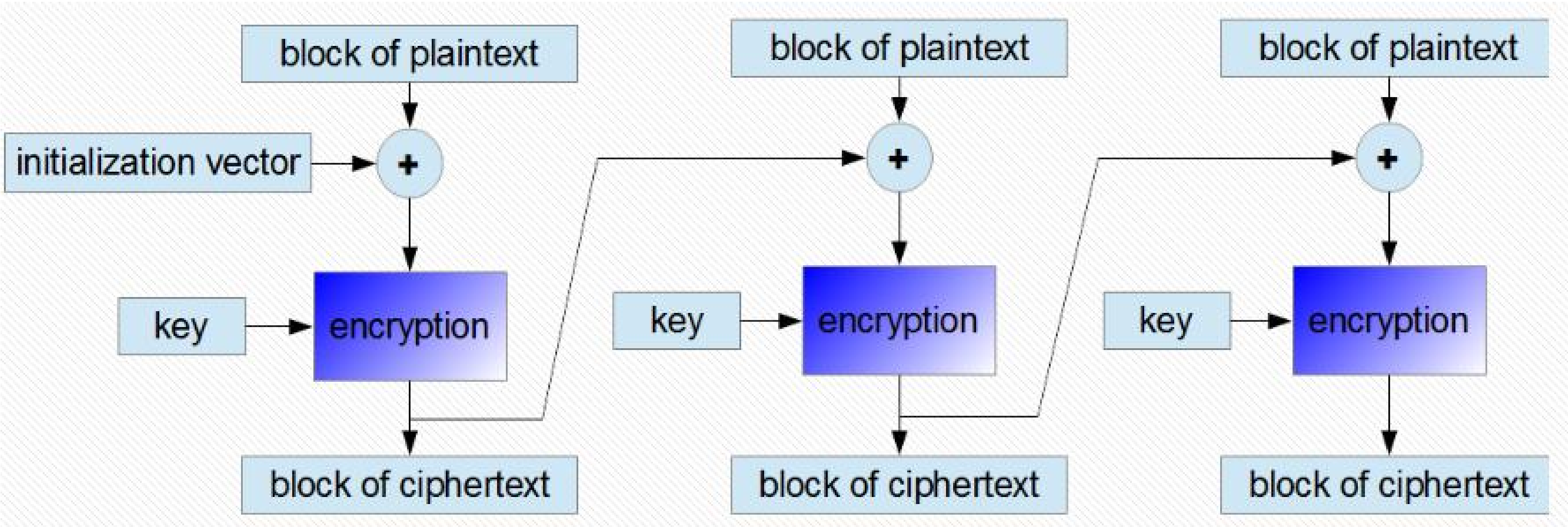
```
JA|CK|__|50|00|00
LD|AS|FP|C9|IO|IO
```

```
Q9|2D|FP|VX|C9|IO
LD|AS|FP|C9|IO|IO
```

ECB: insecurity

Cipher Block Chaining (CBC)



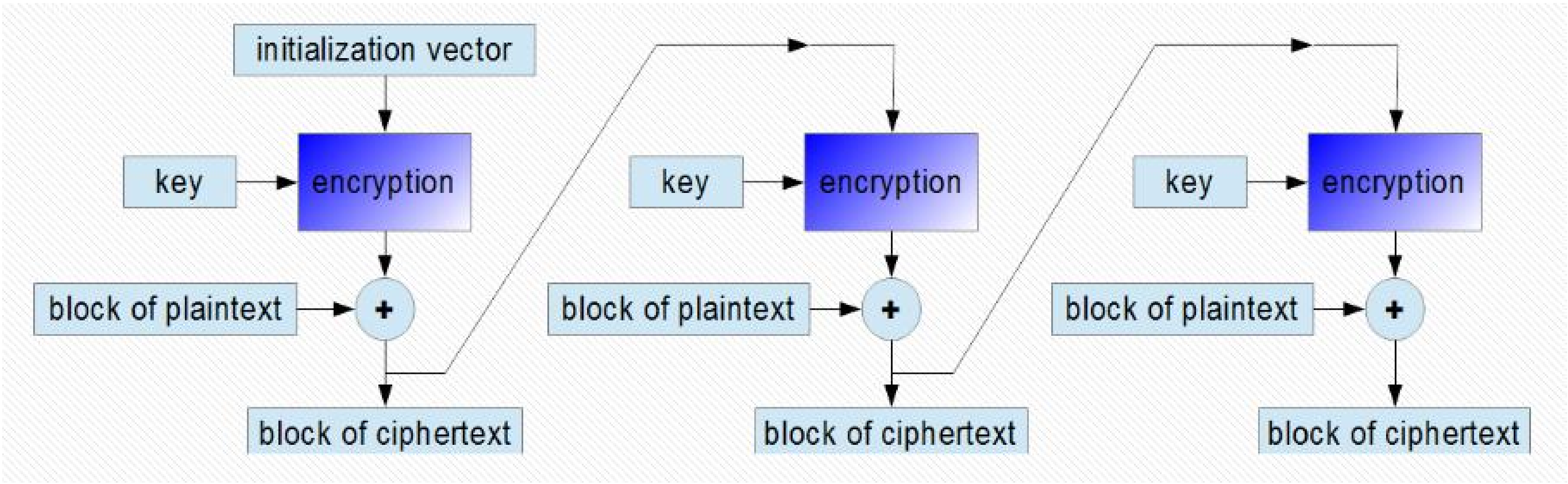
- Strength
  - The encryption of a block depends on current and blocks before
    - Repeated plaintext blocks are encrypted differently
- Weakness
  - Errors in one block propagate to two blocks
    - error in C_j affects M_j and M_{j+1}
  - Encryption cannot be parallelized (decryption: yes)

Initialization Vector

- Initialization Vector (IV)
  - Used along with the key; not secret
  - For a given plaintext, changing either the key, or IV, will produce a different ciphertext

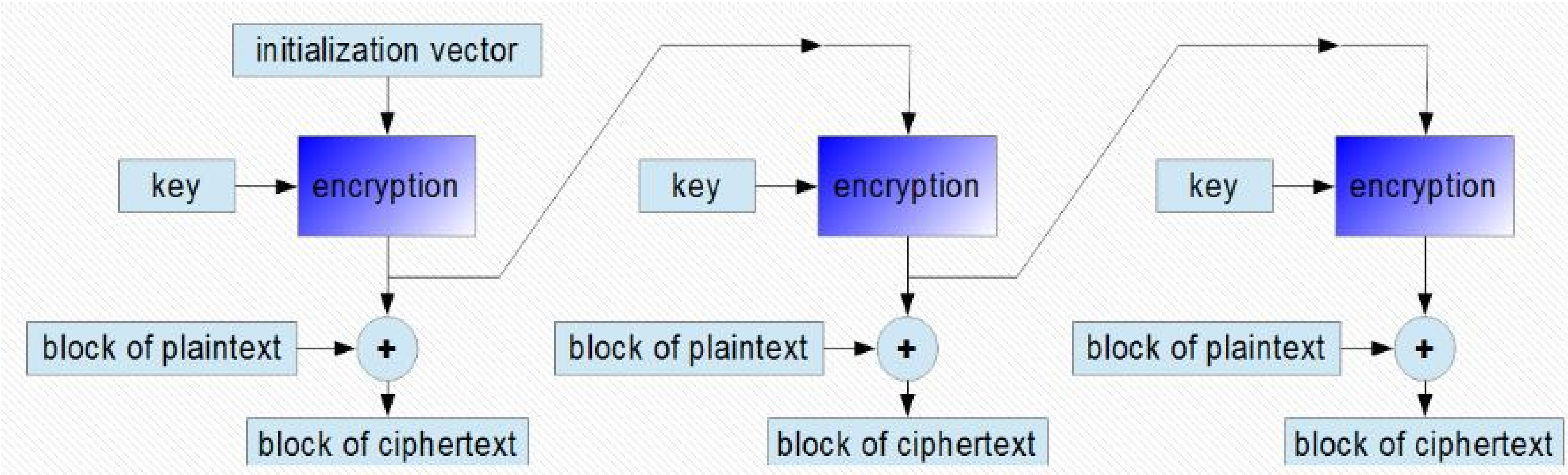
- IV generation and sharing
  - Random; may transmit with the ciphertext
  - Incremental; predictable by receivers

Cipher feedback mode (CFB)



- Compare with One-Time-Pad

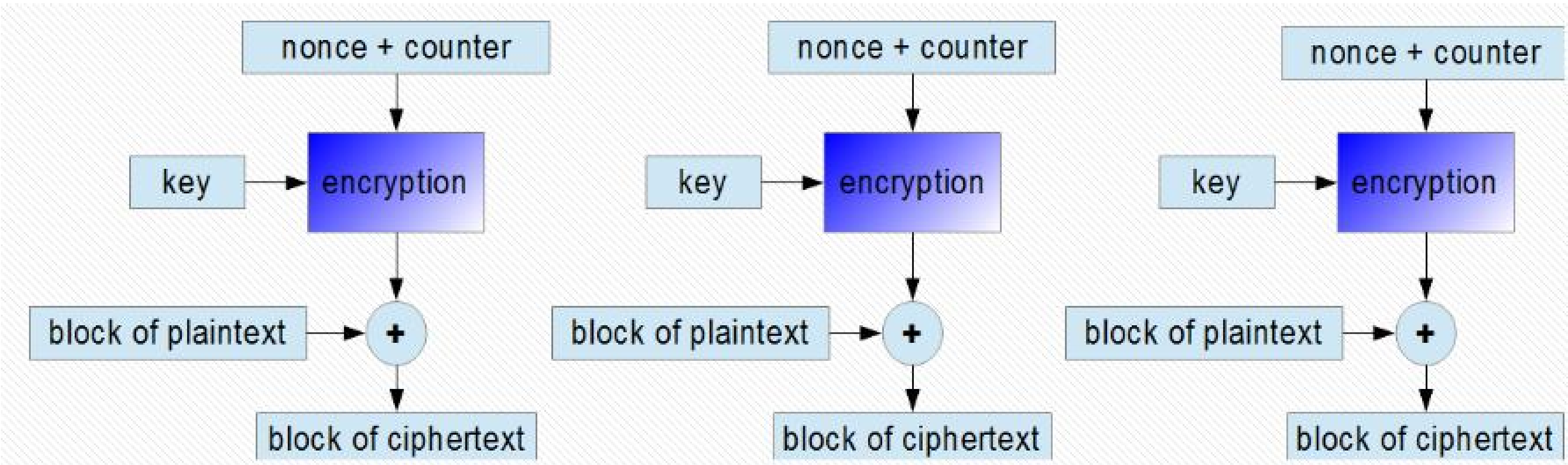
Output feedback mode (OFB)



- Pre-processing possible
- Error propagation limited

- IV: different per message

Counter Mode (CTR)



Operation mode: summary

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Diffie-Hellman Protocol

- For negotiating a secret key using only public communication
  - Does not provide authentication
- Parameters: public information
  - p: a large prime (~512 bits)
  - g: a primitive root mod p, and $g < p$
    - $\{g^k \bmod p\}$ is a finite cyclic group of order p

- g is a primitive root mod p if, for every number a relatively prime to p, there is an integer z such that $a \equiv g^z \bmod p$

2 is a primitive root mod 5

- $2^0 = 1,\ 1 \pmod 5 = 1$, so $2^0 \equiv 1$
- $2^1 = 2,\ 2 \pmod 5 = 2$, so $2^1 \equiv 2$
- $2^3 = 8,\ 8 \pmod 5 = 3$, so $2^3 \equiv 3$
- $2^2 = 4,\ 4 \pmod 5 = 4$, so $2^2 \equiv 4$.

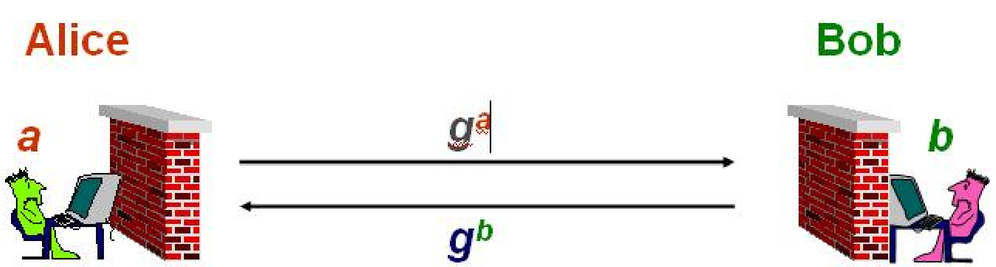
4 is not a primitive root mod 5,

- $4^0 = 1,\ 1 \pmod 5 = 1$
- $4^1 = 4,\ 4 \pmod 5 = 4$
- $4^2 = 16,\ 16 \pmod 5 = 1$
- $4^3 = 64,\ 64 \pmod 5 = 4$

Diffie-Hellman Protocol

- Alice picks a random number a, Bob b
- Alice computes $A = g^a \bmod p$, Bob $B = g^b \bmod p$
- Alice and Bob exchange A et B
- Alice computes $B^a \bmod p = g^{ba} \bmod p$, Bob $A^b \bmod p = g^{ab} \bmod p$
  - $k = g^{ba} \bmod p$ is the secret key

- Impossible to derive k from A and B: discrete log problem
  - given g, p et n, computationally infeasible to find a: $g^a \bmod p = A$



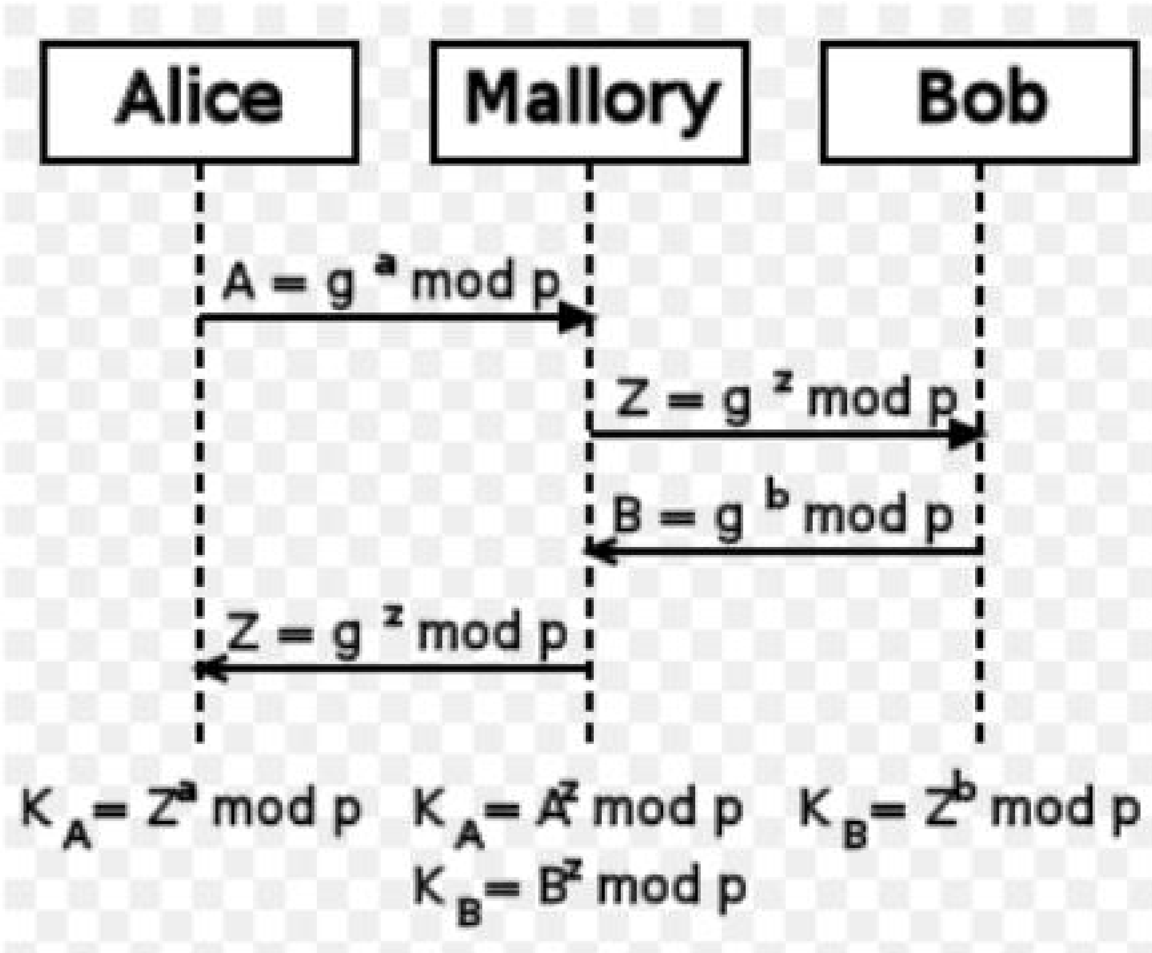
$$g^{ab} = (g^a)^b = (g^b)^a$$

Diffie-Hellman example

- g=3, p=7

Alice		Bob
$a = 3$		$b = 2$
$A = 3^3 \bmod 7 = 6$	$\xrightarrow{6}$	$B = 3^2 \bmod 7 = 2$
	$\xleftarrow{2}$	
$K = 2^3 \bmod 7 = 1$		$K = 6^2 \bmod 7 = 1$

DH man-in-the-middle attack



- DH limitation: not for user authentication
    - You may negotiate key with attacker!

DH: phone-book mode

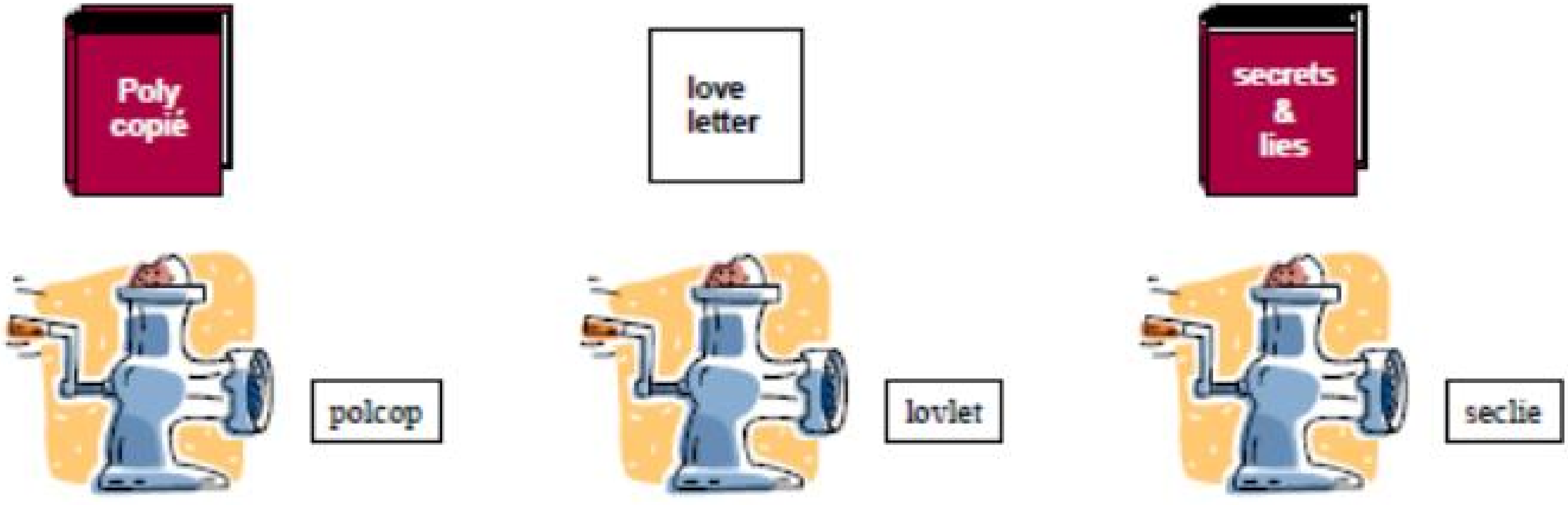
- Alice and Bob each chooses a secret number, generates A and B
- Alice and Bob publish A and B
  - Alice can get Bob's B at any time
  - Bob can get Alice's A at any time
- Alice and Bob can then generate key without communicating
  - but, they must be using the same p and g
- Needs reliability of the published values
  - no one can substitute false values

Test: group DH

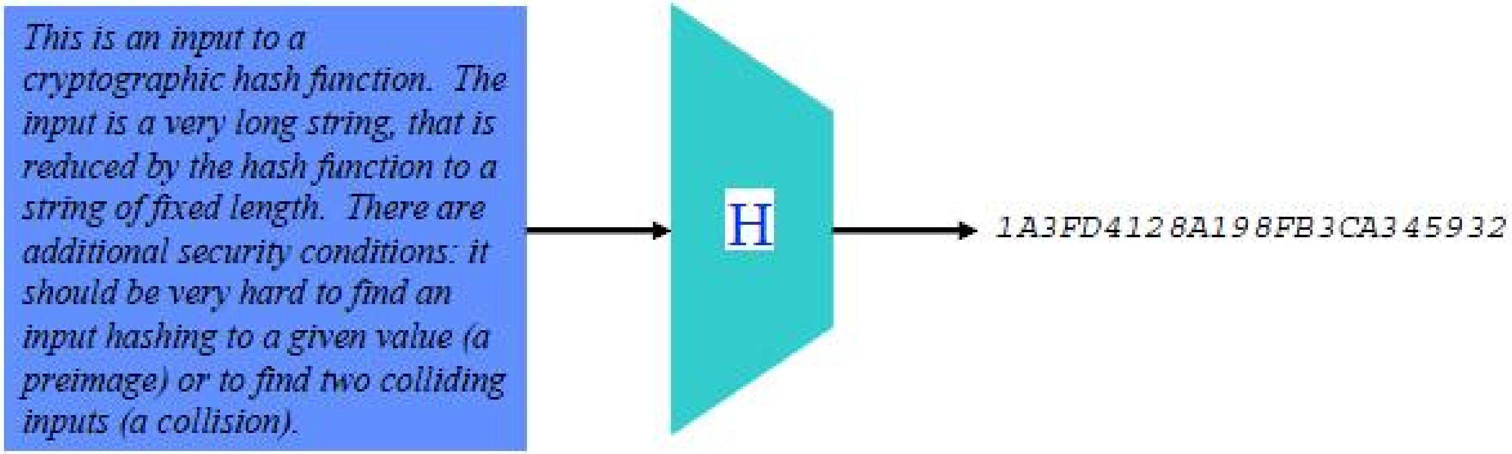
- Alice, Bob, Carole need to negociate a secret key
- Develop a DH-based protocol for them

Hash function

- Arbitrary-length input to fixed-length output

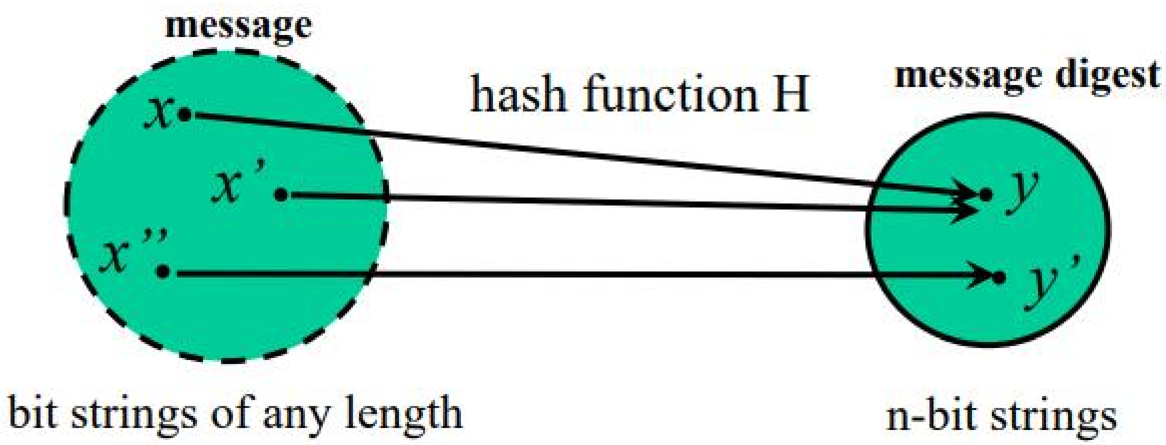


Hash function



- Also known as
  - Message digest
  - One-way transformation
  - One-way function
  - Hash
- Length of H(m) << m
- Usually fixed lengths: 128, 160, 256 bits
- McCarthy's puzzle

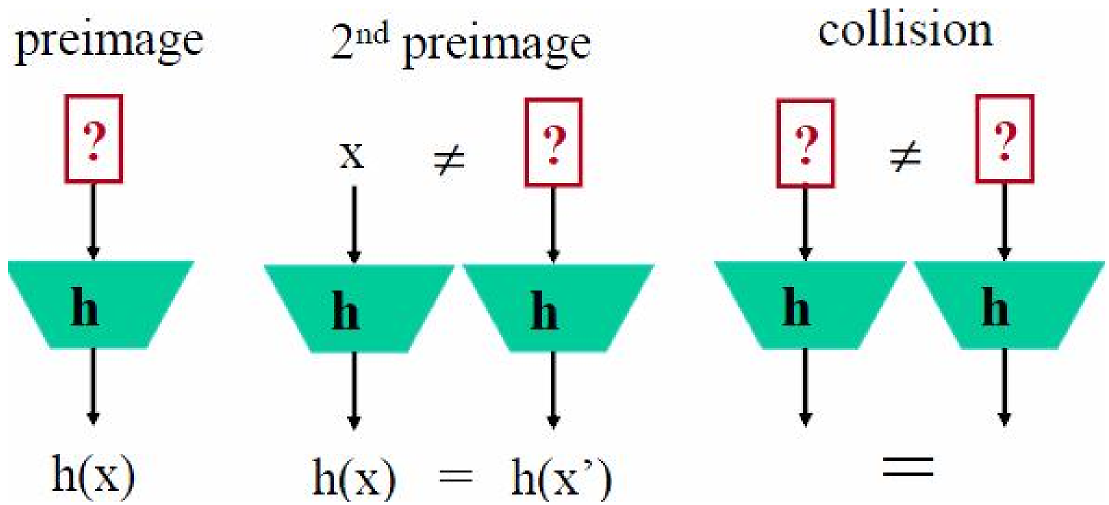
Hash function: desirable properties



- **Performance**: easy to compute H(m)
- **One-way** property: given H(m), computationally infeasible to find m
- **Weak collision resistance**: given H(m), computationally infeasible to find m' such that H(m') = H(m)
- **Strong collision resistance**: computationally infeasible to find m_1, m_2 such that $H(m_1) = H(m_2)$

Hash Function: desirable properties

- One-way property: resistance to pre-image
- Weak collision resistance: resistance to second pre-image
- Strong collision resistance: imply resistance to 2^{nd} pre-image



- Test: are they good hash functions
  - $H(x) = a^x \bmod p$; $H(m) = m1+m2+\ldots$

Length of Hash Image

- Why do we have 128, 160, 256 bits Hash image?
  - Too long: unnecessary overhead
  - Too short: collision

- For k messages, what is minimal Hash image length such that
  - prob. that at least two messages have the same hash < 0.5?

- Number of sand grains: 2^{70}

Birthday Paradox

- What is the smallest group size k such that
  - Prob. at least two people having same birthday > 0.5
  - n=365 days
- P(k people having k different birthdays) $\frac{n\times(n-1)\times\ldots\times(n-k+1)}{n^k}$
- P(at least two people having same birthday)

$$P(n,k)=1-\frac{n\times(n-1)\times\ldots\times(n-k+1)}{n^k}=1-\left[\frac{n-1}{n}\times\frac{n-2}{n}\times\ldots\times\frac{n-k+1}{n}\right]=1-\left[\left(1-\frac{1}{n}\right)\times\left(1-\frac{2}{n}\right)\times\ldots\times\left(1-\frac{k-1}{n}\right)\right]$$

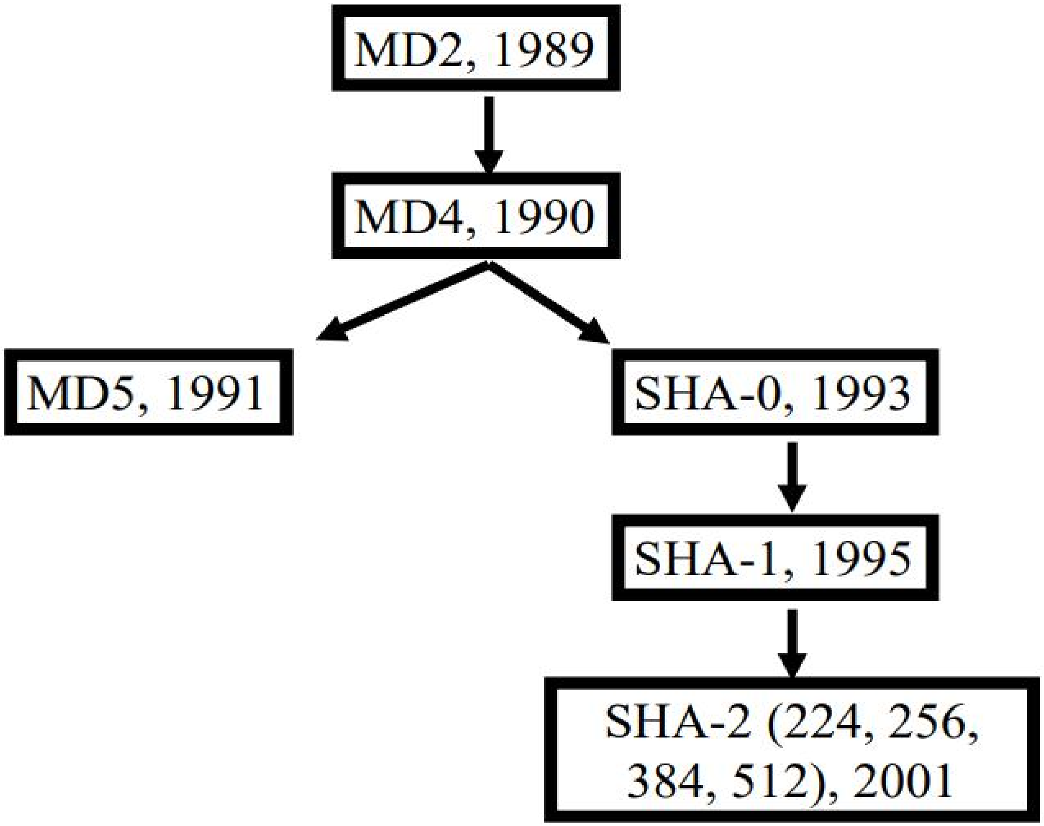
$$P(n,k)=1-e^{-k(k-1)/2n}>1/2$$

$$k>\sqrt{2(\ln 2)n}\approx 1.18\sqrt{n}$$

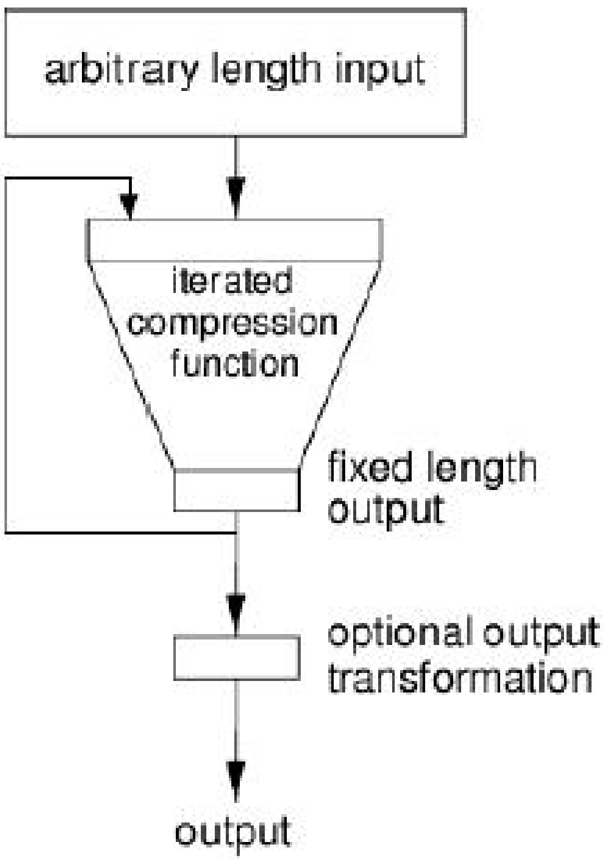
If $n=365$, we get $k>1.18\times\sqrt{365}=22.54$

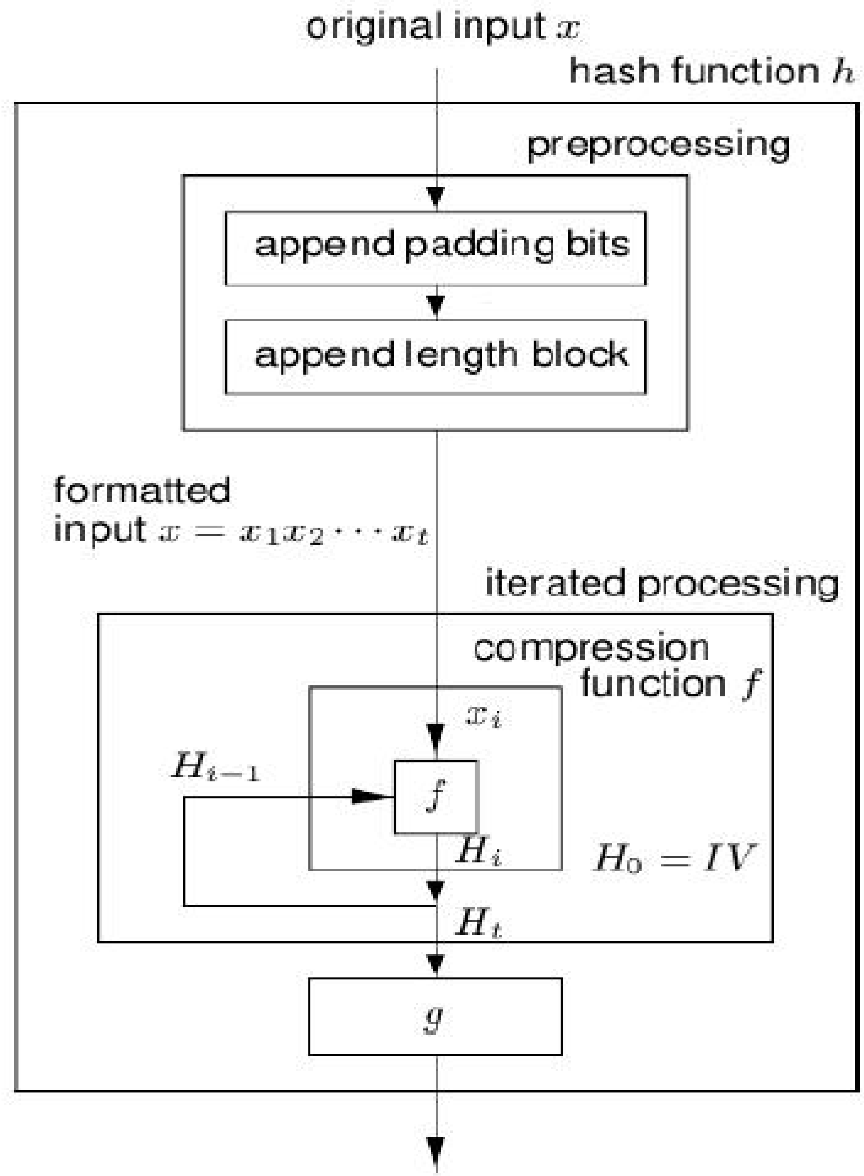
- Back to our minimal Hash image length problem
  - k messages, $n=2^m$ possible images
  - To have prob. of collision > 0.5
    - $k>2^{m/2}$

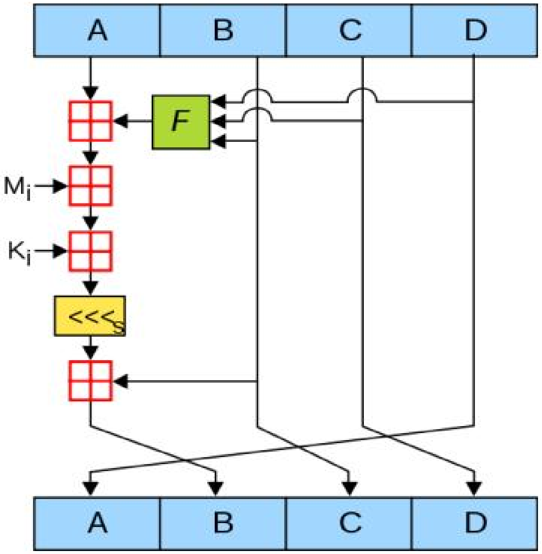
Hash Function Genealogy

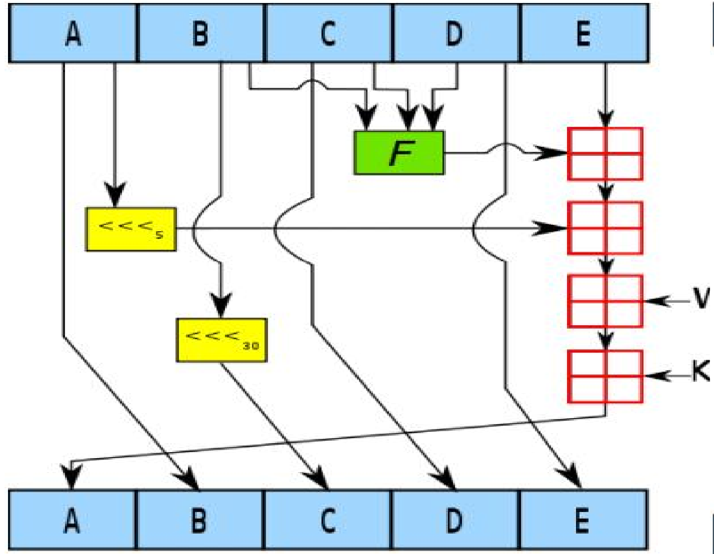


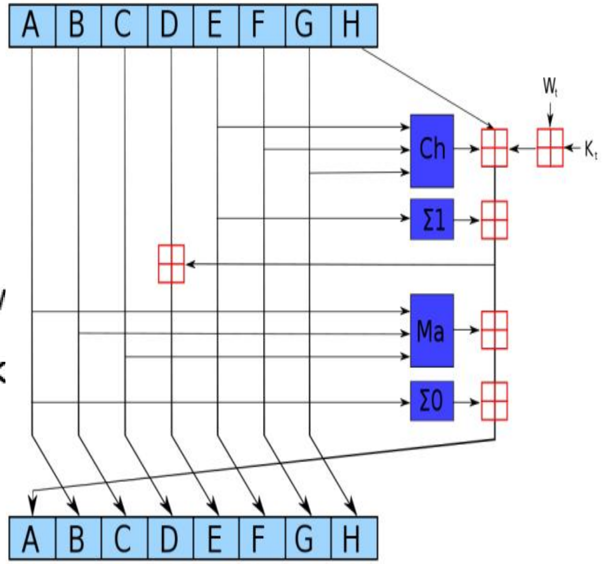
Hash function structure

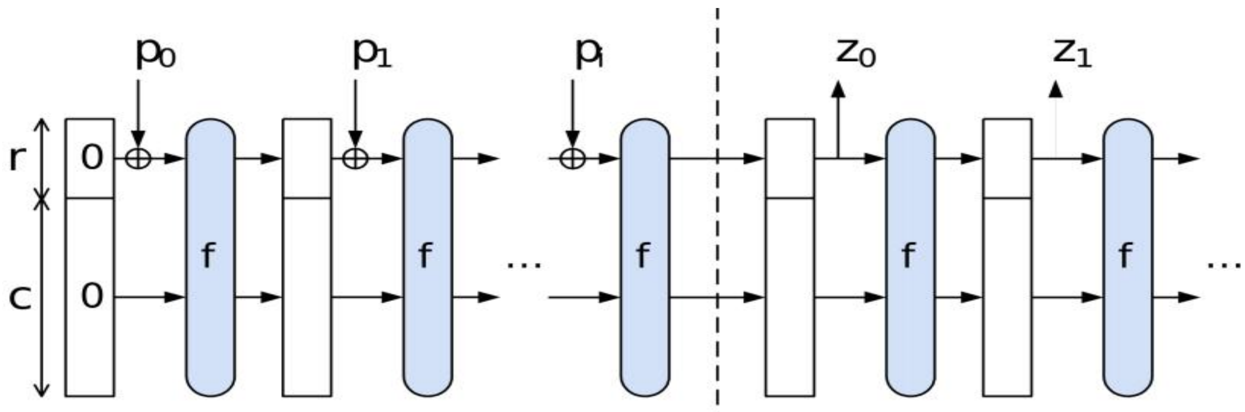




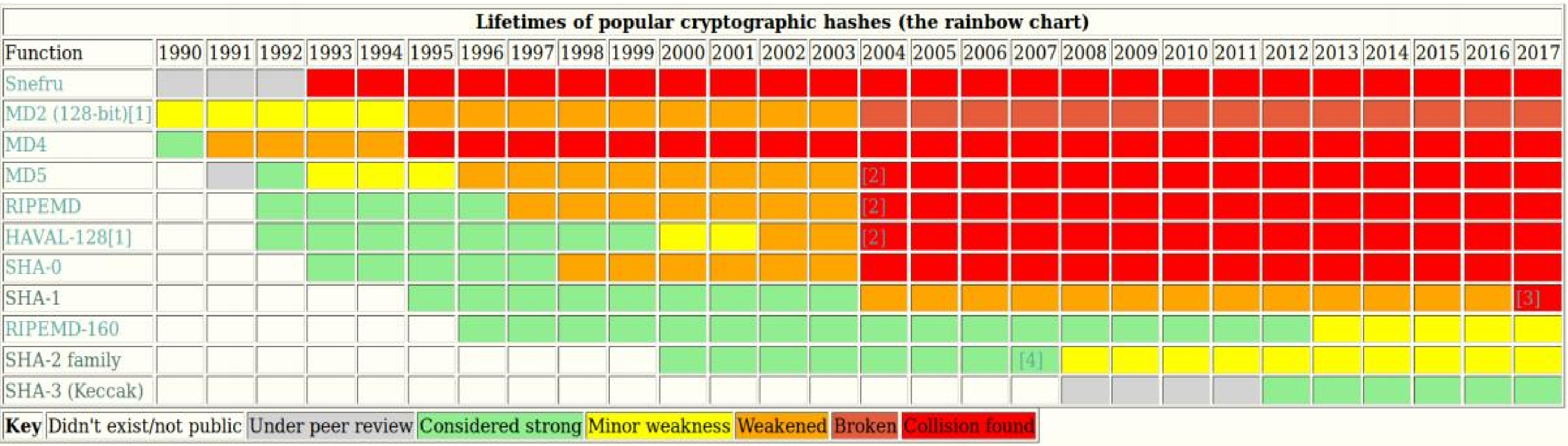

A
B
C
D
F
Mi
Ki
<<<s
A
B
C
D


A
B
C
D
E
F
<<<s
<<<30
W
K
A
B
C
D
E


A
B
C
D
E
F
G
H
Wt
Kt
Ch
Σ1
Ma
Σ0
A
B
C
D
E
F
G
H


p0
p1
pi
z0
z1
r
c
0
0
f
f
f
f
f
...
...

Hash function life cycles



- Lifetime of cryptographic hash functions: ~10 years

Hash function application

- Encryption
  - Protect confidentiality, but not integrity

- Hash function
  - Integrity, authentication
  - Digital signature
  - ......

File authentication

- Objective: detect modification
- Method
  - Stock H(f) separately as f
  - Check H(f)=H(f')
- Why not just store a duplicate copy of F?

User authentication

- Alice wants to authenticate herself to Bob
  - assuming they already share a secret key k

- Protocol

  A->B : I'm Alice

  B->A : R (random number)

  A->B : H(R|k)

- Why not just send k, in plaintext, or H(k)? Why using R?

Message authentication

- Alice wants to authenticate a message to Bob
  - assuming they already share a secret key k

- Protocol
  - A->B: M, R, H(M|k|R)

Commitment Protocol

- A and B play game “odd or even” over the network
  - A picks a number X
  - B picks another number Y
  - A and B “simultaneously” exchange X and Y
  - A wins if X+Y is odd, otherwise B wins

- If A gets Y before deciding X, A can easily cheat
  - How to prevent this?

Commitment Protocol

- A should commit to X before B sends Y

- Protocole :
  1. A->B : H(X)
  2. B->A : Y
  3. A->B : X

- Why is sending H(X) better than sending X?
- Why is sending H(X) enough to prevent A from cheating?
- Why not necessary for B to send H(Y) (instead of Y)?
- What problems are there if:
  - The set of possible values for X is small?

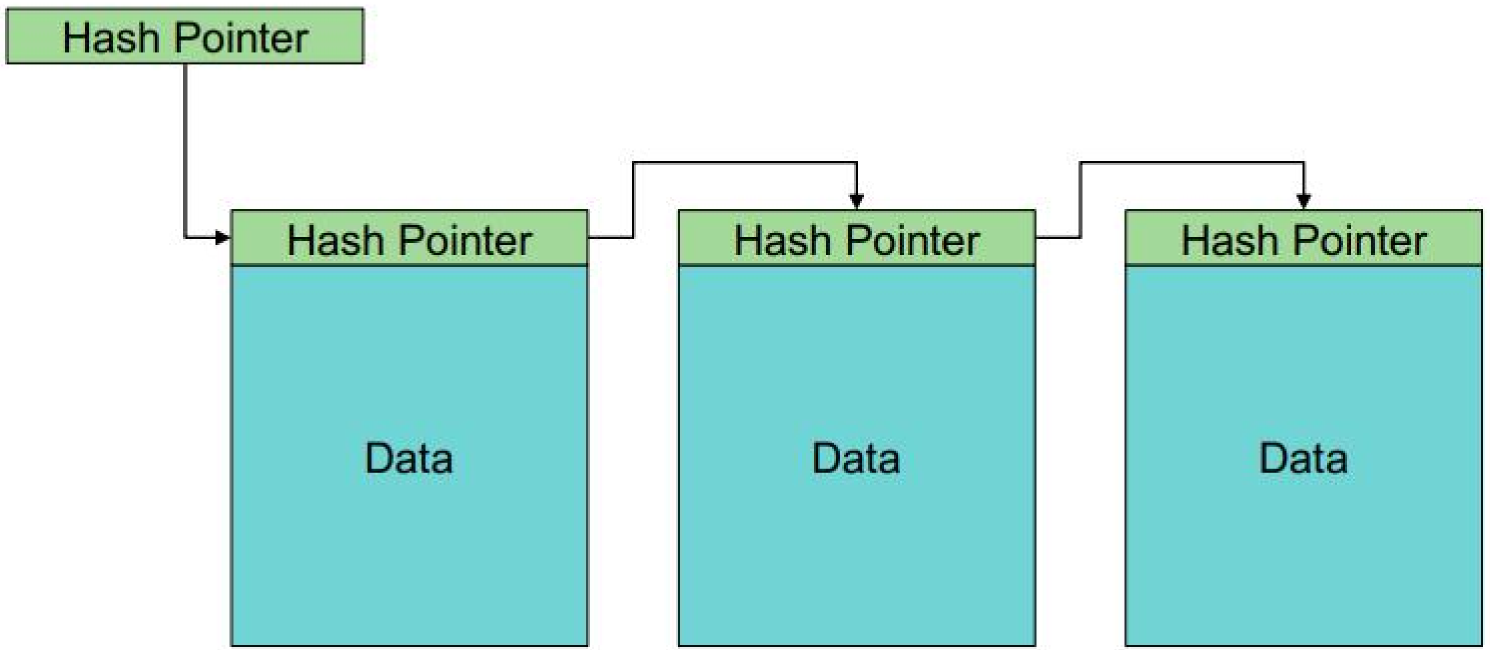
Encryption with Hash

- One-time pad
  - compute bit streams using h, k, and IV
  - $b_1 = h(k|IV)$, $b_i = h(k|b_{i-1})$, …
  - XOR with message blocks

- Mixing in the plaintext: similar to cipher feedback mode (CFB)
  - $b_1 = h(k|IV)$, $c_1 = p_1 \oplus b_1$
  - $b_2 = h(k|c_1)$, $c_2 = p_2 \oplus b_2$

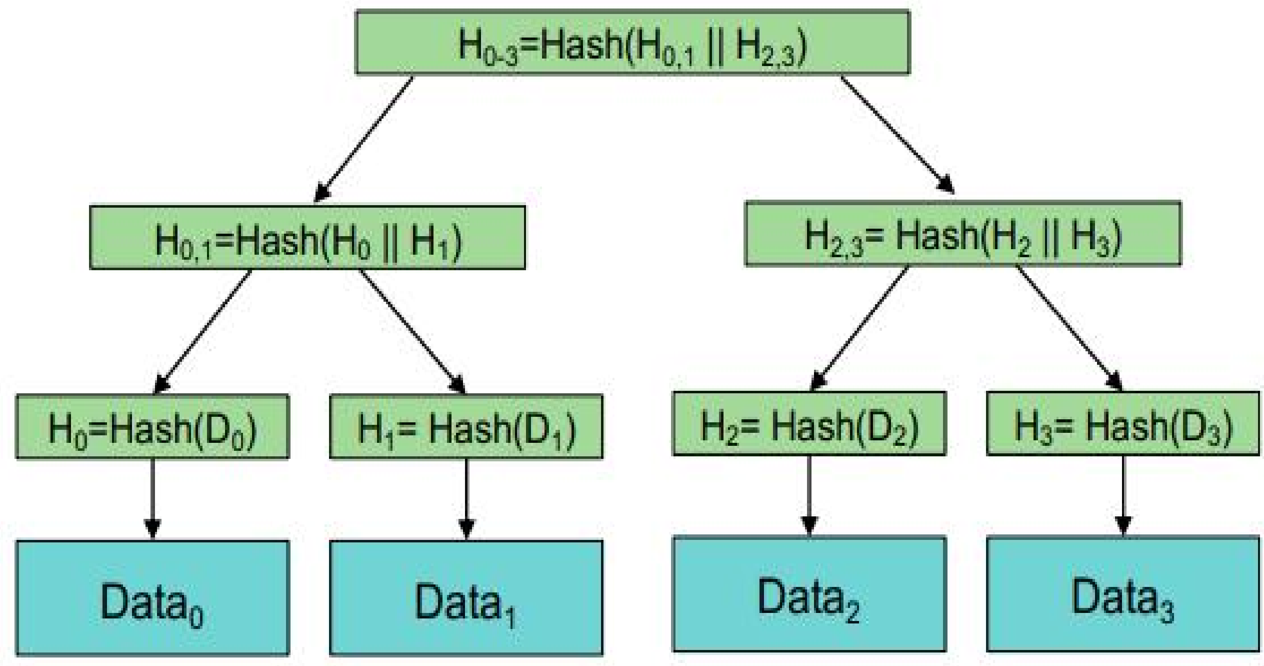
Hash Pointers

- Hash pointer = { pointer, hash(data) }
  - allows to verify data has not been modified
  - the first hash pointer is protected



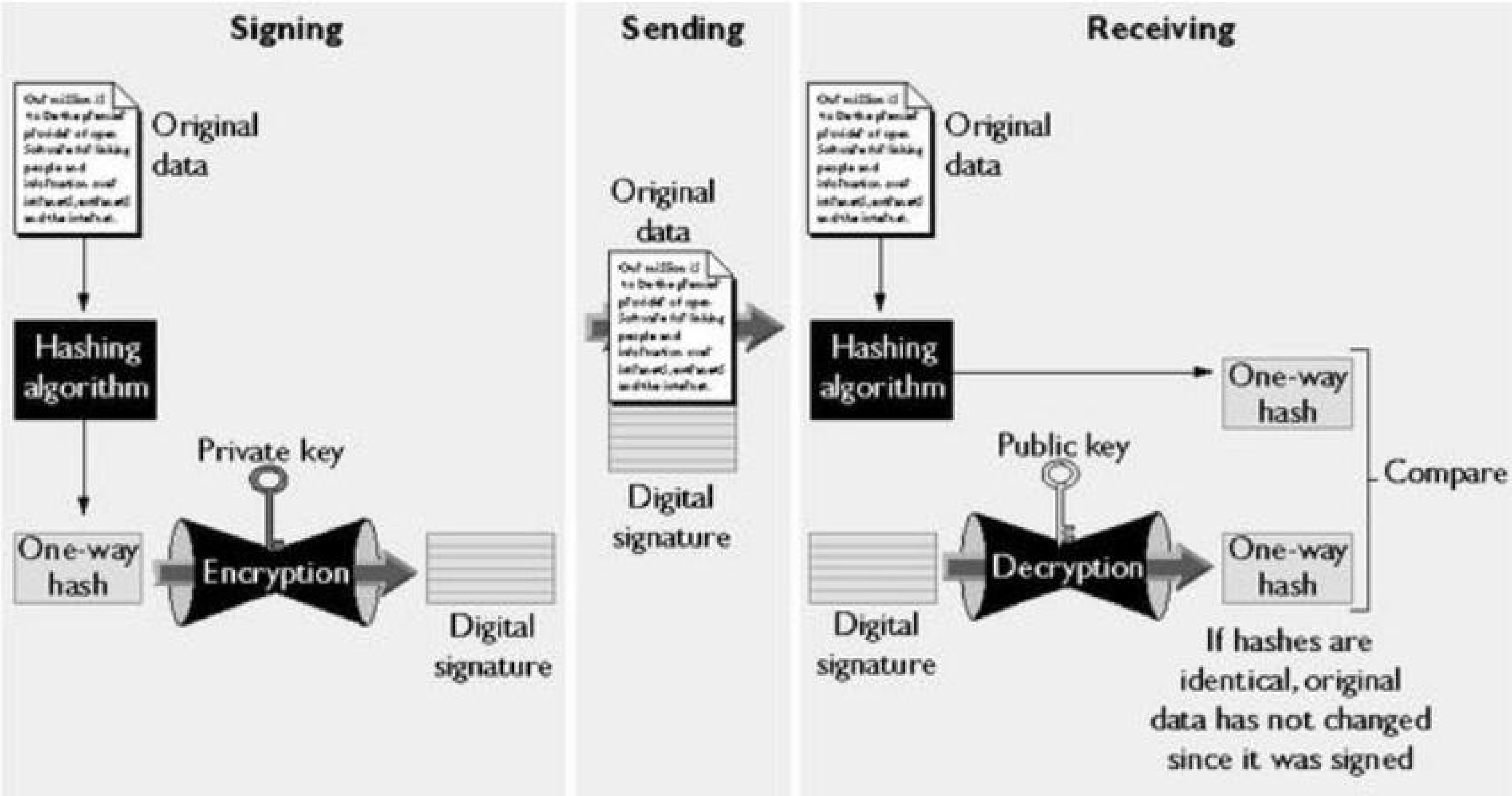
  - Blockchain

Merckle tree



- Only need to examine O($\log_2 n$) hashes to validate data

Digital signature



- No need to encrypt whole message
- Can ensure: integrity, authentication, non-repudiation

Test: signature

- Alice uses the following signature scheme
  - Signature: $s=[h(M)]^d \bmod n$
  - (d,n): Alice's private key

- Find a problem if hash is not used
  - $s=(M)^d \bmod n$

- RSA is homomorphic

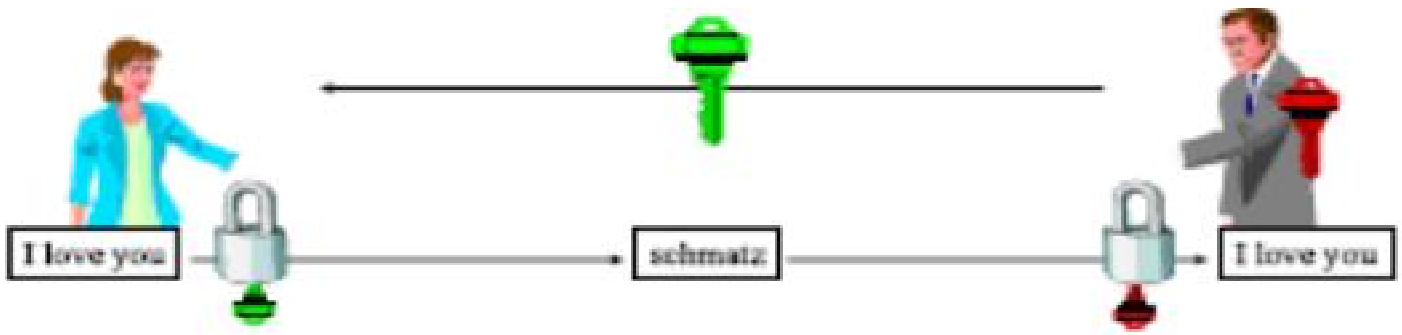
Cryptographic algorithm benchmark

- Pentium II ™ 233 MHz, API BSAFE 4.0 de RSA

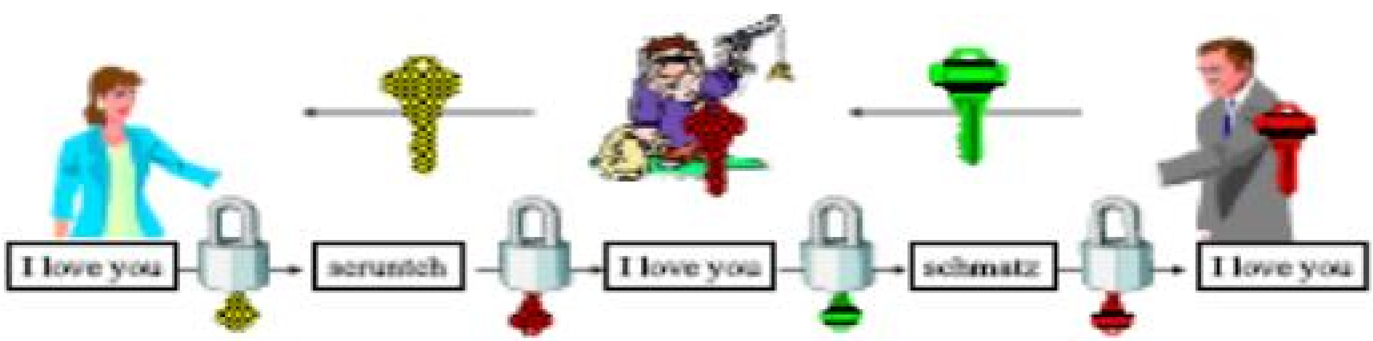
operation	performance
DES key generation	6 μsec
DES encryption	3241 KB/s
DES decryption	3333 KB/s
MD5 hash generation	36 250 KB/s
RSA encryption	4,23 KB/s
RSA decryption	2,87 KB/s
SHA hash generation	36 250 KB/s

Public key infrastructure (PKI): motivation

- Alice wants to communicate with Bob
  - Problem: how to get the public key Bob



- Man-in-the-middle attack

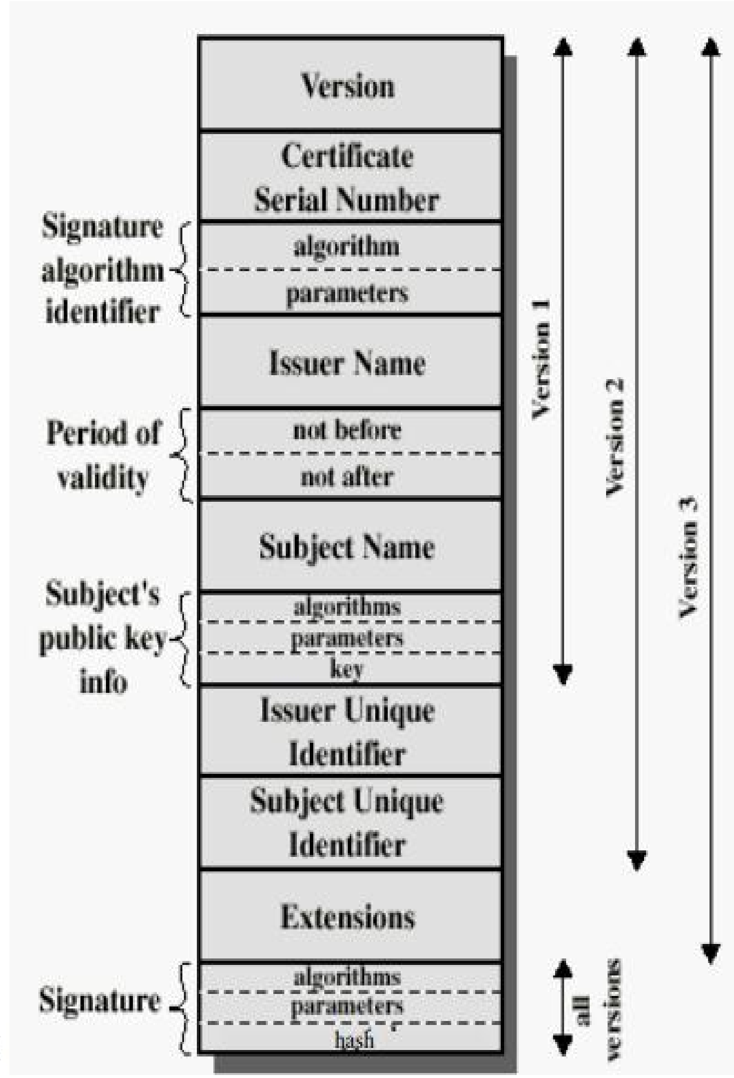


Certificate

- A certificate is a signed message proving that a particular name goes with a public key
- Example:
  - [Alice's public key is 876234]$_{\text{carol}}$
  - [Carol's public key is 676554]$_{\text{Ted}}$
- A certificate is signed by a trusted third party
- Contains following information
  - ID (name and address)
  - Public key
  - Expiration date
  - Signature of the certificate

Certificate: example

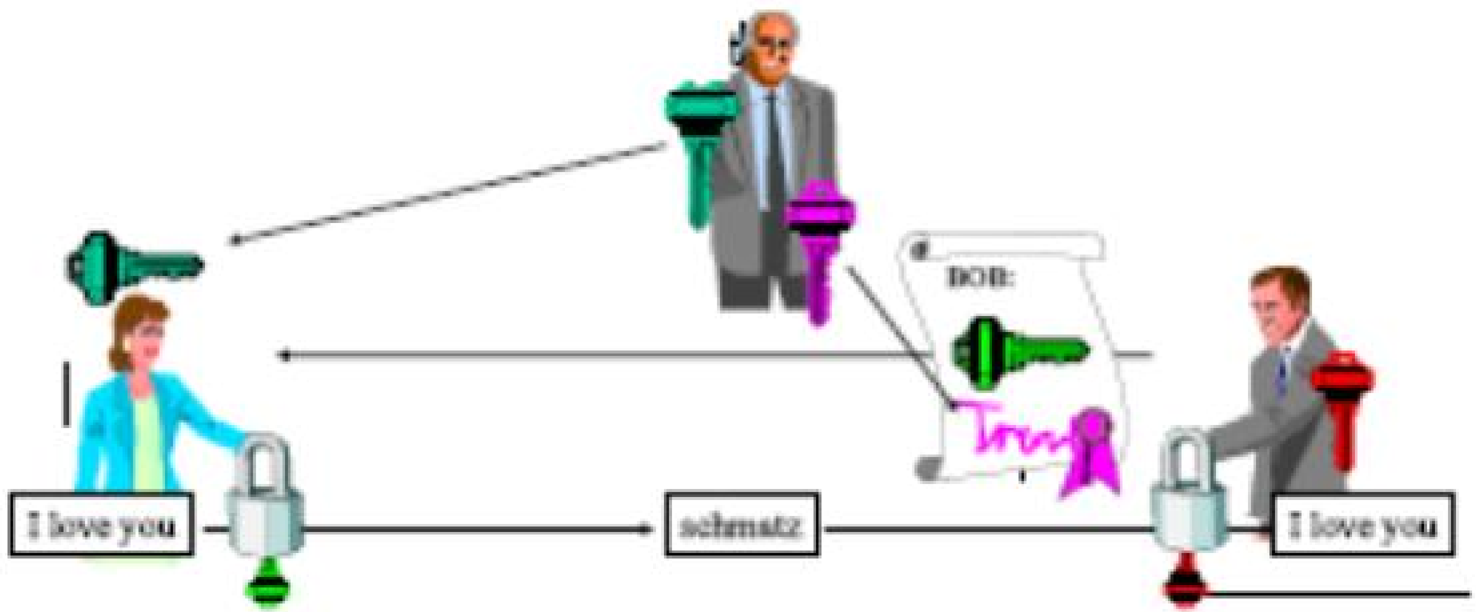
```
Certificate:
  Data:
    Version: v3 (0x2)
    Serial Number: 3 (0x3)
    Signature Algorithm: PKCS #1 MD5 With RSA Encryption
    Issuer: OU=Ace Certificate Authority, O=Ace Industry, C=US
    Validity:
      Not Before: Fri Oct 17 18:36:25 1997
      Not After: Sun Oct 17 18:36:25 1999
    Subject: CN=Jane Doe, OU=Finance, O=Ace Industry, C=US
    Subject Public Key Info:
      Algorithm: PKCS #1 RSA Encryption
      Public Key:
        Modulus:
          00:ca:fa:79:98:8f:19:f8:d7:de:e4:49:80:48:e6:2a:2a:86:
          ed:27:40:4d:86:b3:05:c0:01:bb:50:15:c9:de:dc:85:19:22:
          43:7d:45:6d:71:4e:17:3d:f0:36:4b:5b:7f:a8:51:a3:a1:00:
          98:ce:7f:47:50:2c:93:36:7c:01:6e:cb:89:06:41:72:b5:e9:
          73:49:38:76:ef:b6:8f:ac:49:bb:63:0f:9b:ff:16:2a:e3:0e:
          9d:3b:af:ce:9a:3e:48:65:de:96:61:d5:0a:11:2a:a2:80:b0:
          7d:d8:99:cb:0c:99:34:c9:ab:25:06:a8:31:ad:8c:4b:aa:54:
          91:f4:15
        Public Exponent: 65537 (0x10001)
    Extensions:
      Identifier: Certificate Type
        Critical: no
        Certified Usage:
          SSL Client
      Identifier: Authority Key Identifier
        Critical: no
        Key Identifier:
          f2:f2:06:59:90:18:47:51:f5:89:33:5a:31:7a:e6:5c:fb:36:
          26:c9
  Signature:
    Algorithm: PKCS #1 MD5 With RSA Encryption
    Signature:
      6d:23:af:f3:d3:b6:7a:df:90:df:cd:7e:18:6c:01:69:8e:54:65:fc:06:
      30:43:34:d1:63:1f:06:7d:c3:40:a8:2a:82:c1:a4:83:2a:fb:2e:8f:fb:
      f0:6d:ff:75:a3:78:f7:52:47:46:62:97:1d:d9:c6:11:0a:02:a2:e0:cc:
      2a:75:6c:8b:b6:9b:87:00:7d:7c:84:76:79:ba:f8:b4:d2:62:58:c3:c5:
      b6:c1:43:ac:63:44:42:fd:af:c8:0f:2f:38:85:6d:d6:59:e8:41:42:a5:
      4a:e5:26:38:ff:32:78:a1:38:f1:ed:dc:0d:31:d1:b0:6d:67:e9:46:a8:
       dd:c4
```



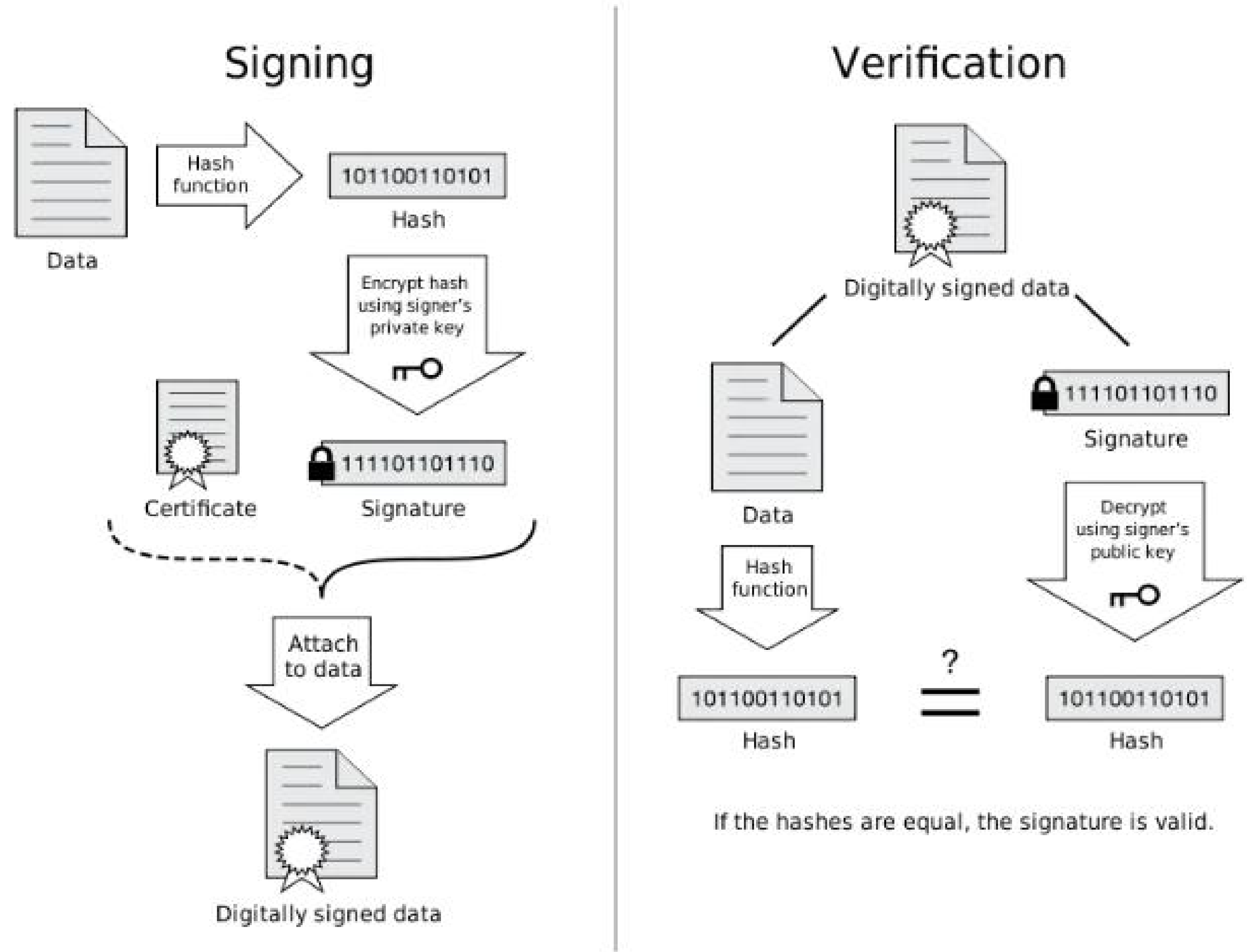
Using certificate

- A trusted third party Trent signs certificate of Bob

- If Alice has public key of Trent, she can verify certificate of Bob
  - Alice trusts Trent



Issue and check certificate



Public key infrastructure (PKI)

- An infrustructure containing all necessary components to securely distribute public keys

- Certificate Authority (CA)
- Certificates
- Registration Authorities (RA)
- A repository for retrieving certificates
- A method of revoking/updating certificates

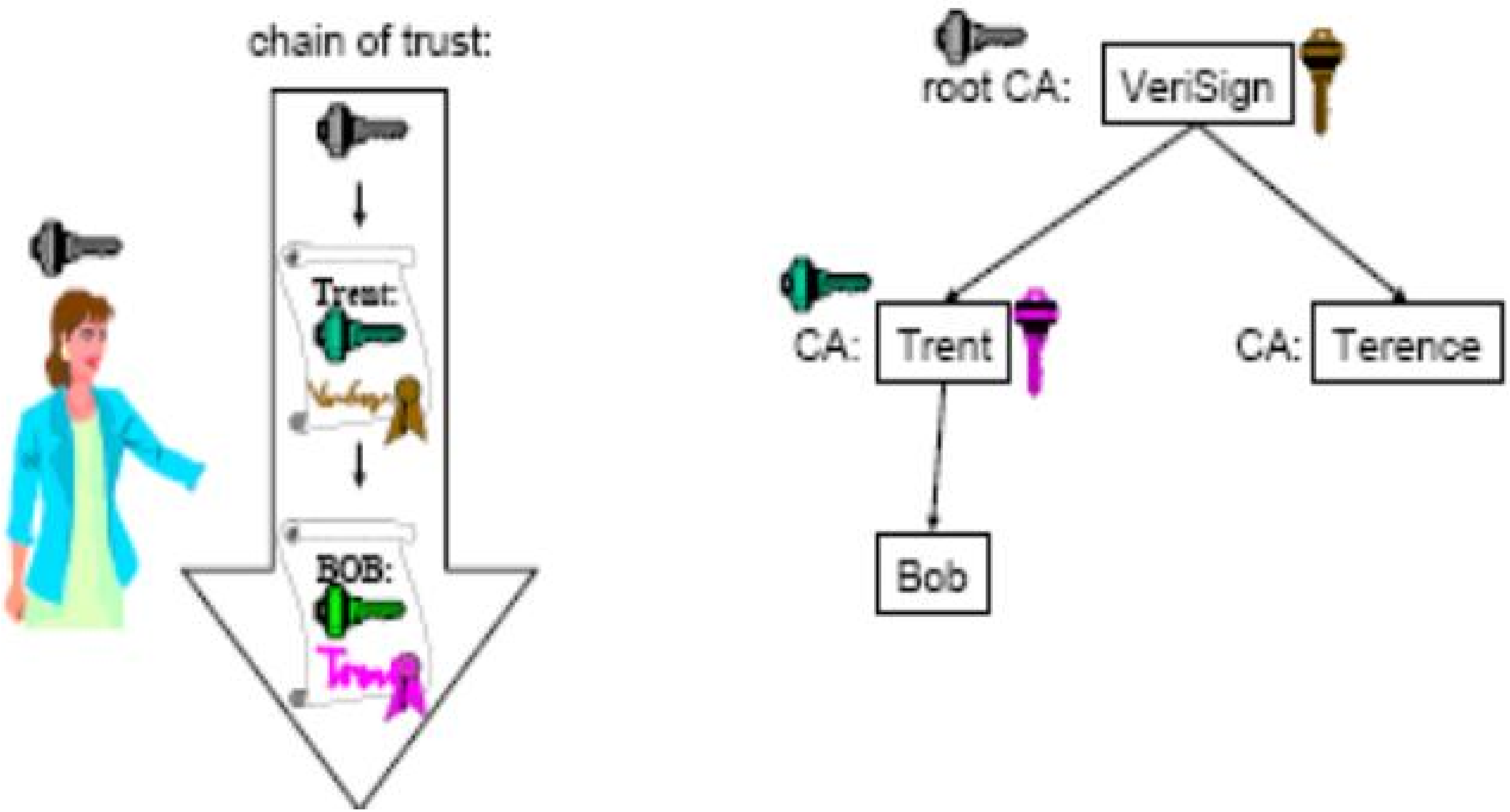
Certification Authorities (CA)

- A trusted entity maintaining public keys for all nodes
  - Each node maintains its own private key

- CA authenticates each new participant Alice physically
- Alice creates a pair of public, private keys
- CA creates and signs certificate of Alice

- A CA can be public or private
- Mutiple CAs can form a trust chain

CA hierharchy



Registration Authorities (RA)

- CA can delegate registration of new participants to RAs
- RA does not have private key of CA
  - Cannot sign certificates
- RA only authenticates physically new participants
- RAs check identities and provide CA with relevant information (identity and public key) to generate certificates

Certificate Revocation

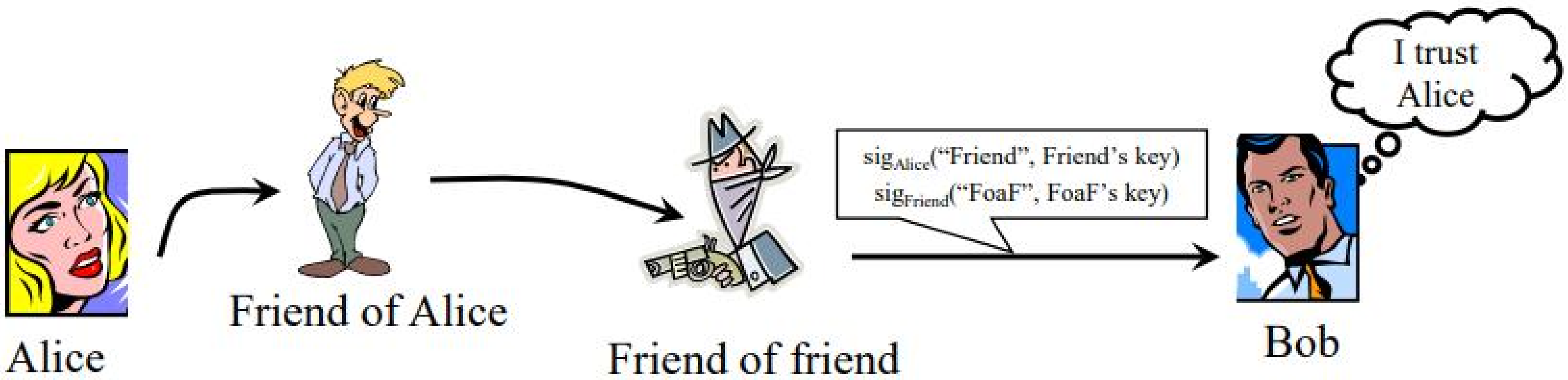
- Certificates may need to be revoked
  - Someone is fired
  - Someone is graduated
  - Someone’s certificate (card) is stolen

- CA maintain a Certificate Revocation List (CRL)
- CRL issued periodically by CA, containing revoked certificates
- Each transaction is checked against CRL

Web of Trust

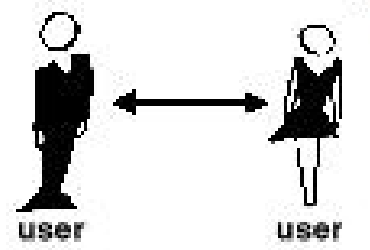
- Problem of PKI
  - Centralized hierarchic model
  - Cannot have a CA for the whole world!

- Web of trust
  - Philosophy: trust friend's friend
  - Trust = sign one's certificate

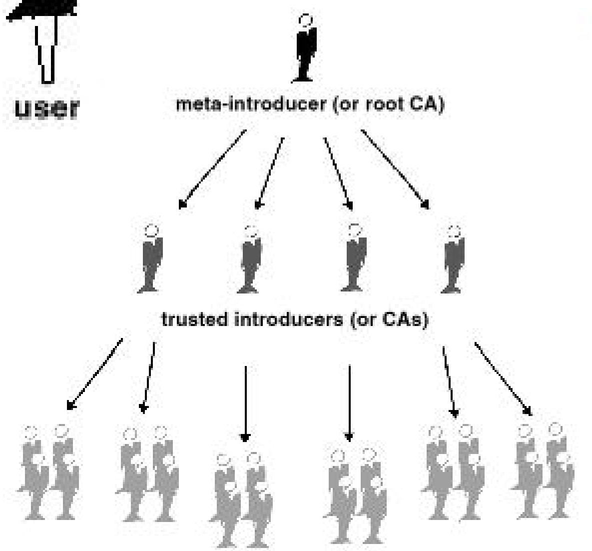


Trust models

- Direct trust



- Hierarchical trust



- Web of trust