An Introduction to Cryptology

Prof. Bart Preneel

Katholieke Universiteit Leuven, Belgium

Bart.Preneel@esat.kuleuven.ac.be

http://www.esat.kuleuven.ac.be/~preneel

Slides used by permission

Comp527 status items

- Hack-a-Vote project
  - Everybody should be working now!

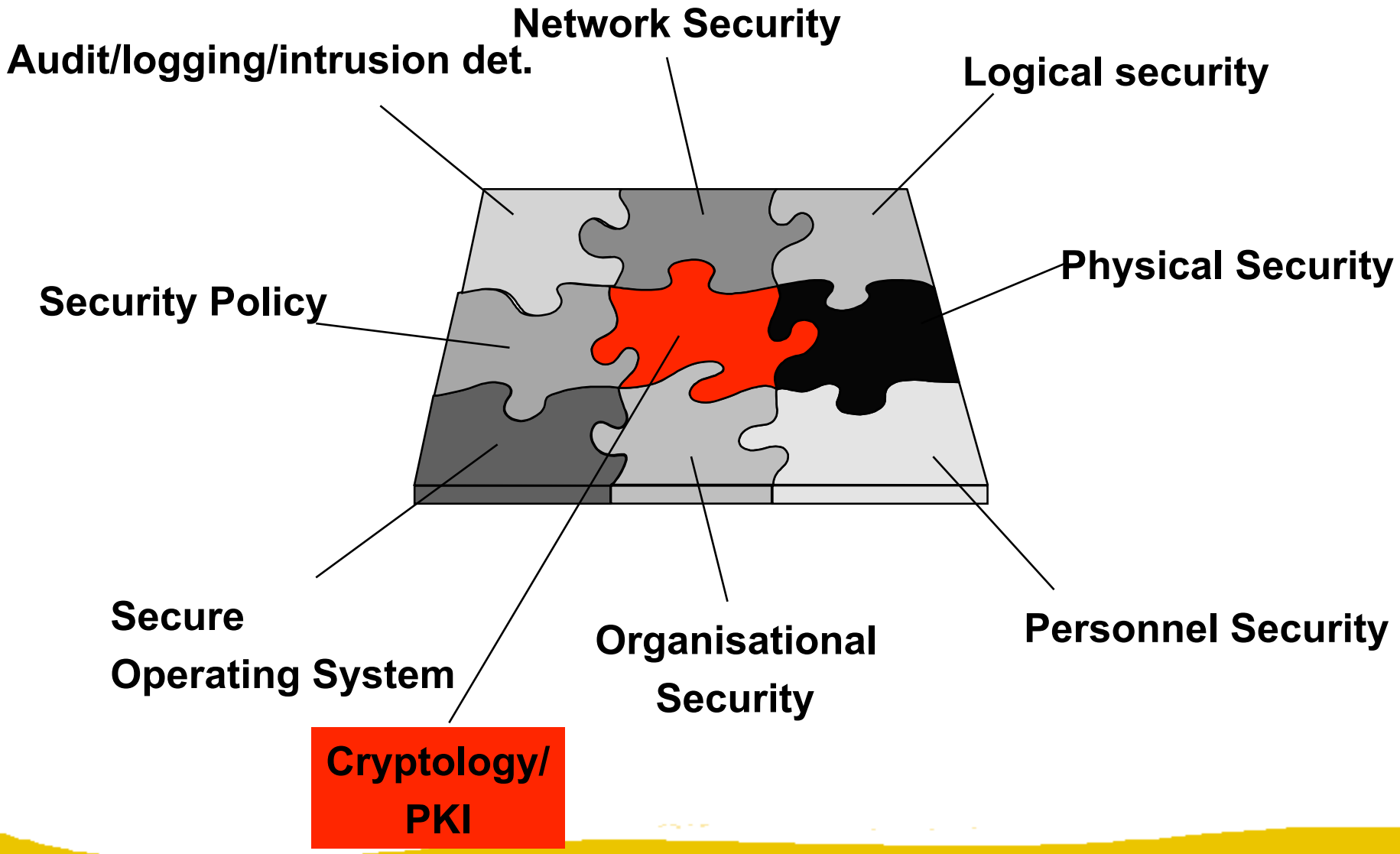
Some books on cryptology

- B. Schneier, Applied Cryptography, Wiley, 1996. Widely popular and very accessible – make sure you get the errata.
- D. Stinson, Cryptography: Theory and Practice, CRC Press, 1995. Solid introduction, but only for the mathematically inclined.
  - 2nd edition, part 1 available in 2002.
- A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997. The bible of modern cryptography. Thorough and complete reference work – not suited as a first text book. All chapters can be downloaded for free at http://www.cacr.math.uwaterloo.ca/hac

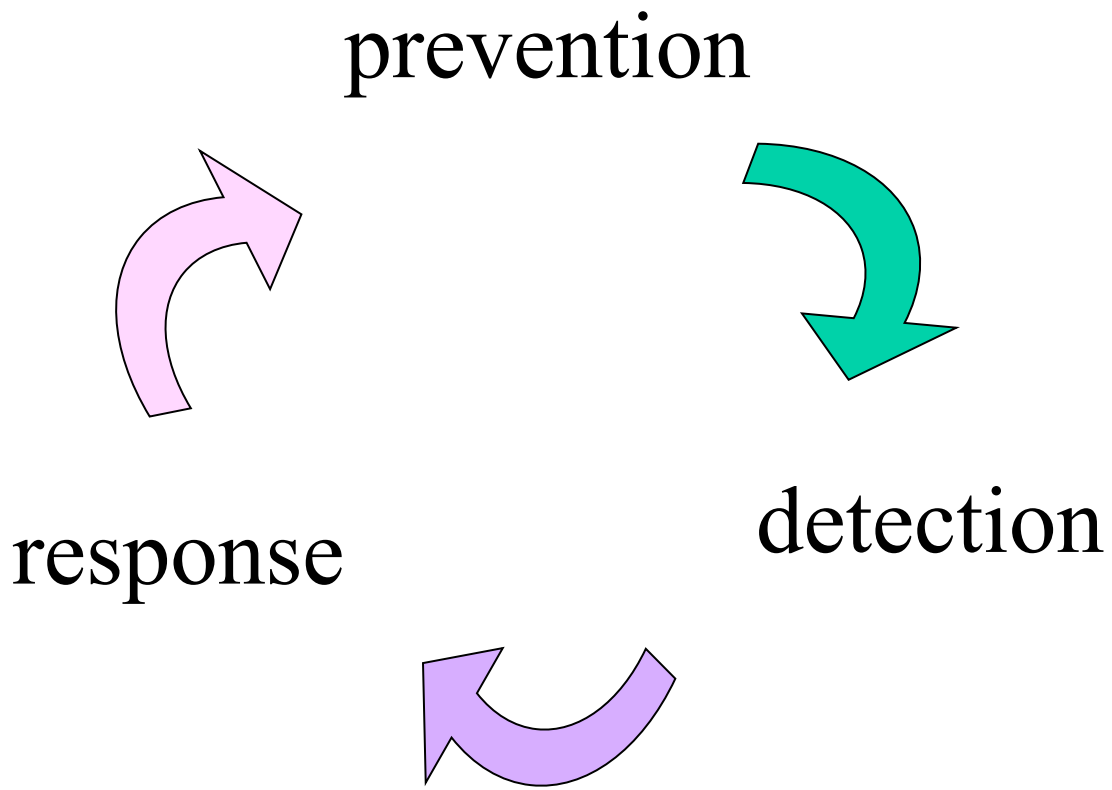
More information: some links

- IACR (International Association for Cryptologic Research): www.iacr.org
- IETF web site: www.ietf.org
- Cryptography faq: www.faqs.org/faqs/cryptography-faq
- links: Ron Rivest, David Wagner, Counterpane www.counterpane.com/hotlist.html
- Digicrime (www.digicrime.org) - not serious but informative and entertaining

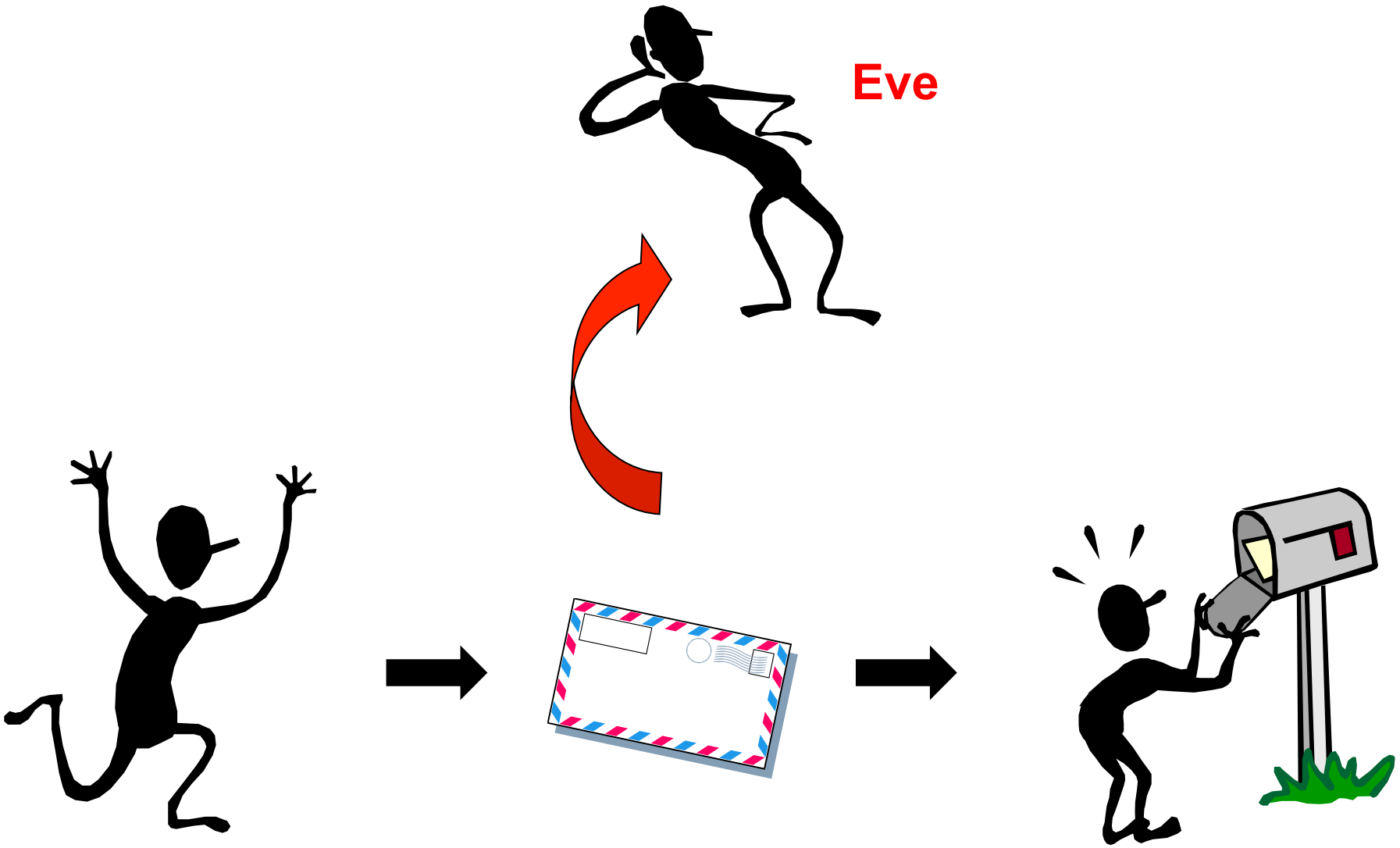
Information security: a puzzle



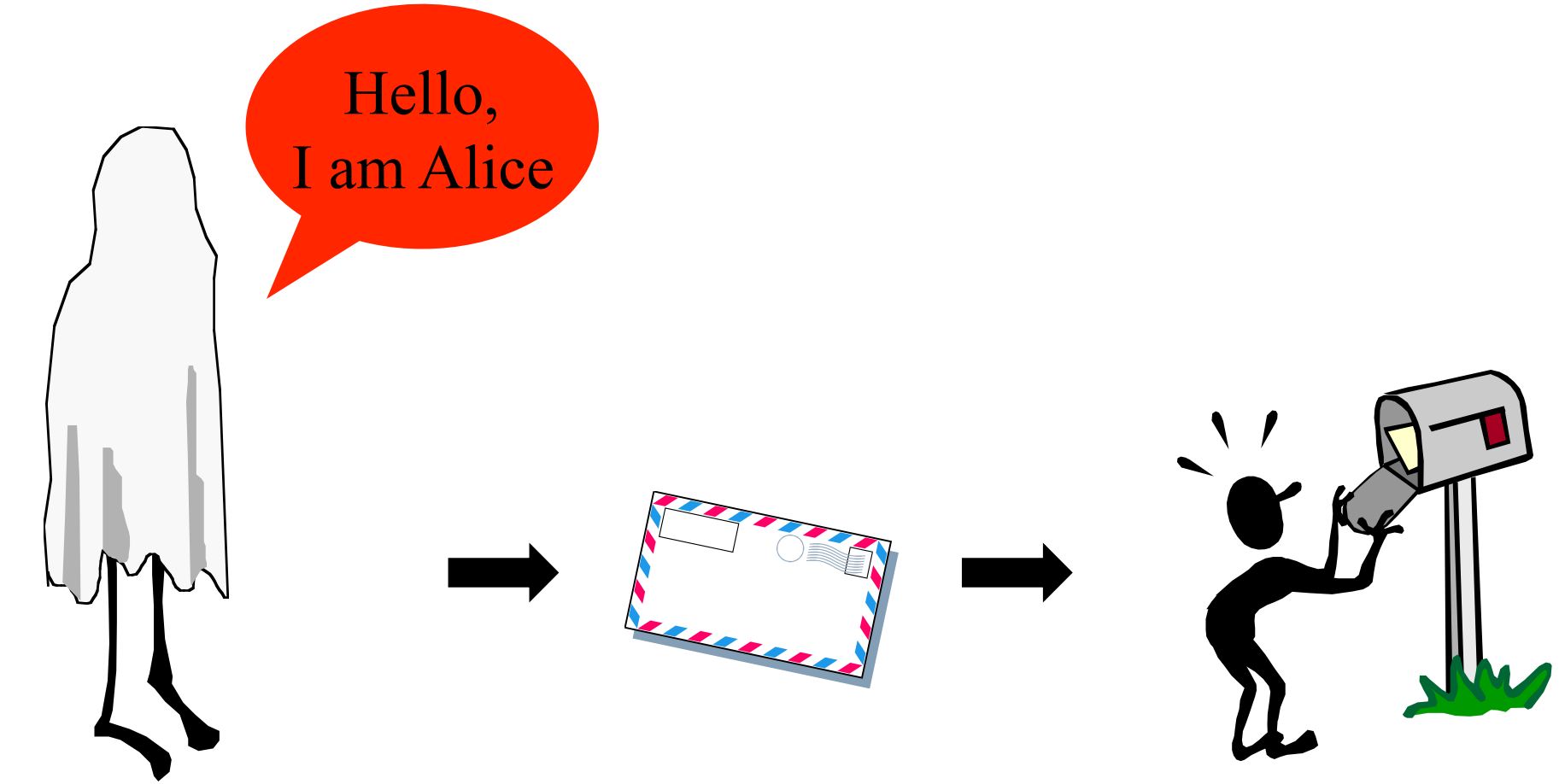
Process approach to security



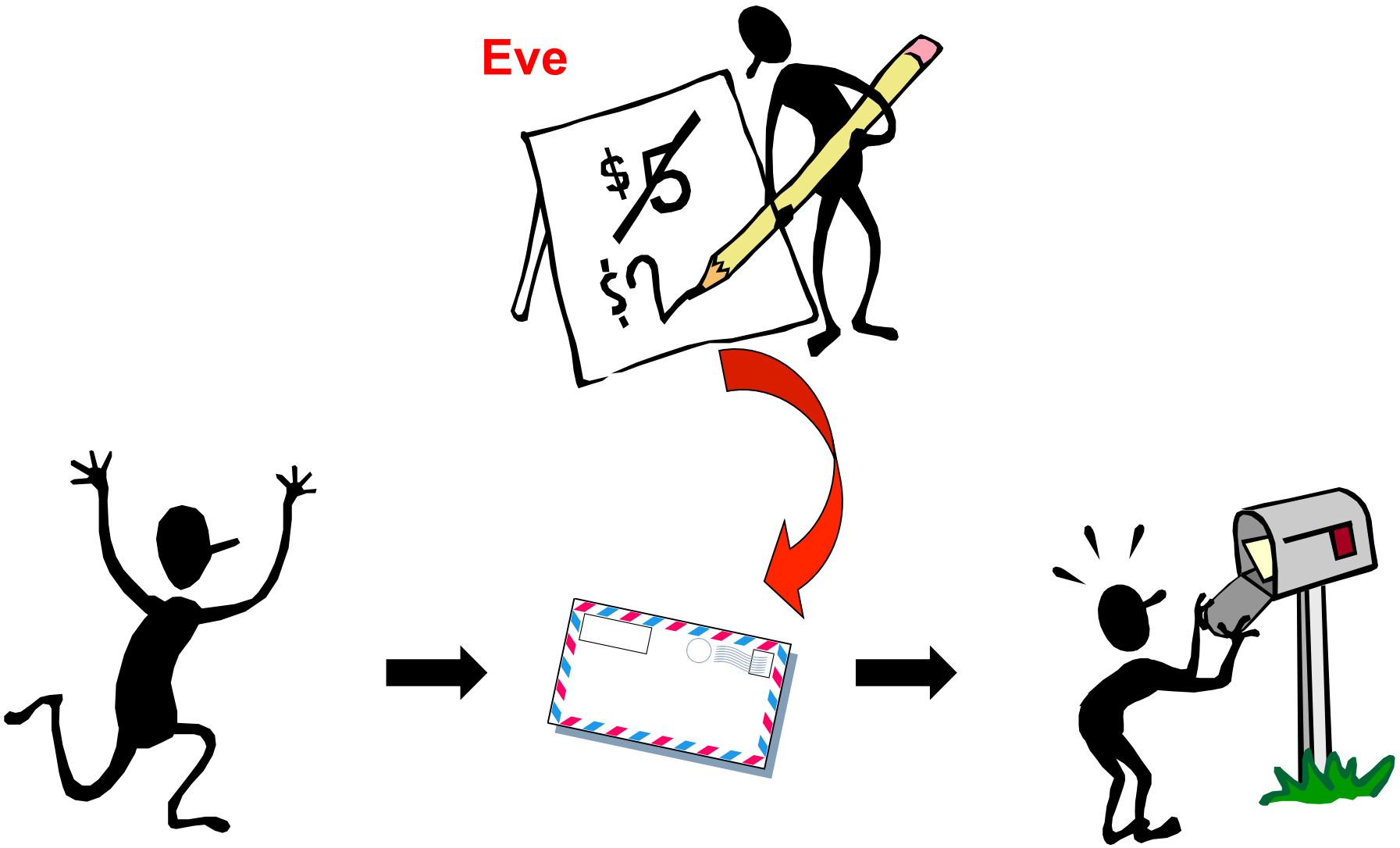
Data confidentiality



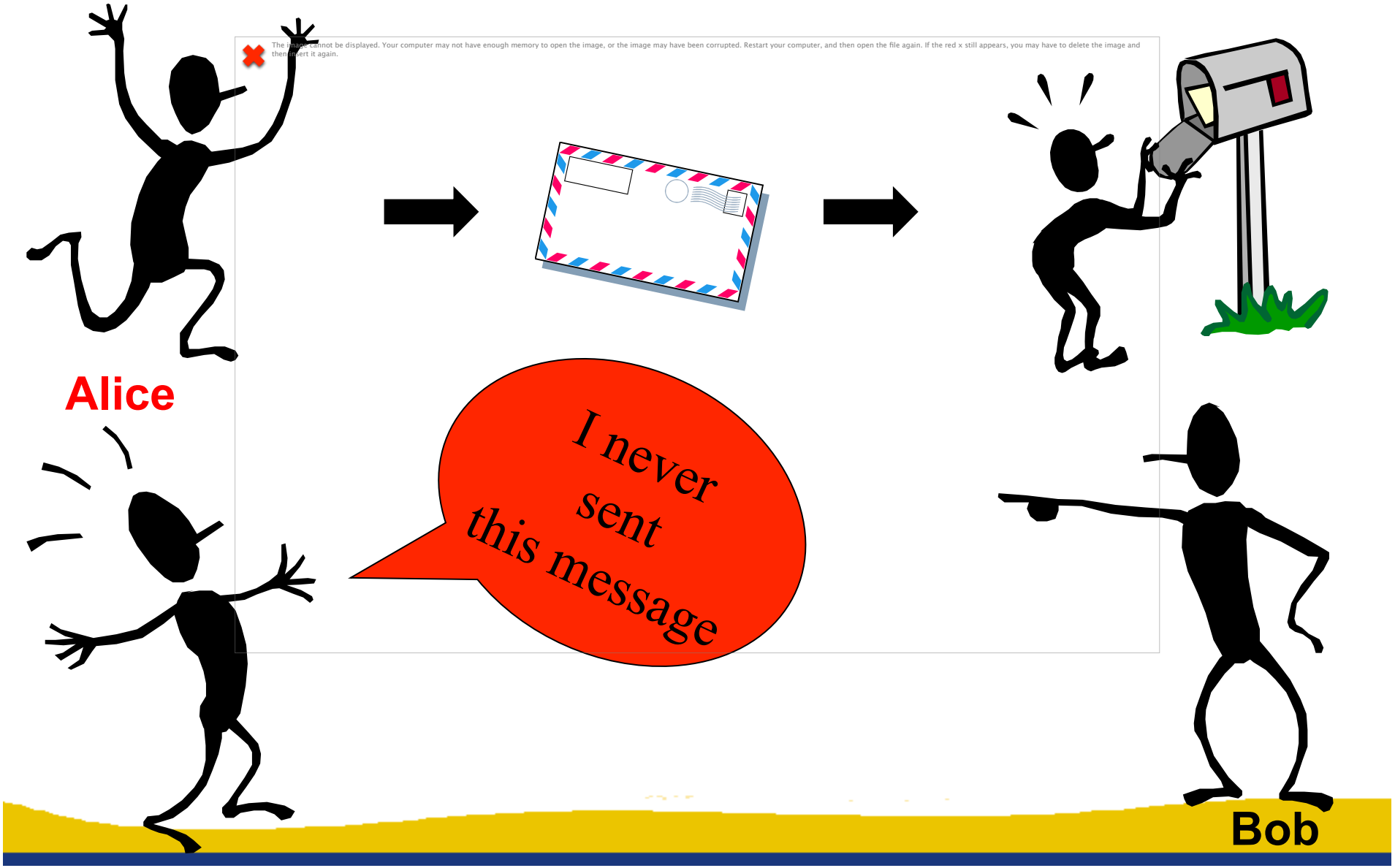
Entity authentication



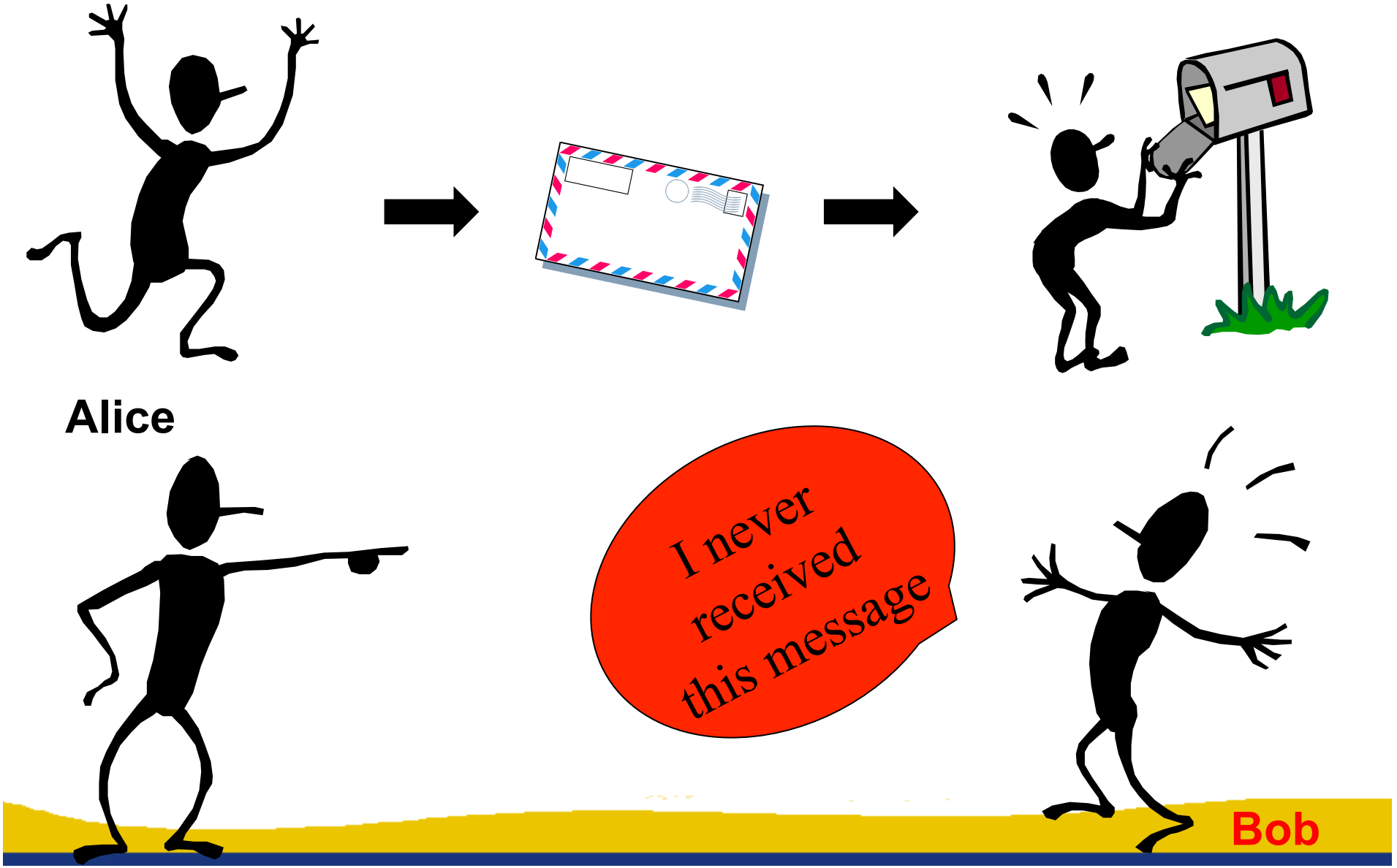
Data authentication



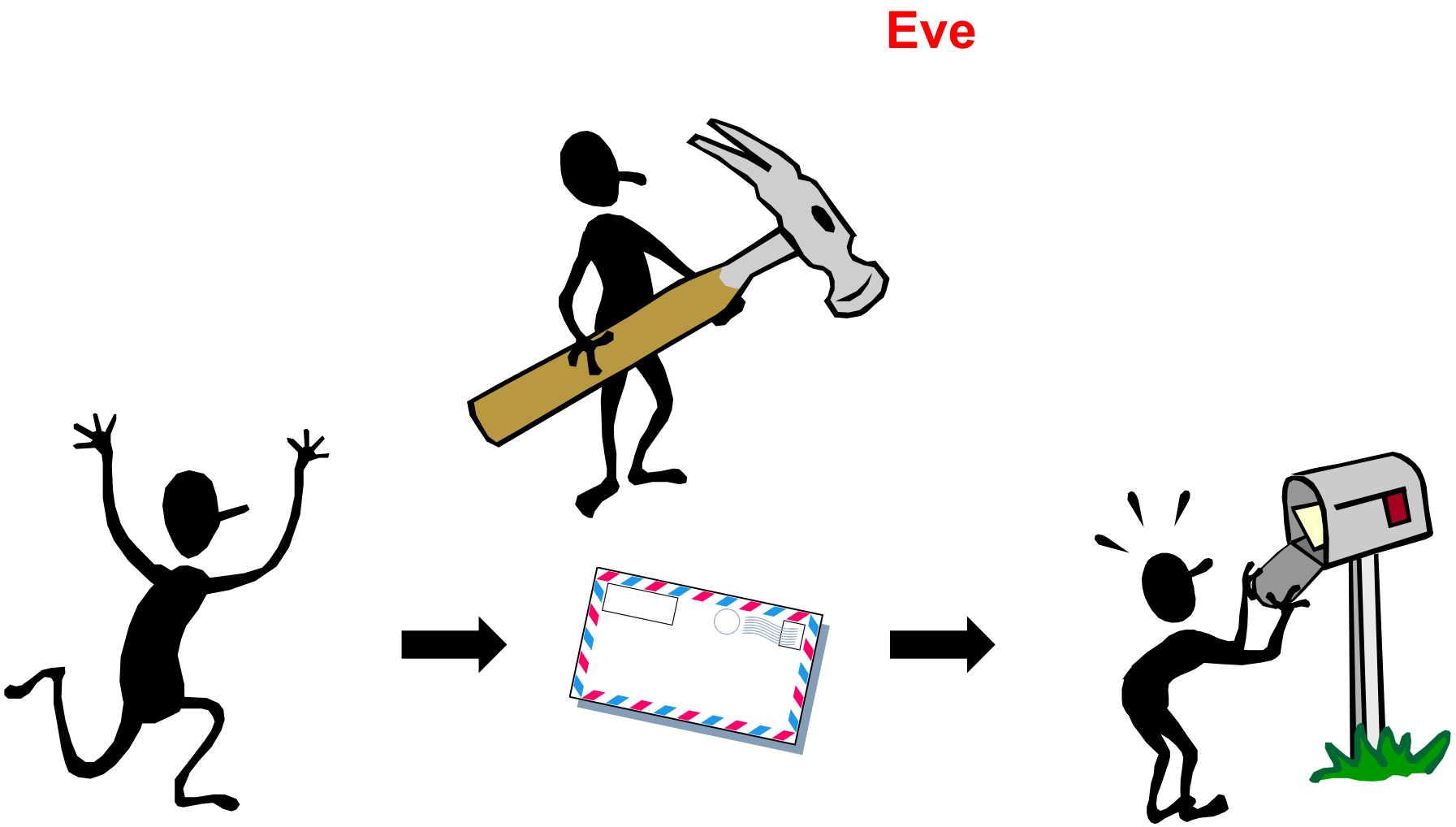
Non-repudiation (origin)



Non-repudiation (receipt)



Denial of service



Definitions

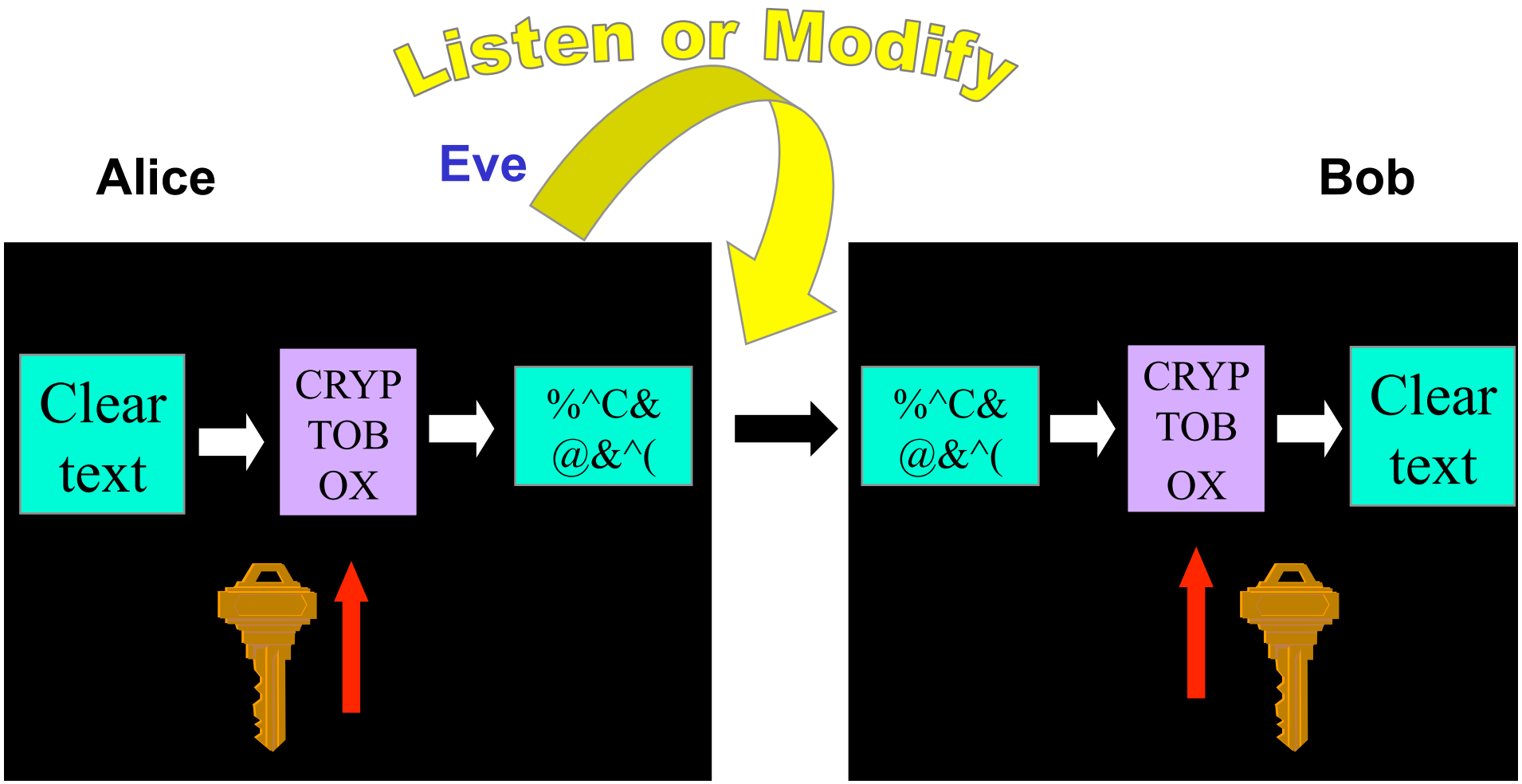
	data	**entities**
confidentiality	encryption	anonymity
authentication	data authentication	identification

Non-repudiation of origin, receipt

Contract signing

Notarisation and Timestamping

Cryptology: basic principles



Symmetric cryptology: confidentiality

- old cipher systems:
  - transposition, substitution, rotor machines
- the opponent and her power
- the Vernam scheme
- A5/1, Bluetooth, RC4
- DES and triple-DES
- AES

Old cipher systems (pre-1900)

- Caesar cipher: shift letters over k positions in the alphabet (k is the secret key)

THIS IS THE CAESAR CIPHER

WKLV LV WKH FDHVDU FLSKHU

- Julius Caesar never changed his key (k=3).

Cryptanalysis example:

```
HJAEG JAWFW FNGQW JMKMJ
IKBFH KBXGX GOHRX KNLNK
JLCGI LCYHY HPISY LOMOL
KMDHJ MDZIZ IQJTZ MPNPM
LNEIK NEAJA JRKUA NQOQN
MOFGL OFBKB KSLVB ORPRO
NPGHM PGCLC LTMWC PSQSP
OQHLN QHDMD MUNXD QTRTQ
PRIMO RIENE NVOYE RUSUR
QSJNP SJFOF OWPZF SVTVS
RTKOQ TKGPG PXQAG TWUWT
```

Old cipher systems (pre-1900) (2)

- Substitutions
  - ABCDEFGHIJKLMNOPQRSTUVW XYZ
  - MZNJSOAXFQGYKHLUCTDVWBI PER
- Transpositions

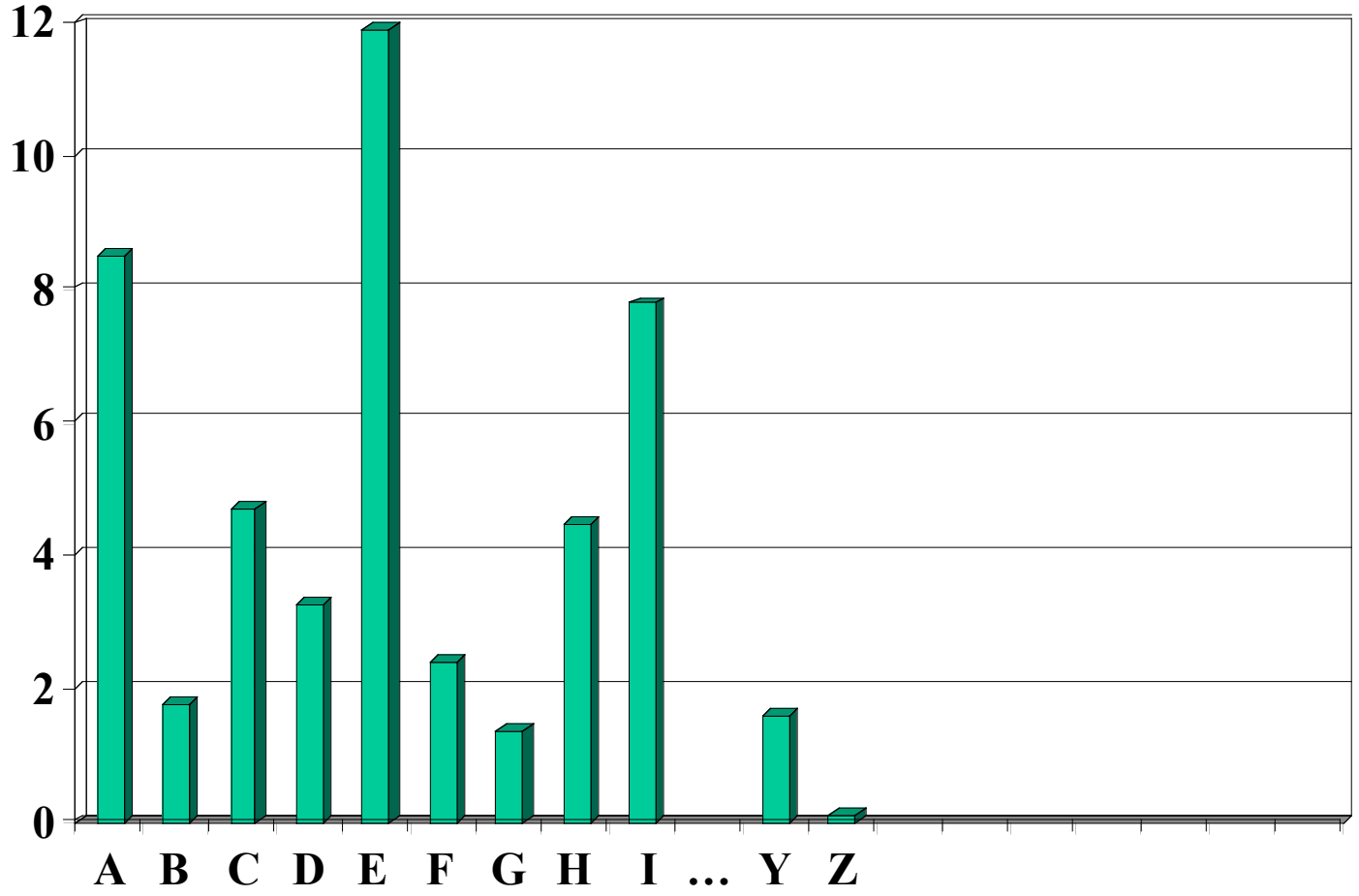
TRANS	ORI S
POSIT	NOTIT
IONS	OSAN P

Security

- there are n! different substitutions on an alphabet with n letters
- there are n! different transpositions of n letters
- n=26: n!
  =403291461126605635584000000 = 4 10^{26} keys
- trying all possibilities at 1 nanosecond per key requires....

Easy to break simple subsitution using statistical techniques

Letter distributions



Assumptions on Eve (the opponent)

- Cryptology = cryptography + cryptanalysis
- Eve knows the algorithm, except for the key (Kerckhoffs's principle)
- increasing capability of Eve:
  - knows some information about the plaintxt (e.g., in English)
  - knows part of the plaintext
  - can choose (part of) the plaintext and look at the ciphertext
  - can choose (part of) the ciphertext and look at the plaintext

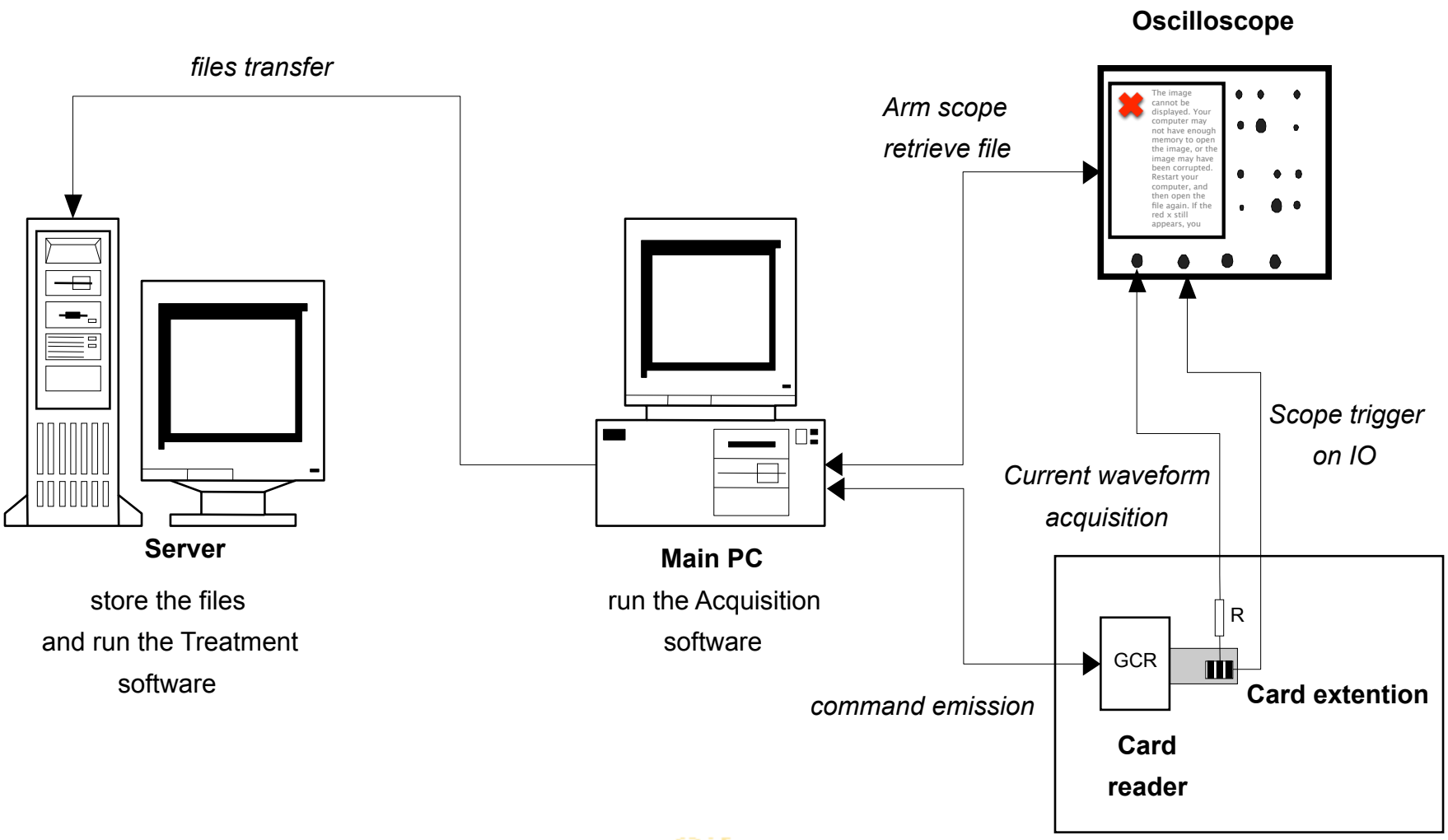
Assumptions on Eve (the opponent)

- A scheme is broken if Eve can deduce the key or obtain additional plaintext
- Eve can always try all possible keys till "meaningful" plaintext appears:
  a brute force attack
  - solution: large key space
- Eve will try to find shortcut attacks (faster than brute force)
  - history shows that designers are too optimistic about the security of their cryptosystems

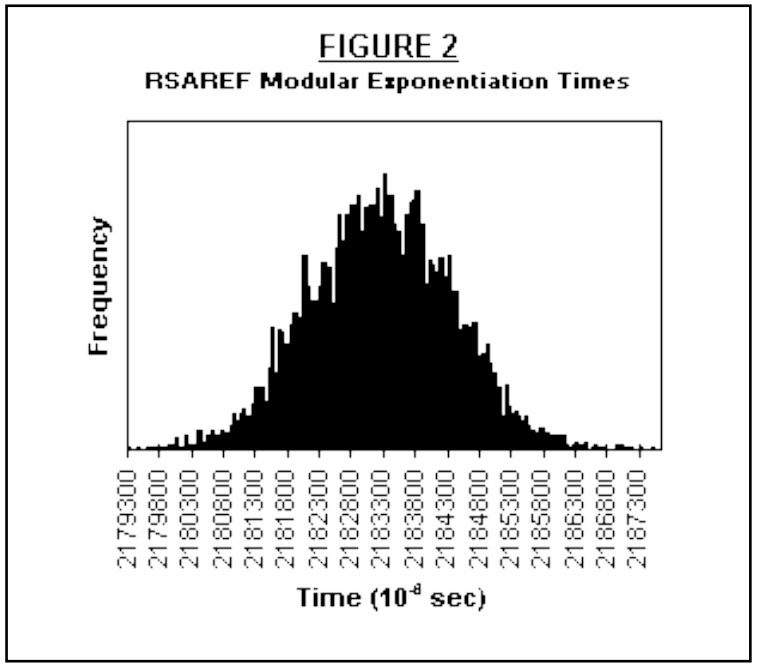
New assumptions on Eve

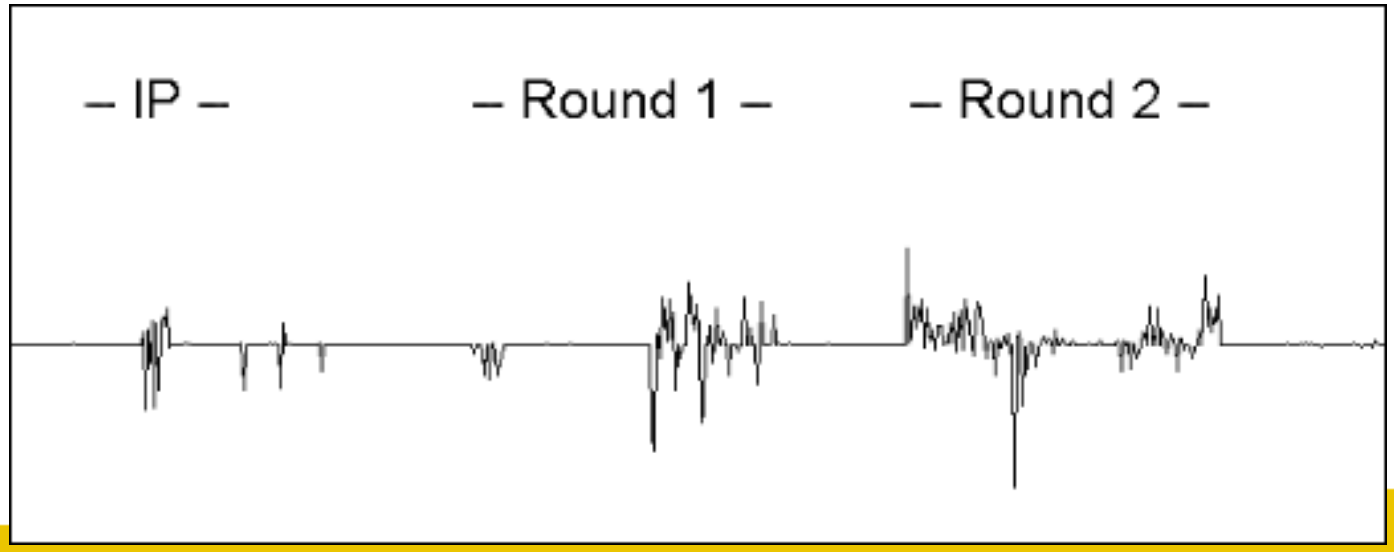
- Eve may have access to side channels
  - timing attacks
  - simple power analysis
  - differential power analysis
  - differential fault analysis
  - electromagnetic interference

Side channel analysis

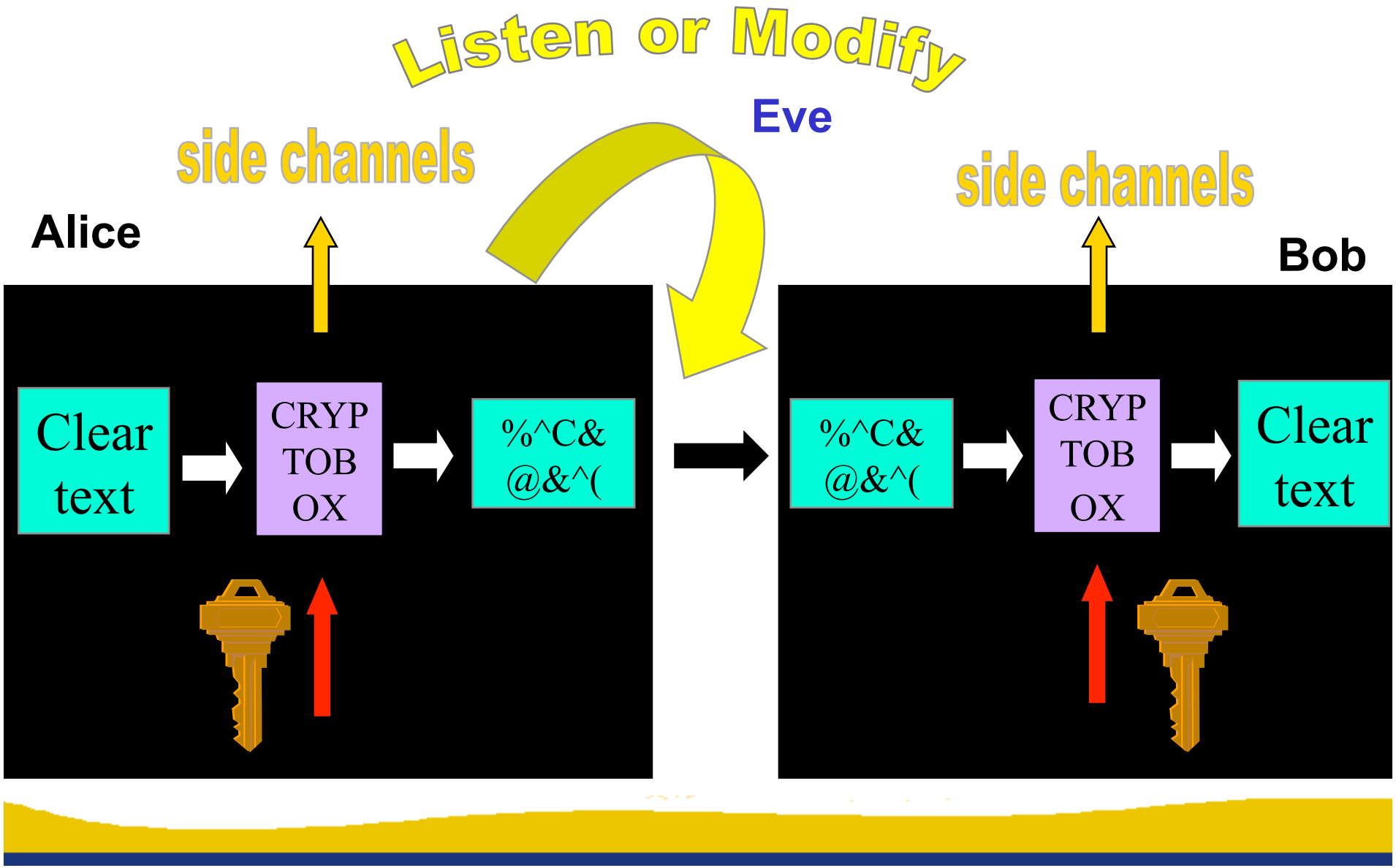


Timing attacks and power analysis

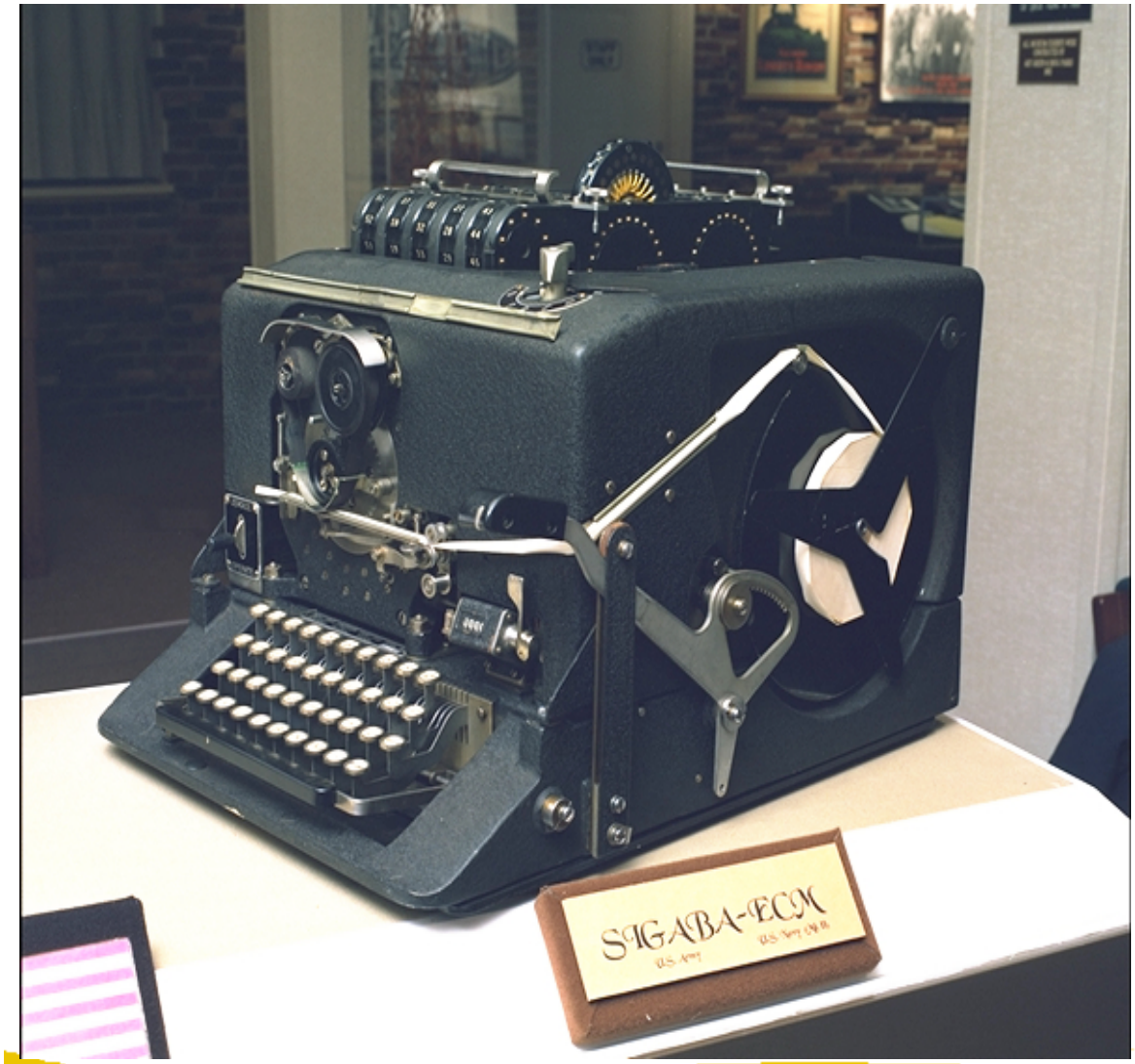




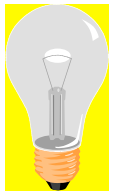
Cryptology + side channels

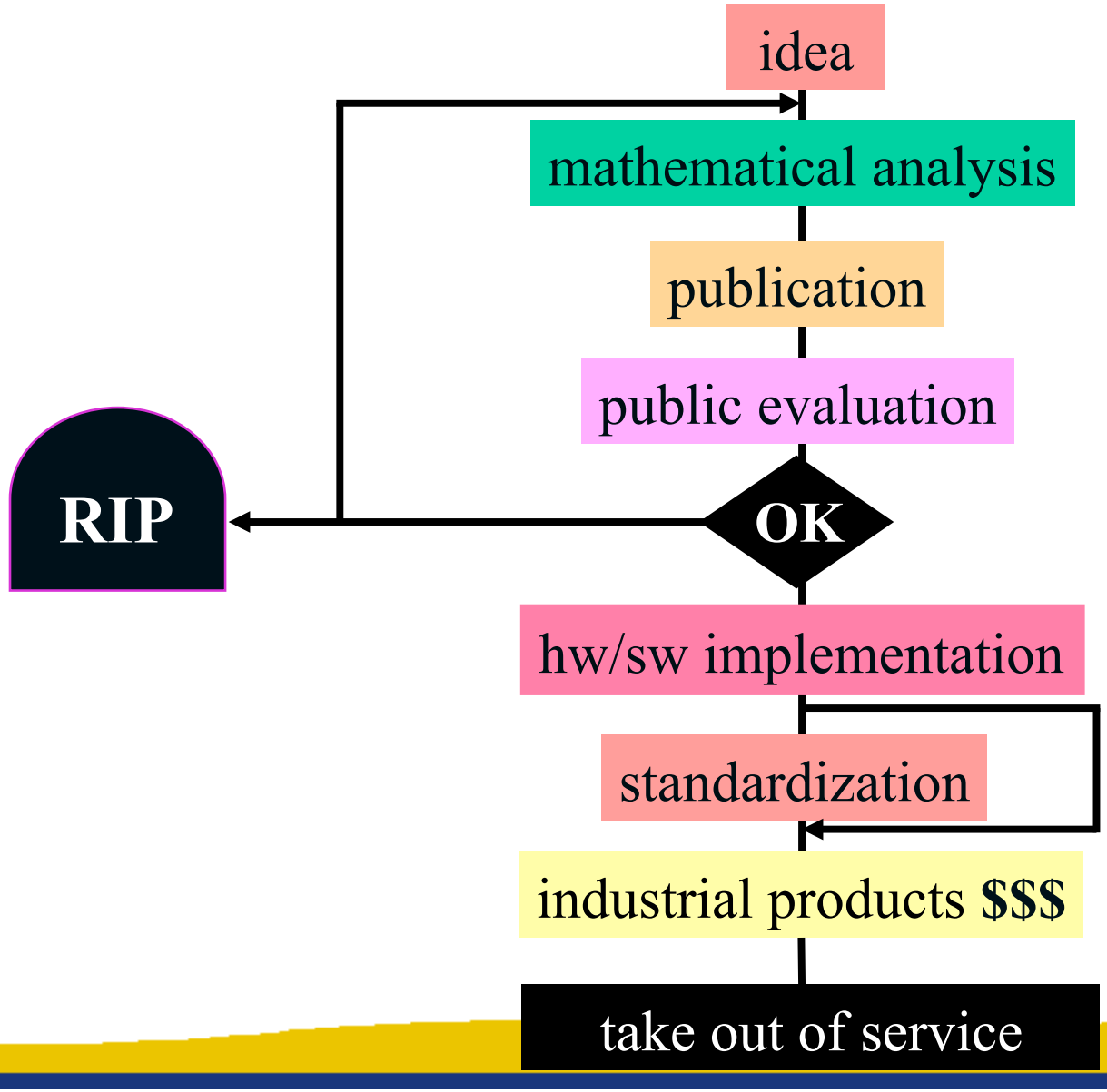


The Rotor machines (WW II)



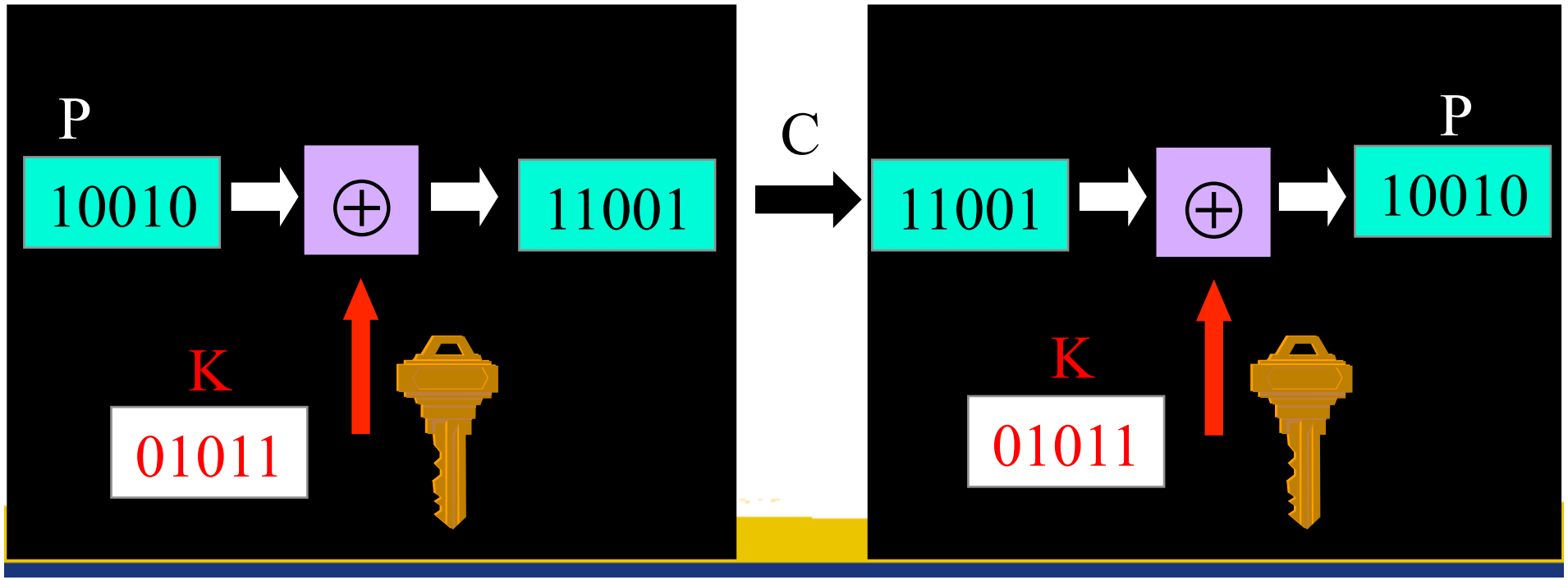
Life cycle of a cryptographic algorithm





Vernam scheme (1917) + Shannon (1948)

- key is random string, as long as the plaintext



Vernam scheme

- perfect secrecy: ciphertext gives opponent no additional information on the plaintext or $H(P|C)=H(P)$
- impractical: key is as long as the plaintext
- but this is optimal: for perfect secrecy $H(K) \geq H(P)$

Three approaches in cryptography

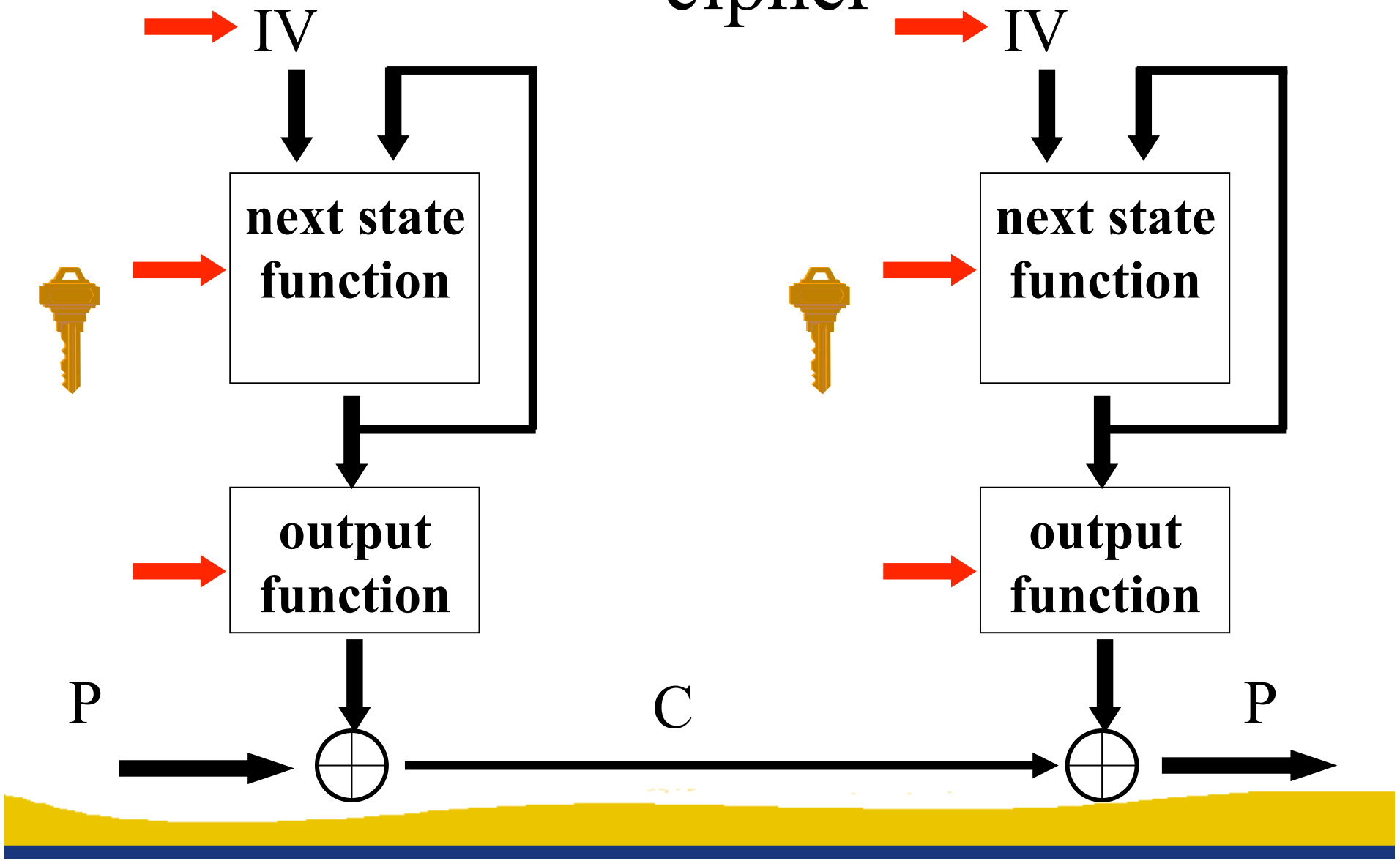
- **information theoretic** security
  - ciphertext only
  - part of ciphertext only
  - noisy version of ciphertext
- **system-based** or practical security
  - also known as "prayer theoretic" security
- **complexity theoretic** security:

  model of computation, definition, proof
  - variant: quantum cryptography

Design of ciphers

- More on this in a week
- For now, the high-level details
  - Symmetric key cryptography
    - Stream ciphers
    - Block ciphers
    - Message authentication codes (MACs)
    - Hash functions
  - Public key cryptography
    - Encryption
    - Digital signatures

Model of a practical stream cipher

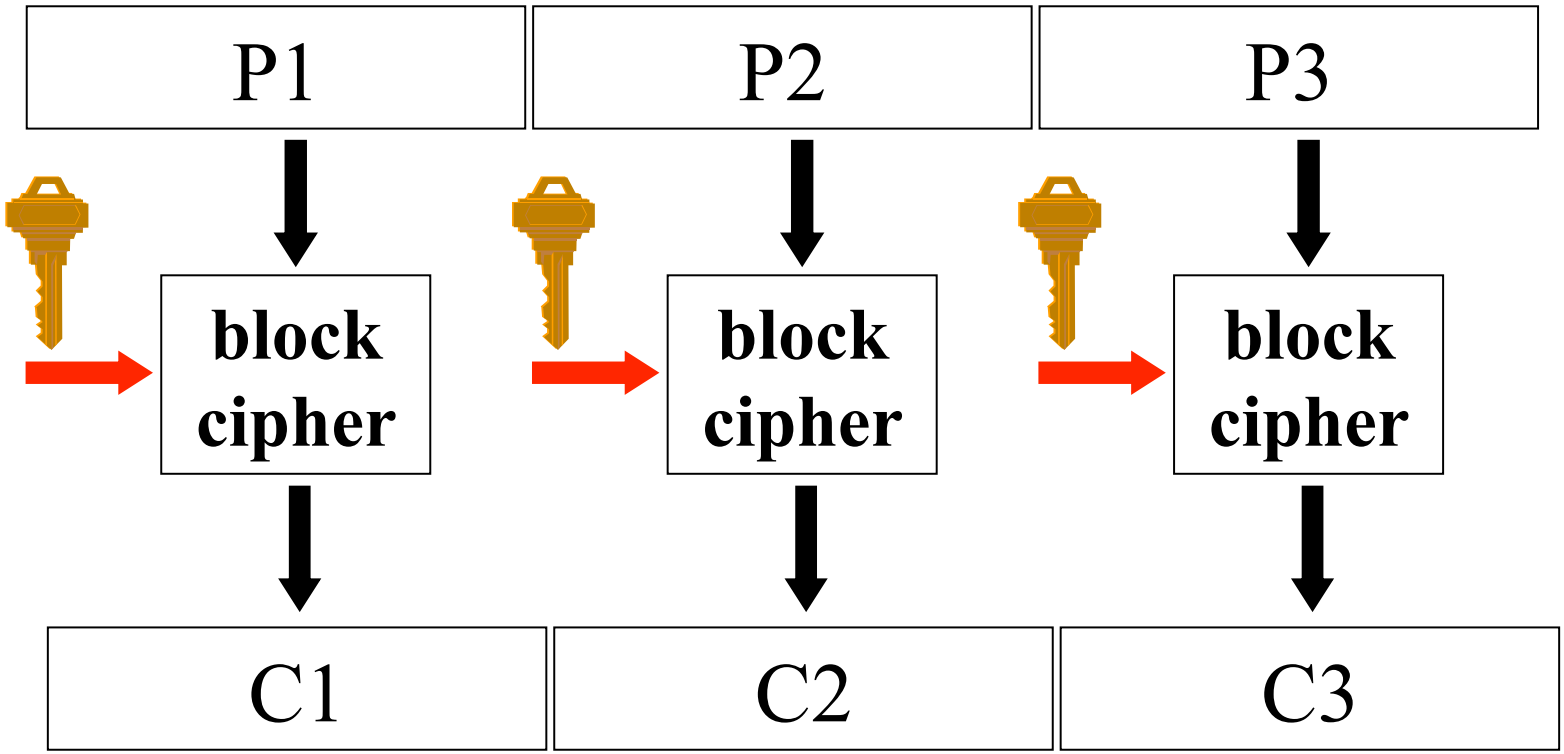


Example stream ciphers

- Bluetooth
- A5 (used in GSM cel phones)
- RC4 (used by most SSL web sites)

- Generally faster than block ciphers
  - Often less secure
  - If you ever reuse a key, the system collapses

Block cipher



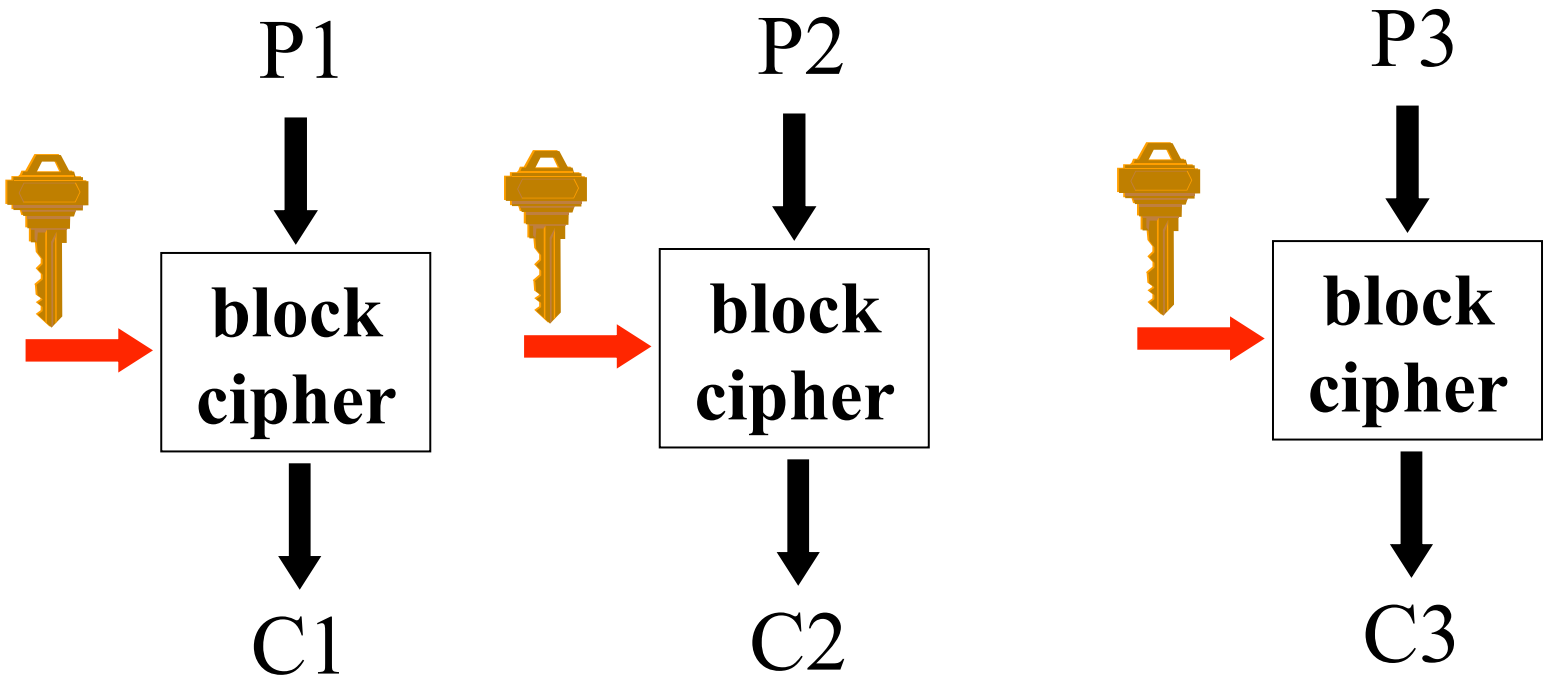
- larger data units: 64…128 bits
- memoryless
- repeat simple operation (round) many times

Example block ciphers

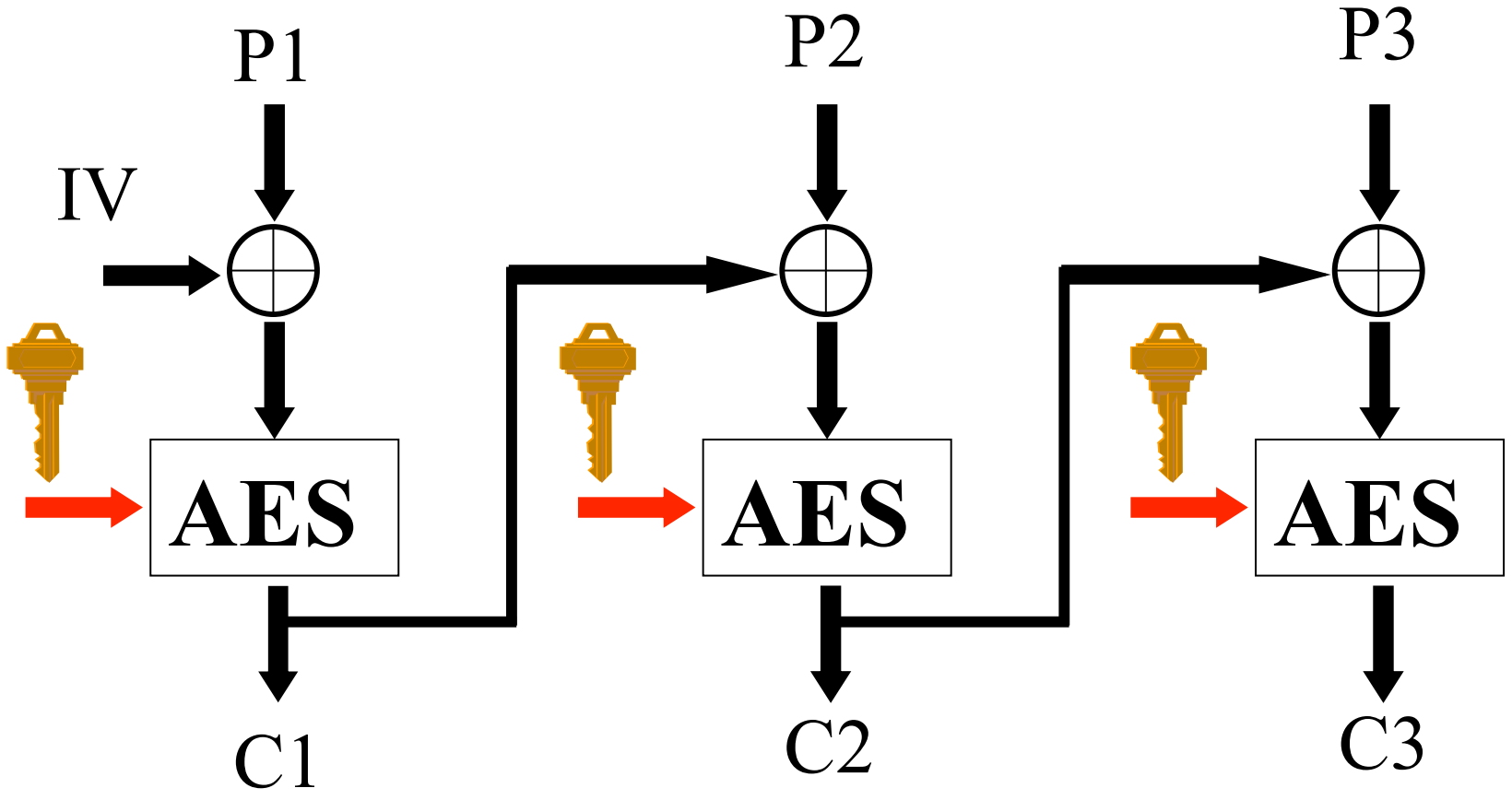
- DES (56-bit), Triple-DES (168-bit)
- AES / Rijndael (several key lengths)
- Many, many others

- Generally slower
- Very versatile: can make stream ciphers, hash functions, many other uses

How NOT to use a block cipher: ECB mode



How to use a block cipher: CBC mode



need random IV

Symmetric cryptology: data authentication

- the problem
- hash functions without a key
  - MDC: Manipulation Detection Codes
- hash functions with a secret key
  - MAC: Message Authentication Codes

Data authentication: the problem

- encryption provides confidentiality:
  - prevents Eve from learning information on the cleartext/plaintext
  - but does not protect against modifications (active eavesdropping)
- Bob wants to know:
  - the **source** of the information (data origin)
  - that the information has not been **modified**
  - (optionally) **timeliness** and **sequence**
- data authentication is typically more complex than data confidentiality

Data authentication: MDC

- MDC (manipulation detection code)
- Protect short hash value rather than long text

- (MD5)
- SHA-1
- SHA-256, -512
- RIPEMD-160

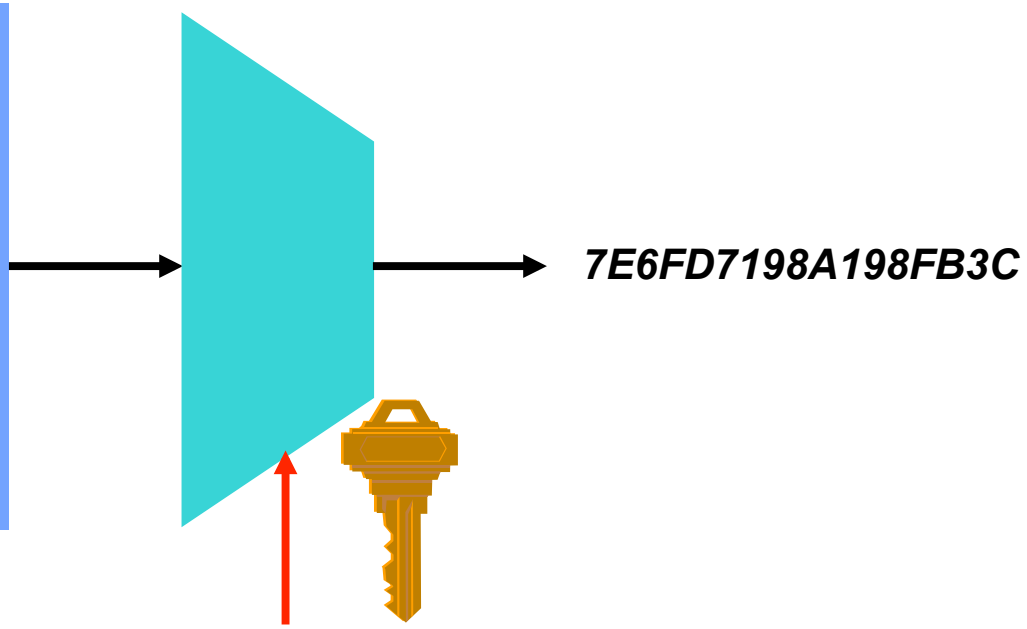
This is an input to a cryptographic hash function. The input is a very long string, that is reduced by the hash function to a string of fixed length. There are additional security conditions: it should be very hard to find an input hashing to a given value (a preimage) or to find two colliding inputs (a collision).

Data authentication: MAC

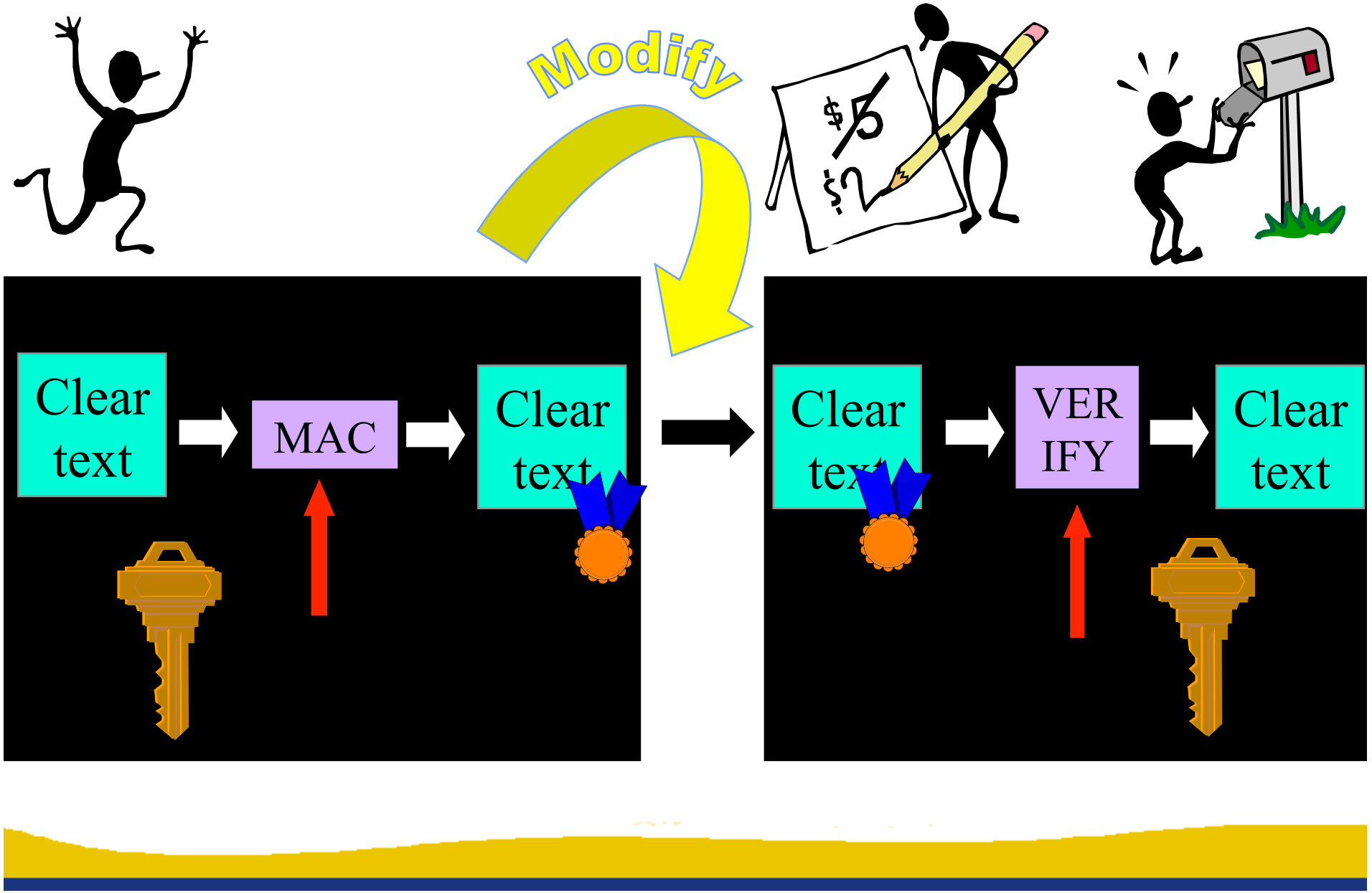
- Replace protection of authenticty of (long) message by protection of secrecy of (short) key
- Add MAC to the plaintext

- CBC-MAC
- HMAC

This is an input to a MAC algorithm. The input is a very long string, that is reduced by the hash function to a string of fixed length. There are additional security conditions: it should be very hard for someone who does not know the secret key to compute the hash function on a new input.



MAC algorithms



Public-key cryptology

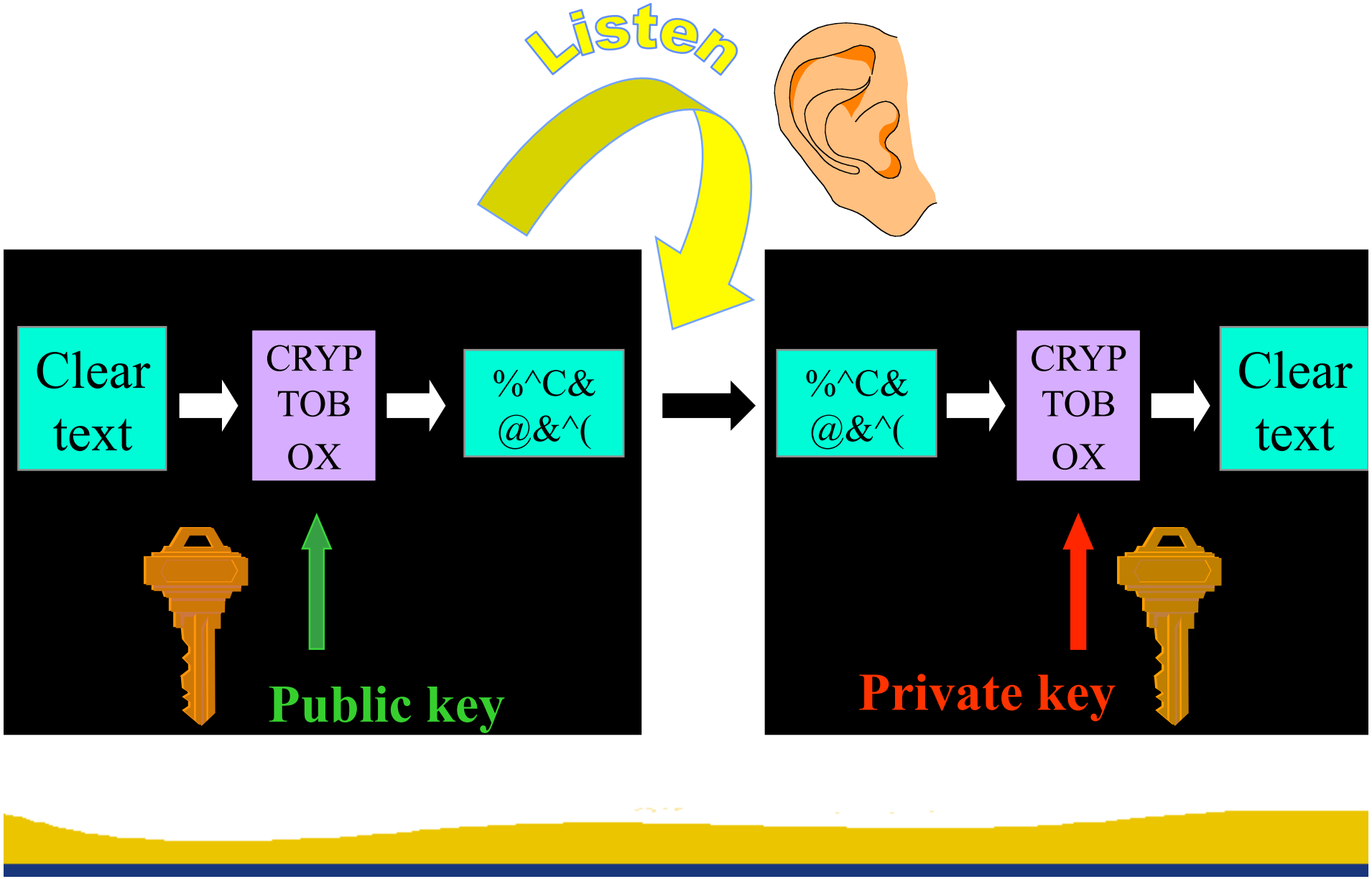
- the problem
- public-key encryption
- digital signatures
- an example: RSA
- advantages of public-key cryptology

Limitation of symmetric cryptology

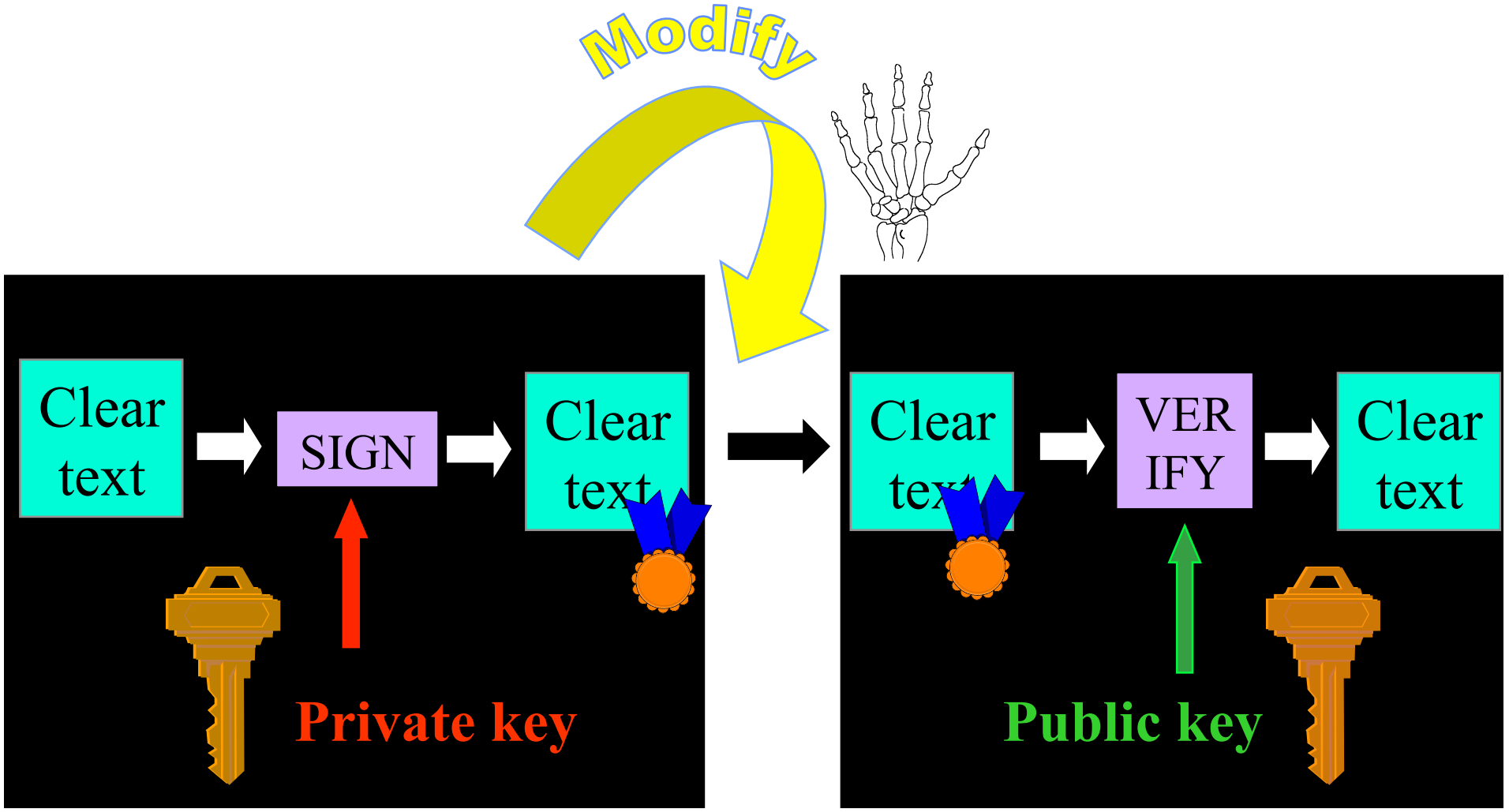
- Reduce security of information to security of keys

- But: how to establish these secret keys?
  - Cumbersome and expensive
  - Or risky: all keys in 1 place
- Do we really need to establish secret keys?

Public-key cryptology: encryption



Public key cryptology: digital signature



Advantages of public-key cryptology

- Reduce protection of information to protection of authenticity of public keys
- Confidentiality without establishing secret keys
  - extremely useful in an open environment
- Data authentication without shared secret keys: digital signature
  - sender and receiver have different capability
  - third party can resolve dispute between sender and receiver

Disadvantages of public-key cryptology

- Calculations in software or hardware two to three orders of magnitude slower than symmetric algorithms
- Longer keys: 1024 bits rather than 56…128 bits
- What if factoring is easy?

Key establishment

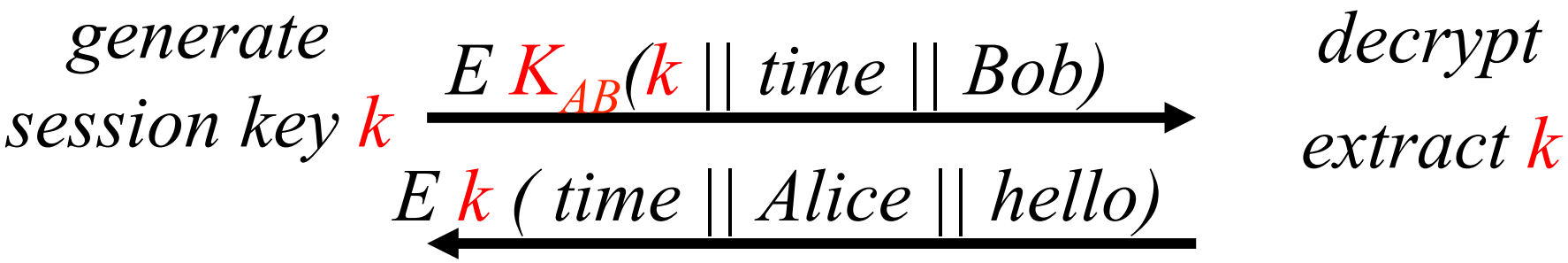
- The problem
- How to establish secret keys using secret keys?
- How to establish secret keys using public keys?
  - Diffie-Hellman and STS
- How to distribute public keys? (PKI)

Key establishment: the problem

- Cryptology makes it easier to secure information, by replacing the security of information by the security of keys
- The main problem is how to establish these keys
  - 95% of the difficulty
  - integrate with application
  - if possible transparent to end users

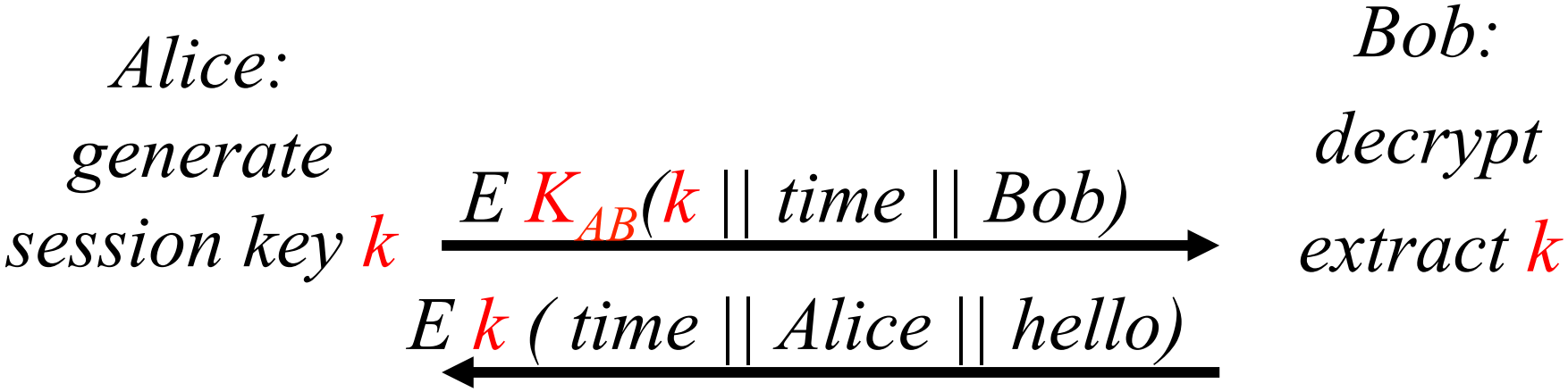
Point-to point symmetric key distribution

- Before: Alice and Bob share long term secret K_{AB}



- After: Alice and Bob share a short term key *k*
  - which they can use to protect a specific interaction
  - which can be thrown away at the end of the session

Attacks on the protocol



- Bob: proof that message is from Alice?
  - Proof that message isn't a repeat?
- Alice: proof that reply is from Bob?
  - Reply is in response to first message?

Protocol disambiguation

- Be explicit about names in messages
  - Alice names Bob in message to him
    - Harder for an adversary to forward to 3rd party
- Be explicit about responses
  - A response should say what it responds to

Message freshness

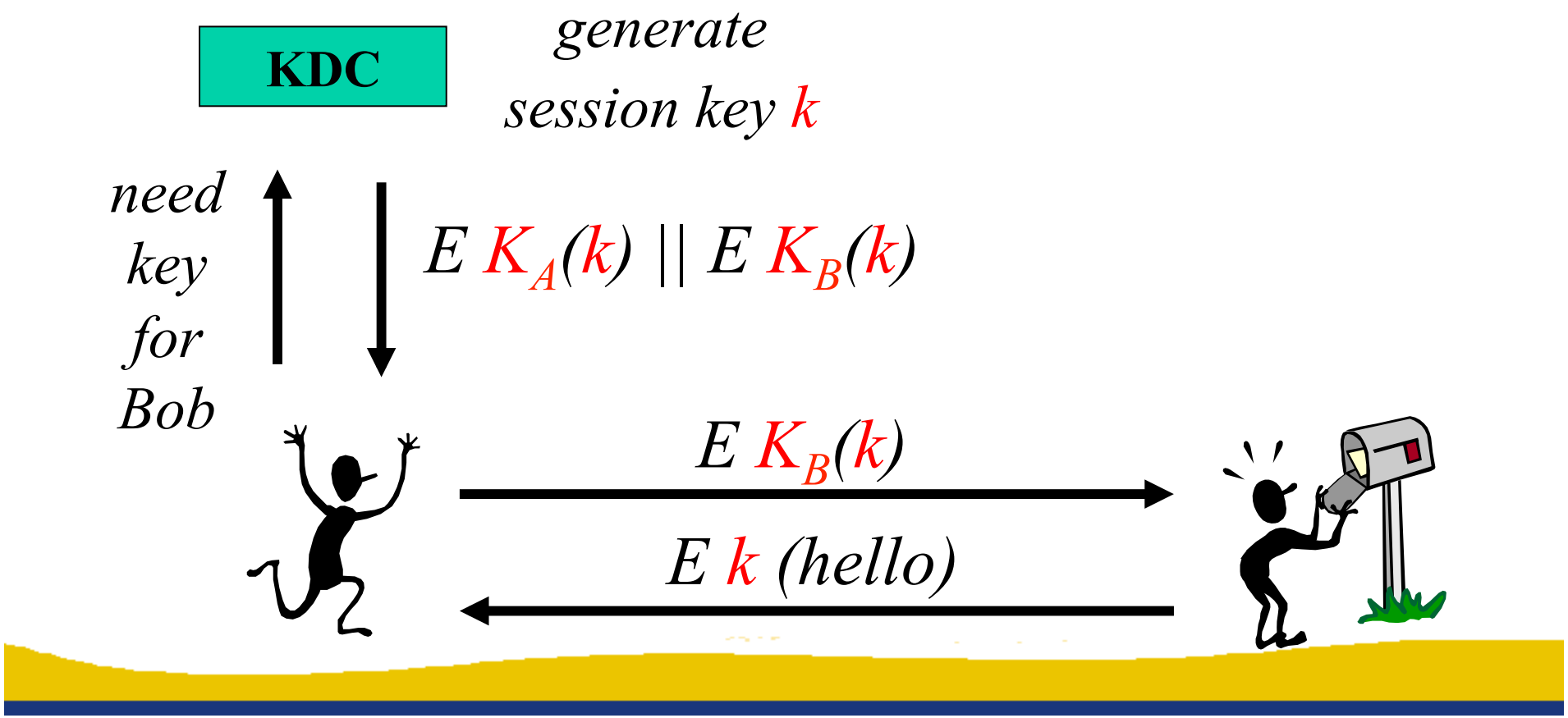
- Make sure a message isn't a repeat
  - Nonce handshakes (used by *cryptyc*, SSL)
    - A→B: N
    - B→A: $E_K(N)$
  - Timestamps (used by Kerberos)
  - Counters
    - A→B: $E_K(i{+}{+})$
  - Hash chains (used by S/Key)
    - A→B: $H(H(\ldots H(x)\,)) = H^{n-1}(x)$
    - B: $H(H^{n-1}(x)) \rightarrow H^n(x)$?

Banking example

- Checks have:
  - Unique check number (can't deposit twice)
  - Handwritten signature ("hard" to forge)
  - Source account number
  - Destination account name
  - Destination signature
    - proof that payment was accepted

Symmetric key distribution with 3rd party

- Before (KDC=Key Distribution Center)
  - Alice shares a long term secret with KDC: K_A
  - Bob shares long term secret with KDC: K_B



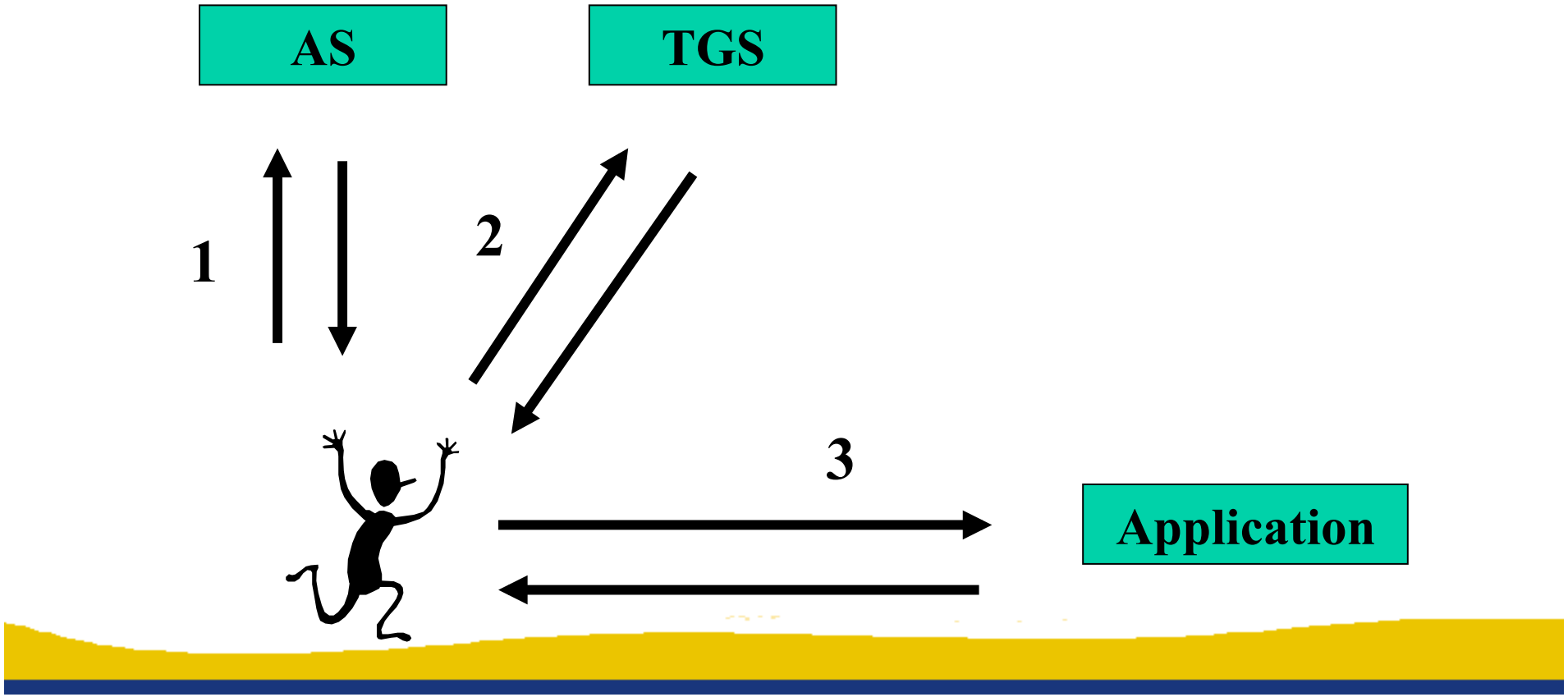
Symmetric key distribution with 3rd party(2)

- After: Alice and Bob share a short term key ***k***

- Need to trust third party!
- Single point of failure in system

Kerberos/Single Sign On (SSO)

- Alice uses her password only once per day

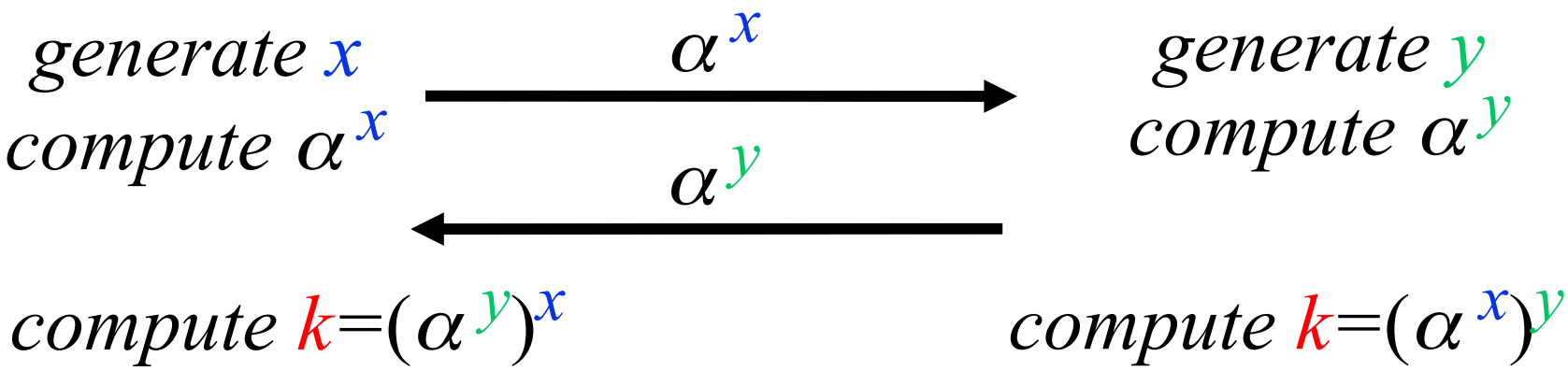


Kerberos/Single Sign On (2)

- Step 1: Alice gets a "day key" K_A from AS (Authentication Server)
  - based on a Alice's password (long term secret)
  - K_A is stored on Alice's machine and deleted in the evening
- Step 2: Alice uses K_A to get application keys k_i from TGS (Ticket Granting Server)
- Step 3: Alice can talk securely to applications (printer, file server) using application keys k_i

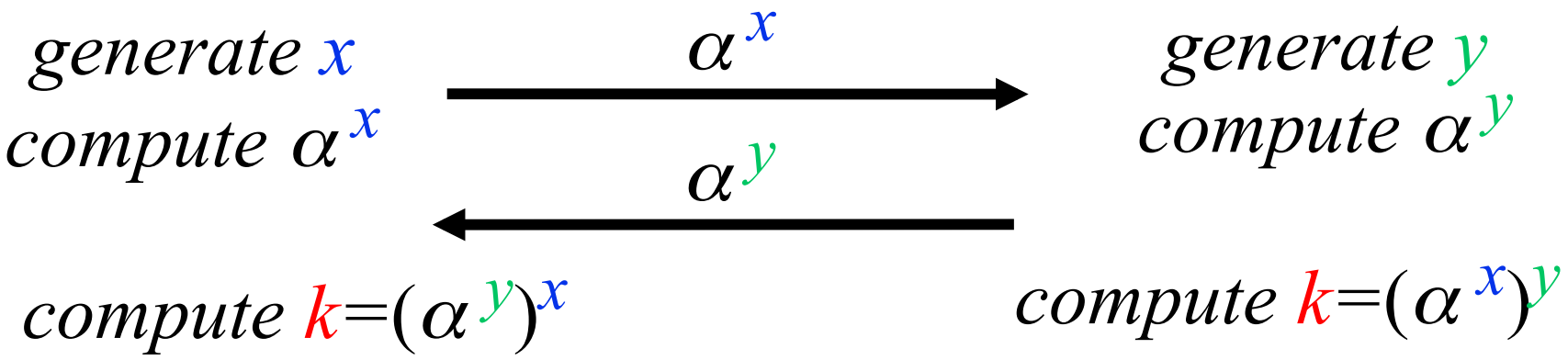
A public-key distribution protocol: Diffie-Hellman

- Before: Alice and Bob have never met and share no secrets; they know a public system parameter α



- After: Alice and Bob share a short term key k
  - Eve cannot compute k : in several mathematical structures it is hard to derive x from α^x (this is known as the discrete logarithm problem)

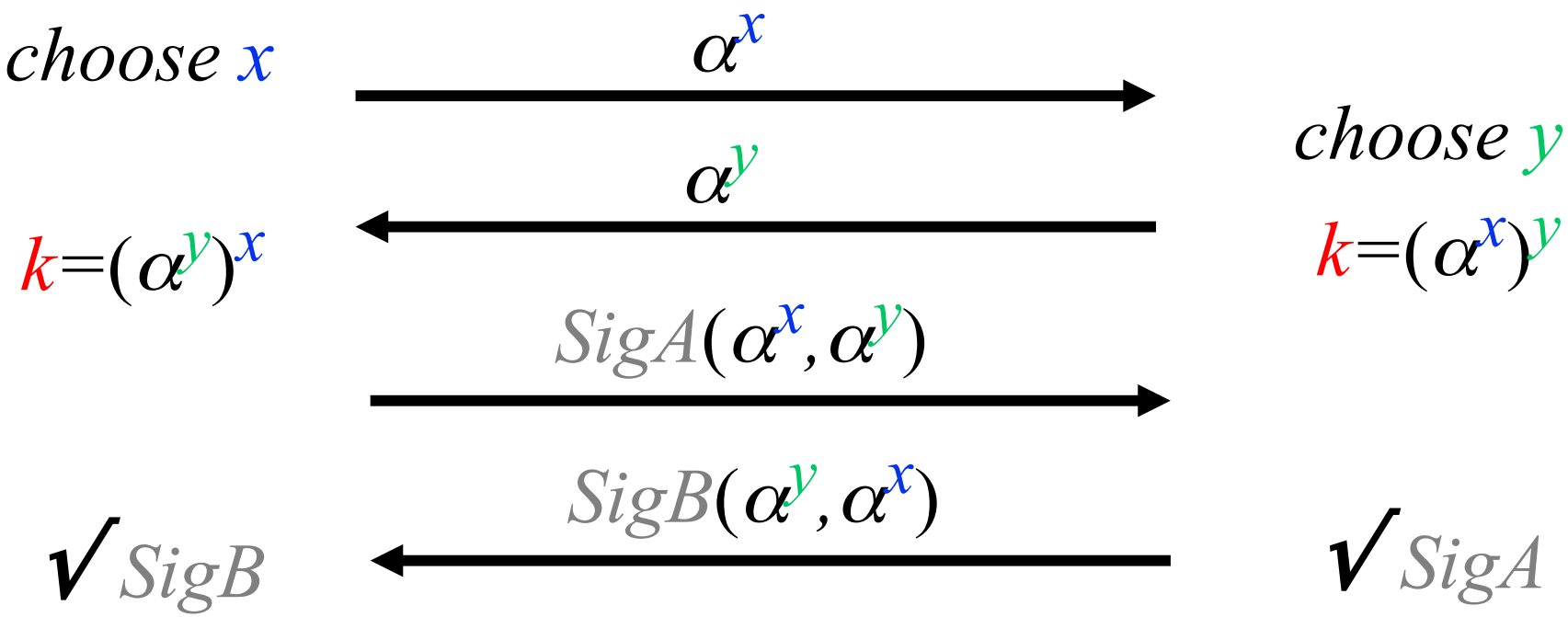
Diffie-Hellman (continued)



- BUT: How does Alice know that she shares this secret key ***k*** with Bob?
- Answer: Alice has no idea at all about who the other person is! The same holds for Bob.

Station to Station protocol (STS)

- The problem can be fixed by adding digital signatures
- This protocol plays a very important role on the Internet (under different names)



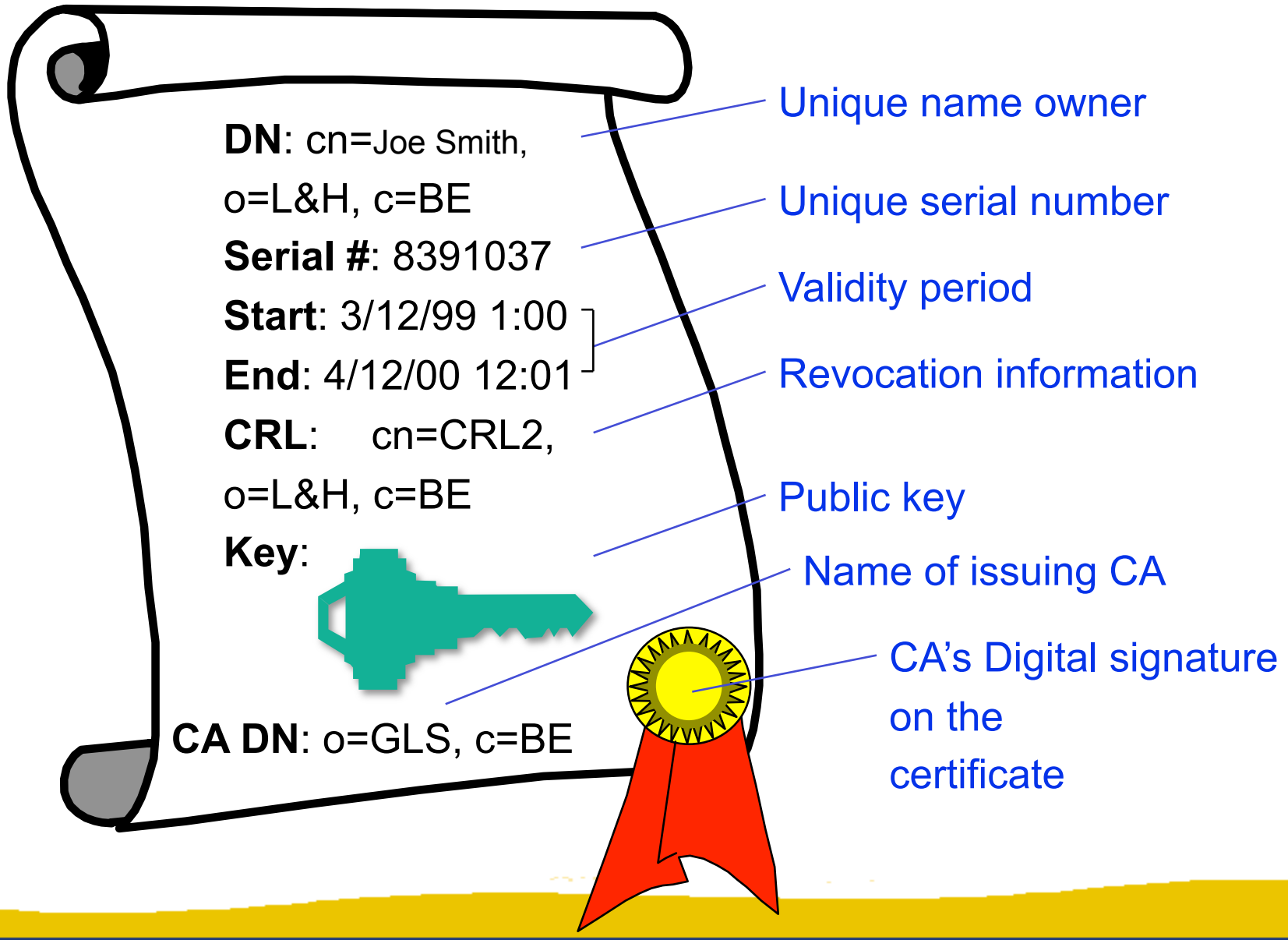
Distribution of public keys

- How do you know whose public key you have?
- Where do you get public keys?
- How do you trust public keys?
- What should you do if your private key is compromised?

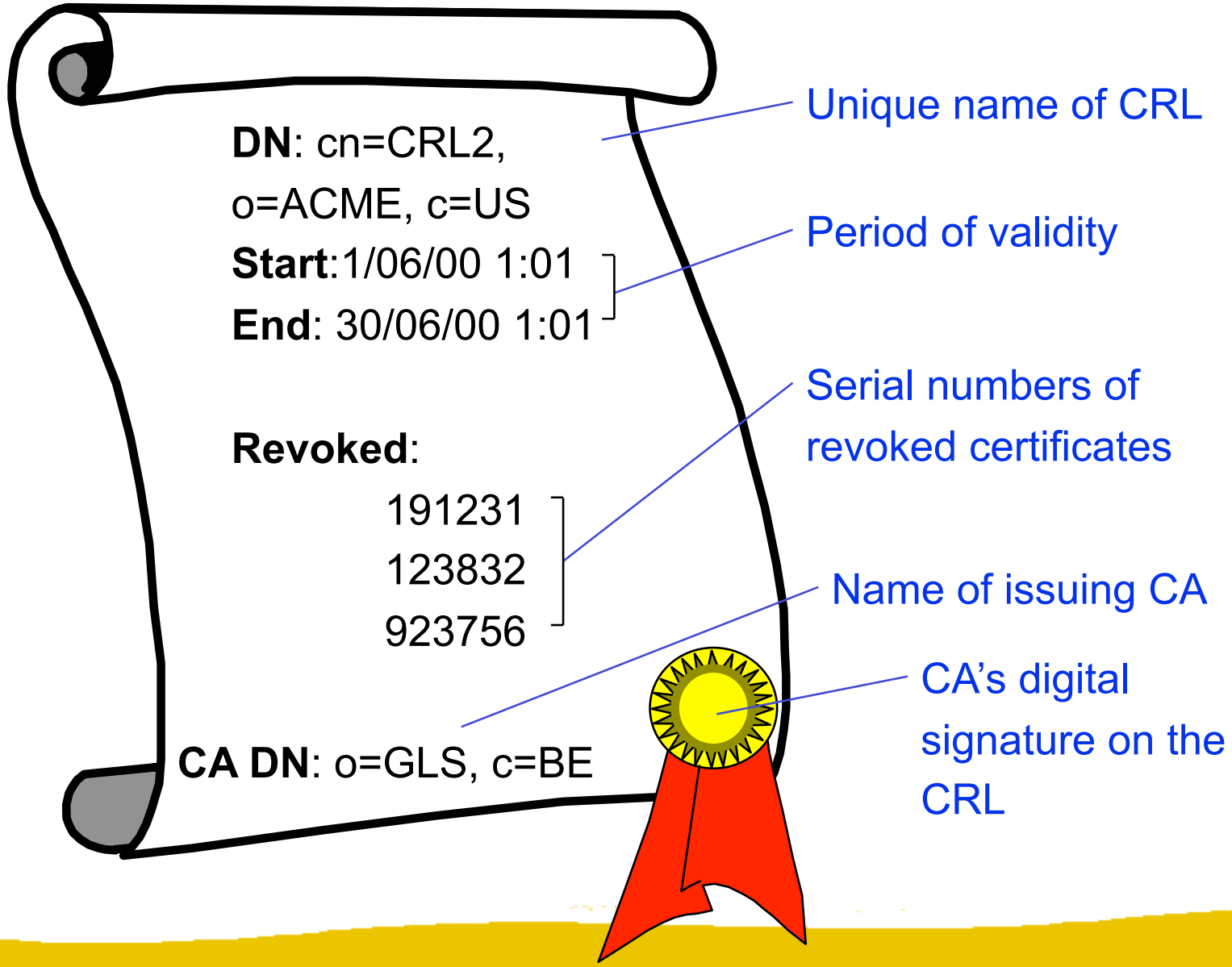
reduce protection of public key of many users to knowledge of a single public key of a Certification Authority (CA)

digital certificates &
Public Key Infrastructure (PKI)

Public Key Certificates



Certificate Revocation List

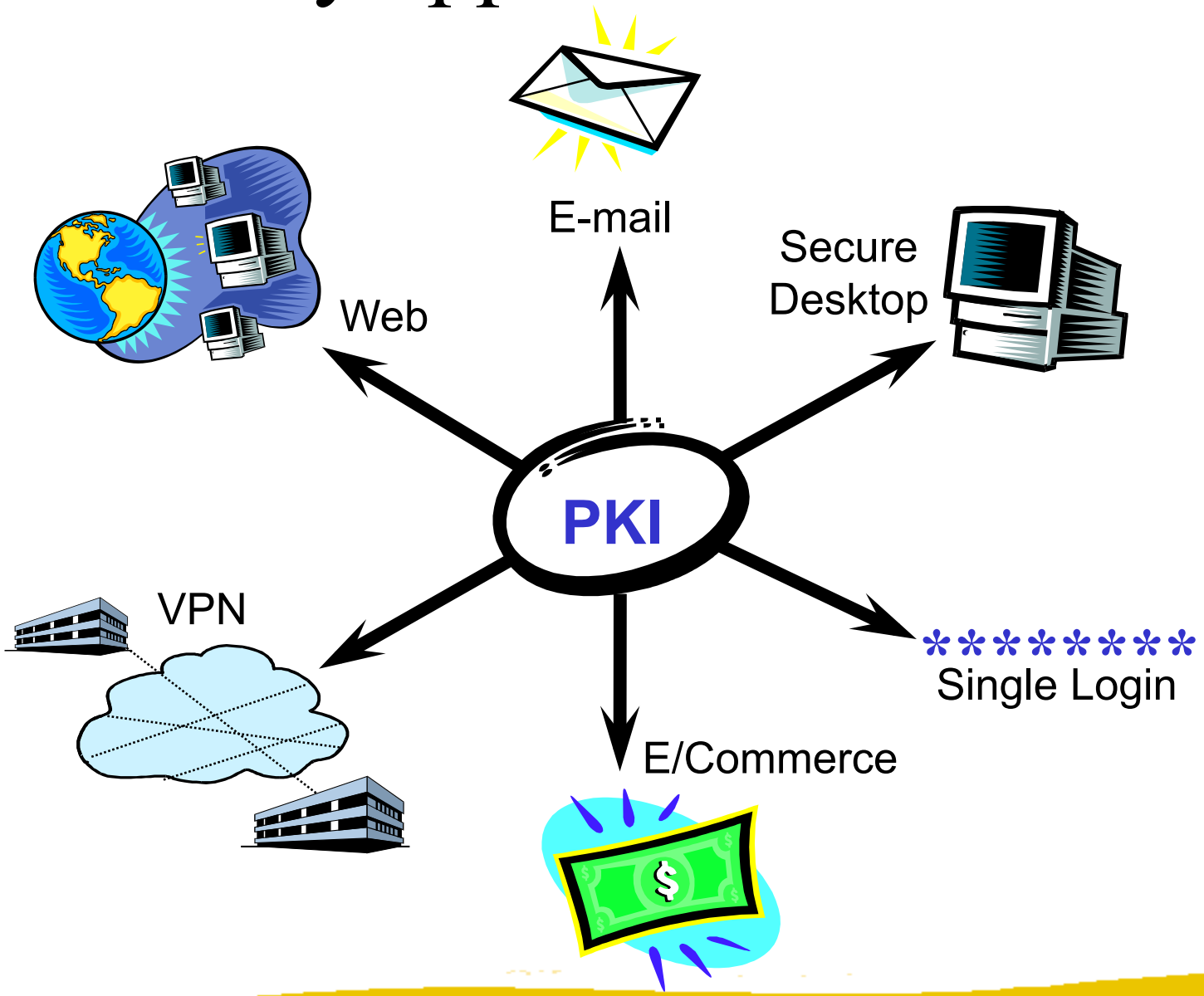


Essential PKI Components

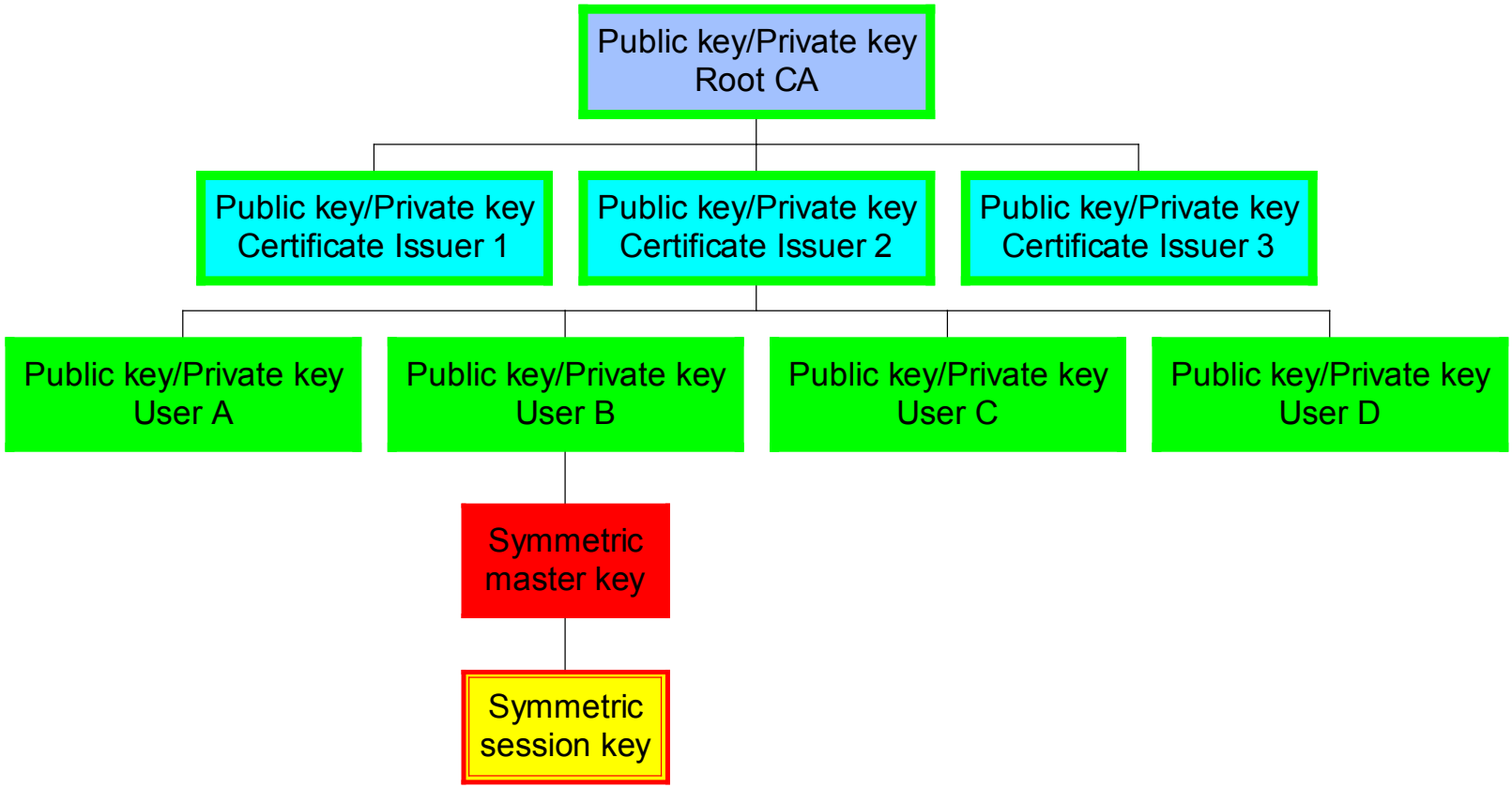
- Certification Authority
- Revocation system
- Certificate repository (“directory”)

- Key backup and recovery system
- Support for non-repudiation
- Automatic key update
- Management of key histories
- Cross-certification
- PKI-ready application software

PKI-ready application software



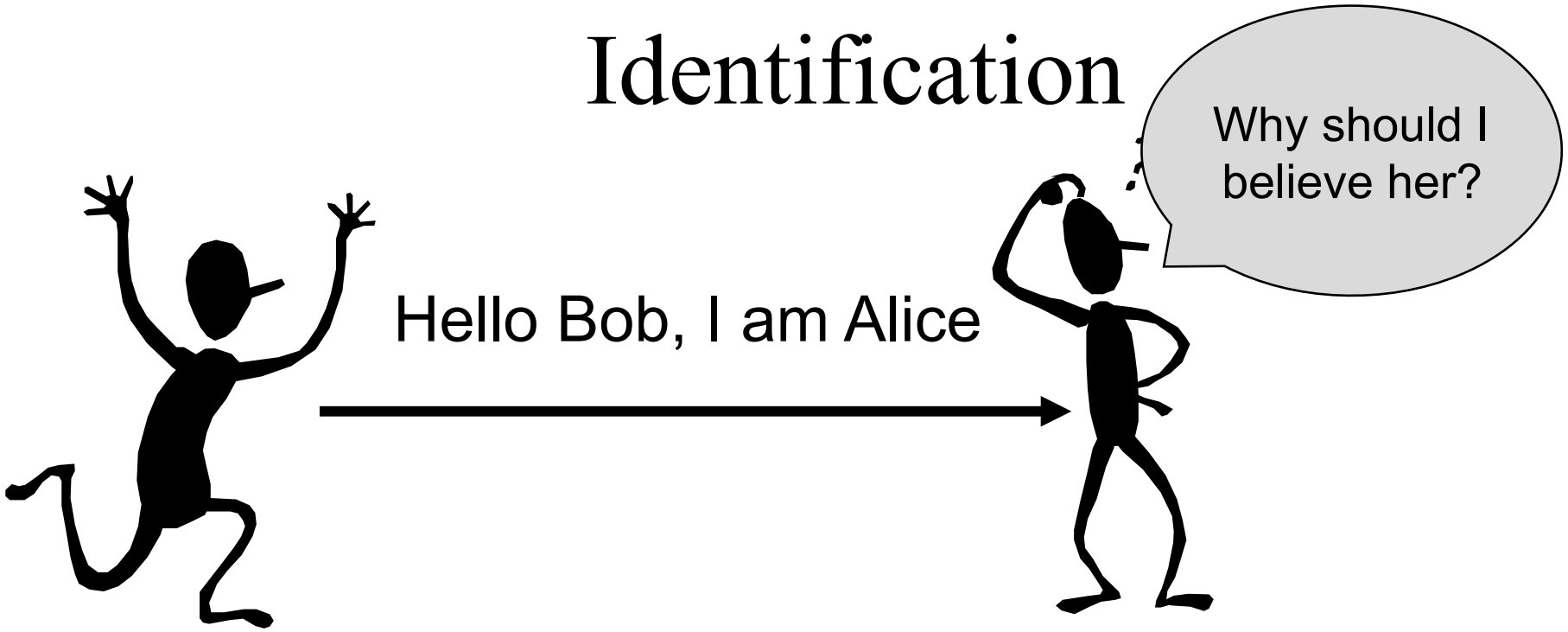
Example of a key hierarchy



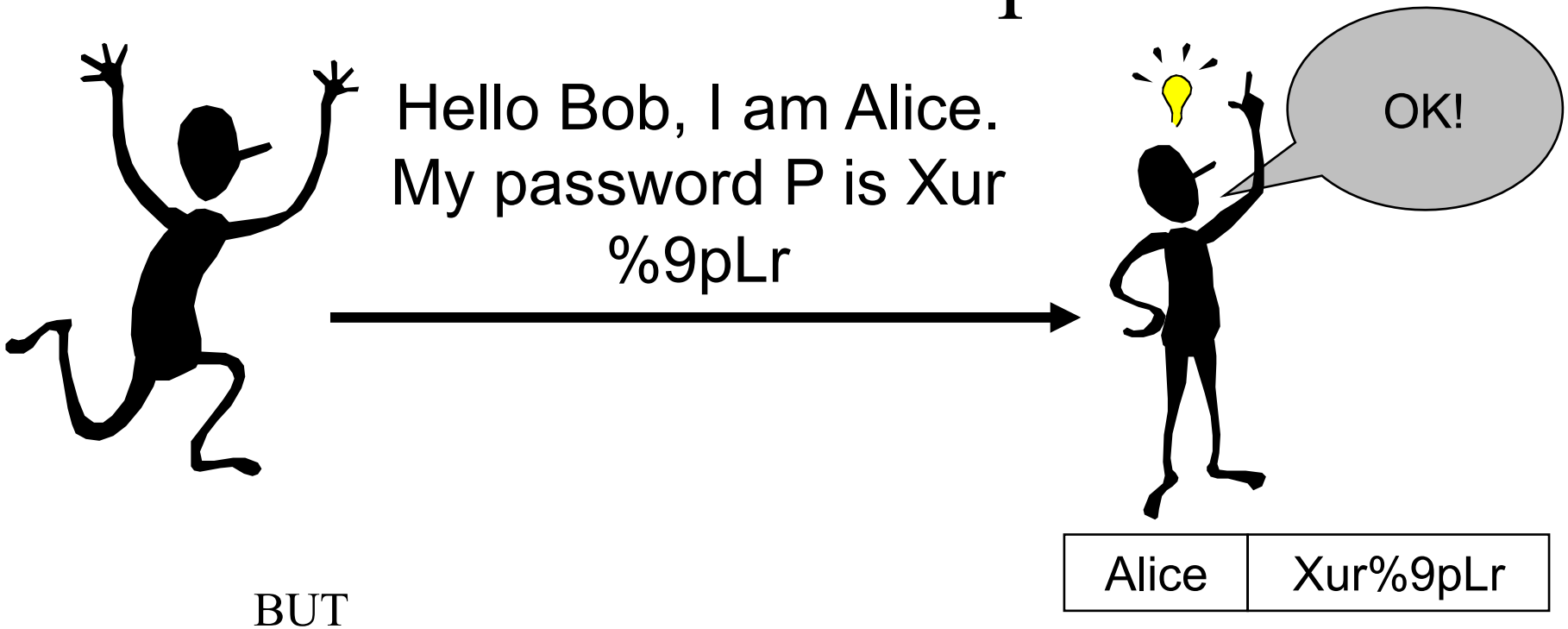
Identification

- the problem
- passwords
- challenge response with symmetric key and MAC
- challenge response with signatures

Identification



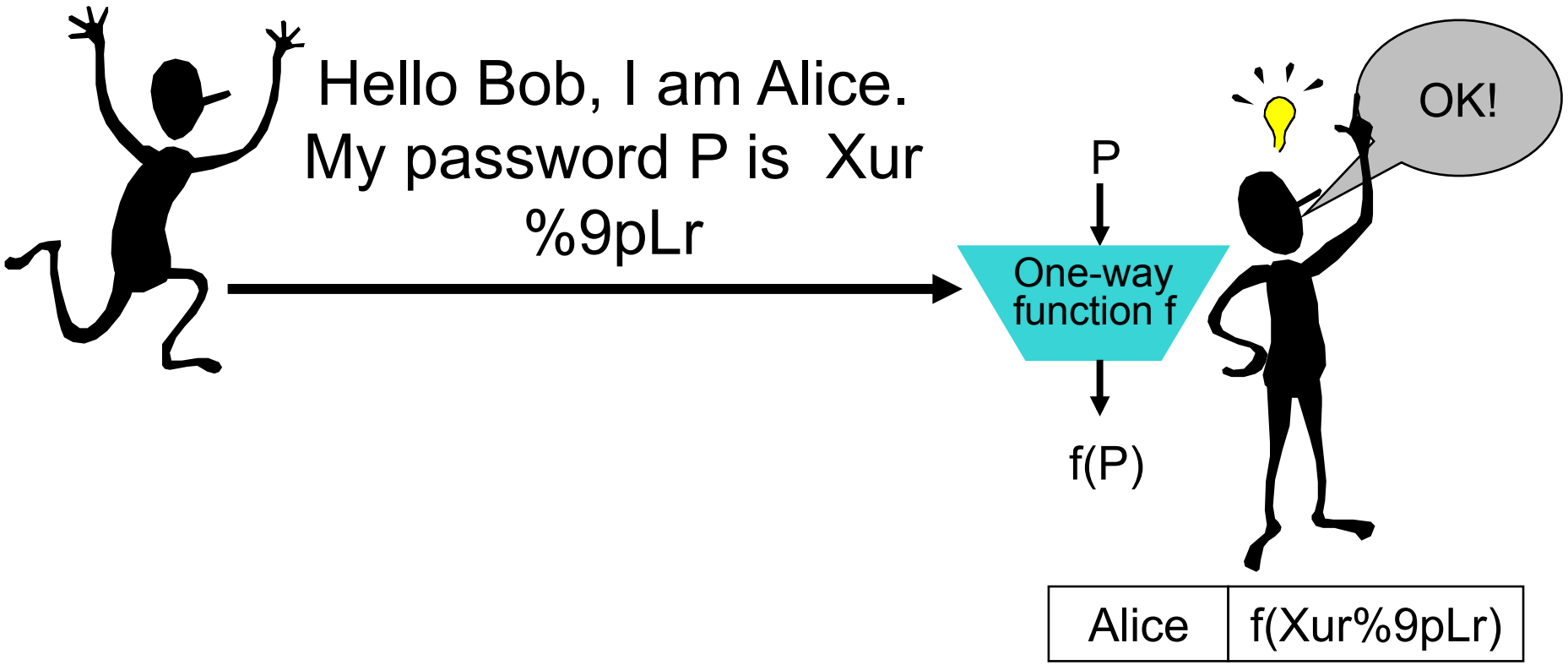
Identification with passwords



BUT

- Eve can guess the password
- Eve can listen to the channel and learn Alice's password
- Bob needs to know Alice's secret
- Bob needs to store Alice's secret in a secure way

Improved identification with passwords



Bob stores f(P) rather than Alice's secret P

- it is difficult to deduce P from f(P)

Variations on passwords

- Challenge-response (Bob must know *pw*)
  - A→B: "Hi, I'm Alice"
  - B→A: *N*
  - A→B: *Hash*(*N*, *pw*)
- Many, many recent schemes
  - Variations on Diffie-Hellman, etc.