Entity authentication and symmetric key establishment

Prof. Bart Preneel

COSIC

Bart.Preneel(at)esatDOTkuleuven.be

http://homes.esat.kuleuven.be/~preneel

February 2007

Outline

- 1. Cryptology: protocols
  - identification/entity authentication
  - key establishment
- 2. Public Key Infrastructures
- 3. Secure Networking protocols
  - Internet Security: email, web, IPSEC, SSL
- 4. How to use cryptography well
- 5. New developments in cryptology

Definitions (ctd)

Confidentiality
Integrity
Availability

	data	entities
confidentiality	encryption	anonymity
authentication	data authentication	identification

Authorisation

Non-repudiation of origin, receipt

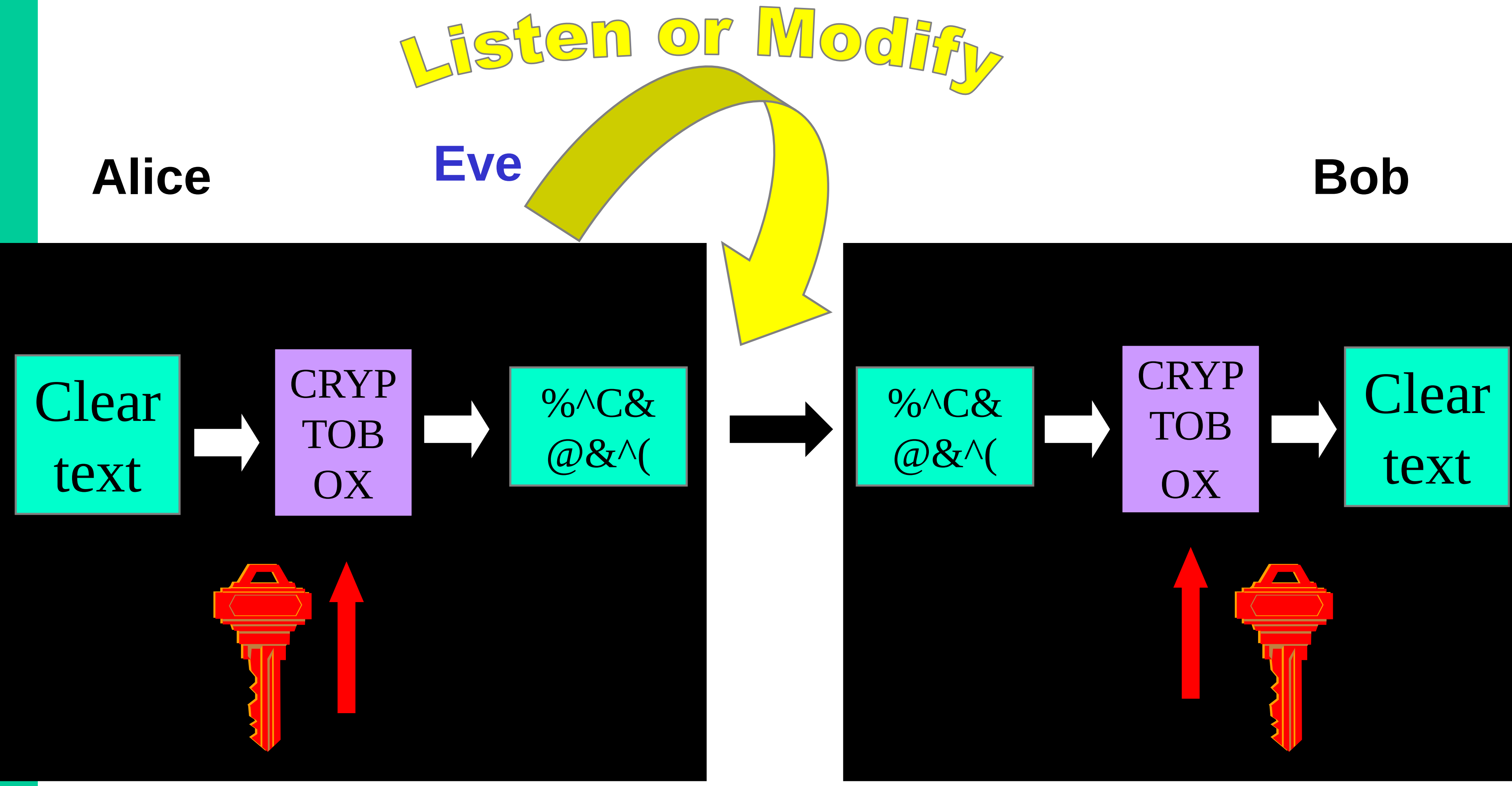
Contract signing

Notarisation and Timestamping

E-voting, e-auction,…

Don't use the word authentication without defining it

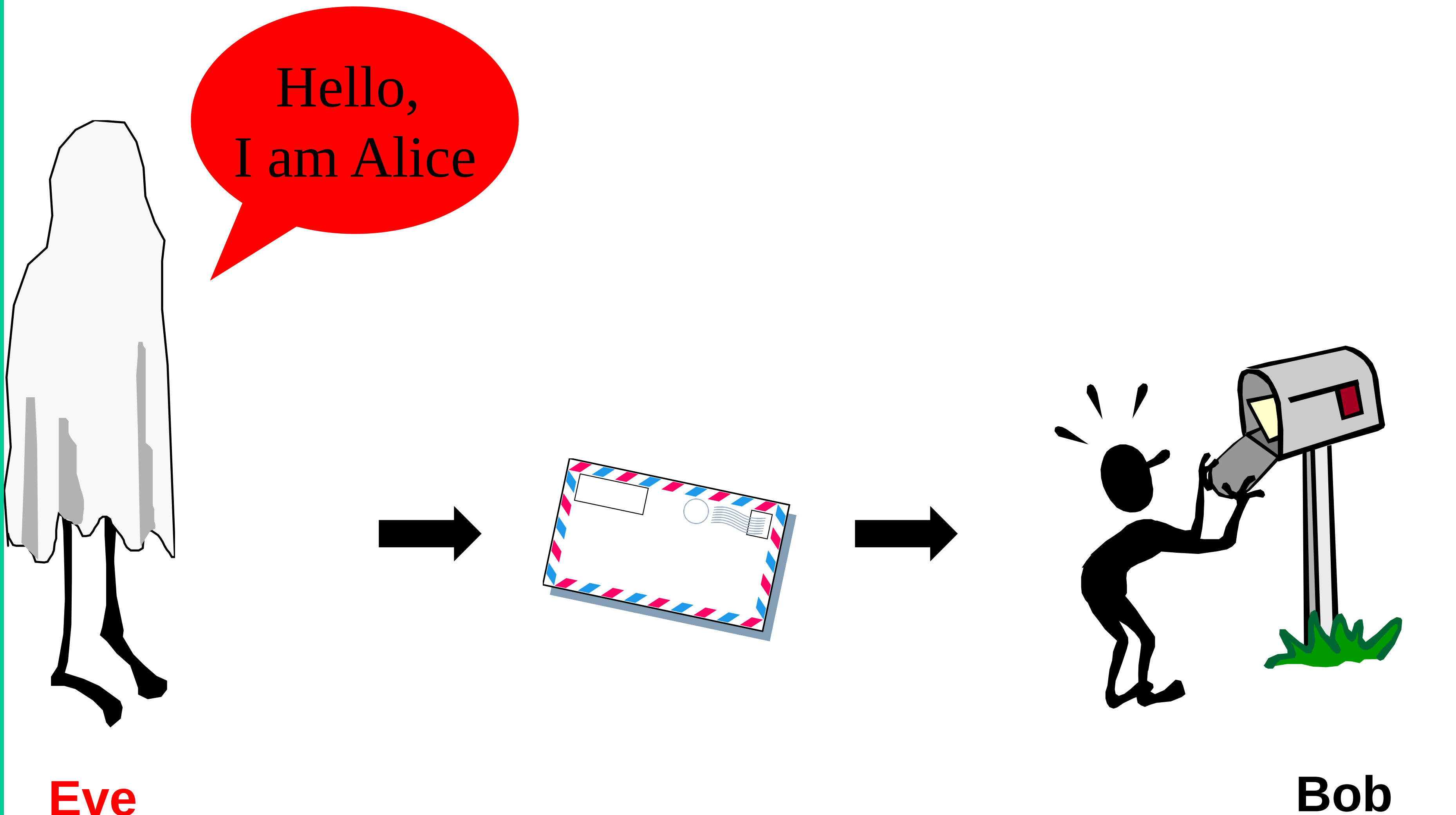
Cryptology: basic principles



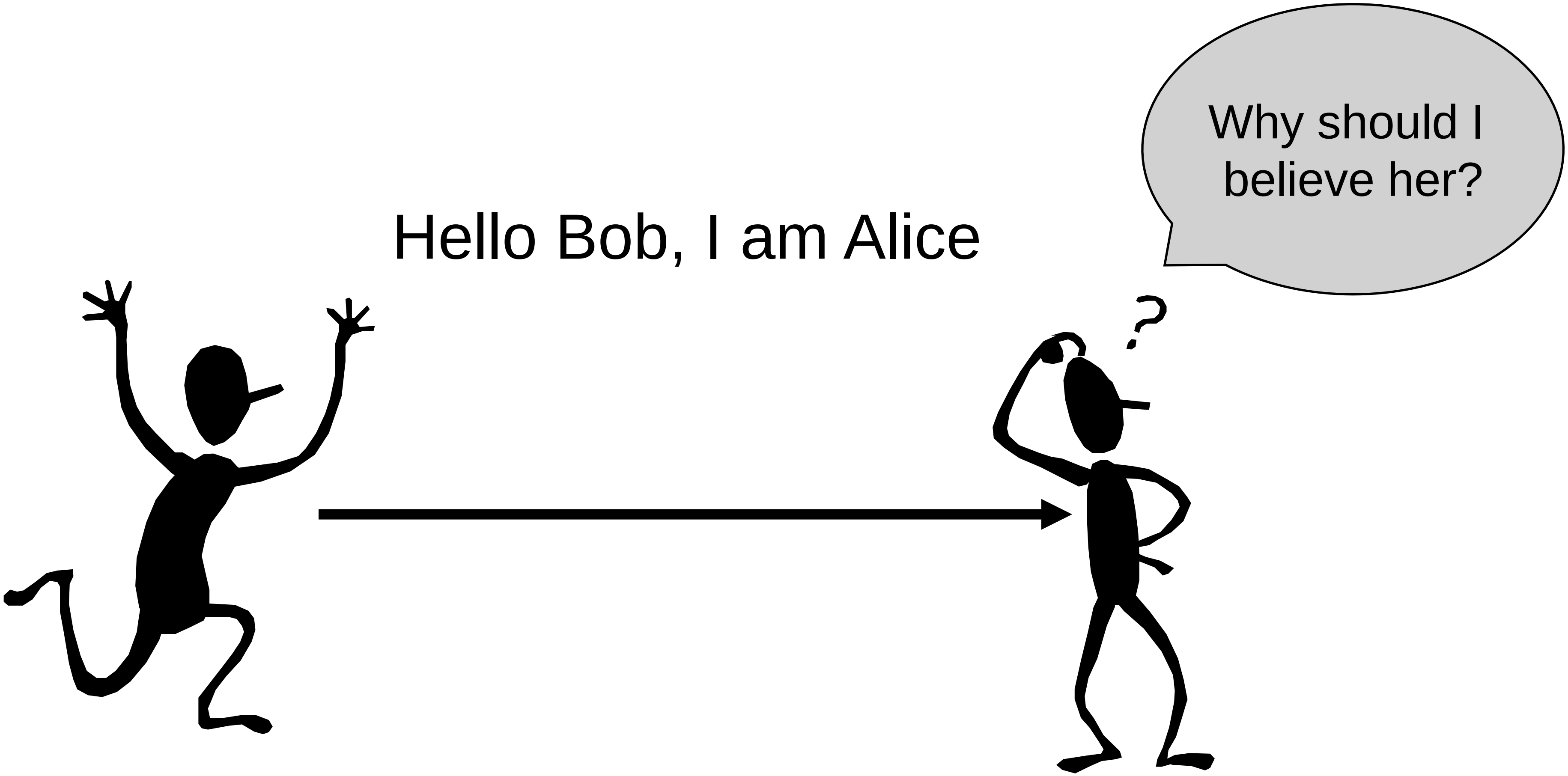
Identification

- the problem
- passwords
- challenge response with symmetric key and MAC (symmetric tokens)
- challenge response with public key (signatures, ZK)
- biometry
- symmetric key establishment and Kerberos
- public key establishment

Entity authentication



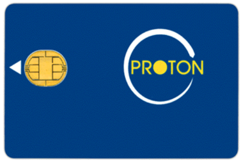
Entity authentication

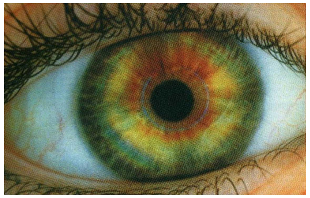


Identification is based on one or more of the following elements:

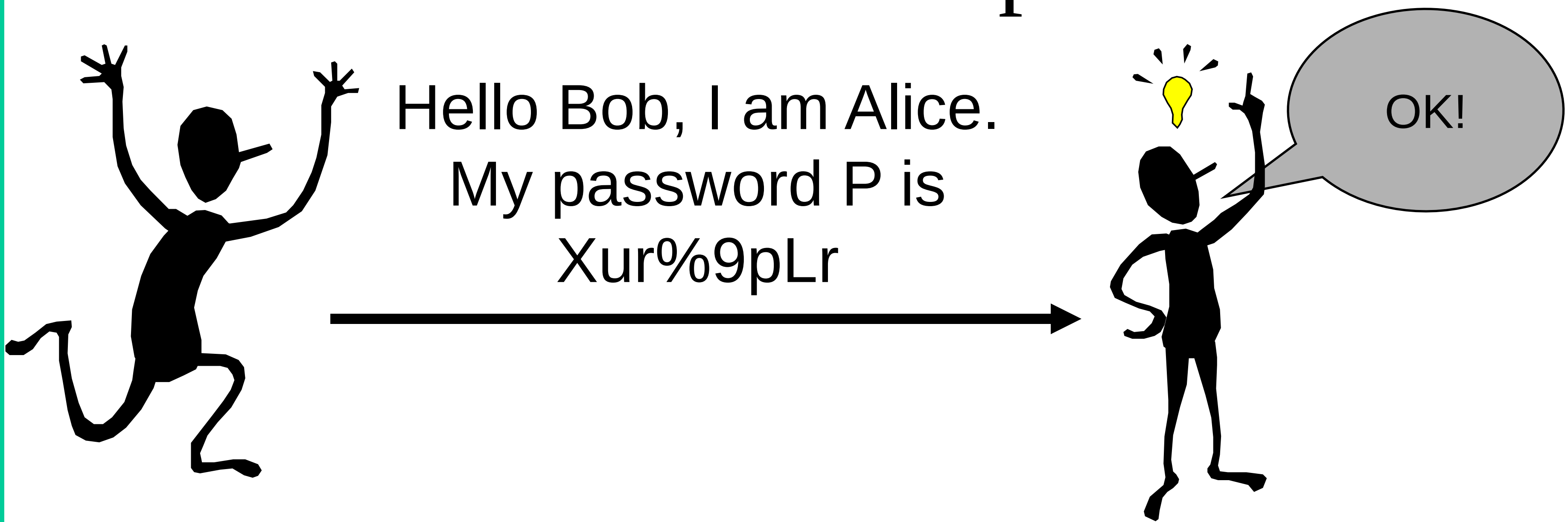
- what someone knows
  - password, PIN
- what someone has
  - magstripe card, smart card
- what someone is (biometrics)
  - fingerprint, retina, hand shape,...
- how someone does something
  - manual signature, typing pattern
- where someone is
  - dialback

ert5^r$#89Oy





Identification with passwords

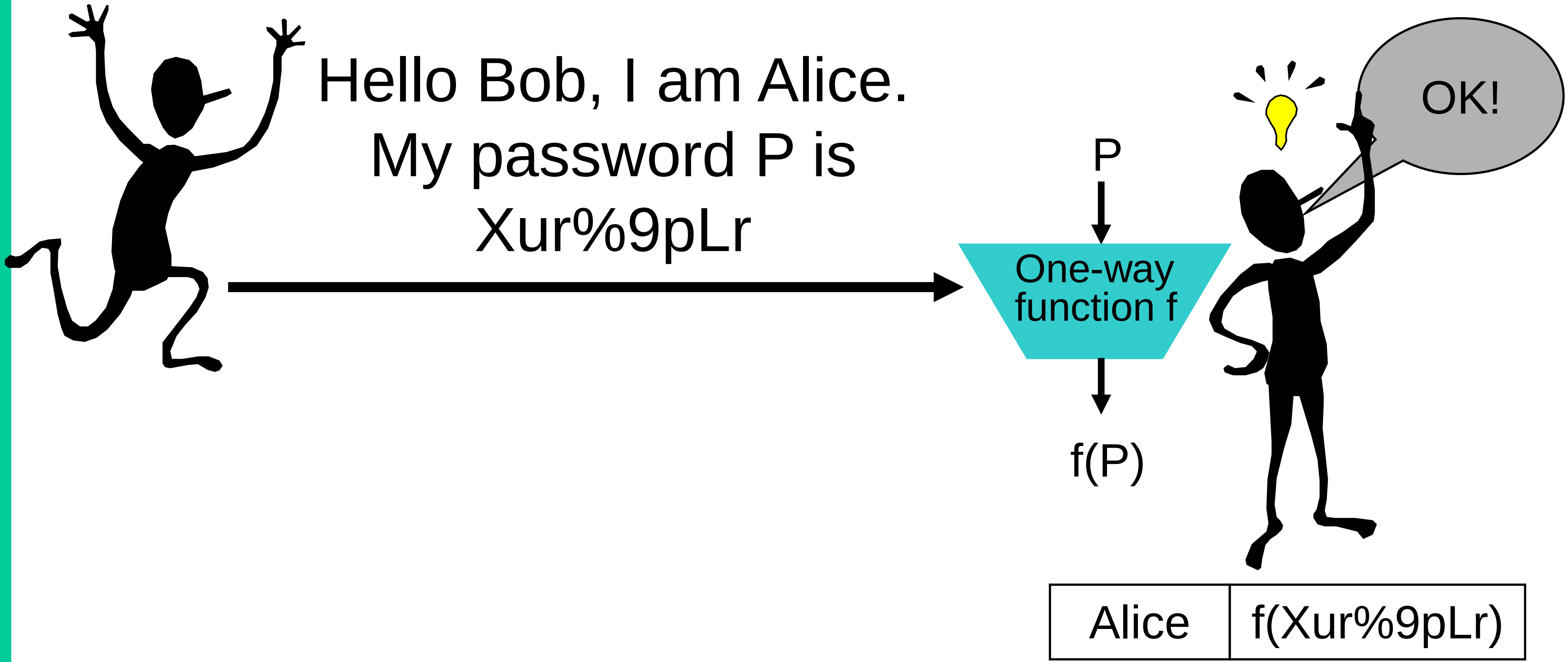


Alice	Xur%9pLr

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

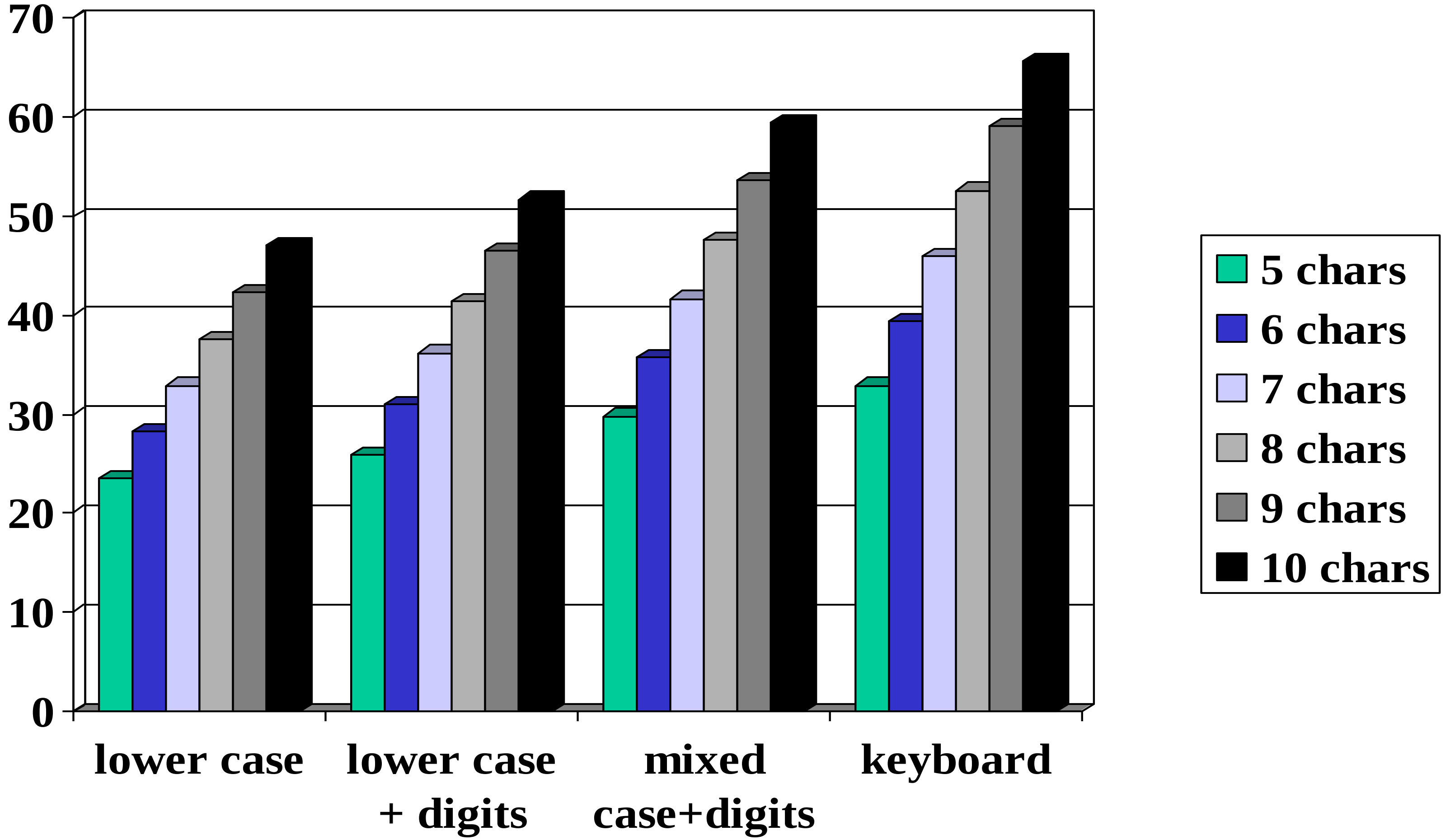
Improved identification with passwords



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

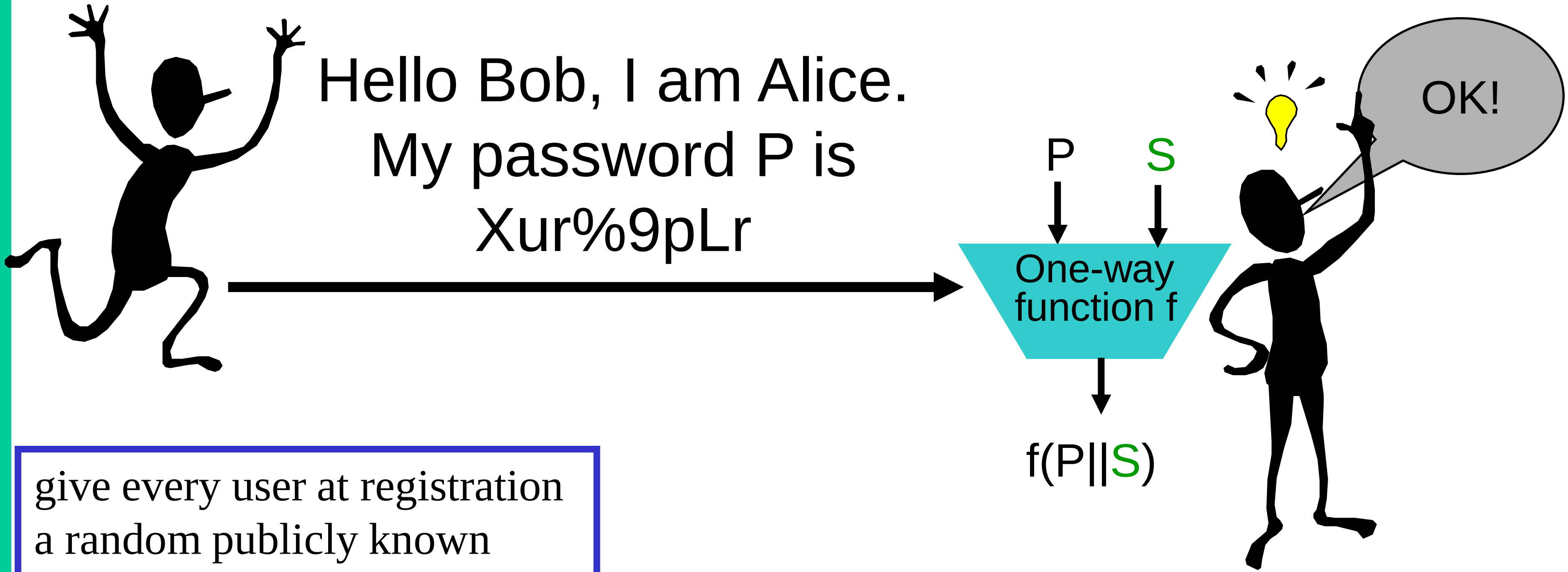
- it is difficult to deduce P from f(P)

Password entropy: effective key length



Problem: passwords from dictionaries

Improved+ identification with passwords



give every user at registration a random publicly known value S (salt)

Alice	f(Xur%9pLr||987&*)|| 987&*)

Bob stores f(P,S) || S rather than Alice’s secret P

- it is harder to attack the passwords of all users simultaneously

Example: UNIX

- Function f() = DES applied 25 times to the all zero plaintext $DES_K(DES_K(\ldots DES_K(000..0)))$ with as key the password (8 7-bit characters)
- Salt: 12-bit modification to DES
- etc/passwd public
- PC: 1 million passwords/second
- But time-memory tradeoff…

Problem: human memory is limited

- Solution: store key K on magstripe, USB key, hard disk
- Stops guessing attacks

But this does not solve the other problems related to passwords

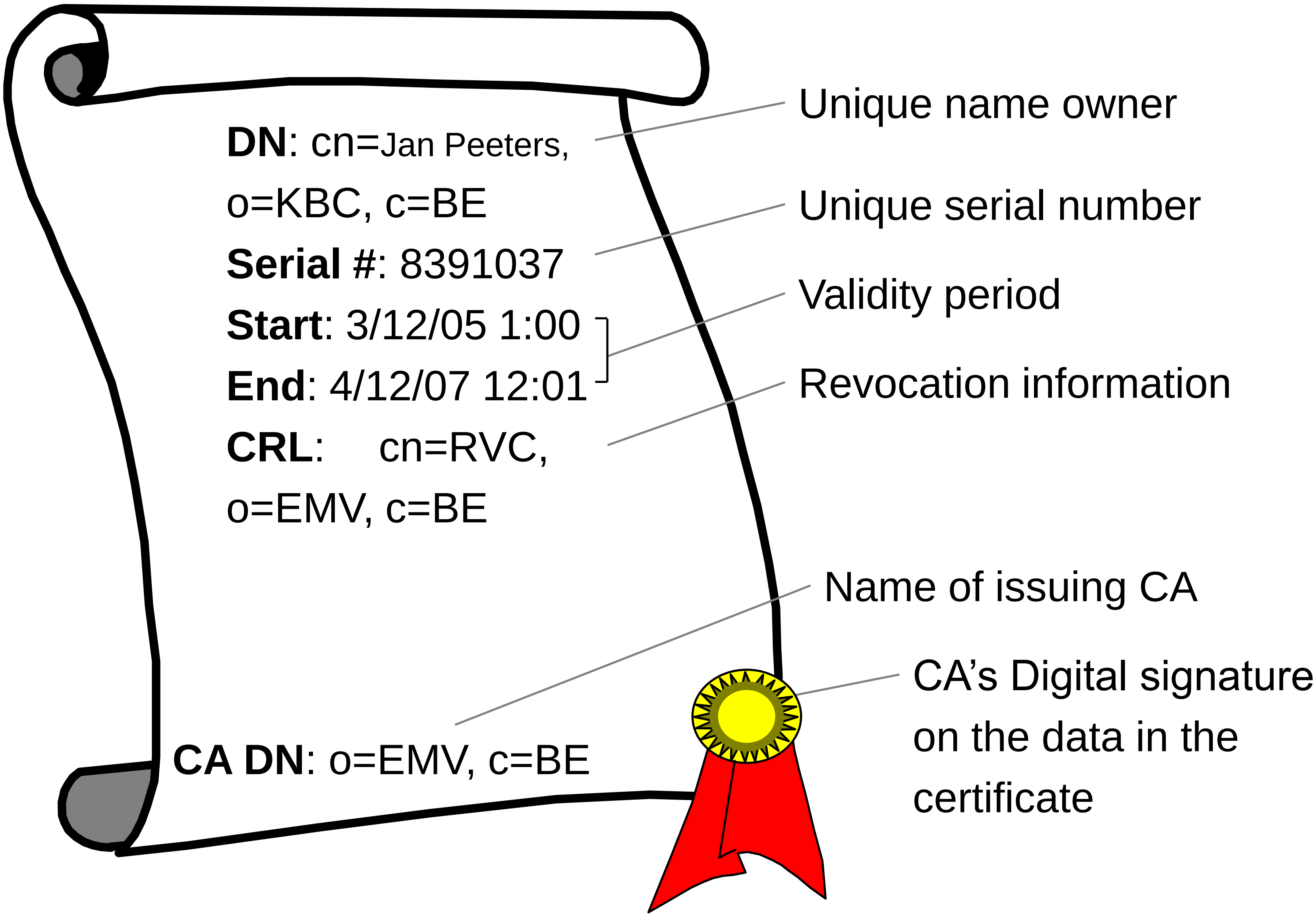
And now you identify the card, not the user….

Improvement: Static Data Authentication

- Replace K by a signature of a third party CA (Certification Authority) on Alice's name: SigSK_{CA} (Alice) = special certificate

- Advantage: can be verified using a public string PK_{CA}
- Advantage: can only be generated by CA
- Disadvantage: signature = 40..128 bytes
- Disadvantage: can still be copied/intercepted

“Certificate” for static data authentication



Entity authentication with symmetric token

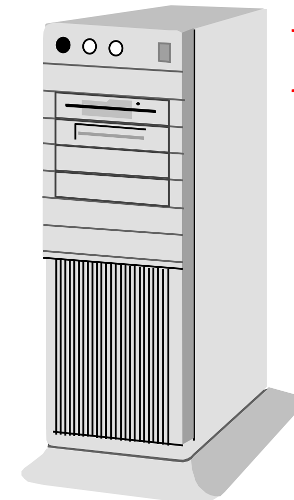
Challenge response protocol

K

random number r

or

MAC$_K$(r)

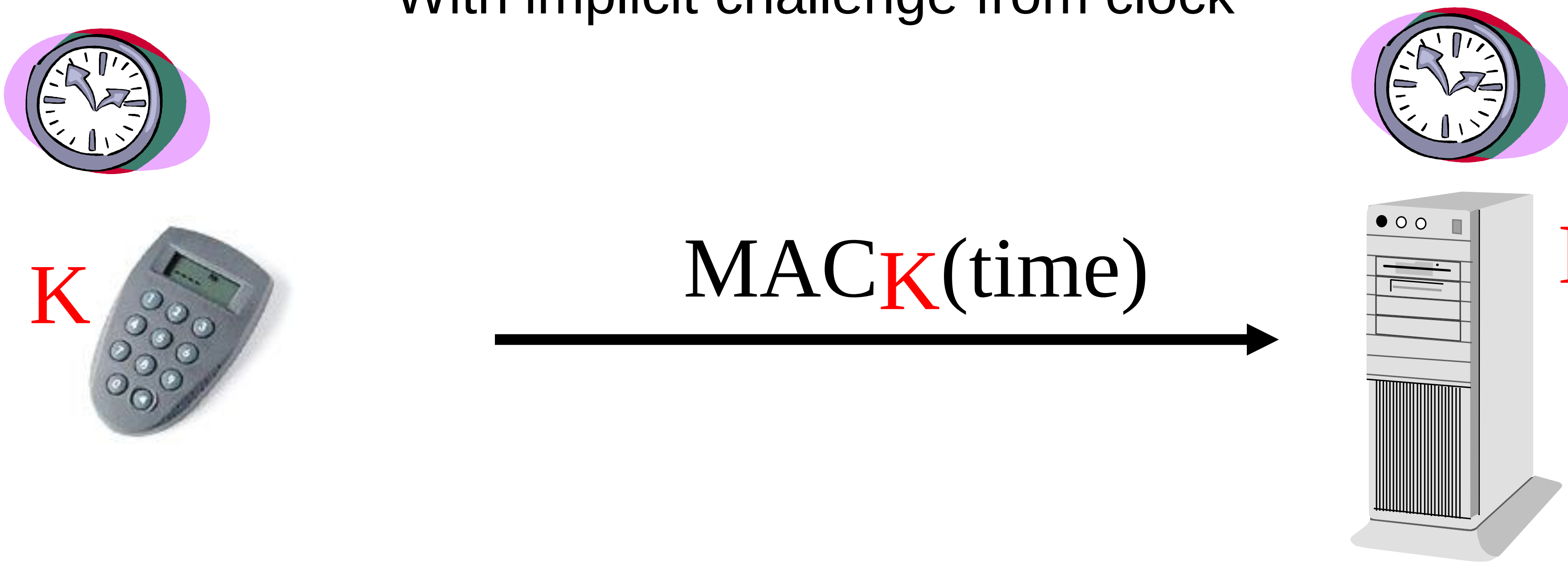


K

- Eavesdropping no longer effective
- Bob still needs secret key K

Entity authentication with symmetric token

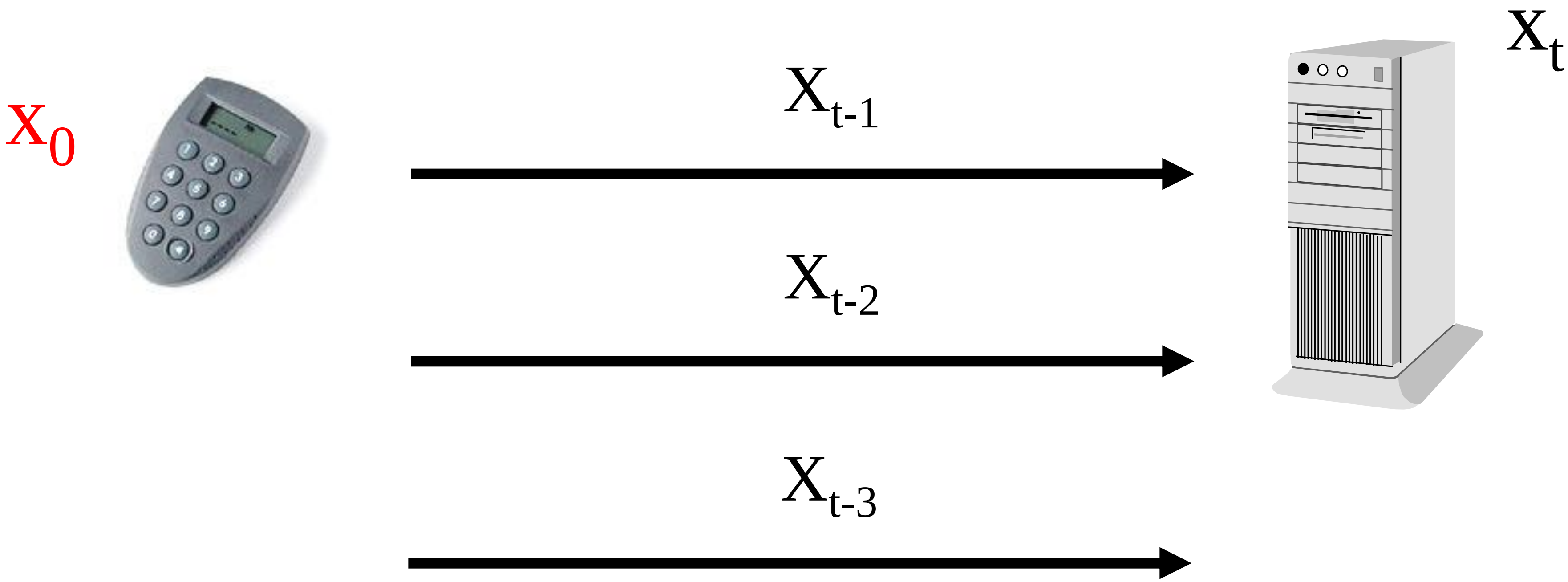
With implicit challenge from clock

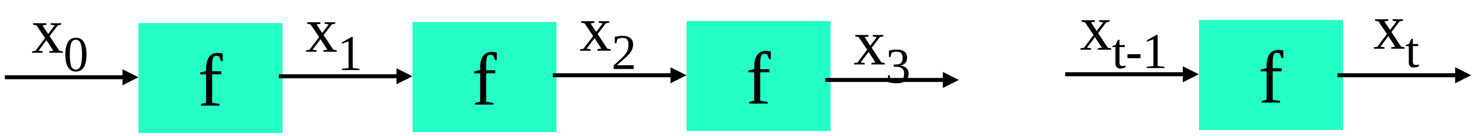


- Eavesdropping no longer effective
- Bob still needs secret key K
- resynchronization mechanism needed

Lamport's one-time passwords

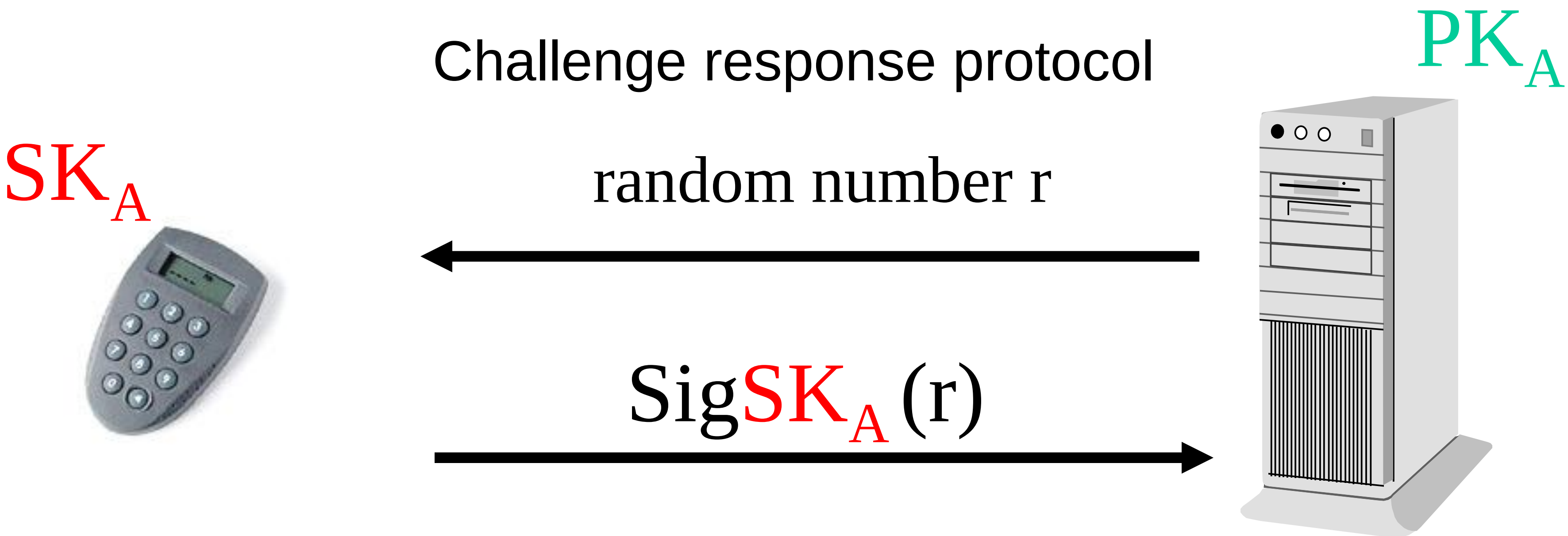
iterated one-way function





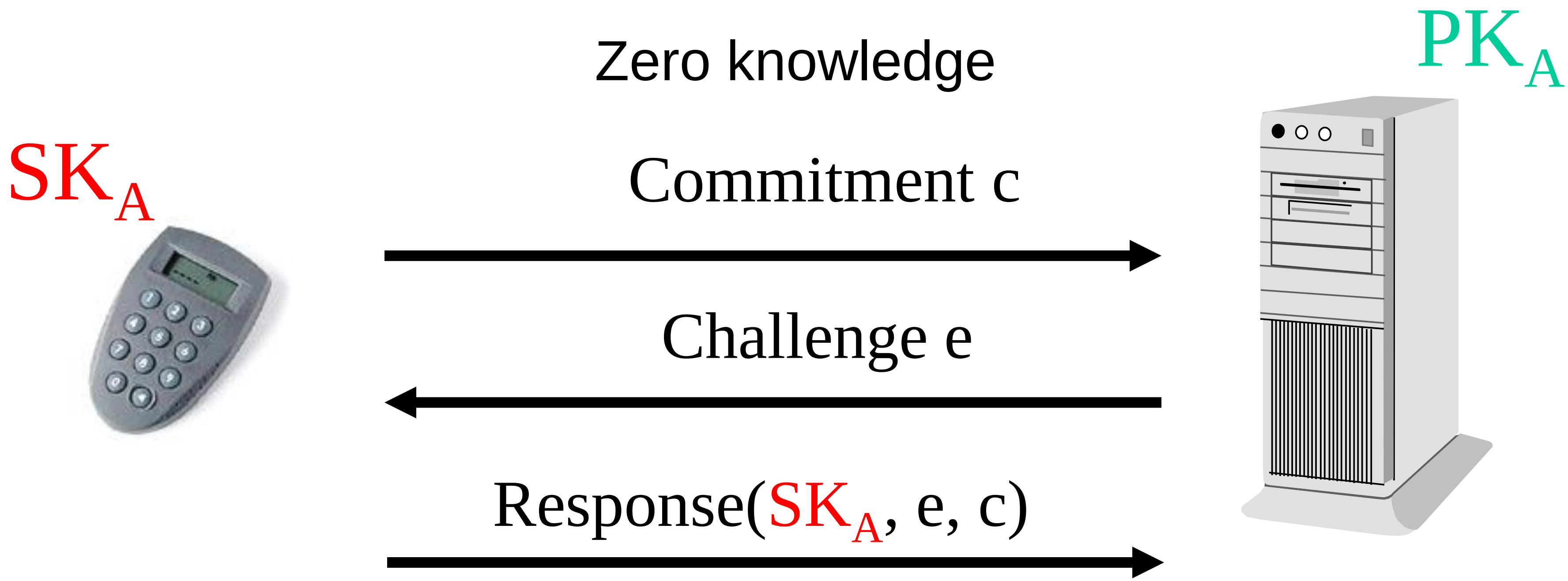
- Disadvantage: only works with one Bob

Entity authentication with public key token



- Eavesdropping no longer effective
- Bob no longer needs a secret – only PK_A

Entity authentication with ZK



- Mathematical proof that Bob only learns that he is talking to Alice (1 bit of information)
- Bob cannot use this information to convince a third party that he is/was talking to Alice

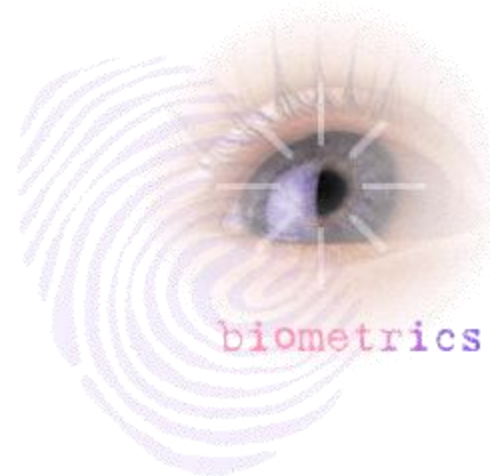
Mutual authentication

- Many applications need entity authentication in two directions
- !! This is not complete the same as 2 parallel unilateral protocols for entity authentication

2 stage authentication

- Local: user to device
- Device to rest of the world

Biometry

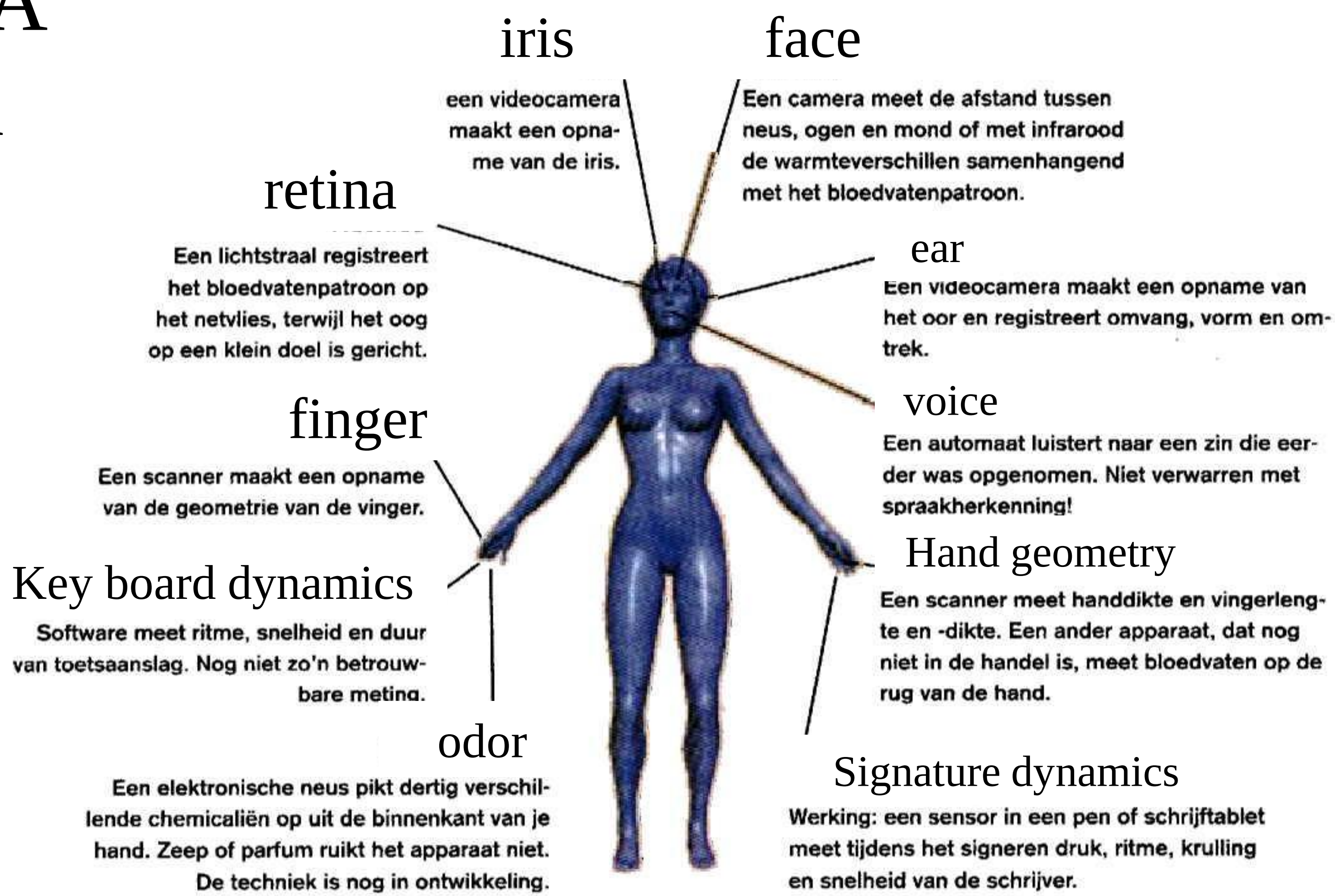


- Based on our unique features

- Identification or verification
  - Is this Alice?
  - Check against watchlist
  - Has this person ever registered in the system?

Some unique features

DNA
skin
…

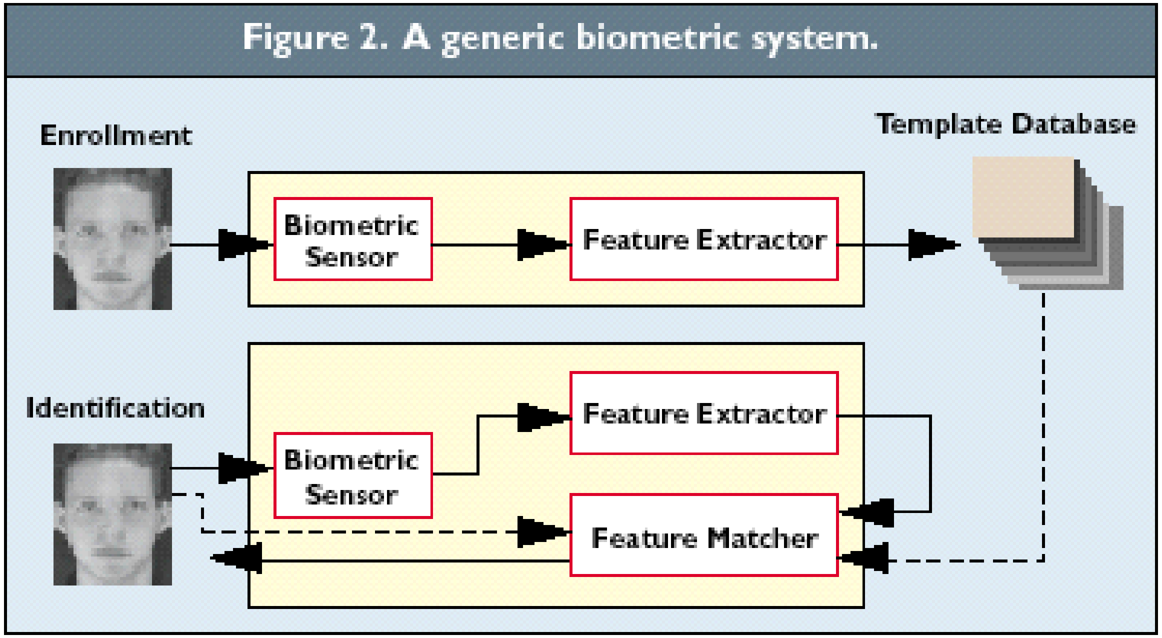


Biometric procedures

- Registration
- Template extraction

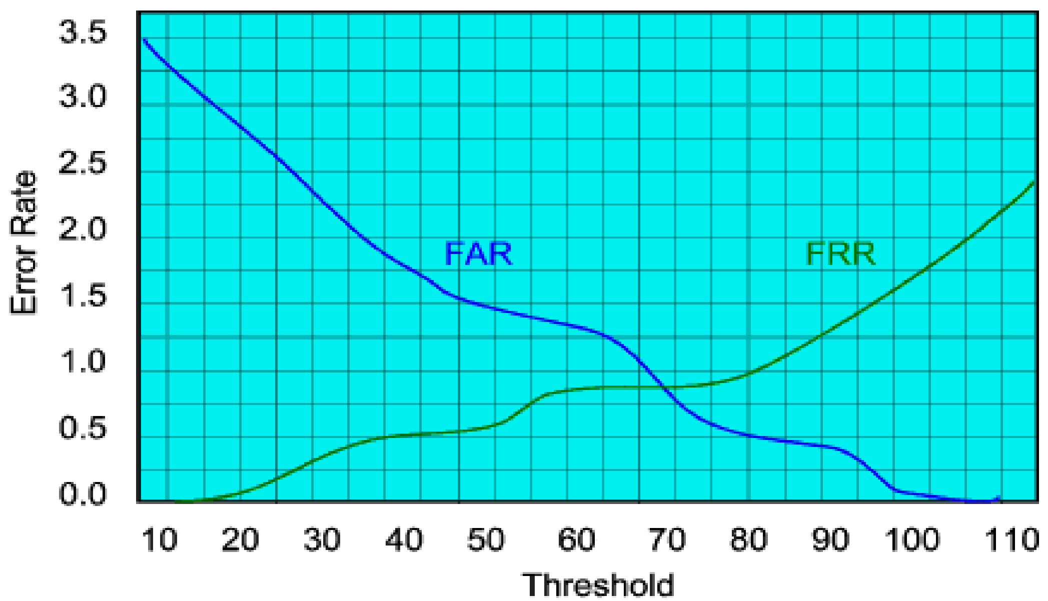
- Measurement
- Processing
- Template matching

- Link with applications

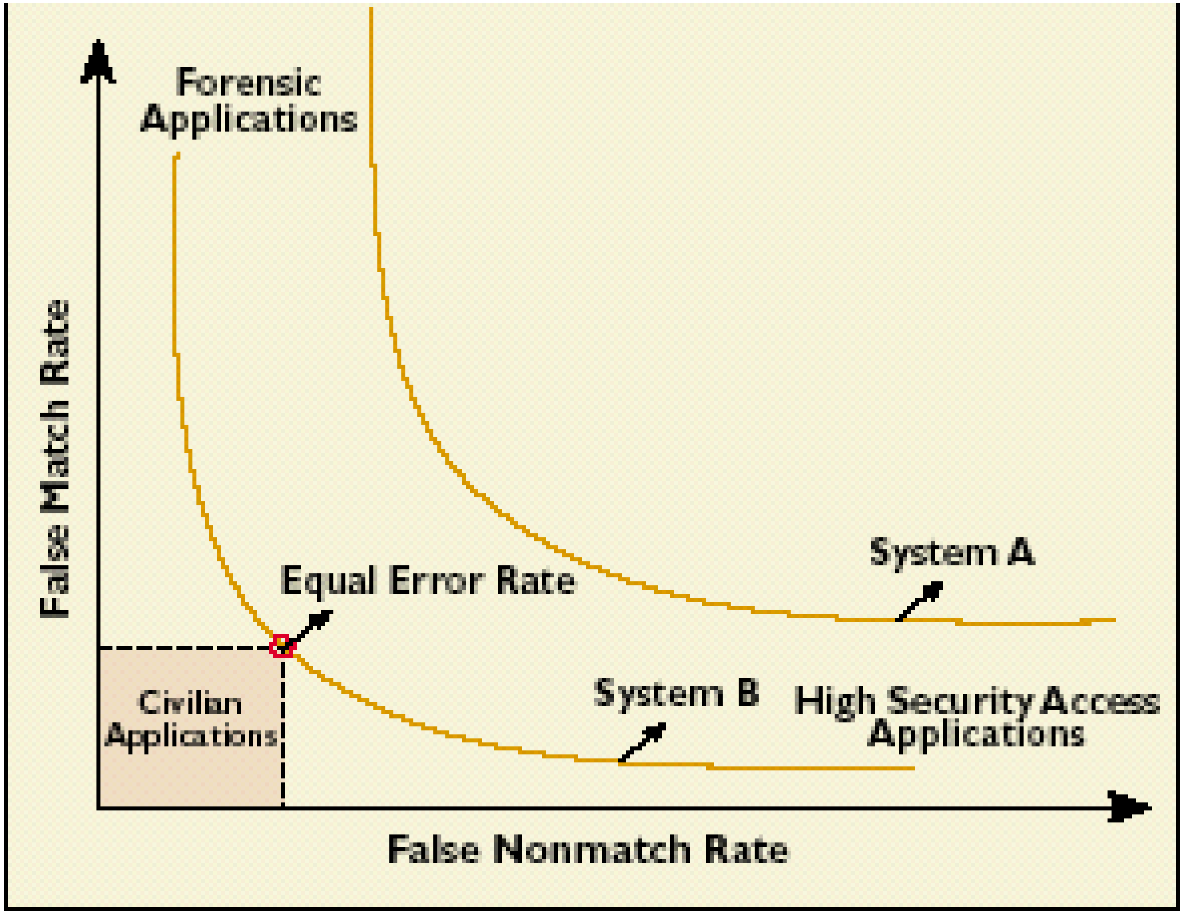


Robustness/performance

- Performance evaluation
  - False Acceptance Ratio or False Match Rate
  - False Rejection Ratio or False Non-Match Rate
- Application dependent

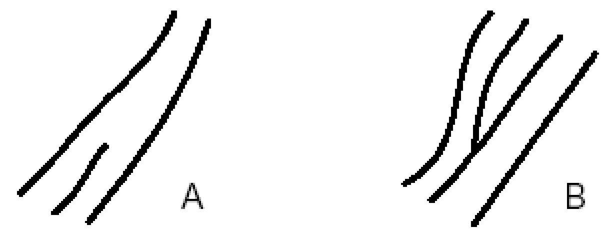


Robustness/performance (2)

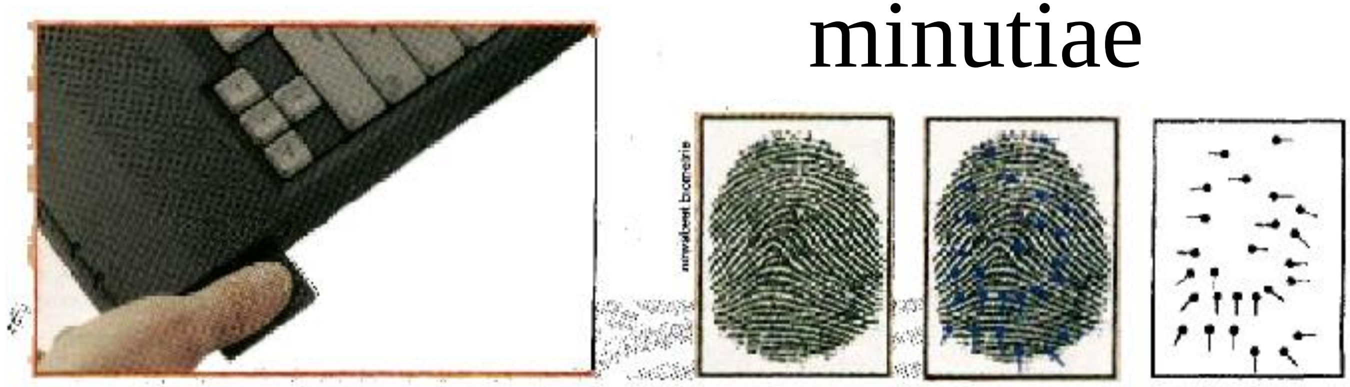


Fingerprint

- Used for PC/laptop access
- Widely available
- Reliable and inexpensive
- Simple interface



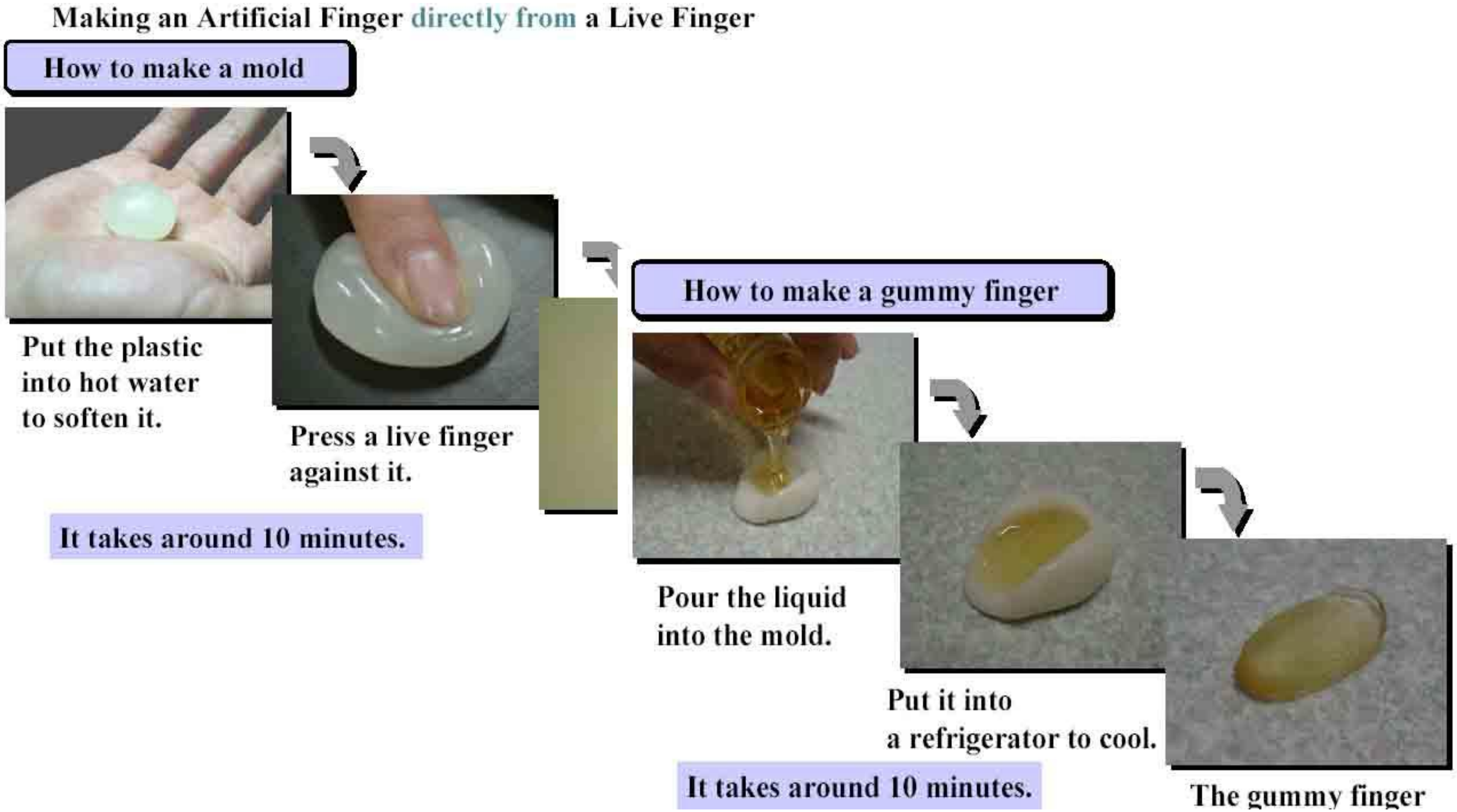
minutiae



Fingerprint (2)

- Small sensor
- Small template (100 bytes)
- Commercially available
  - Optical/thermical/capacitive
  - Liveness detection
- Problems for some ethnic groups and some professions
- Connotation with crime

Fingerprint (3): gummy fingers



Hand geometry

- Flexible performance tuning
- Mostly 3D geometry
- Example: 1996 Olympics

Voice recognition

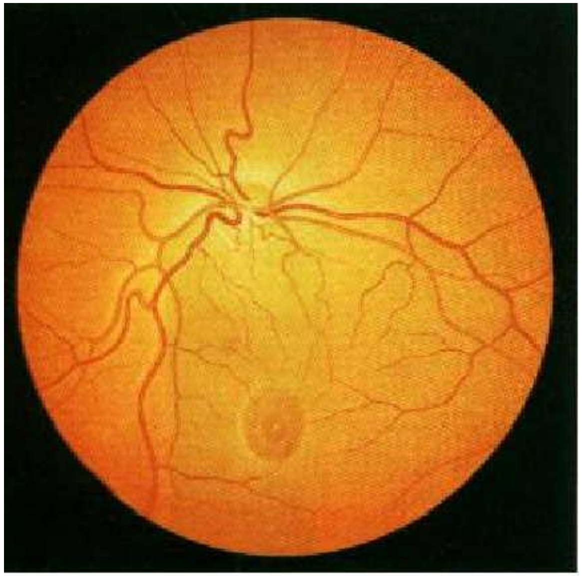
- Speech processing technology well developed
- Can be used at a distance
- Can use microphone of our gsm
- But tools to spoof exist as well
- Typical applications: complement PIN for mobile or domotica

Iris Scan

- No contact and fast
- Conventional CCD camera
- 200 parameters
- Template: 512 bytes
- All etnic groups
- Reveals health status

Retina scan

- Stable and unique pattern of blood vessels
- Invasive
- High security



Manual signature

- Measure distance, speed, accelerations, pressure
- Familiar
- Easy to use
- Template needs continuous update
- Technology not fully mature

Facial recognition

- User friendly
- No cooperation needed
- Reliability limited
- Robustness issues
  - Lighting conditions
  - Glasses/hair/beard/...

Comparison

Feature	Uniqueness	Permanent	Performance	Acceptability	Spoofing
Facial	Low	Average	Low	High	Low
Fingerprint	High	High	High	Average	High
Hand geometry	Average	Average	Average	Average	Average
Iris	High	High	High	Low	High
Retina	High	Average	High	Low	High
Signature	Low	Low	Low	High	Low
Voice	Low	Low	Low	High	Low

Biometry: pros and cons

- Real person
- User friendly
- Cannot be forwarded
- Little effort for user

- Secure implementation: derive key in a secure way from the biometric

- Privacy (medical)
- Intrusive?
- Cannot be replaced
- Risk for physical attacks
- Hygiene
- Does not work everyone, e.g., people with disabilities
- Reliability

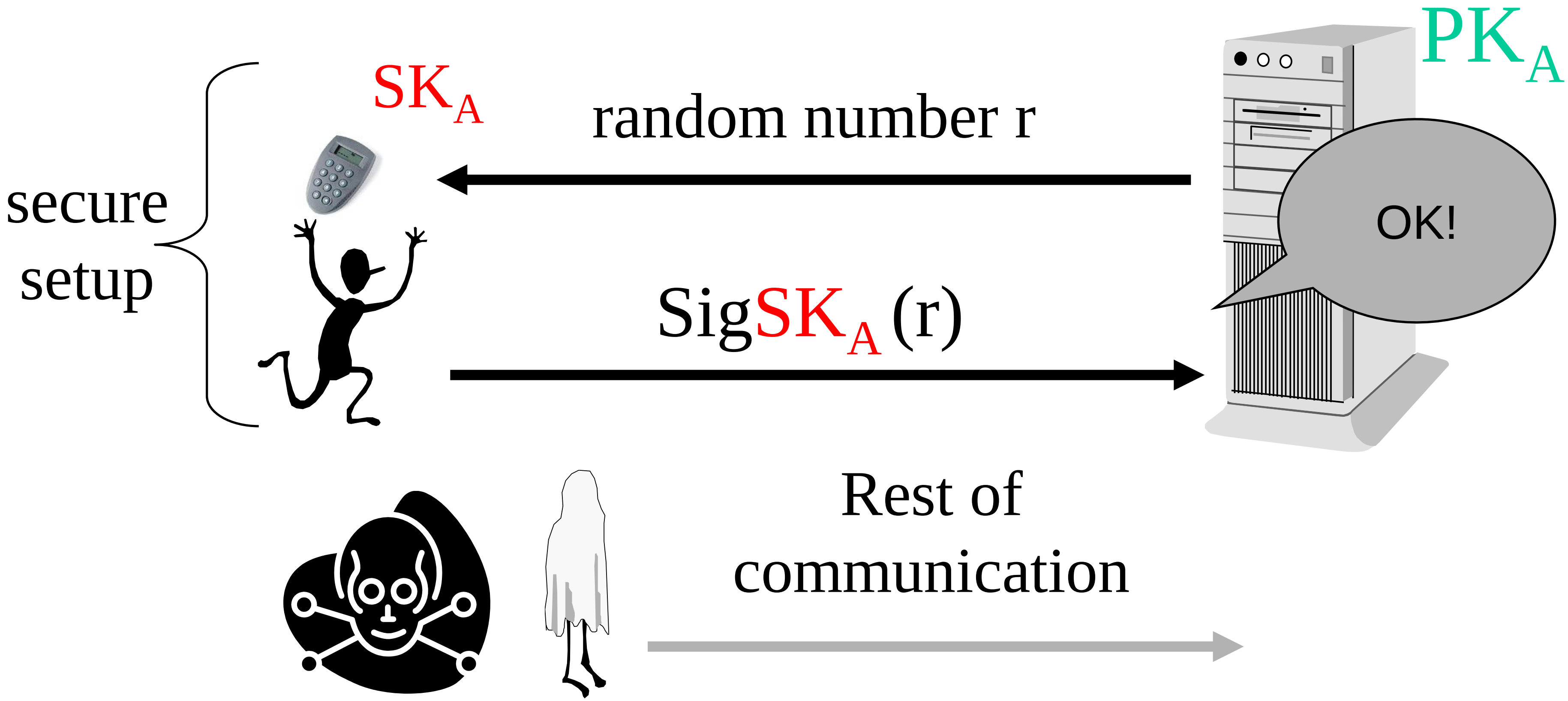
- No cryptographic key

Location-based authentication

- Dial-back: can be defeated using fake dial tone
- IP addresses and MAC addresses can be spoofed
- Mobile/wireless communications: operator knows access point, but how to convince others?
- Trusted GPS?

Limitations of entity authentication

- Establish who someone is
- Establish that this person is active
- But what about keeping authenticity alive?



Solution

- Authenticated key agreement

- Run a mutual entity authentication protocol
- Establish a key
- Encrypt and authenticate all information exchanged using this key

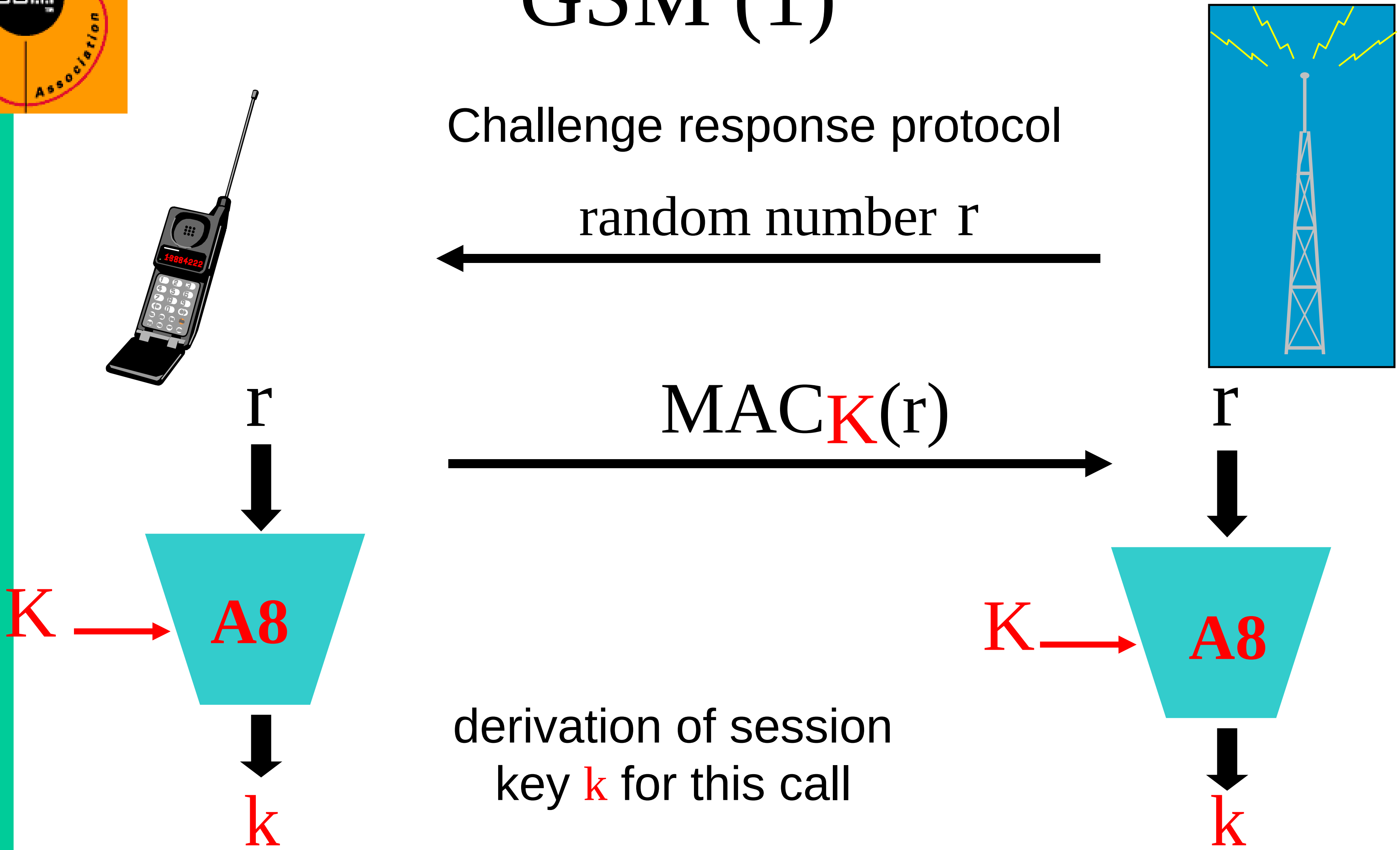
Key establishment

- 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

GSM (1)



encrypt all data with k

GSM (2)

- SIM card with long term secret key K (128 bits)
- secret algorithms
  - A3: MAC algorithm
  - A8: key derivation algorithm
  - A5.1/A5.2: encryption algorithm
- anonimity: IMSI (International Mobile Subscriber Identity) replaced by TIMSI (temporary IMSI)
  - the next TIMSI is sent (encrypted) during the call set-up

Point-to point symmetric key distribution

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

generate session key k

$E K_{AB}(k \,||\, time \,||\, Bob)$ →

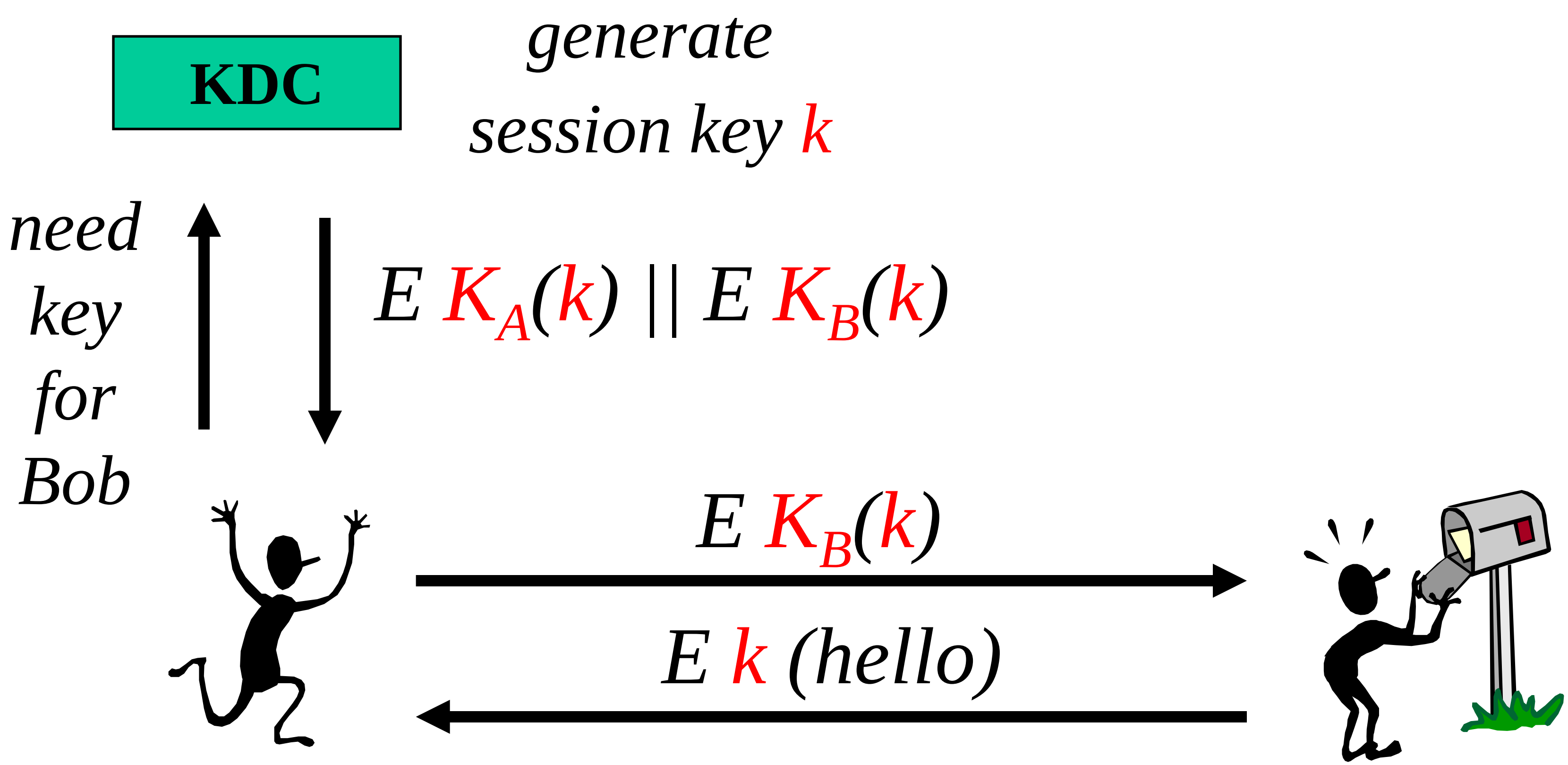
← $E k\,(time \,||\, Alice \,||\, hello)$

decrypt extract k

- 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
- Alice and Bob have also authenticated each other

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



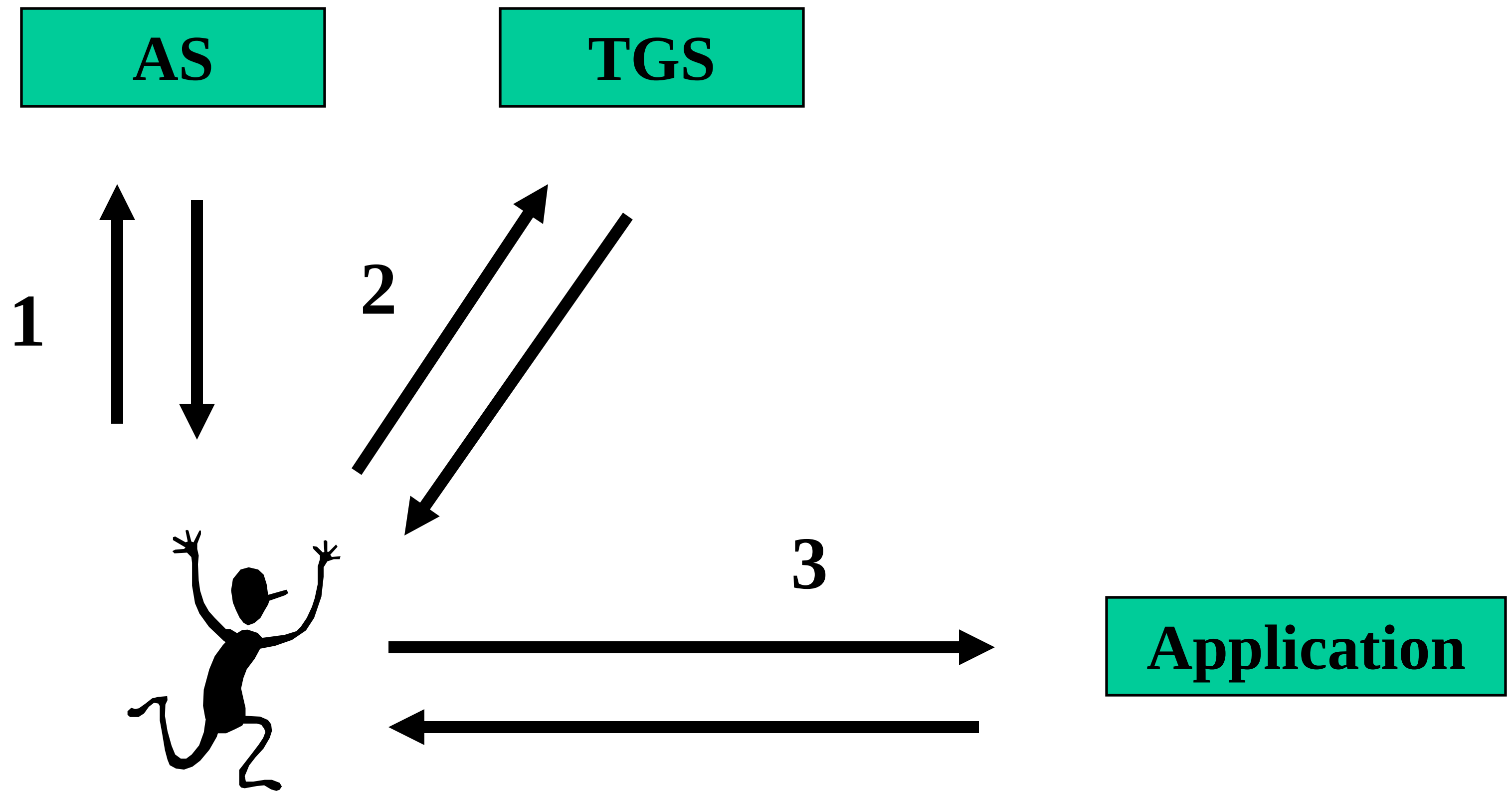
Symmetric key distribution with 3rd party(2)

- 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

Alice		Bob
generate x	α^x →	*generate* y
compute α^x	← α^y	*compute* α^y
compute $k=(\alpha^y)^x$		*compute* $k=(\alpha^x)^y$

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

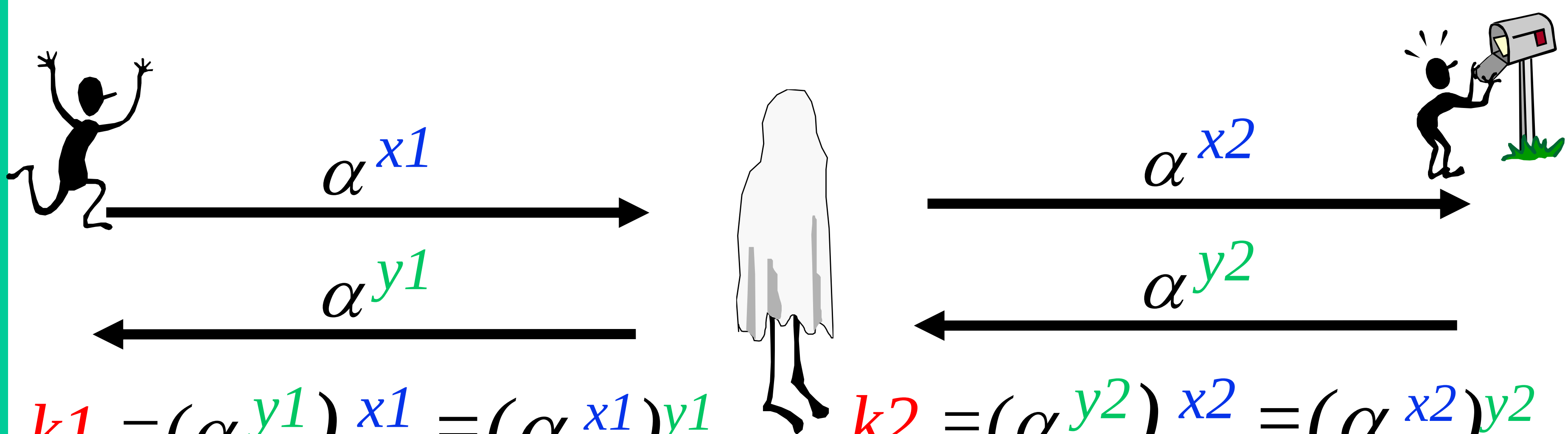
Diffie-Hellman (continued)

Alice		Bob
generate x	α^x →	*generate* y
compute α^x	← α^y	*compute* α^y
compute $k=(\alpha^y)^x$		*compute* $k=(\alpha^x)^y$

- 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.

Meet-in-the middle attack

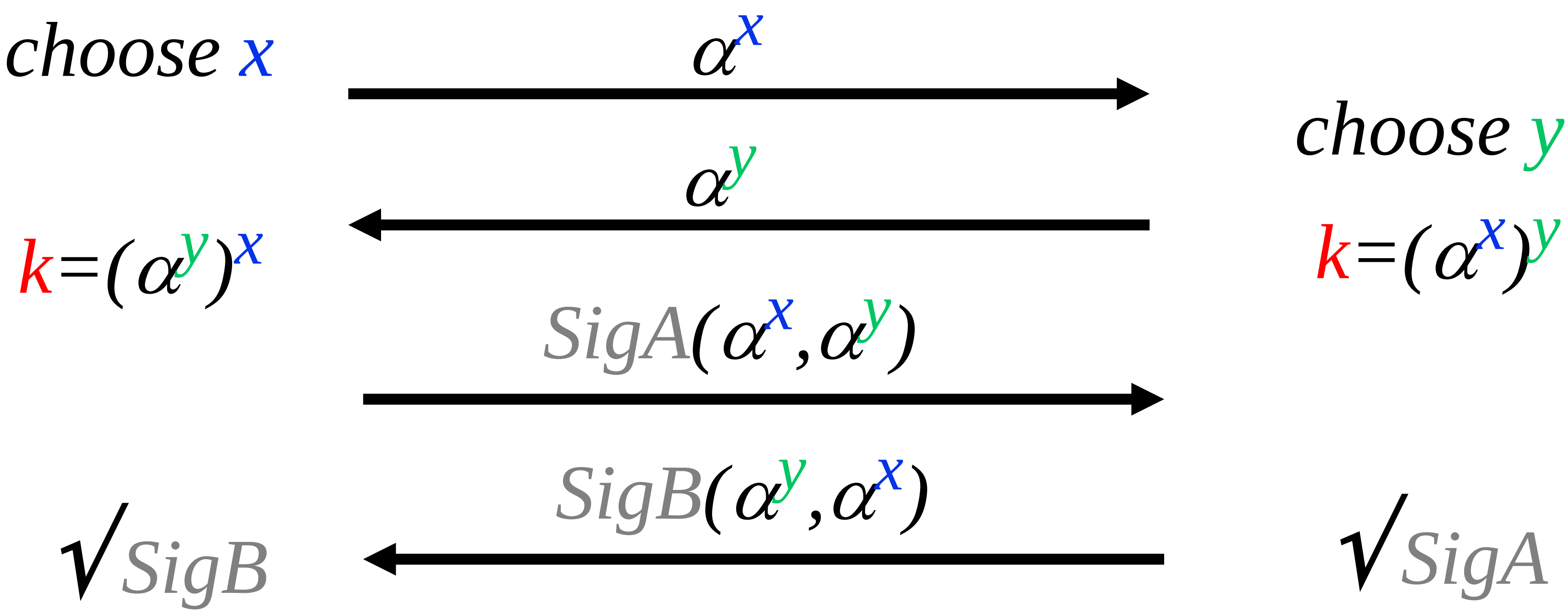
- Eve shares a key *k1* with Alice and a key *k2* with Bob
- Requires *active* attack



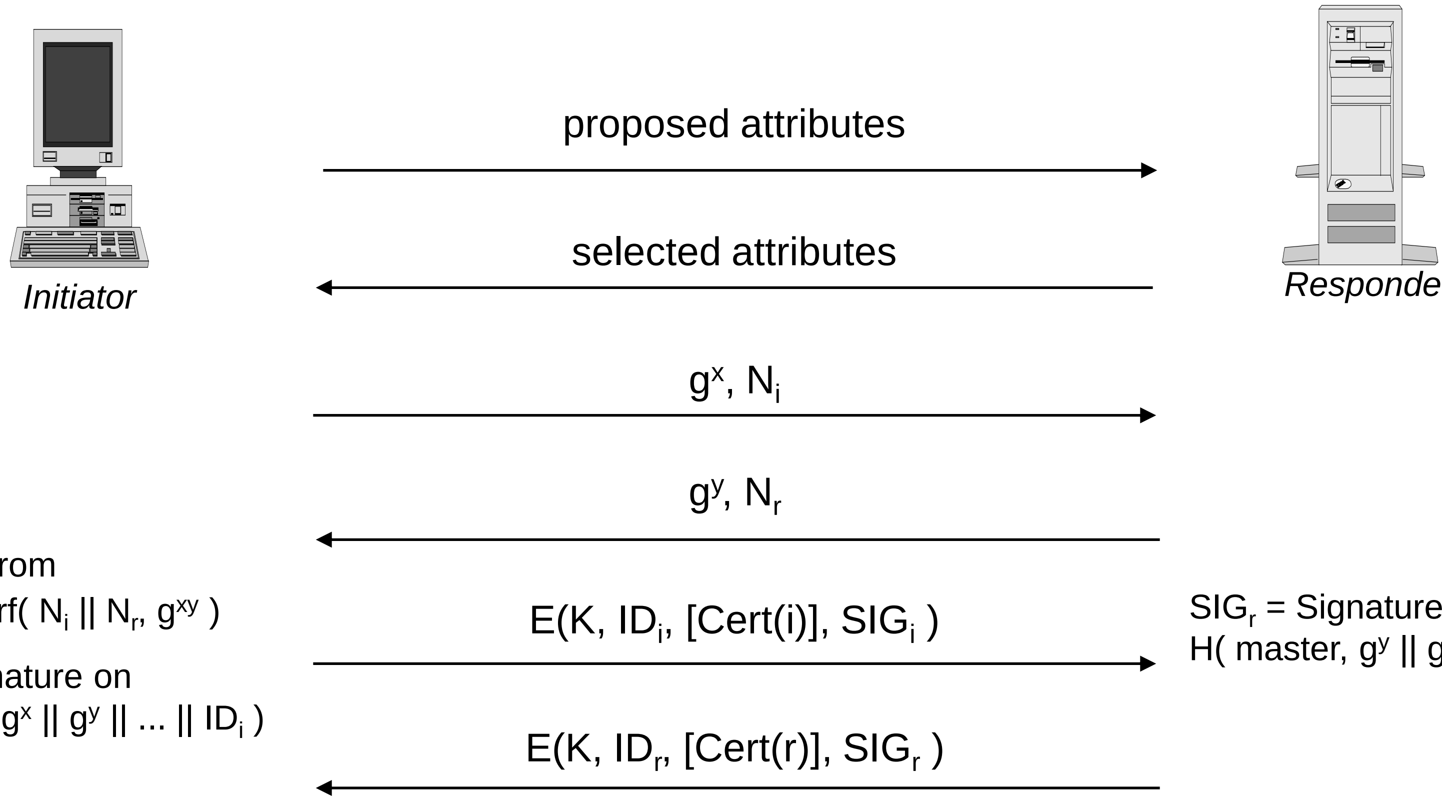
$$k1 = (\alpha^{y1})^{x1} = (\alpha^{x1})^{y1} \qquad k2 = (\alpha^{y2})^{x2} = (\alpha^{x2})^{y2}$$

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)



IKE - Main Mode with Digital Signatures



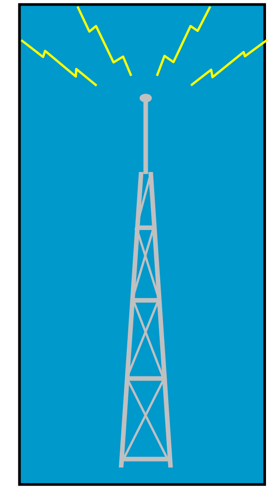
K derived from
master = prf(N_i || N_r, g^{xy})

SIG_i = Signature on
H(master, g^x || g^y || ... || ID_i)

SIG_r = Signature on
H(master, g^y || g^x || ... || ID_r)

H is equal to prf or the hash function tied to the signature algorithm
(all inputs are concatenated)

Key establishment in future mobile systems



random number r ←

SigA(r || B), r’ →

√ *SigA*

SigB(r || r’ || A || B) ←

√ *SigB*

[+] slightly more efficient (ECC)

Key transport using RSA

generate k

$E_{PKB}(k)$

$E_{PKB}(k)$ →

decrypt using SKB to obtain k

- How does Bob know that ***k*** is a fresh key?
- How does Bob know that this key ***k*** is coming from Alice?
- How does Alice know that Bob has received the key ***k*** and that Bob is present (entity authentication)?

Key transport using RSA (2)

generate k

$E_{PKB}(k)$

$E_{PKB}(k \parallel t_A)$ →

decrypt using SKB to obtain k

- Freshness is solved with a timestamp t_A

Key transport using RSA (3)

generate k

$Sig_{SKA}(E_{PKB}(k \parallel t_A))$ →

decrypt using SKB and verify using PKA

- Alice authenticates by signing the message
- There are still attacks (signature stripping…)

Key transport using RSA (4): X.509

generate k

$Sig_{SKA}(B \| t_A \| E_{PKB}(A \| k))$
$\| t_A \| E_{PKB}(A \| k)$

→

decrypt using SKB and verify using PKA

Mutual: B can return a similar message including part of the first message

Problem (compared to D-H/STS): lack of **forward secrecy**

If the long term key *SKB* of Bob leaks, all past session keys can be recovered!

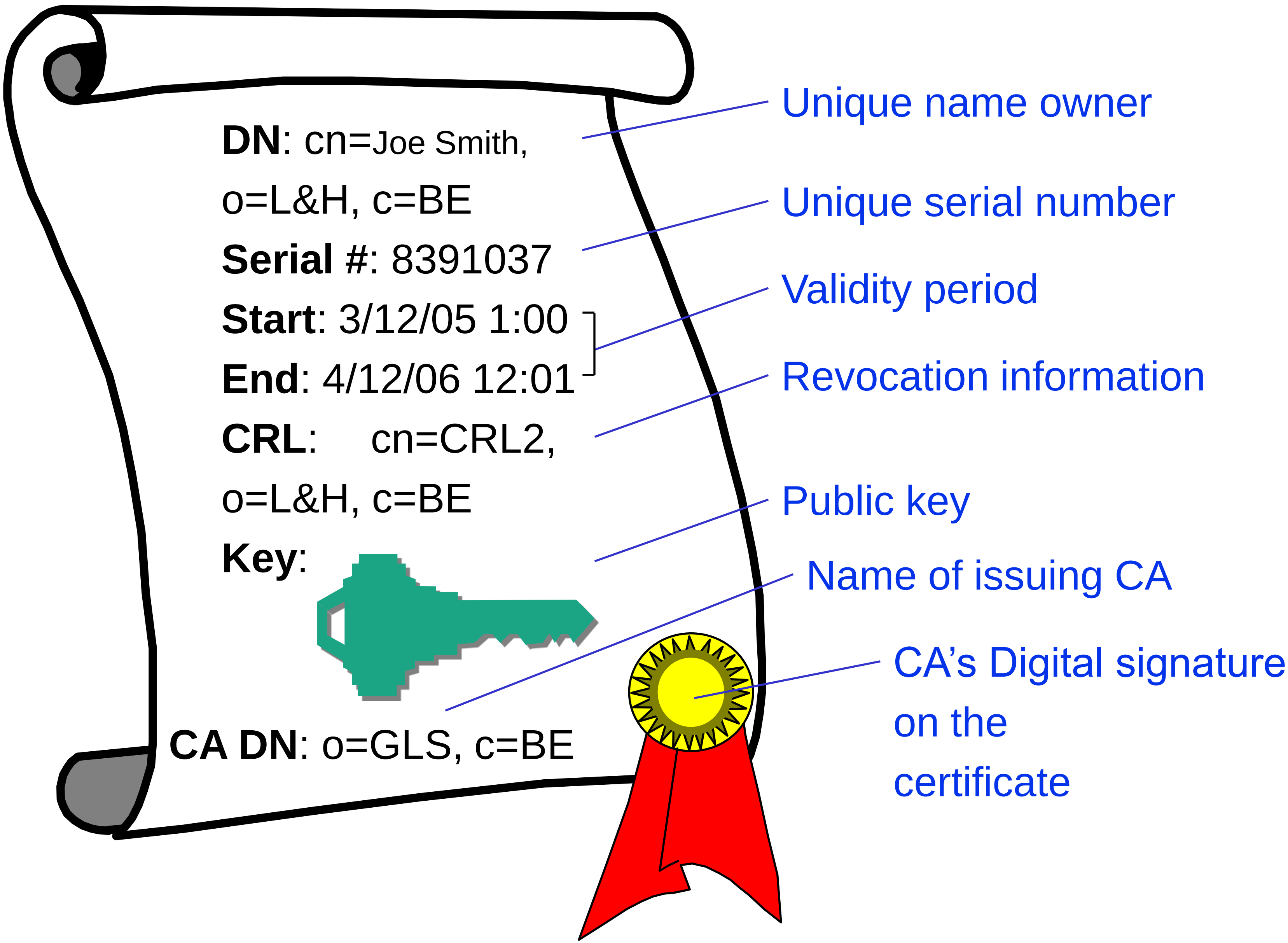
Distribution of public keys

- 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?

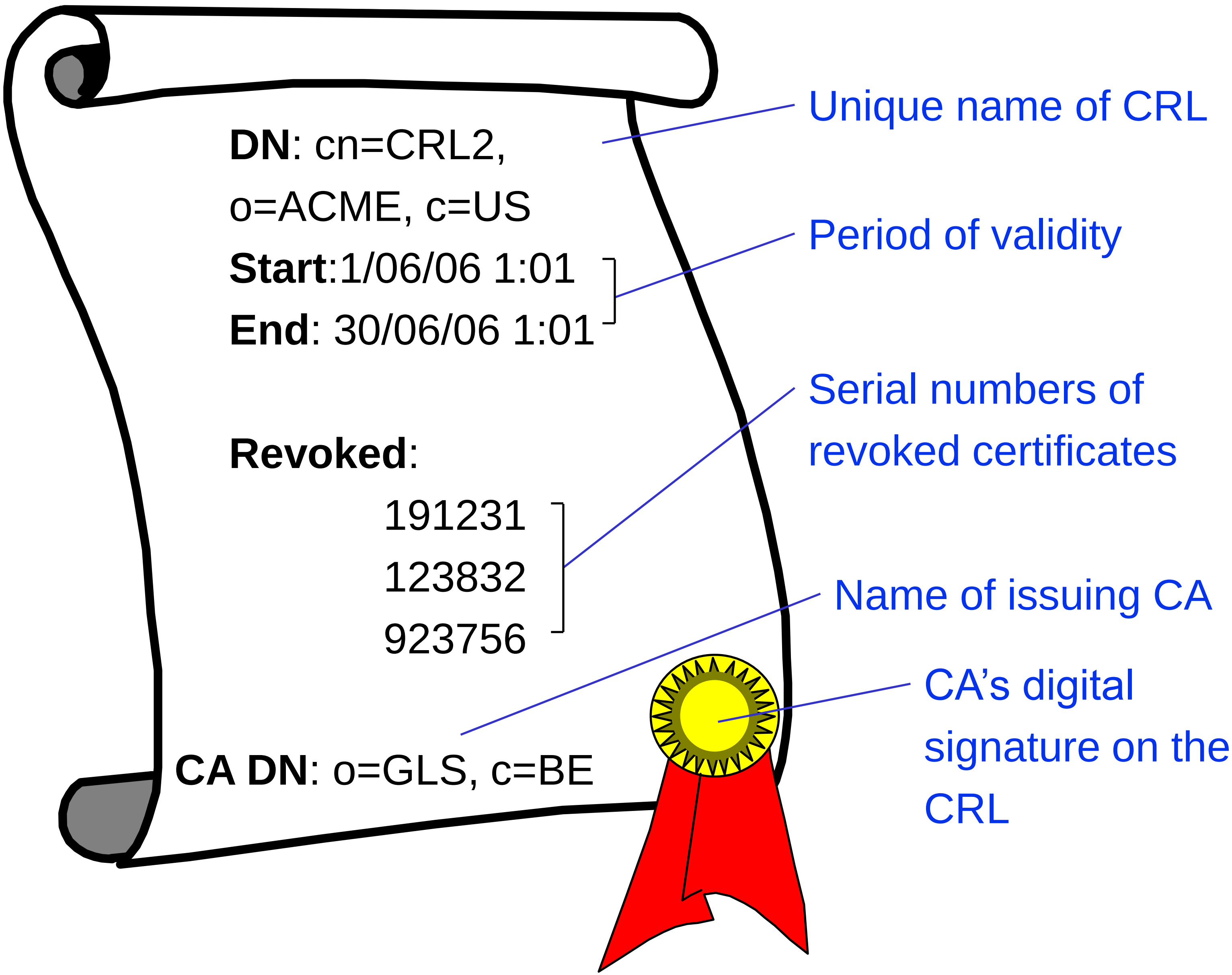
reduce protection of public key of many users to knowledge of a single public key of a Certification Authority (CA)

digital certificates &
Public Key Infrastructure (PKI)

Public Key Certificates



Certificate Revocation List

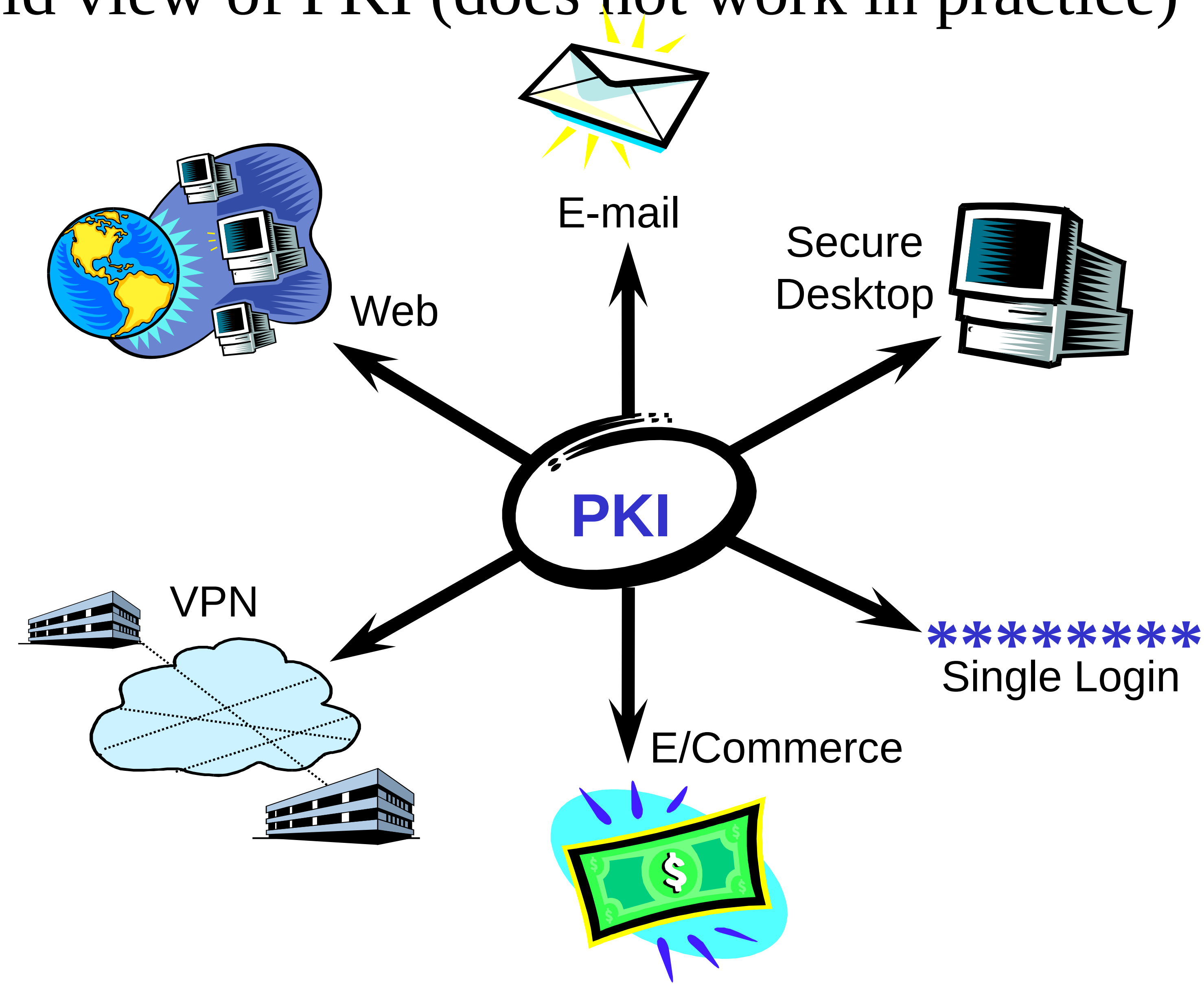


Essential PKI Components

- 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: old view of PKI (does not work in practice)



Example of a key hierarchy