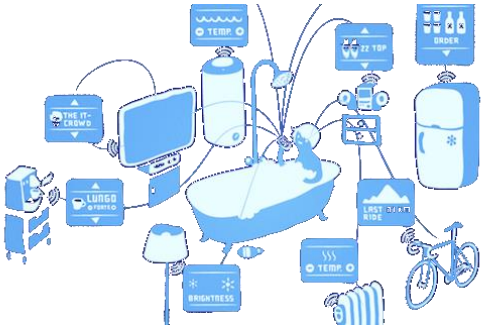
SAPIENZA
UNIVERSITÀ DI ROMA

Green Wireless Sensor Network Security

Angelo Capossele

WSN towards Internet of Things



"Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts" (EPoSS 2008)

"A world-wide network of interconnected objects uniquely addressable, based on standard communication protocols." (EPoSS 2008)

"*IoT can be understood as an enabling framework for the interaction between a bundle of heterogeneous objects and also as a convergence of technologies.*" (Farideh Ganji, Ernesto Morales Kluge and Bernd Scholz-Reiter 2010)

CISCO

50 Billions of objects by 2020

Concern about Privacy

Security on WSN

Secure communication between nodes

Confidentiality — *Integrity*

Authentication

Requirements

Scenario

Military

Healthcare

Industrial control

Different constraints

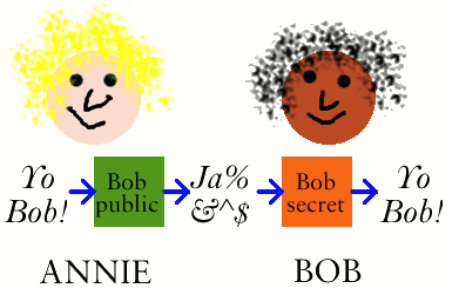
TelosB

Imote2

Mica

First contribution

How to establish a secure channel between nodes?

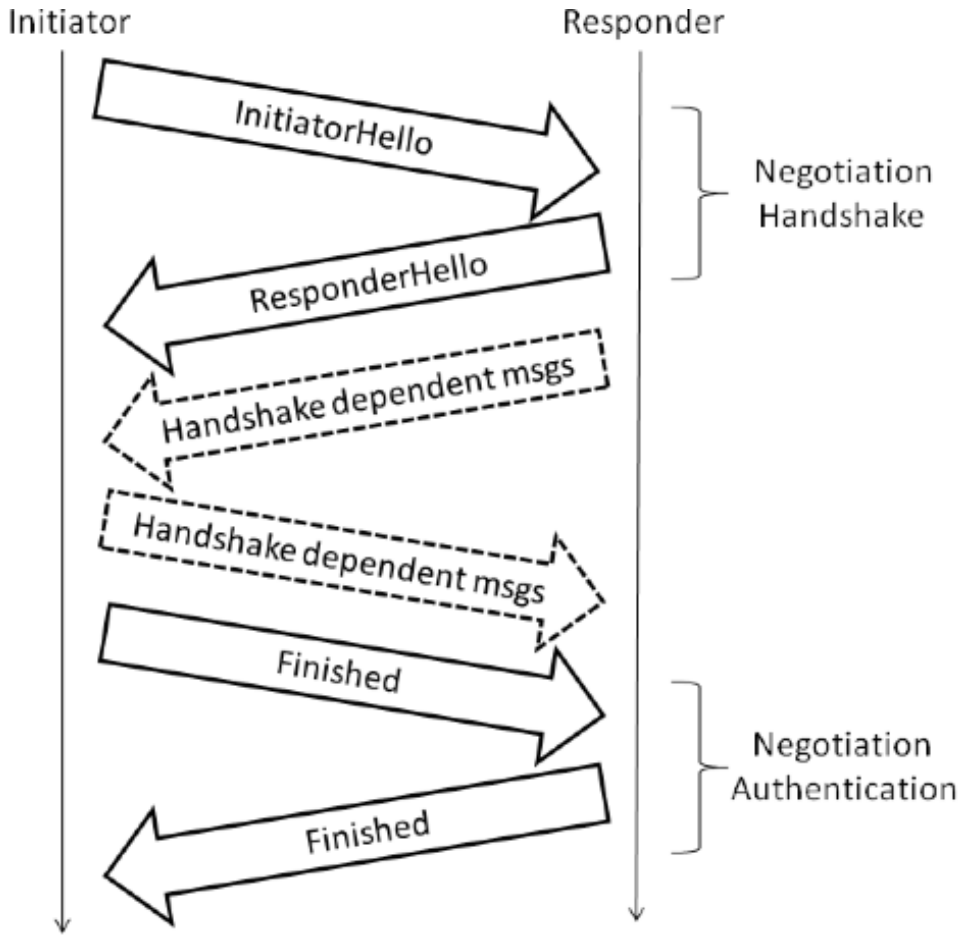


- A new protocol for *key management* and *cipher suite negotiation* based on TLS standard protocol.
- Implementation on two platform (MICA2, TelosB)
- Development of cryptographic library for TinyOS 2.x

It supports different key management mechanisms: RSA, ECC, IBC

Performance evaluation { *Energy consumption overhead*; *Message overhead*

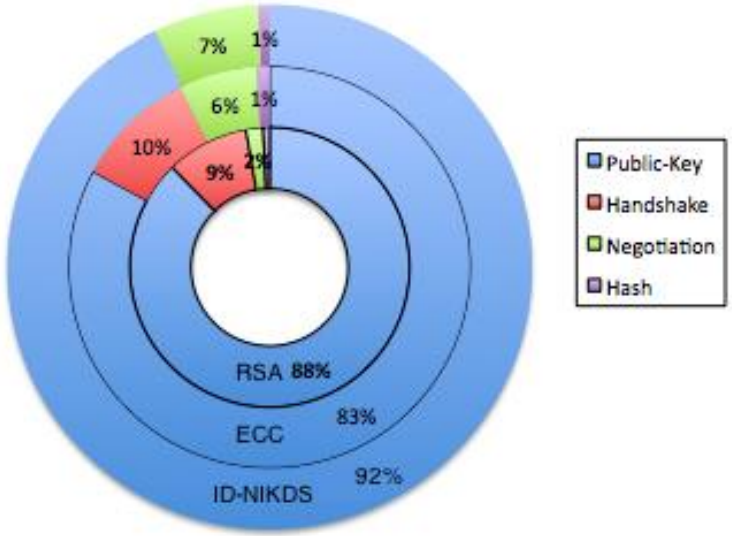
Handshake



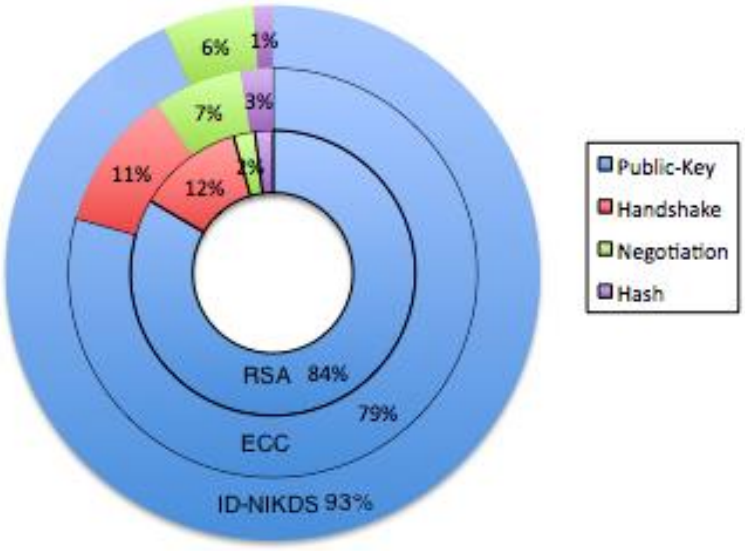
- The InitiatorHello message fits a 28 byte packet
- Handshake message depends on which mechanism is used: RSA - ECC - IBC
- The Finished message contains an HMAC of previously messages

Performance Assessment

Decomposition of energy consumption



MICA2



TelosB

How much do we pay to support the negotiation?

- The cost imposed by public key cryptography is dominant
- With IBC we can minimize the cost of messages exchanged

Our Solution

- It enables the negotiation of key exchange protocol
- It is decoupled from encrypted/authentication message delivery
- It is security association oriented
- It minimizes message transmission overhead and energy consumption

The advantages that the *negotiation* introduces, *flexibility* in primis, are worth the minimal extra cost

Limitations

- Single, known recipient of data

  - Unknown recipient?
  - Many recipients?
  - More may join system later?

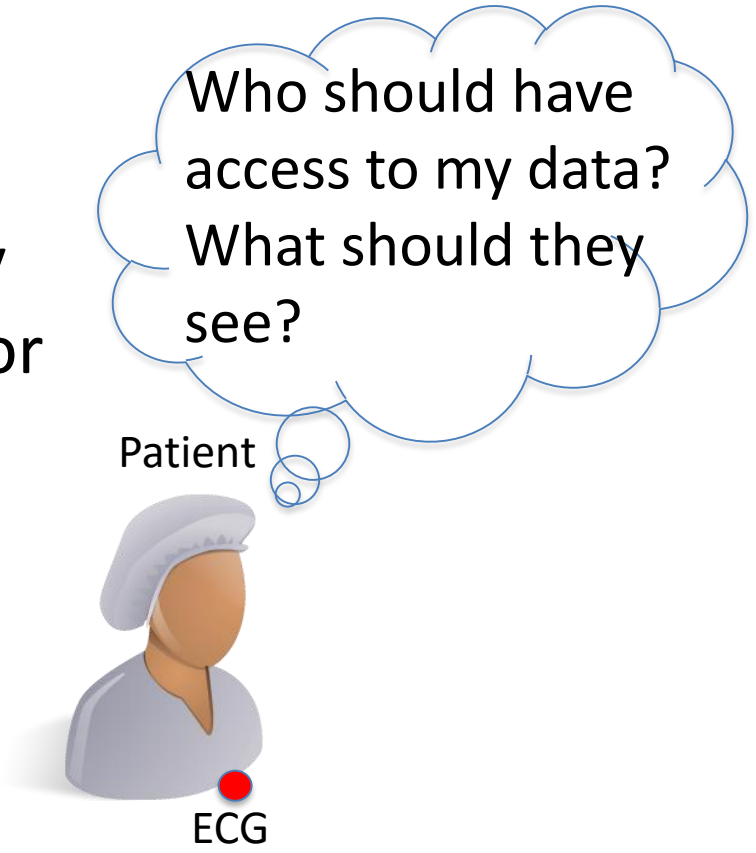
- All-or-nothing data sharing

  - Should each recipient see everything?

Functional encryption

- Data access can be required by doctors, nurses, hospital staff or researchers

- The information is not revealed to all the parties



CiphertextPolicy-ABE

- Ciphertexts: associated with access forumlas


(A OR B) AND C

- Secret Keys: associated with attributes

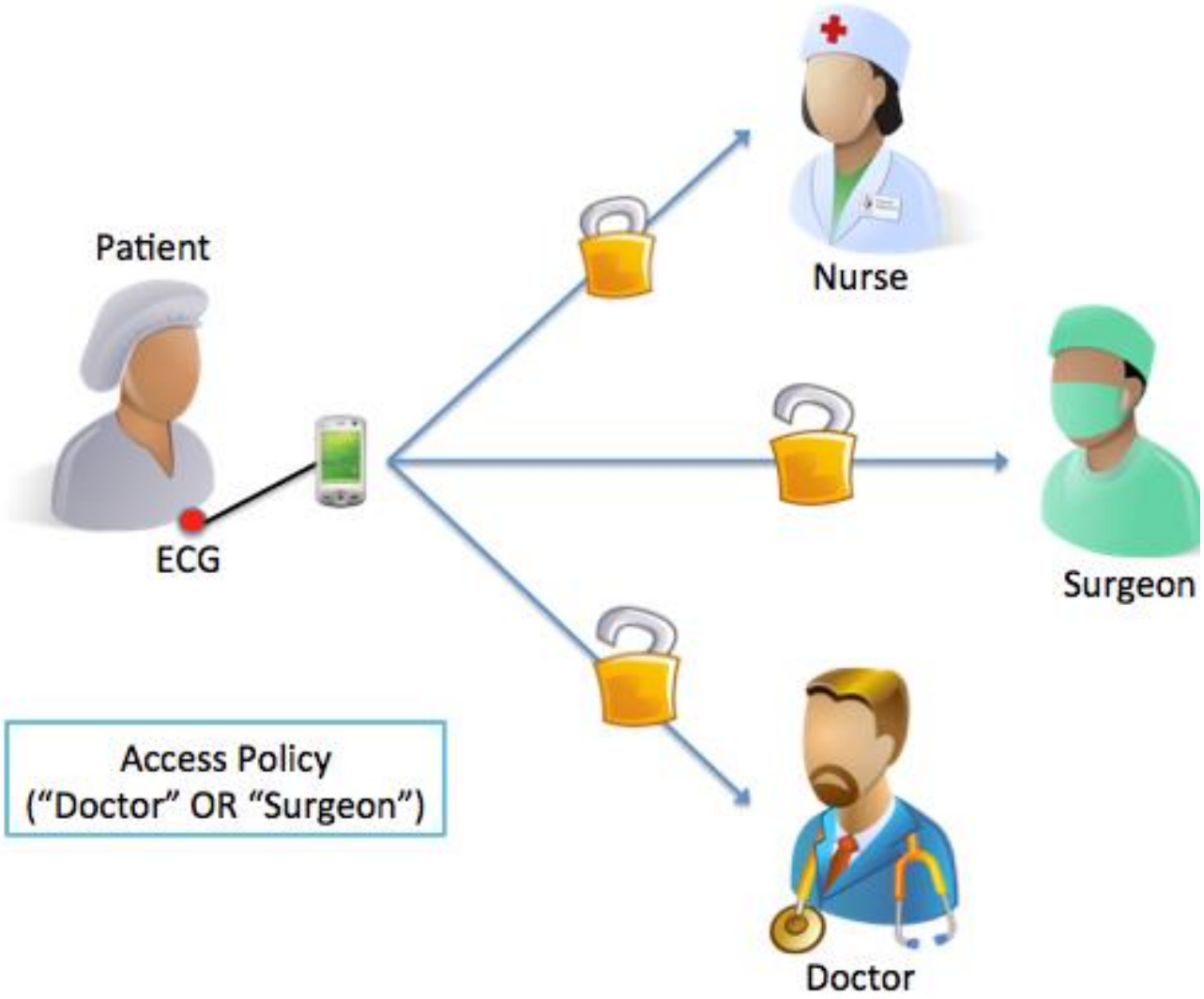

{A, C}

- Decryption:


{A, C}


(A OR B) AND C

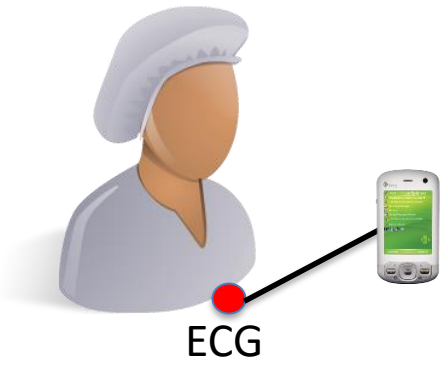
Example


Patient
ECG
Nurse
Surgeon
Doctor
Access Policy
("Doctor" OR "Surgeon")

AGREE

Protocol:

1. event detection: a value is sampled;
2. context evaluation: based on the value sampled, definition of event type (e.g., normal or critical);
3. generation of access control matrix: using LSSS;
4. derivation of key to encrypt data: using Ciphertext-Policy ABE;
5. data encryption: using AES 128;
6. send encrypted data + policy;



PRO vs CONS

- Only one encryption

PRO vs CONS

- Only one encryption
- Access Policies linked to the data

Confidentiality + Access control

PRO vs CONS

- Only one encryption
- Access Policies linked to the data
- Multi-Authority

Hospital

Government

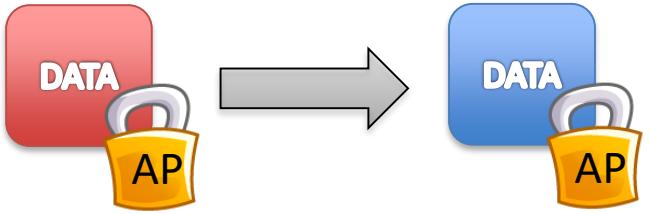
Research Institute

Individuals

PRO vs CONS

- Only one encryption
- Access Policies linked to the data
- Multi-Authority
- Context-aware

Critical events



Doctor OR Nurse

Doctor OR Nurse OR Paramedic

PRO vs CONS

- Only one encryption
- Access Policies linked to the data
- Multi-Authority
- Context-aware
- Long Time Execution
- High Energy Consumption

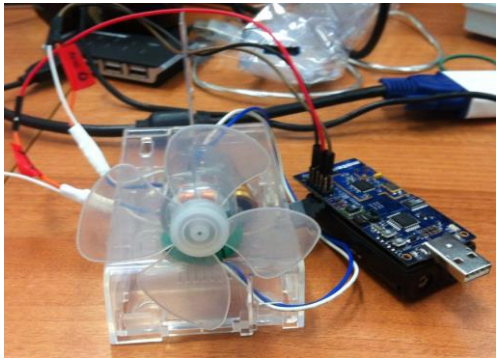
Generation

Energy consume in mJ

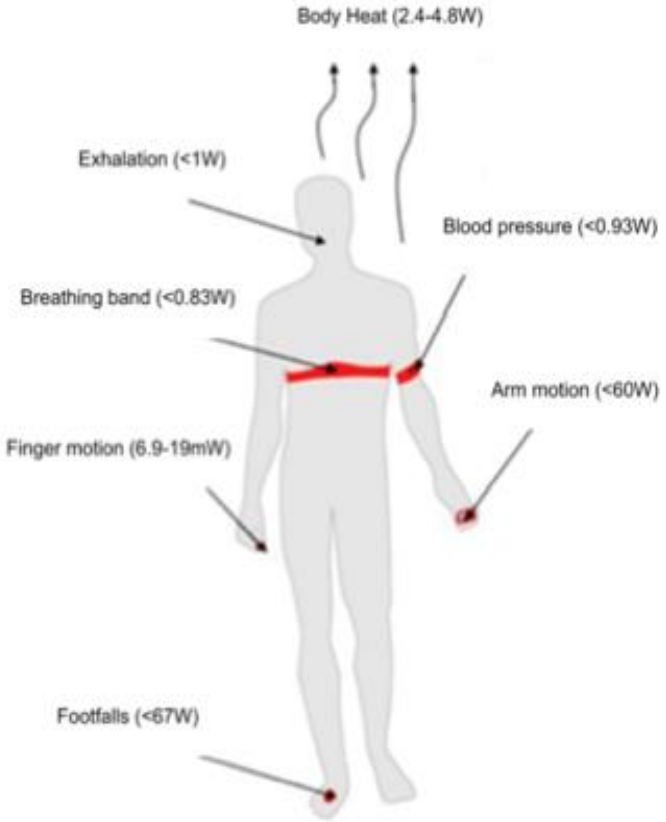
Attributes	Policy Length [bytes]	Tx [bytes]	Scalar Multi.	Energy (Telos B)	Energy (Mica2)
3	349	$209 + \|key\|$	10	115.3	645.1
5	569	$345 + \|key\|$	16	184.6	1032.2
7	797	$489 + \|key\|$	22	253.8	1419.3
9	1033	$641 + \|key\|$	28	323.0	1806.3
11	1277	$801 + \|key\|$	34	392.2	2193.4

Energy sources

Solar

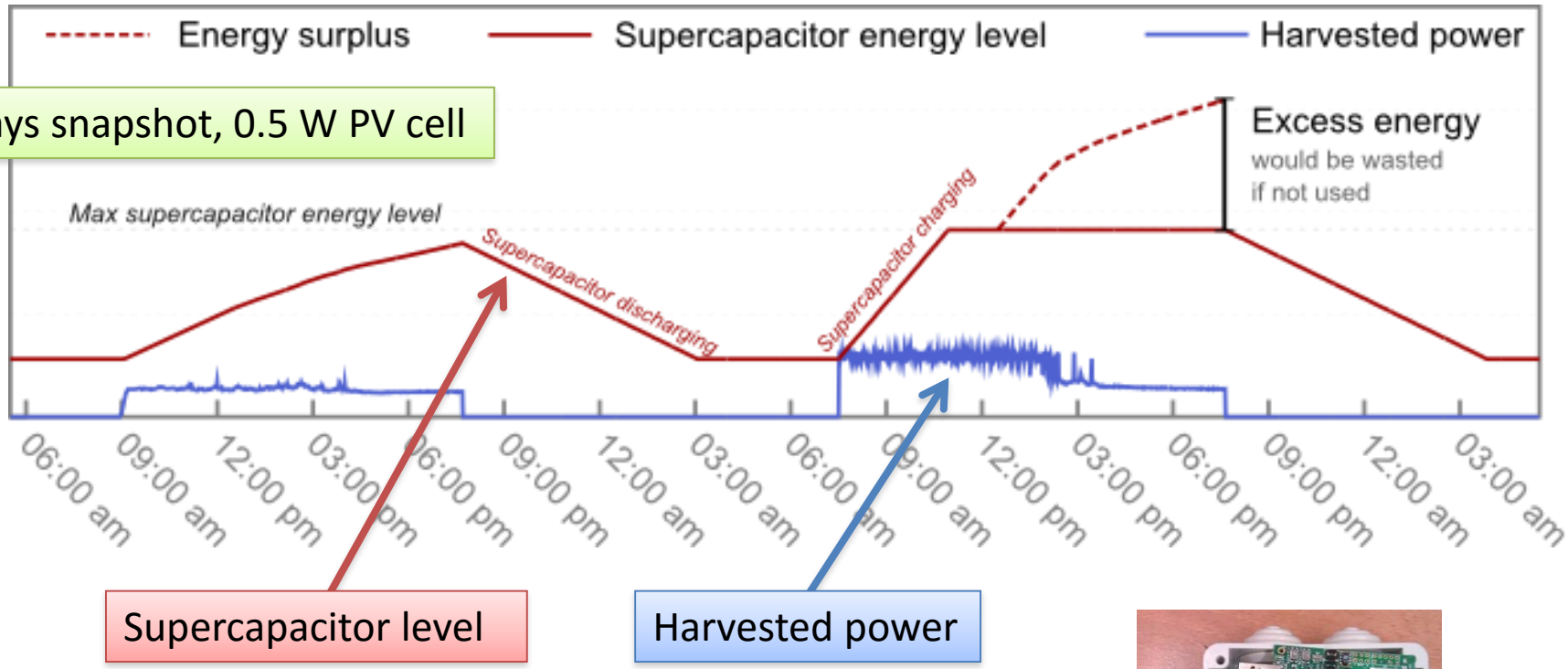


Wind



Human body

Indoor light energy harvesting



Harvest light energy indoor from:

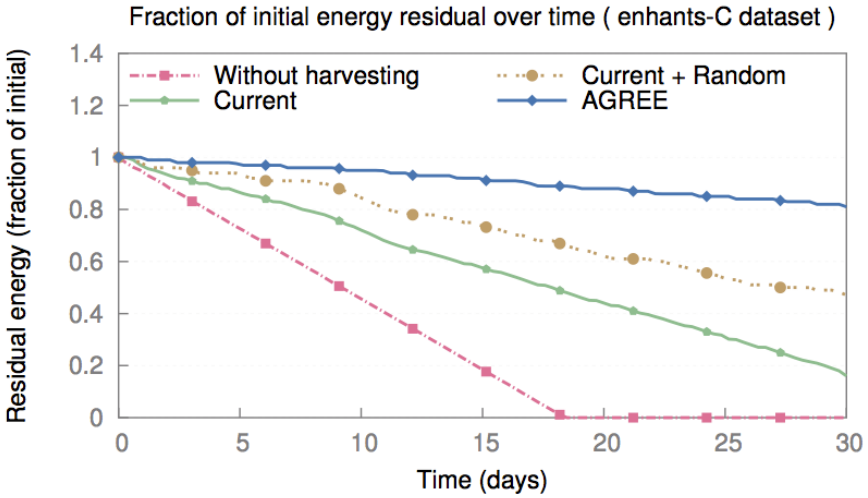
- artificial light generated by ceiling and table lamps
- solar light entering the room from the windows

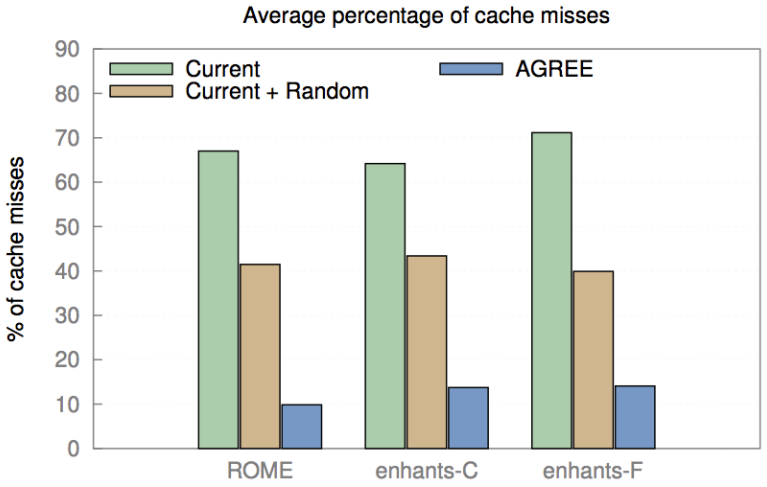
PRO vs CONS

- Only one encryption
- Access Policies linked to the data
- Multi-Authority
- Context-aware
- g Time Execution
- n Energy Consumption

Generation

Pre-computation + energy harvesting + caching



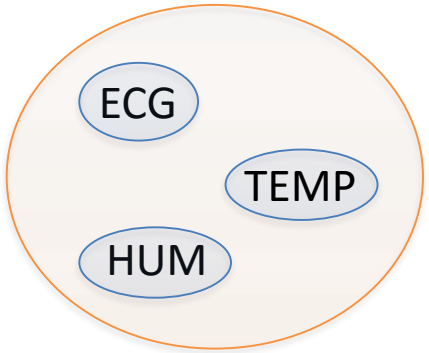


Interesting idea

- Modern sensors are equipped with flash memories which make memory consumption a less critical requirement
- Emerging energy harvesting technologies provide occasional energy peaks which could be exploited for anticipating otherwise costly computational tasks

Combine pre-computation techniques + energy harvesting

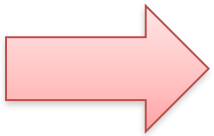
Data from sensors

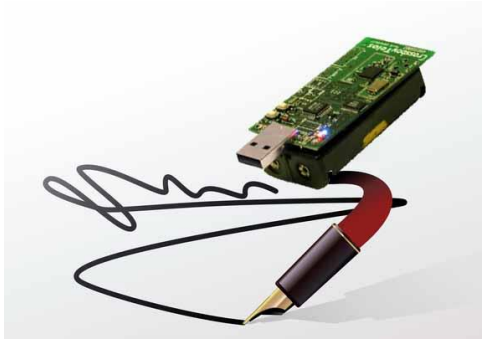


Authentication

+

Integrity

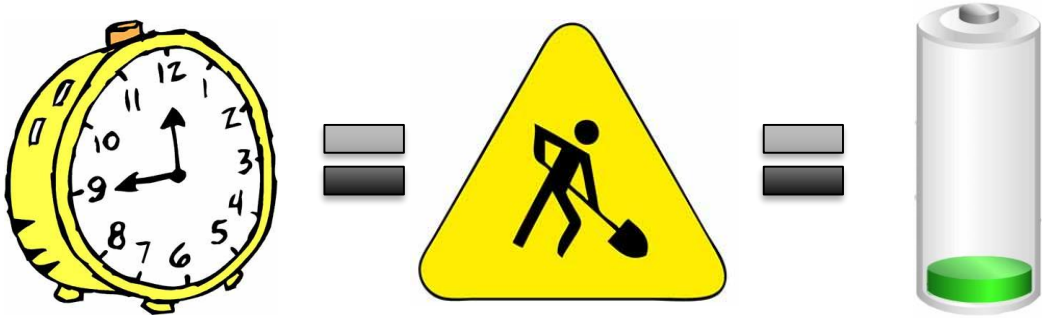




Standard Digital Signature

Standard ECDSA

Standard ECDSA
≈2 seconds



Time | CPU work | Less battery life

Most expansive computation on ECDSA is modular exponentiation $\mathbf{g^r}$

Boyko, Peinado and Venkatesan (BPV)

speeds up the computation by preliminary precomputing, and storing in a table, a number *n* of randomly chosen pairs (x_i, g^{xi}).

$$(x_1, g^{x1}) \mid (x_2, g^{x2}) \mid (\ldots) \mid (x_n, g^{xn})$$

$$(r, g^r) = \left(\sum r_i, \prod g^{ri}\right)$$

I-BPV generator

- Hidden Subset Sum problem (HSS)
- Affine HSS when used with ECDSA

Given integers $M, b_1, \cdots, b_m \in \mathbb{Z}_M$,
find $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}_M$, such that each b_1 is some subset sum of $\alpha_1, \cdots, \alpha_n$ modulo M

Dimension of BPV table and subset size

The BPV generator can be improved

- Random walk on a Cayley graph expander
- I-BPV output essentially follows the uniform distribution
- Memory usage much smaller than before, fits current FLASH
- With proper parameters, security of I-BPV depends on its resistance to birthday attacks

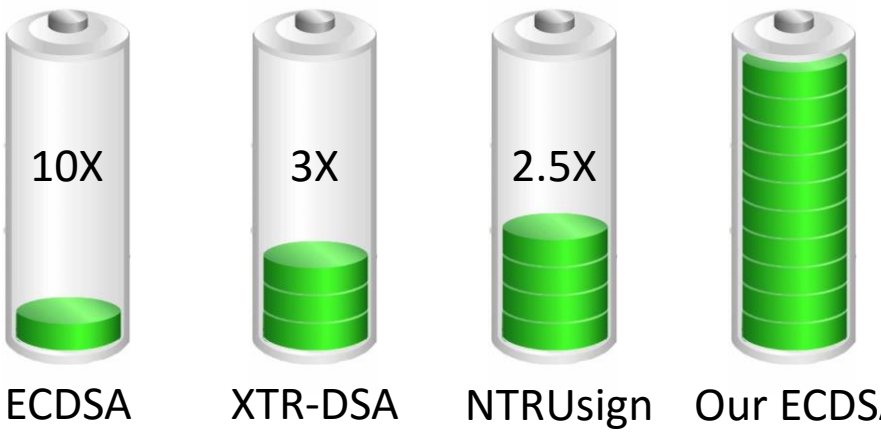
The extra table stored in I- BPV will be five times smaller than the table in BPV, reducing the space overhead of the generator

Comparision

Comparision with NTRUSign, other optimizations of ECDSA, and XTR-DSA.
Parameters: $n = 160$, $n_e = 32$, $k = 8$

Author(s)	Scheme	ROM	RAM	Sig	k_{priv}	k_{pub}	t_{sign}	$E_{CPU}(t_{sign})$
Gura et al.,	RSA	7.4kB	1.1kB	128B	128B	131B	10.99s	263.8mJ
Liu et al.,	ECDSA	19.3kB	1.5kB	40B	21B	40B	2.001s	14.8mJ
Driessen et al.,	NTRUSign	11.3kB	542kB	127B	383B	127B	0.619s	22.3mJ
	ECDSA	43.2kB	3.2kB	40B	21B	40B	0.918s	22.0mJ
	XTR-DSA	24.3kB	1.6kB	40B	20B	176B	0.965s	23.2mJ
This work	ECDSA	18.2kB	1.2kB	40B	21B	40B	0.346s	8.1mJ

Energy Consumption



Modular exponentiation

2.24 s

0.25 s

Anonymizing Network Technologies

Some slides modified from Dingledine, Mathewson, Syverson, Xinwen Fu, and Yinglin Sun

Problem

- Internet surveillance like traffic analysis reveals users privacy.
- Encryption does not work, since packet headers still reveal a great deal about users.
- End-to-end anonymity is needed.

- Solution: a distributed, anonymous network

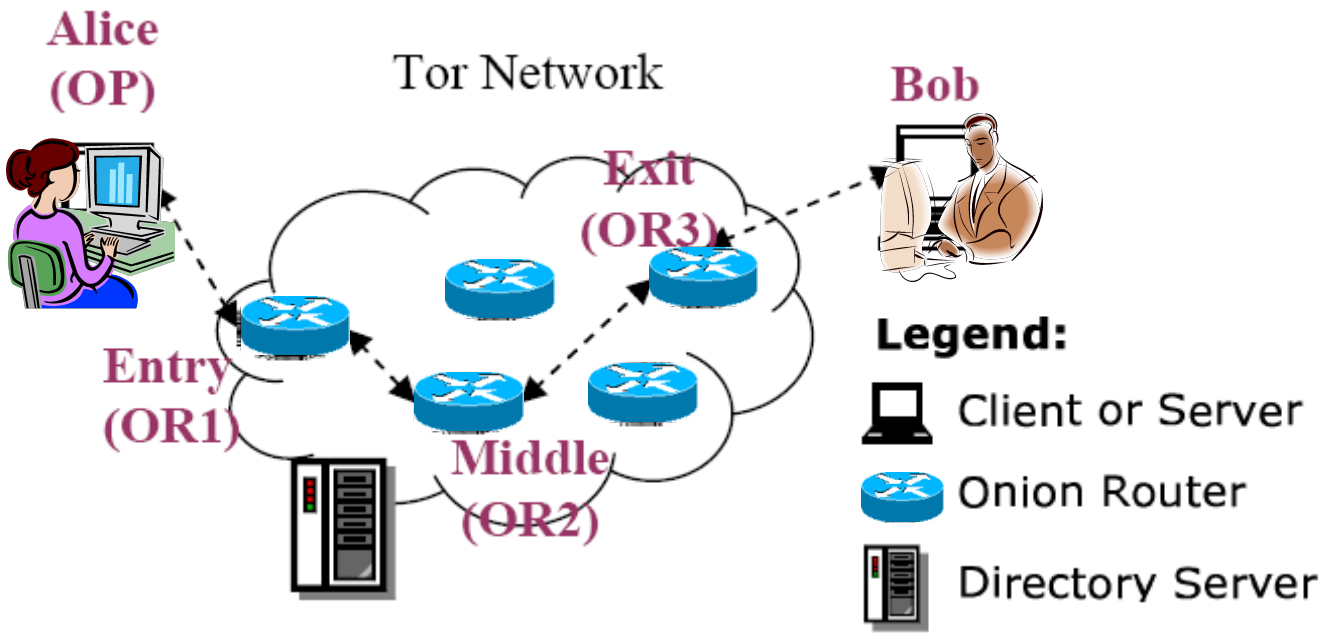
What is Tor

- Tor is a distributed anonymous communication service using an overlay network that allows people and groups to improve their privacy and security on the Internet.
- Individuals use Tor to keep websites from tracking them, or to connect to those internet services blocked by their local Internet providers.
- Tor's hidden services let users publish web sites and other services without needing to reveal the location of the site.

Design

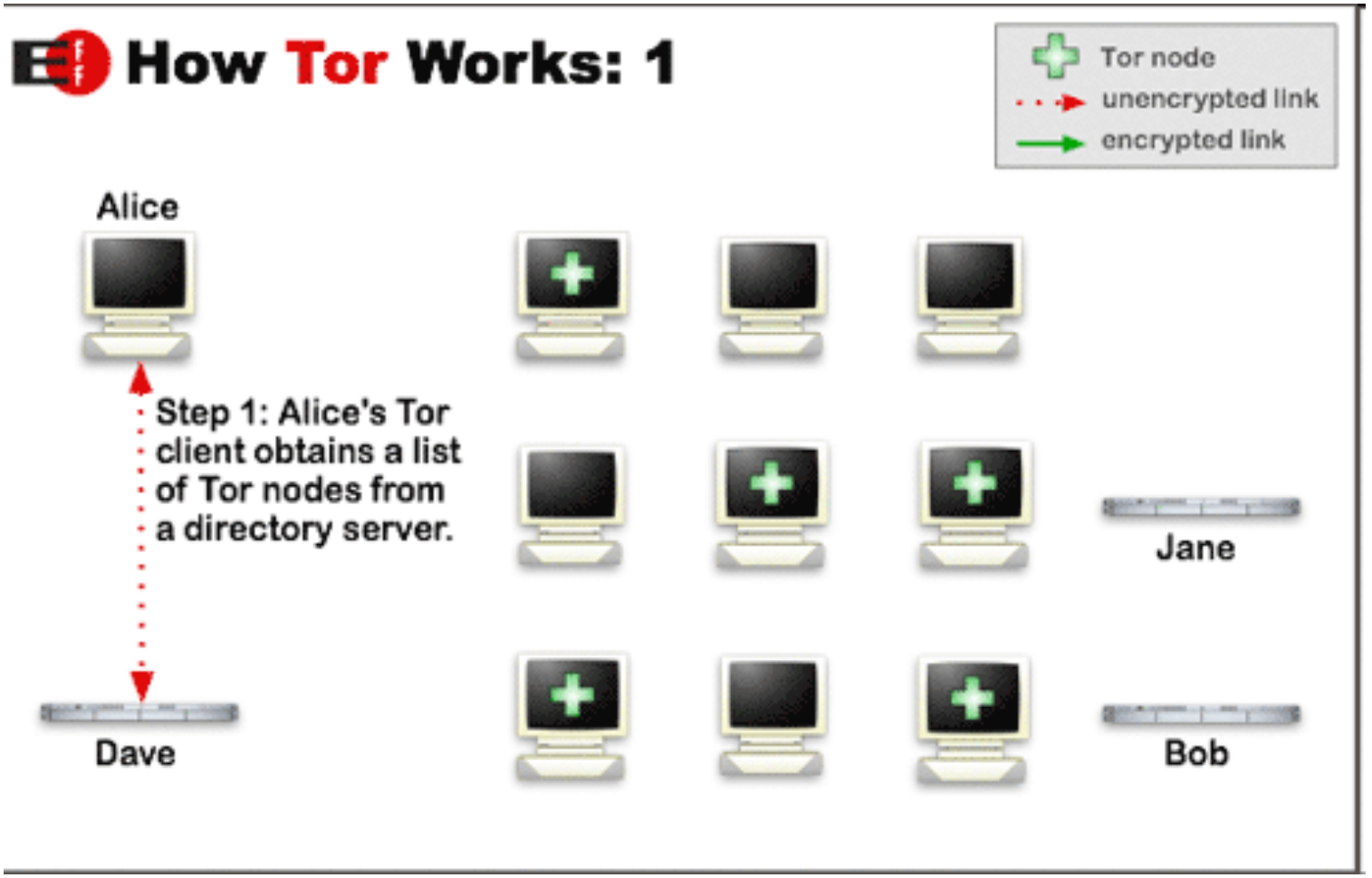
- Overlay network on the user level
- Onion Routers (OR) route traffic
- Onion Proxy (OP) fetches directories and creates virtual circuits on the network on behalf of users.
- Uses TCP with TLS
- All data is sent in fixed size (bytes) cells

Components of Tor

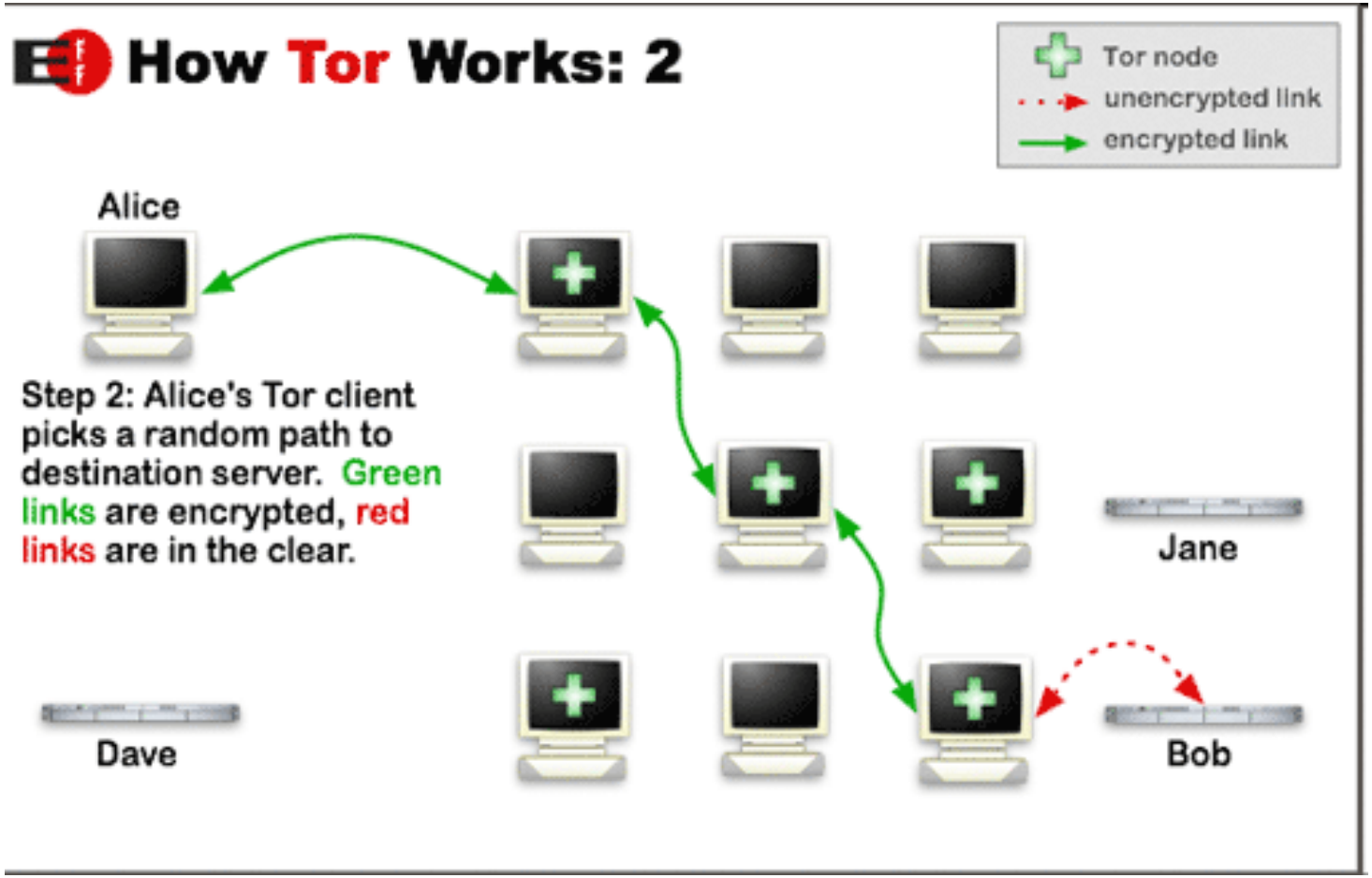


- **Client**: the user of the Tor network
- **Server**: the target TCP applications such as web servers
- **Tor (onion) router**: the special proxy relays the application data
- **Directory server**: servers holding Tor router information

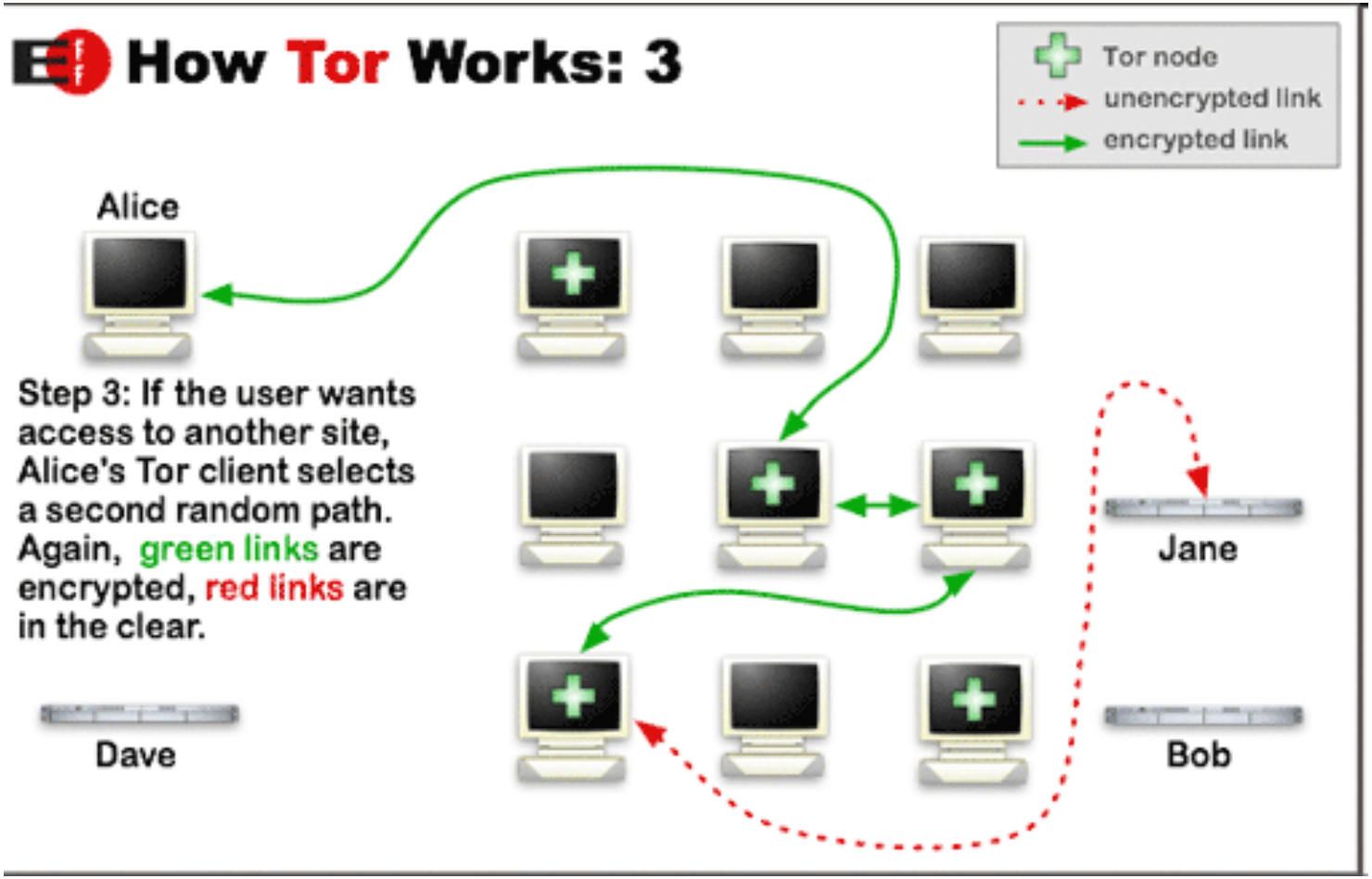
How does Tor work?



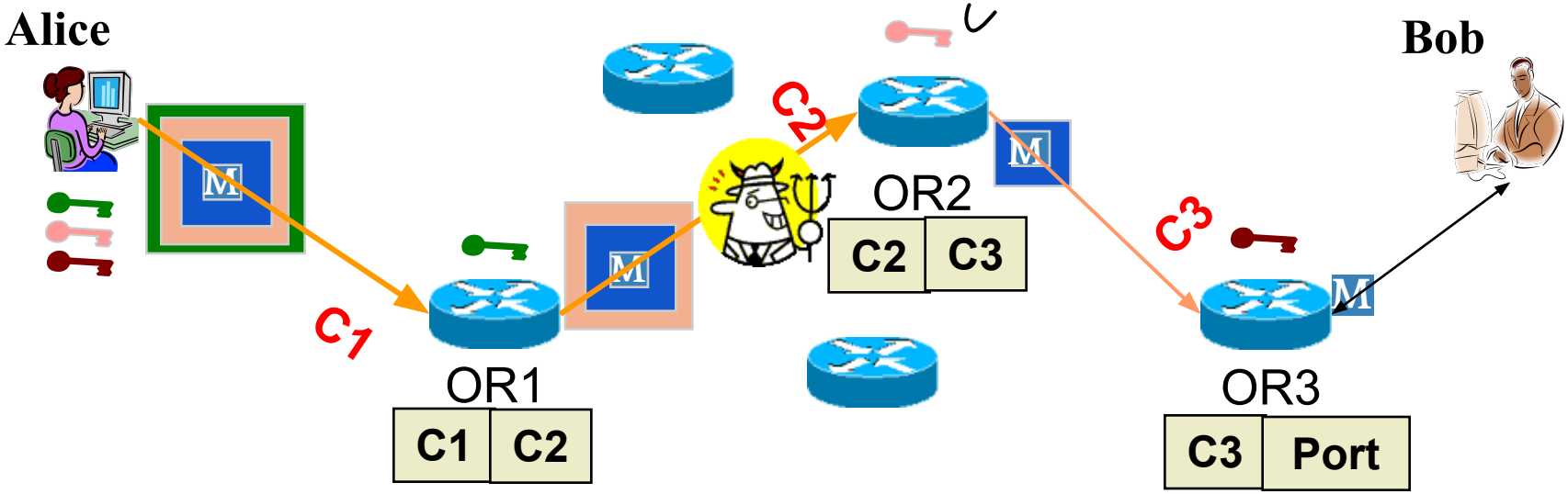
How does Tor work?



How does Tor work?

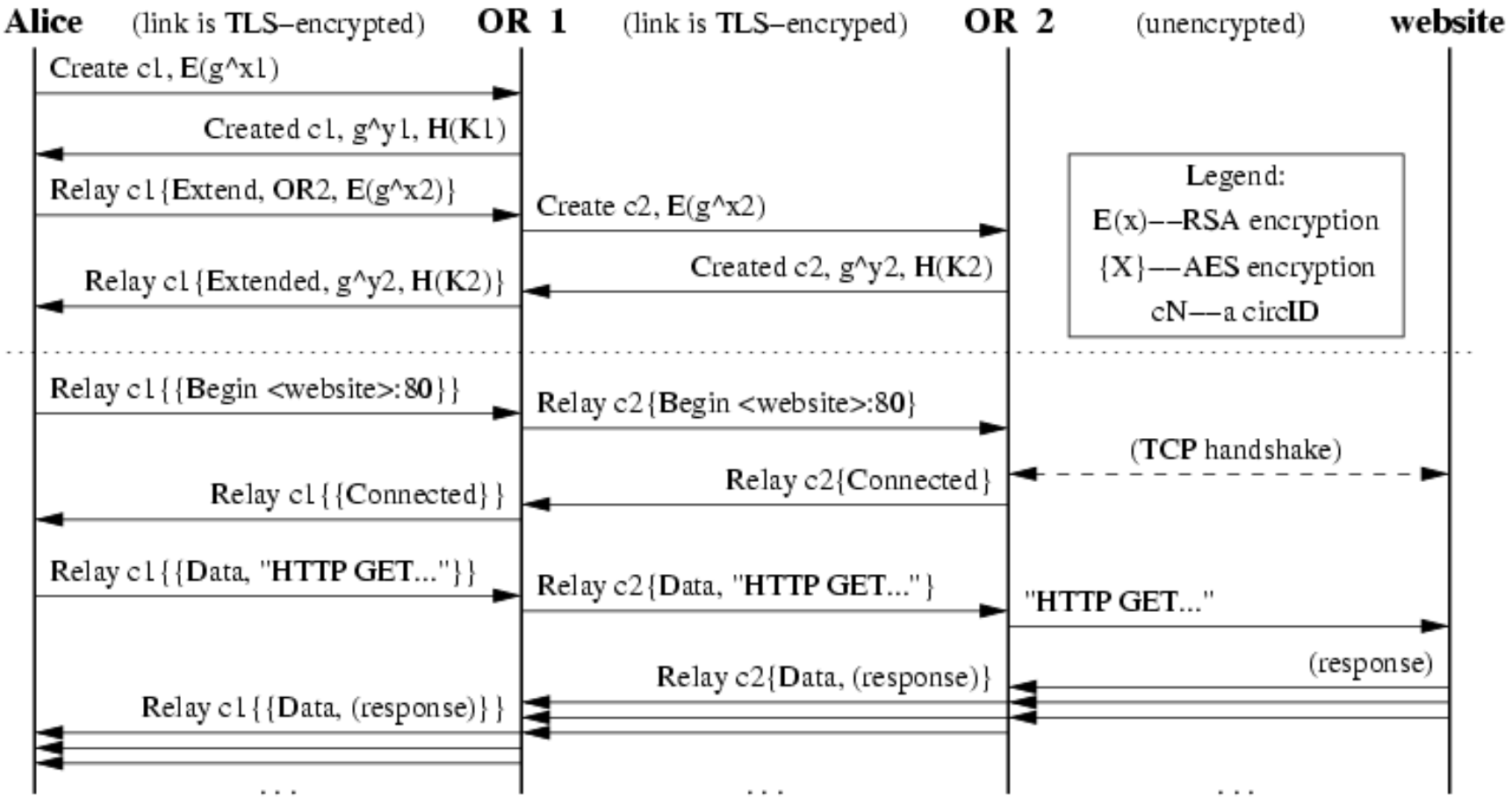


How Tor Works? --- Onion Routing



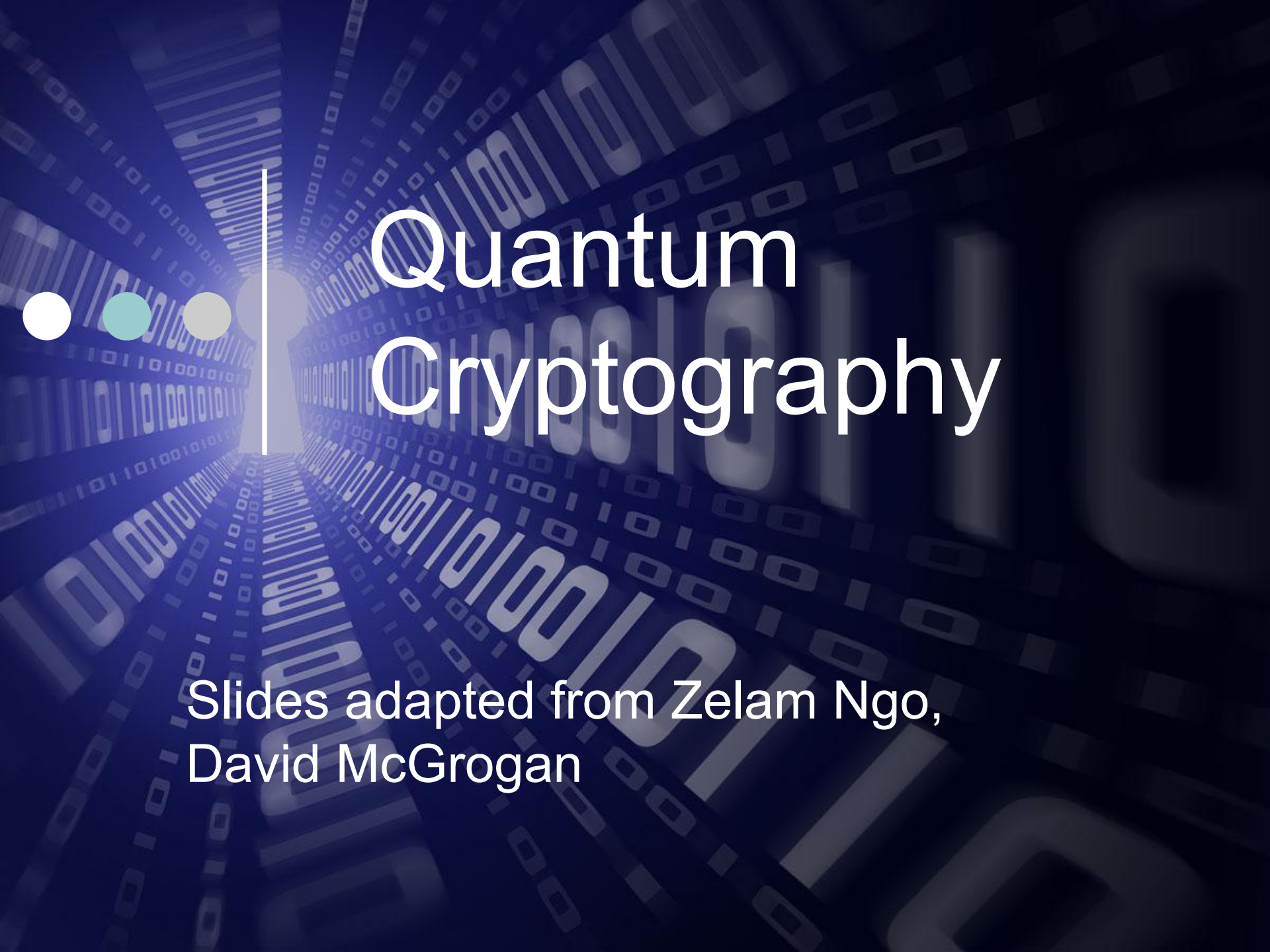
- A circuit is built incrementally one hop by one hop
- Onion-like encryption
  - Alice negotiates an AES key with each router
  - Messages are divided into equal sized cells
  - Each router knows only its predecessor and successor
  - Only the Exit router (OR3) can see the message, however it does not know where the message is from

Commands in Use



Additional functionality

- Integrity checking
  - Only done at the edges of a stream
  - SHA-1 digest of data sent and received
  - First 4 bytes of digest are sent with each message for verification



Quantum Cryptography

Slides adapted from Zelam Ngo, David McGrogan

Motivation

- Age of Information
- Information is valuable
- Protecting that Information

Quantum Security Benefits

- Provably Secure
- Evidence of Tampering

History

- Stephen Wiesner wrote “Conjugate Coding” in the late sixties
- Charles H. Bennett and Gilles Brassard revived the field in 1982 by combining quantum process with public key cryptography.

Fundamentals

- Measurement causes perturbation
- No Cloning Theorem
- Thus, measuring the qubit in the wrong basis destroys the information.

BB84

- Set-up
  - Alice
    - Has the ability to create qubits in two orthogonal bases
  - Bob
    - Has the ability to measure qubits in those two bases.

BB84

- Alice
  - Encodes her information randomly in one of the two bases…
    - For example,

Basis A	Basis B
$\|0\rangle = 0$	$\|+\rangle = 0$
$\|1\rangle = 1$	$\|-\rangle = 1$

BB84

Alice prepares 16 bits

0101100010101100

in the following bases,

BAABAABAAAABBBBA

Thus the following states are sent to Bob:

+10-10+0101+--+0

BB84

Alice's bits	0101100010101100
Alice's bases	BAABAABAAAABBBBA
States sent	+10-10+0101+--+0

Bob receives the stream of qubits and measures each one in a random basis:

ABAABAAABABBBBAB

BB84

Alice's bits	0101100010101100
Alice's bases	BAABAABAAAABBBBA
States sent	+10-10+0101+--+0
Bob's bases	ABAABAAABABBBBAB

So Bob gets

1-00-0+0+0-+--1+

BB84

Alice's bits	0101100010101100
Alice's bases	BAABAABAAAABBBBA
States sent	+10-10+0101+--+0
Bob's bases	ABAABAAABABBBBAB
Bob's results	1-00-0+0+0-+--1+

Then Alice and Bob compare their measurement bases, not the results, via a public channel.

BB84

Bob receives the stream of qubits and measures each one in a random basis:

ABAABAAABABBBBAB

So he gets,

00*0*0*+--**

Then Alice and Bob compare their measurement bases, not the results, via a public channel.

BB84

- So Bob and Alice are left with 7 useable bits out of 16

_ _ 0 _ _ 0 _ 0 _ 0_ 0 1 1 _ _

These bits will be the shared key they use for encryption.

BB84

- Now enter Eve… She wants to spy on Alice and Bob.
- So she intercepts the bit stream from Alice, measures it, and prepares a new bit stream to Bob based on her measurements…

BB84

So how do we know when Eve is being nosy?

Well… Eve doesn't know what bases to measure in, so she would have to measure randomly and 50% of the time she will be wrong…

BB84

- Thus, of the bits Bob measures in the correct bases, there is 50% that eve had changed the basis of the bit. And thus it is equally likely that Bob measure 0 or 1 and thus an error is detected 25% of the time.

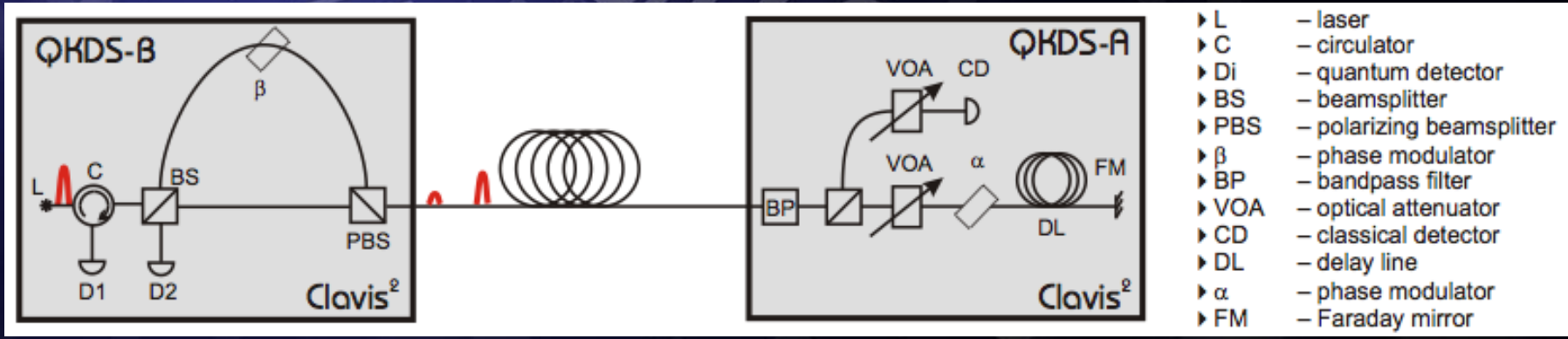
- Eve is found in the errors!

BB84

- In a world with perfect transmissions, all Bob and Alice have to do is publicly compare a few bits to determine if any error exists.
- Errors exist in reality, thus the only way to detect Eve is to notice an increase in errors.
- Thus the transmission process must not have an error rate higher than 25%.

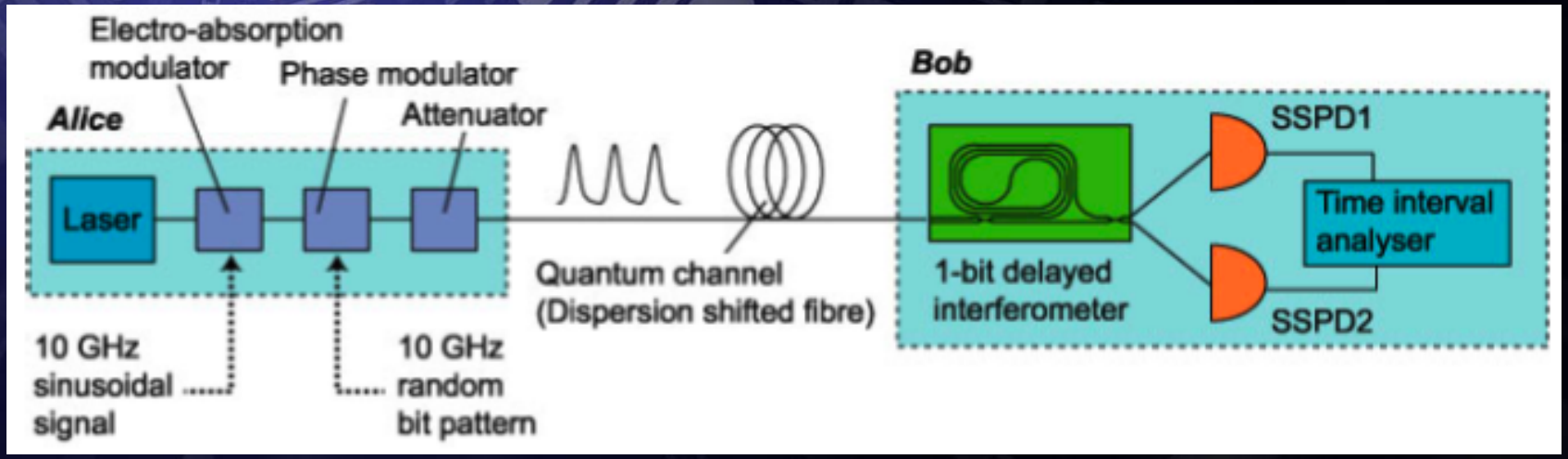
Current State of Affairs

- Commercial quantum key distribution products exist



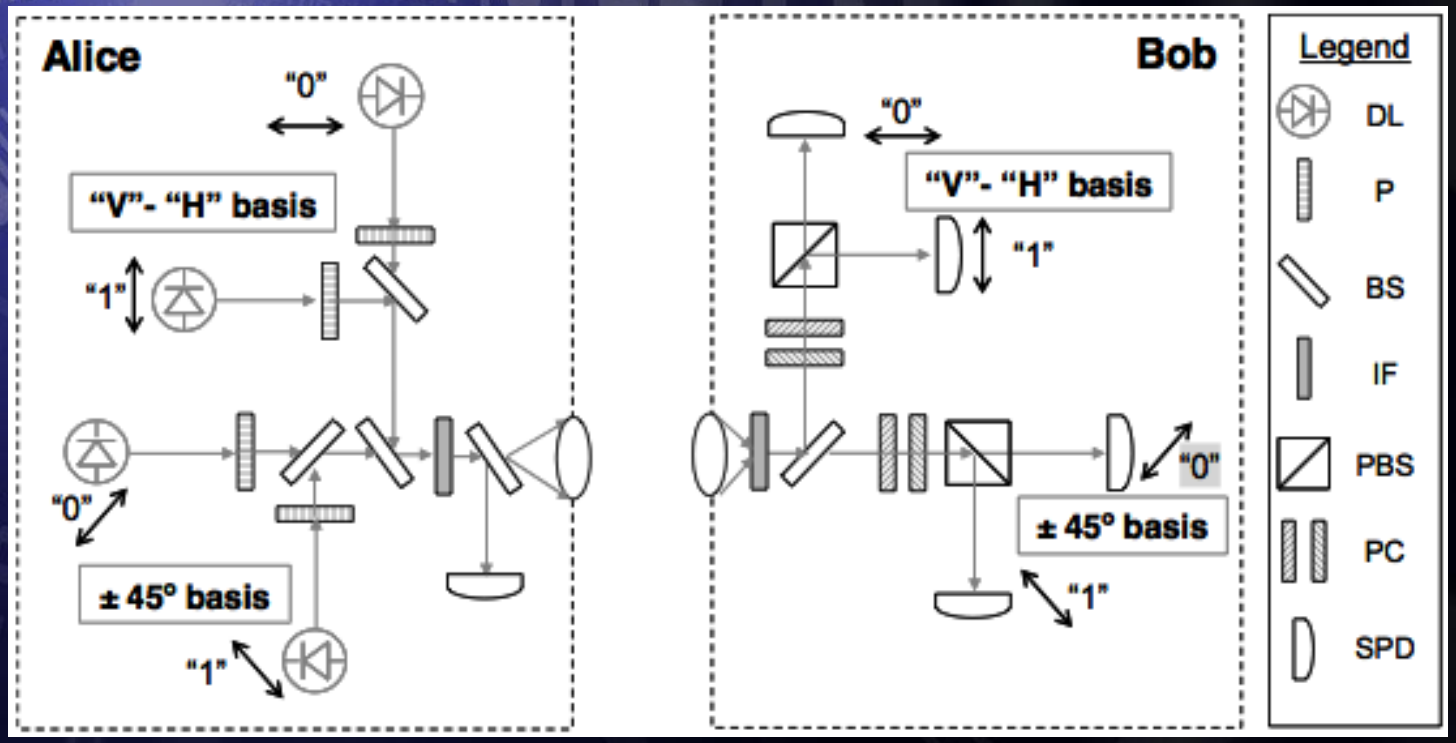
Current State of Affairs

- Current fiber-based distance record: 200 km (Takesue et al)



Current State of Affairs

- Demonstrated free-space link: 10 km



Future Prospects

- Ground-to-satellite, satellite-to-satellite links
- General improvement with evolving qubit-handling techniques, new detector technologies

Computer Security: Principles and Practice

Chapter 3: User Authentication

slides prepared by Dr Lawrie Brown (UNSW@ADFA) for
“Computer Security: Principles and Practice”

Password authentication

- Widely used user authentication method
  - user provides name/login and password
  - system compares password with that saved for specified login
- Authenticates ID of user logging and
  - that the user is authorized to access system
  - determines the user's privileges
  - is used in discretionary access control

Password vulnerabilities

- offline dictionary attack
- specific account attack (user john)
- popular password attack (against a wide range of IDs)
- password guessing against single user (w/ previous knowledge about the user)
- workstation hijacking
- exploiting user mistakes
- exploiting multiple password use
- electronic monitoring

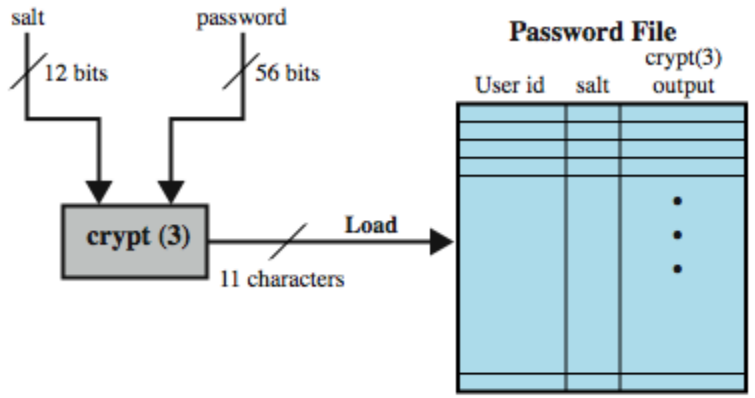
Countermeasures for password vulnerability

- stop unauthorized access to password file
- intrusion detection measures
- account lockout mechanisms
- policies against using common passwords but rather hard to guess passwords
- training & enforcement of policies
- automatic workstation logout
- encrypted network links

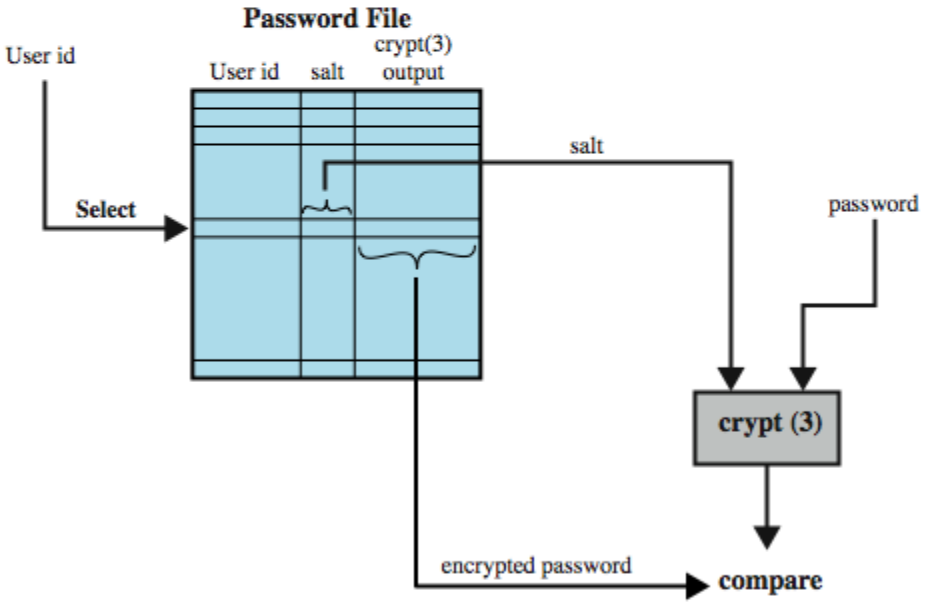
Countermeasures for password vulnerability

- It is worthwhile to study/research password and password vulnerabilities
  - Most common
  - Still the most efficient

Use of hashed passwords



(a) Loading a new password



(b) Verifying a password

Why a salt value?

- Prevents duplicate passwords from being visible in the password file
- Increases the difficulty of offline dictionary attacks
- Nearly impossible to tell if a person used the same password on multiple systems

UNIX Implementation

- Original scheme
  - 8 character password form 56-bit key
  - 12-bit salt used to modify DES encryption into a one-way hash function
  - output translated to 11 character sequence
- Now regarded as woefully insecure
  - e.g. supercomputer, 50 million tests, 80 min
- Sometimes still used for compatibility

Improved implementations

- Have other, stronger, hash/salt variants
- Many systems now use MD5
  - with 48-bit salt
  - password length is unlimited
  - is hashed with 1000 times inner loop
  - produces 128-bit hash
- OpenBSD uses Blowfish block cipher based and hash algorithm called Bcrypt
  - uses 128-bit salt to create 192-bit hash value

Password Cracking

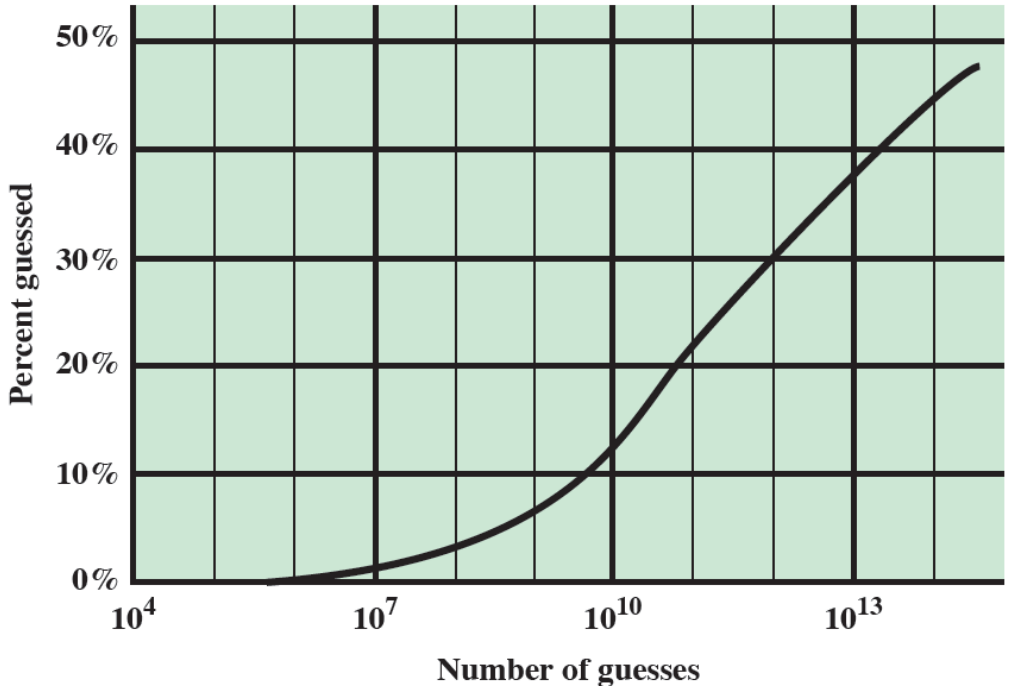
- Dictionary attacks
  - try each word then obvious variants in large dictionary against hash in password file
- **Rainbow** table attacks
  - a large dict of possible passwords
  - for each password:
    - precompute tables of hash values for all salts
    - a mammoth table of hash values: e.g. 1.4GB table cracks 99.9% of alphanumeric Windows passwords in 13.8 secs
  - not feasible if larger salt values used

Password choices/concerns

- users may pick short passwords
  - e.g. 3% were 3 chars or less, easily guessed
  - system can reject choices that are too short
- users may pick guessable passwords
  - so crackers use lists of likely passwords
  - e.g. one study of 14000 encrypted passwords guessed nearly 1/4 of them
  - would take about 1 hour on fastest systems to compute all variants, and only need 1 break!

Another case study

- An analysis of passwords used by 25,000 students
- Over 10% recovered after 10^10 guesses



Password File Access Control

- Can block offline guessing attacks by denying access to encrypted passwords
  - make available only to privileged users
  - often using a separate shadow password (for su only)
- Still have vulnerabilities
  - exploit O/S bug
  - accident with permissions making it readable
  - users with same password on other systems
  - access from unprotected backup media
  - sniff passwords in unprotected network traffic

Using Better Passwords

- Clearly have problems with passwords
- Goal to eliminate guessable passwords
  - Still easy for user to remember
- Techniques
  - user education
  - computer-generated passwords
  - reactive password checking (periodic checking)
  - proactive password checking (at the time of selection)

Proactive Password Checking

- Rule enforcement plus user advice, e.g.
  - 8+ chars, upper/lower/numeric/punctuation
  - may not suffice
- Password cracker
  - list of bad passwords
  - time and space issues
- Markov Model
  - generates guessable passwords
  - hence reject any password it might generate
- Bloom Filter
  - use to build table based on dictionary using hashes
  - check desired password against this table

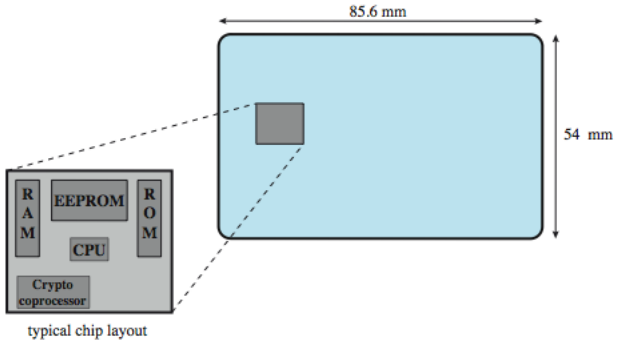
Token-based authentication

- Object user possesses to authenticate, e.g.
  - memory card (magnetic stripe)
  - smartcard

Memory Card

- store but do not process data
- magnetic stripe card, e.g. bank card
- electronic memory card
- used alone for physical access (e.g., hotel rooms)
- some with password/PIN (e.g., ATMs)
- Drawbacks of memory cards include:
  - need special reader
  - loss of token issues
  - user dissatisfaction (OK for ATM, not OK for computer access)

Smartcard

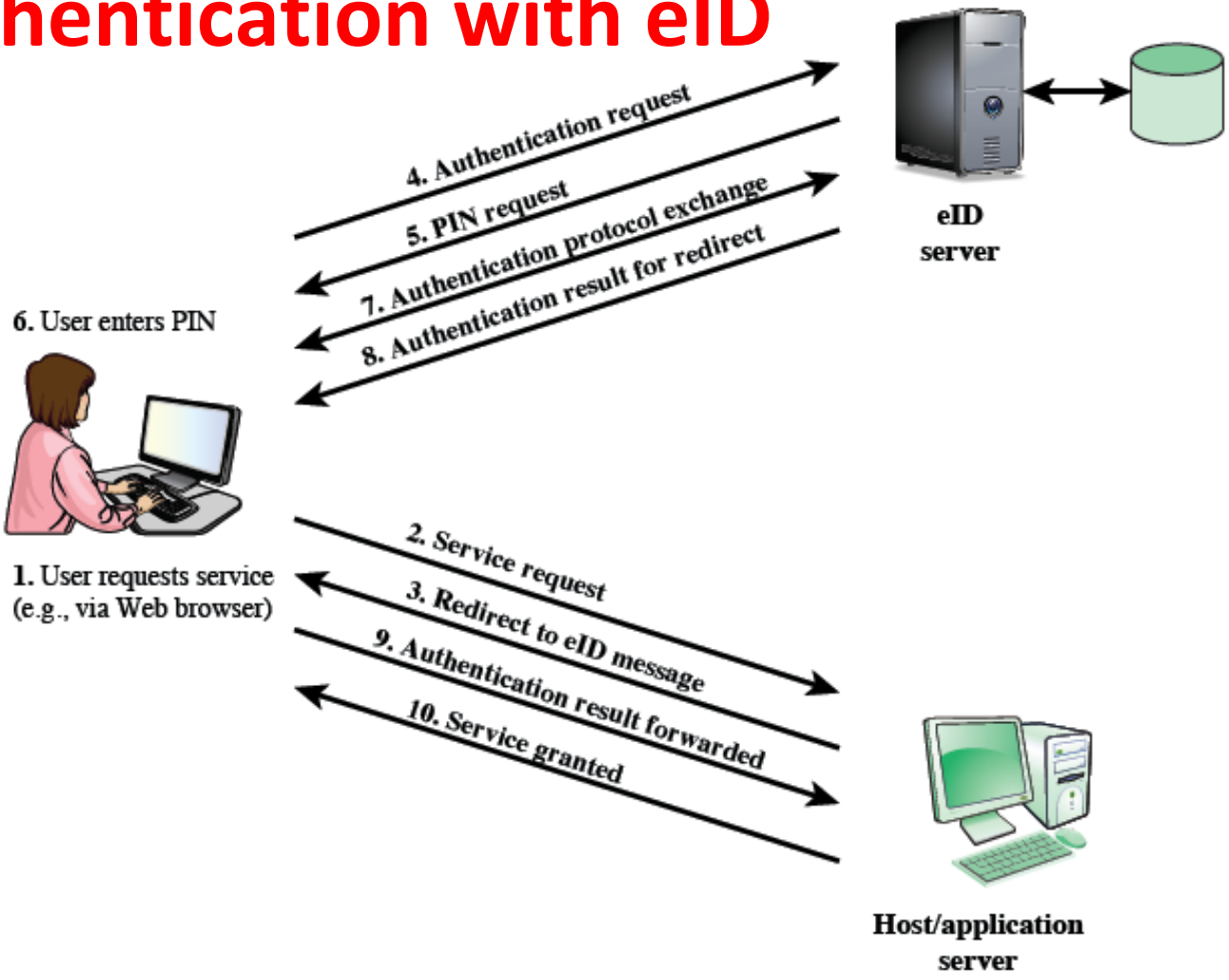


- credit-card like
- has own processor, memory, I/O ports
  - ROM, EEPROM, RAM memory
- executes protocol to authenticate with reader/computer
  - static: similar to memory cards
  - dynamic: passwords created every minute; entered manually by user or electronically
  - challenge-response: computer creates a random number; smart card provides its hash (similar to PK)
- also have USB dongles

Electronic identify cards

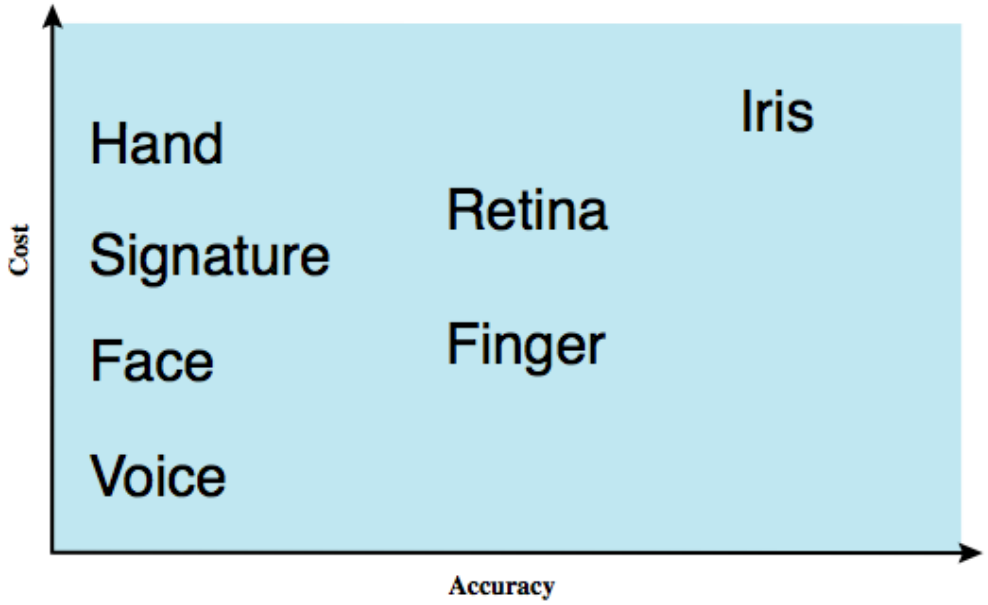
- An important application of smart cards
- A national e-identity (eID)
- Serves the same purpose as other national ID cards (e.g., a driver's licence)
  - Can provide stronger proof of identity
  - A German card
    - Personal data, Document number, Card access number (six digit random number), Machine readable zone (MRZ): the password
    - Uses: ePass (government use), eID (general use), eSign (can have private key and certificate)

User authentication with eID



Biometric authentication

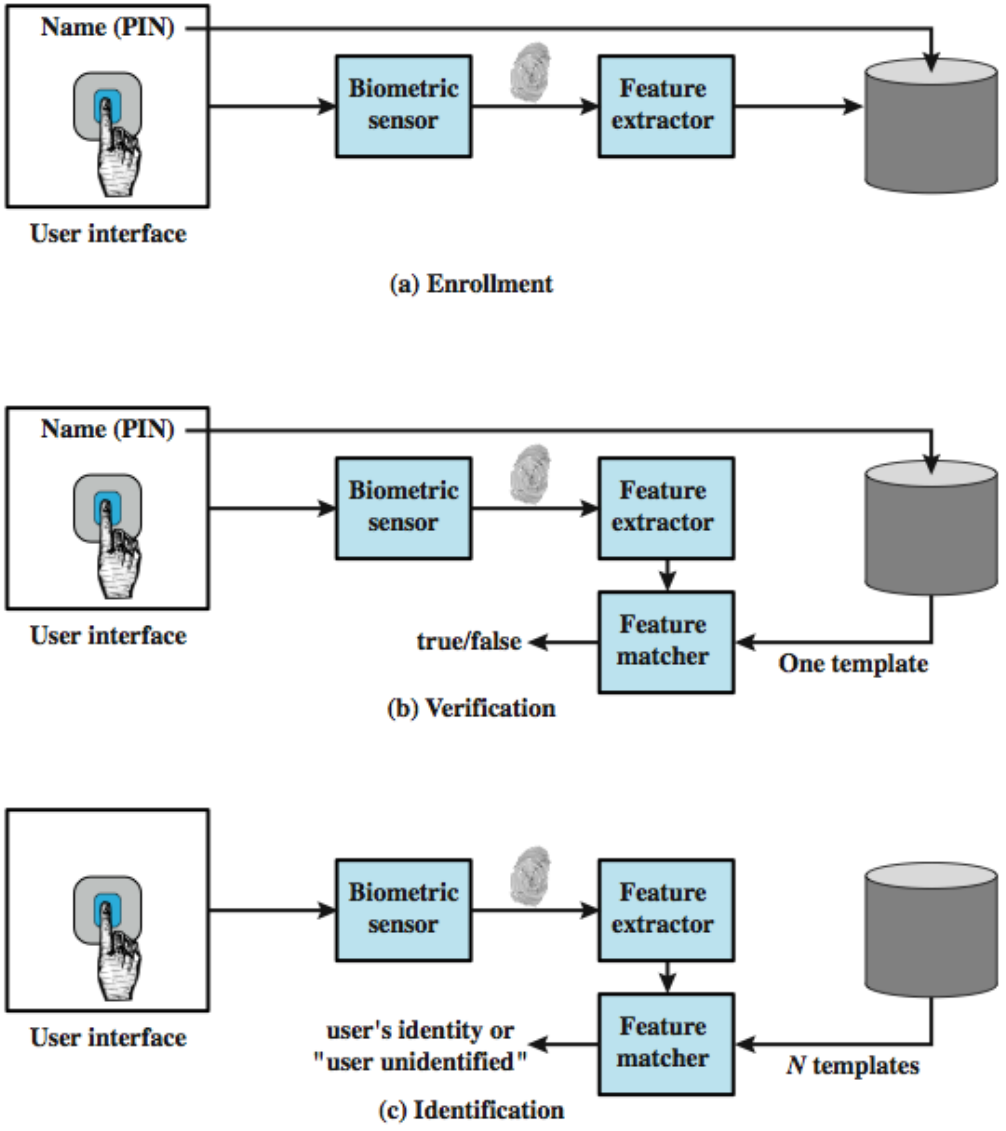
- Authenticate user based on one of their physical characteristics:
  - facial
  - fingerprint
  - hand geometry
  - retina pattern
  - iris
  - signature
  - voice



Operation of a biometric system

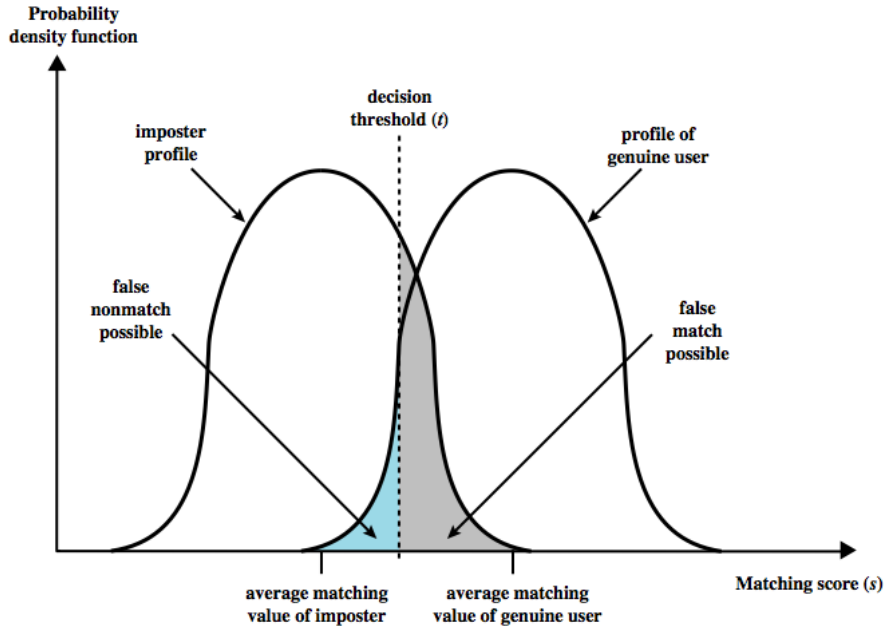
Verification *is analogous to user login via a smart card and a PIN*

Identification *is biometric info but no IDs; system compares with stored templates*



Biometric Accuracy

- The system generates a matching score (a number) that quantifies similarity between the input and the stored template
- Concerns: sensor noise and detection inaccuracy
- Problems of false match/false non-match



Biometric Accuracy

- Can plot characteristic curve (2,000,000 comparisons)
- Pick threshold balancing error rates