Computer Security

Alberto Pace

alberto.pace@cern.ch

CERN Data Management Group

Part 1: An introduction to Cryptography

Alberto Pace

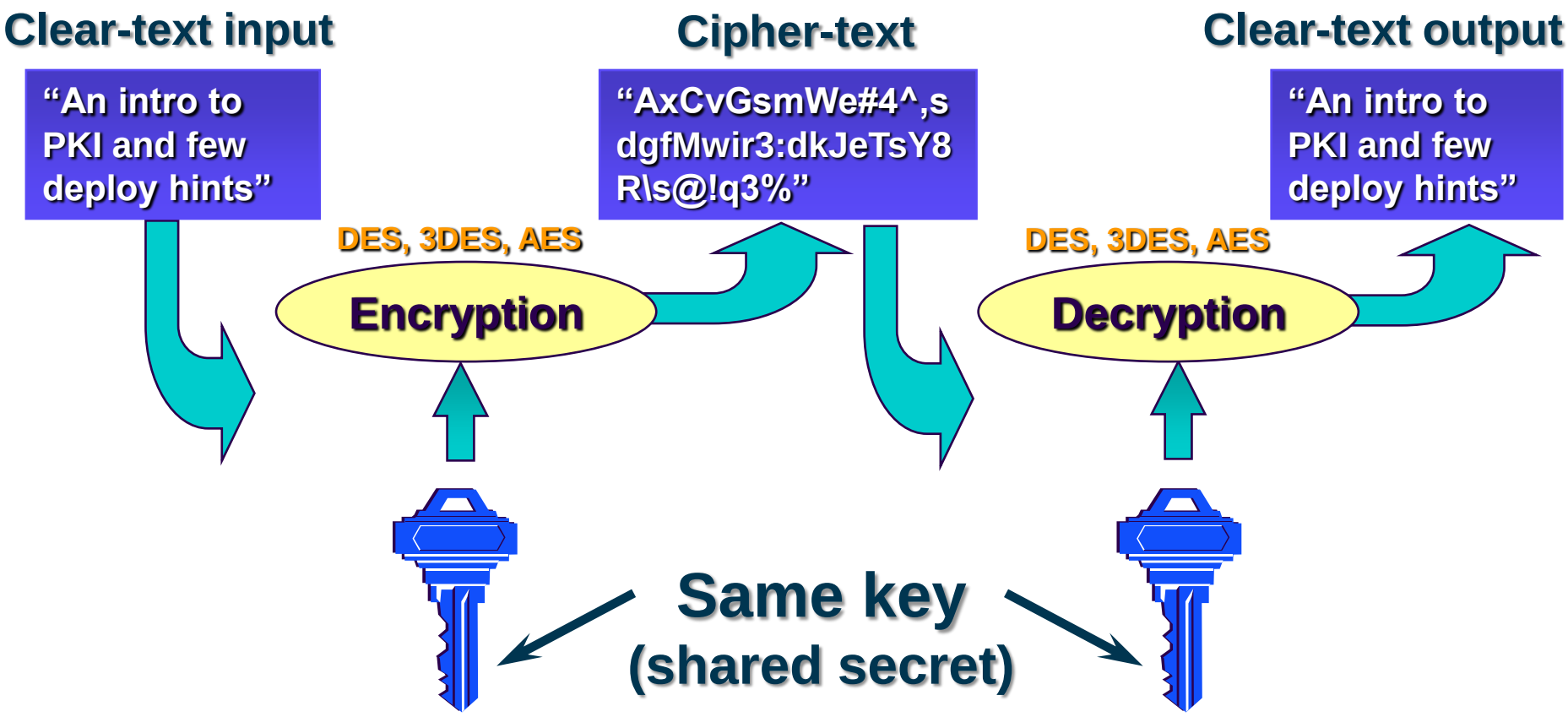
alberto.pace@cern.ch

CERN Data Management Group

What does Cryptography solve?

- **Confidentiality**
  - Ensure that nobody can get knowledge of what you transfer even if listening the whole conversation
- **Integrity**
  - Ensure that message has not been modified during the transmission
- **Authenticity, Identity, Non-repudiation**
  - You can verify that you are talking to the entity you think you are talking to
  - You can verify who is the specific individual behind that entity
  - The individual behind that asset cannot deny being associated with it

Symmetric Encryption



Example: XOR function

XOR	0	1
0	0	1
1	1	0

Encryption

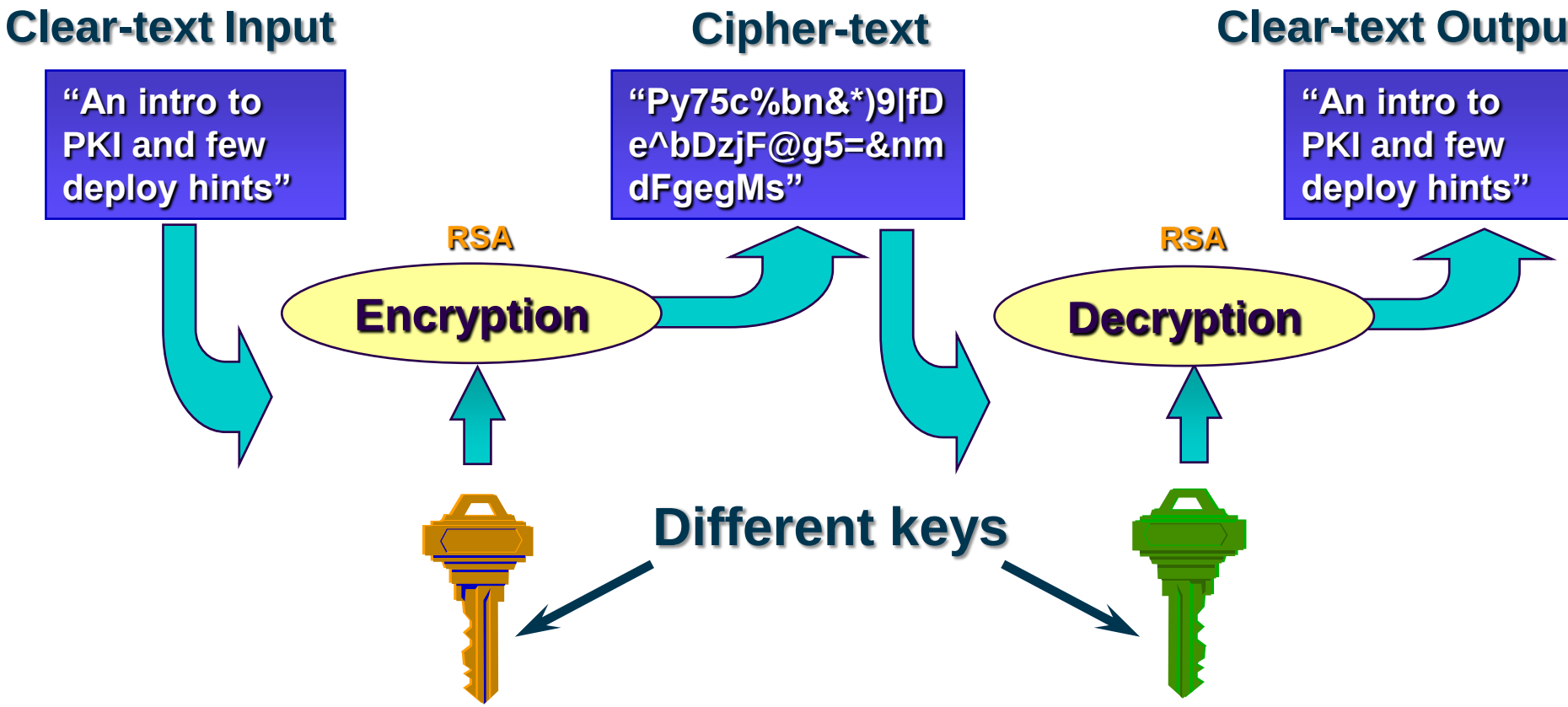
Message	1	1	0	1	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1
Key	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0
Cyper	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1

Decryption

Cypher	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1
Key	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0
Message	1	1	0	1	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1

Secure if Key length = Message Length

Asymmetric Encryption



CERN School of Computing

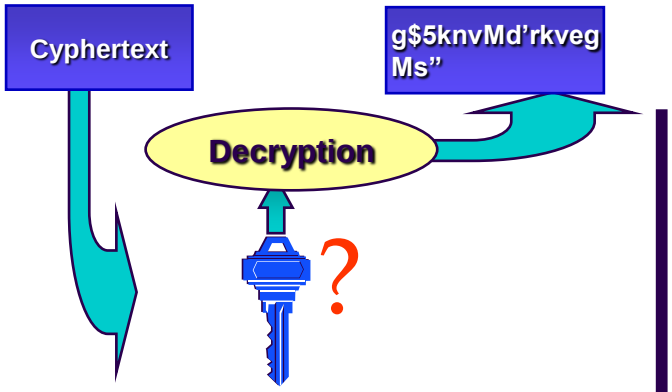
Asymmetric Encryption

- **Things to remember**
  - The relation between the two keys is unknown and from one key you cannot gain knowledge of the other, even if you have access to clear-text and cipher-text
  - The two keys are interchangeable. All algorithms make no difference between public and private key. When a key pair is generated, any of the two can be public or private (in theory but not in practice)

CERN School of Computing

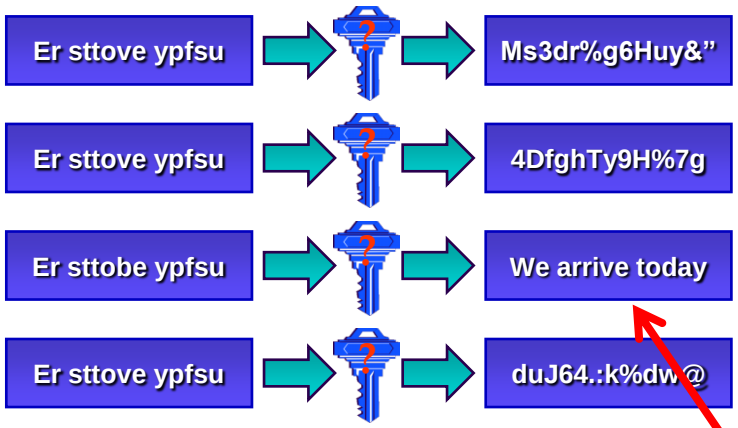
What "Cracking" means ...

- **Cracking Asymmetric encryption is like solving a (difficult) mathematical problem that is entirely defined**
  - Find x, y so that
    x * y = 55491390292407720175546138652590303070607716961484 89
  - (Answer: 283341988972178712821759 9, 19584598277756992630240051 1)
- **Cracking Symmetric encryption requires a way to verify that your "supposed" decryption is correct**
  - Guessing the message may rely on supposed redundancy
  - Compression is important because it removes potential redundancy in the cyphertext



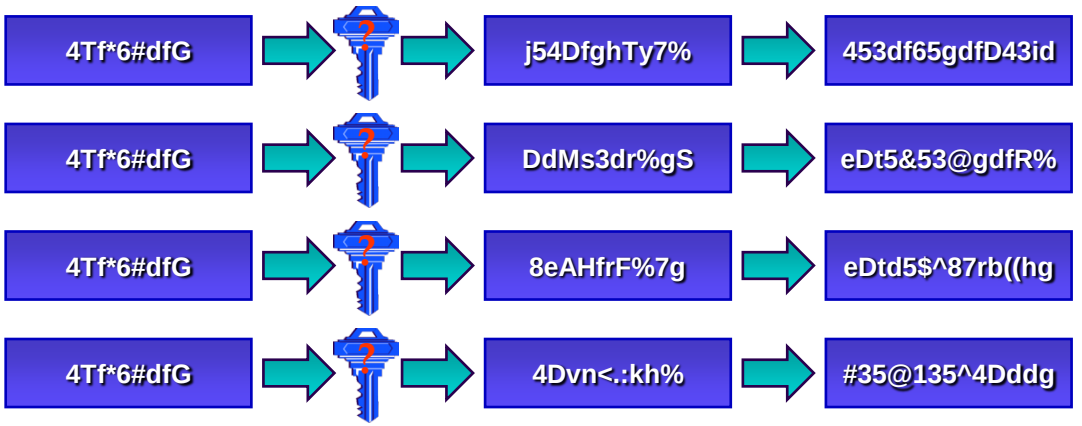
Cracking symmetric encryptions

Uncompressed message
(redundancy may appear in the cyphertext)



Dictionary attacks possible

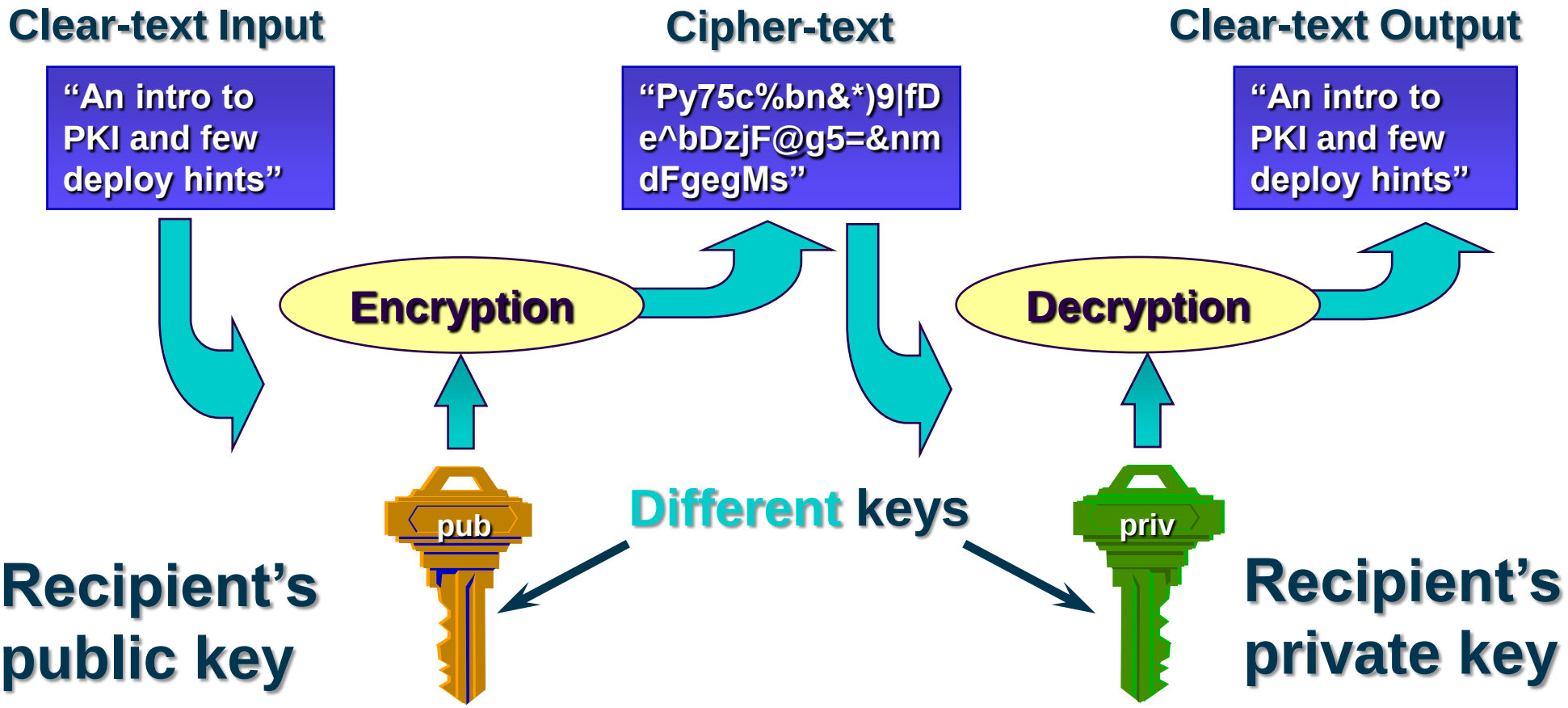
Compressed message – no redundancy
(in the original cyphertext)



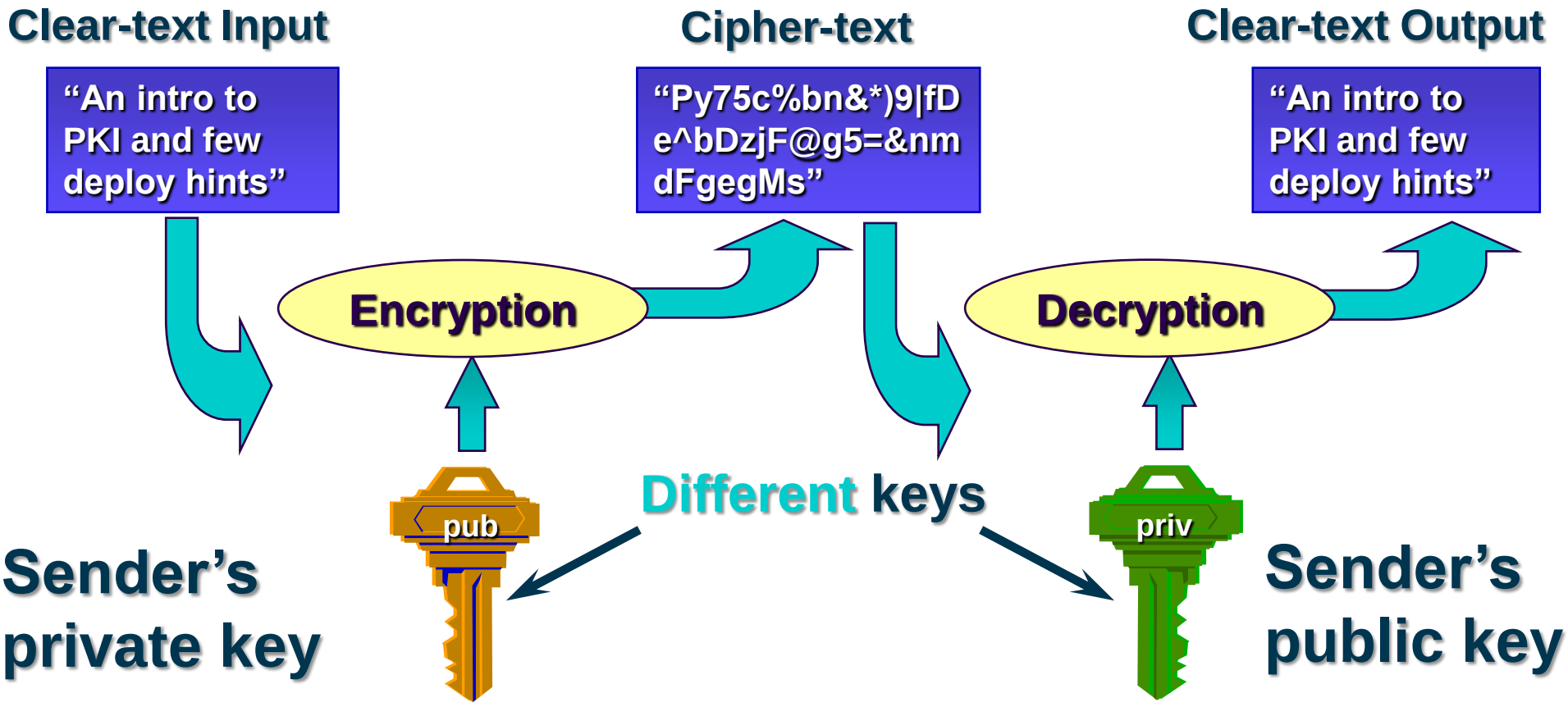
Only brute force attacks

Hmmm … could be this one

Example: Confidentiality

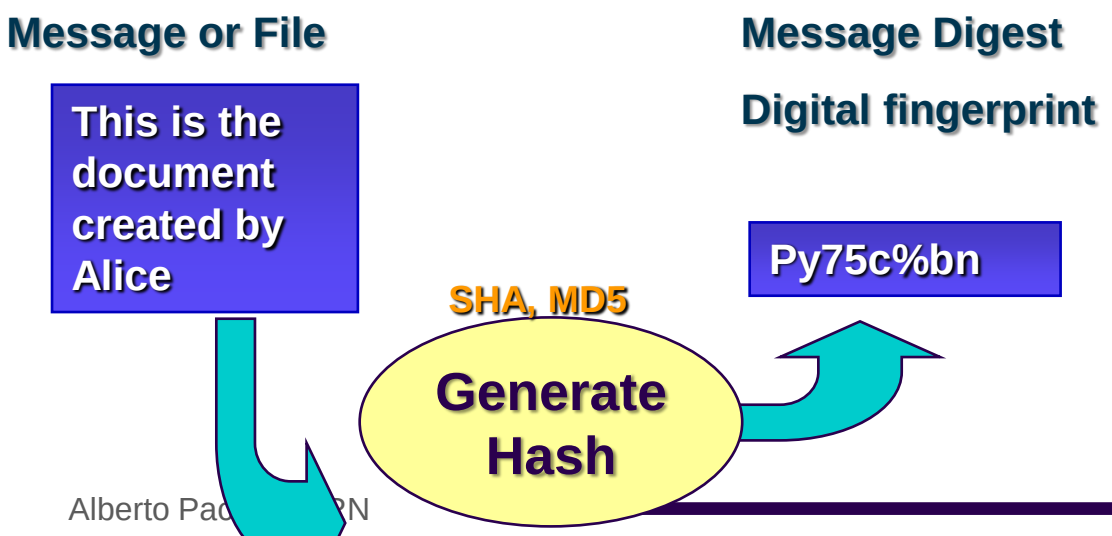


Example: Authenticity

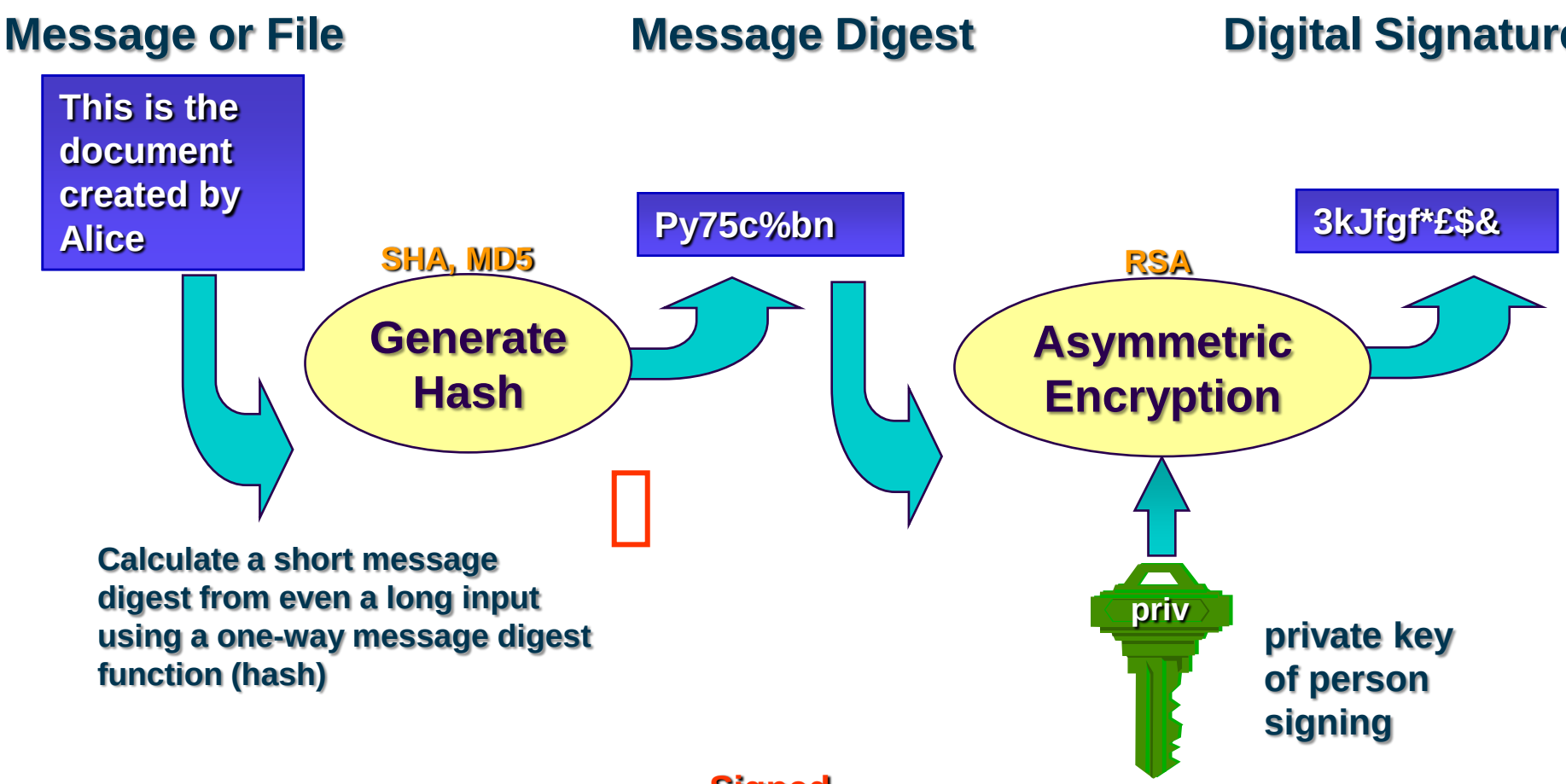


Cryptographic Hash Functions

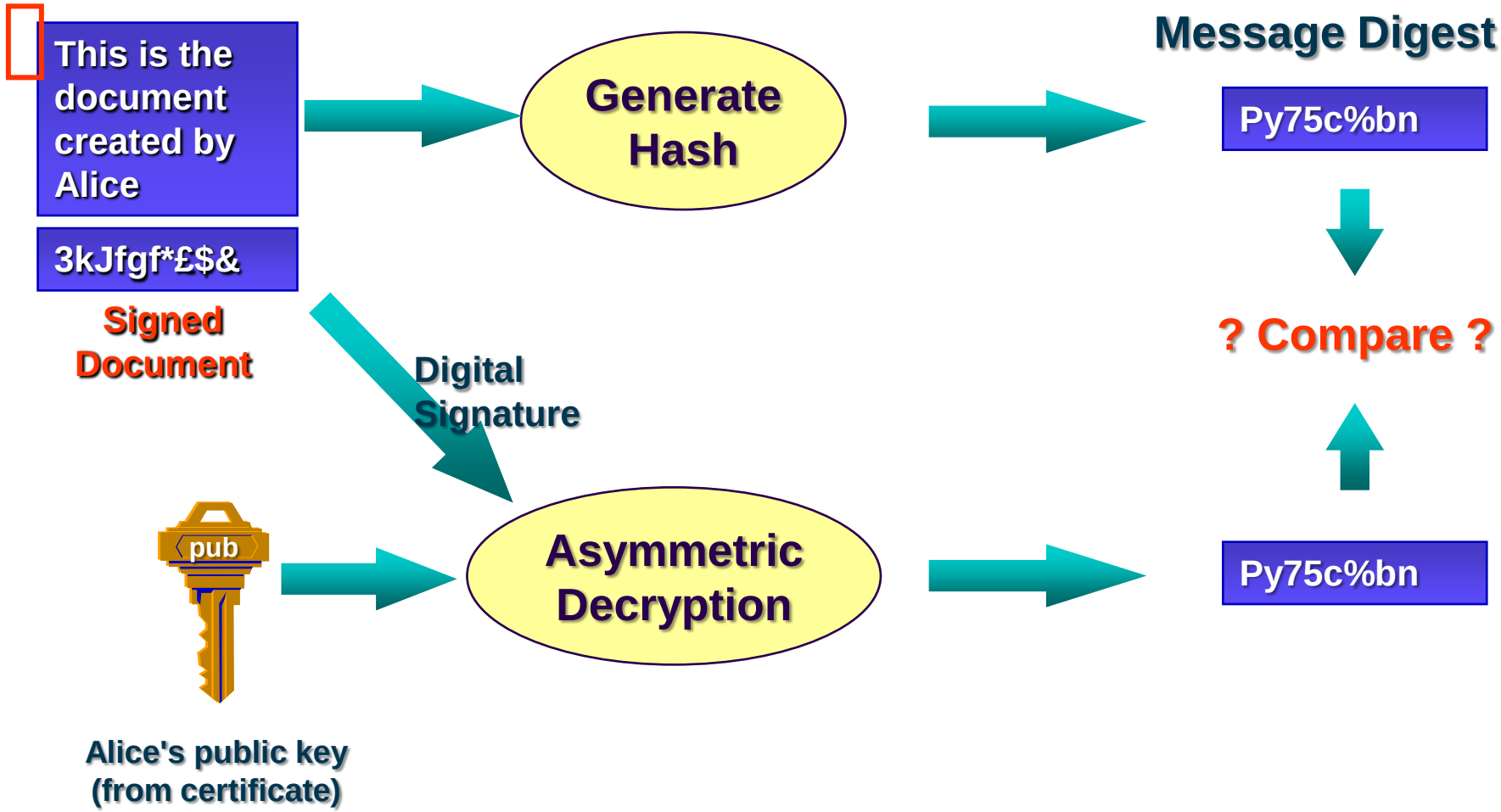
- **A transformation that returns a fixed-size string, which is a short representation of the message from which it was computed**
  - Any (small) modification in the message generates a modification in the digest
- **Should be efficiently computable and impossible to:**
  - find a (previously unseen) message that matches a given digest
  - find "collisions", wherein two different messages have the same message digest



Example: Integrity
Creating a Digital Signature

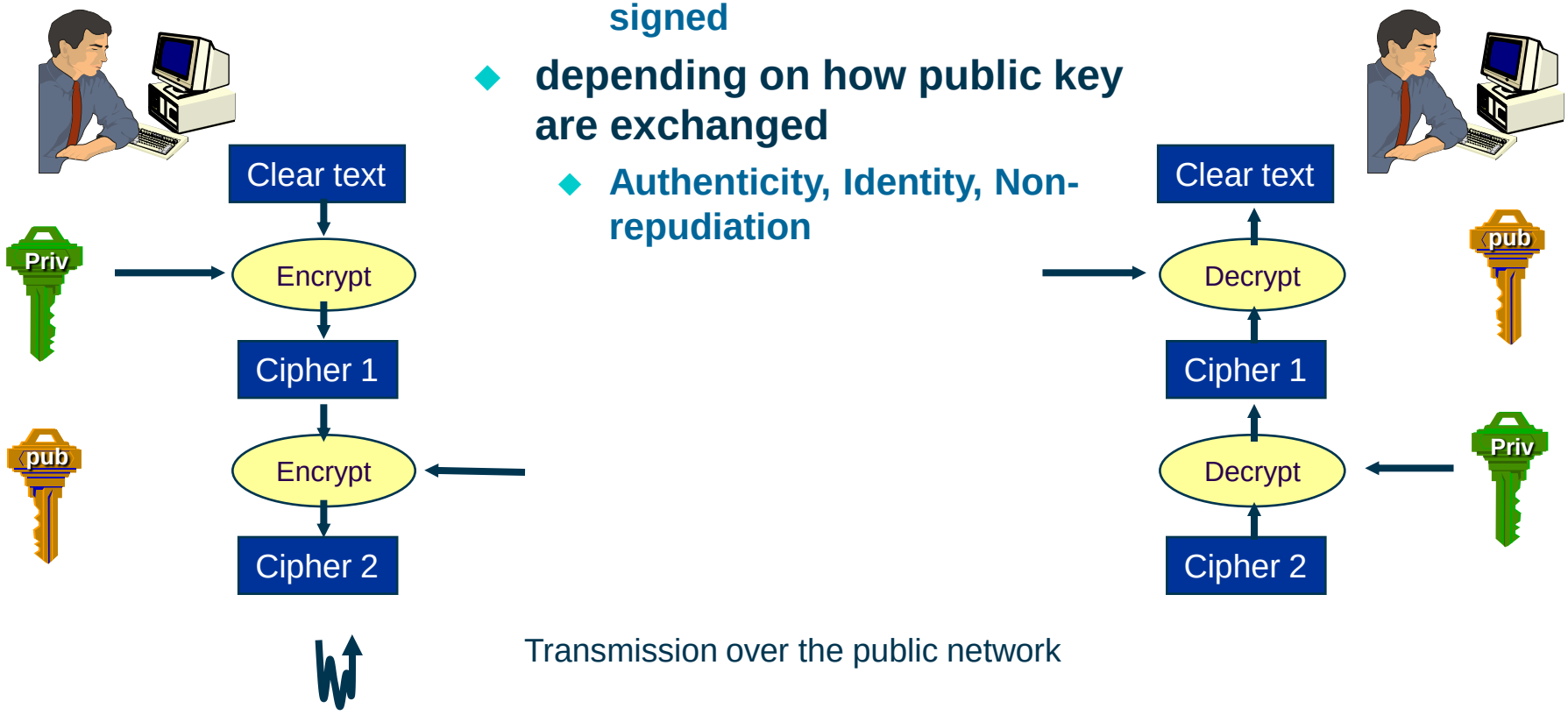


Verifying a Digital Signature



Example: SSL (simplified)

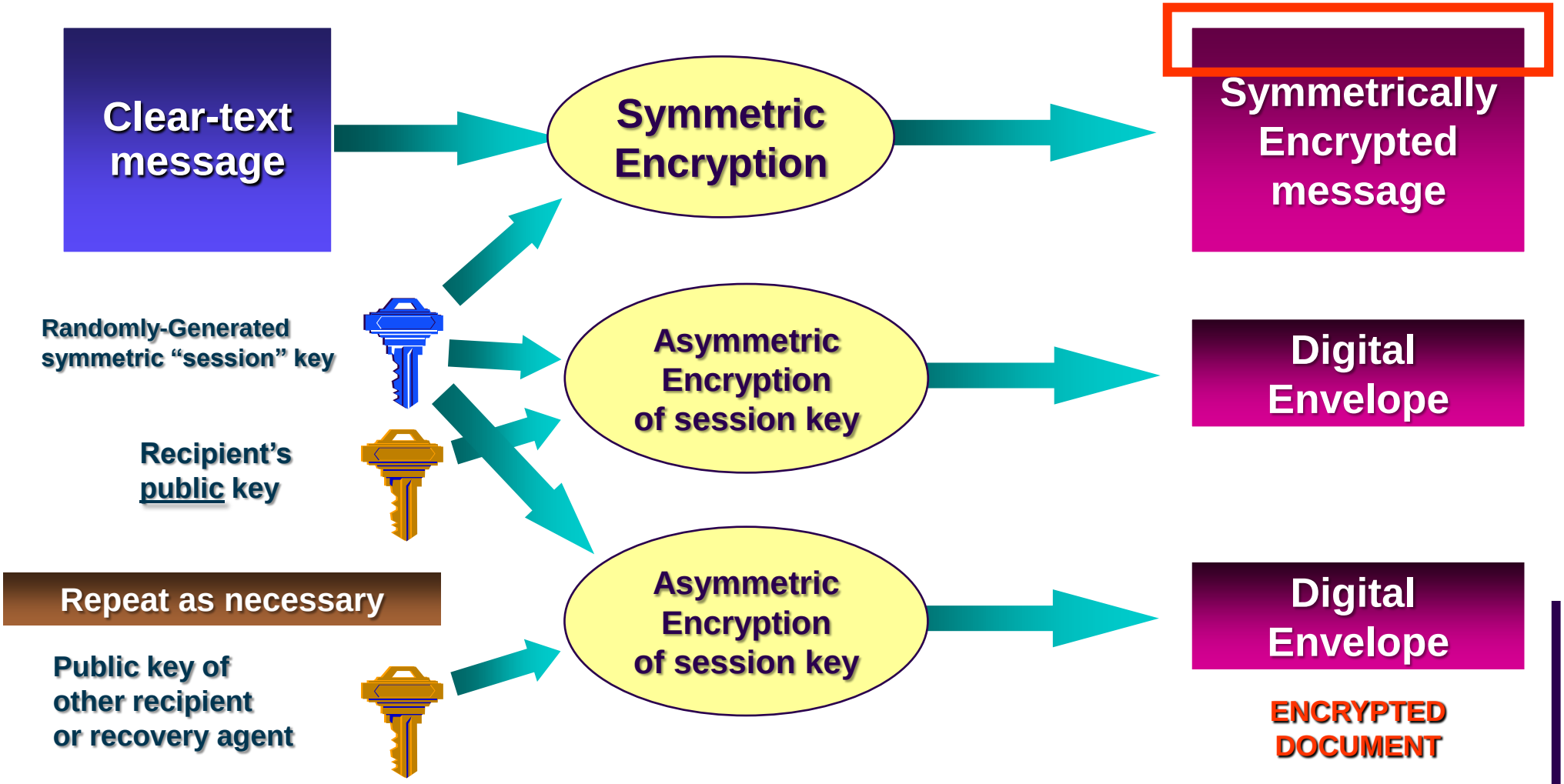
- Ensures confidentiality
  - And integrity if digitally signed
- depending on how public key are exchanged
  - Authenticity, Identity, Non-repudiation



Transmission over the public network

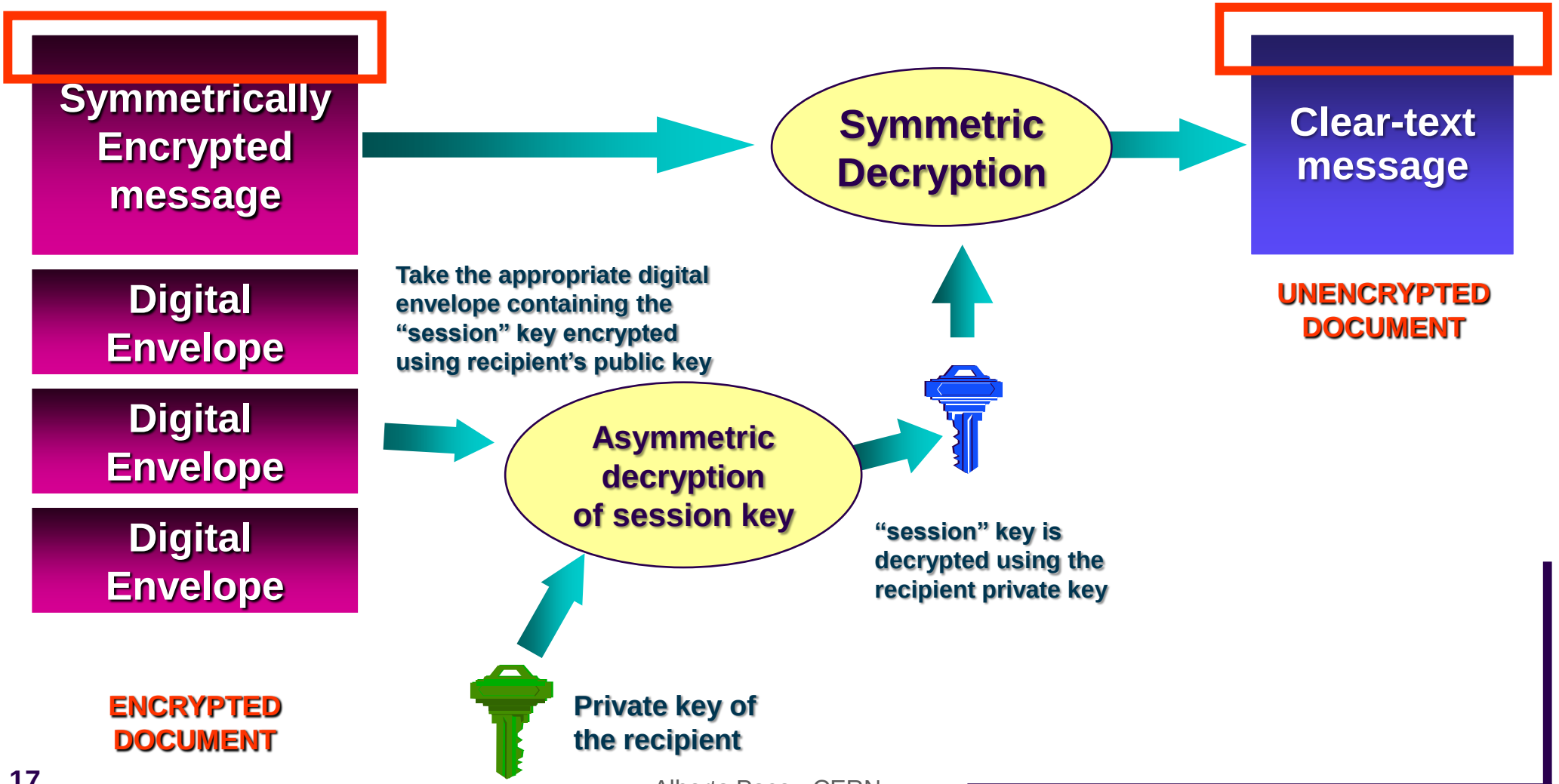
CERN School of Computing

Real World: Hybrid Encryption (typical for encrypted file storage)



CERN School of Computing

Real World: Hybrid Decryption



Cryptography Security

- **Kerckhoff's Principle**
  - The security of the encryption scheme must depend only on the secrecy of the key and not on the secrecy of the algorithms
- **The algorithms should be known and published**
  - They should have resisted to hacking for quite some time
  - They are all based on the fact that some calculations are difficult to reverse (probabilistic impossible)
- **But design and key length matter (brute force attacks)**
  - This means that DES, 3DES, AES , RSA, ECC, MD5, SHA are not immune to attacks
  - They all have a certain strength you should be aware of

Part 2: An introduction to Public Key Infrastructure (PKI)

Alberto Pace

alberto.pace@cern.ch

CERN Data Management Group

CERN School of Computing

Is cryptography enough ?

- **We just showed that cryptography solves the problem of confidentiality, Integrity, (Authenticity, Identity, Non-repudiation)**
  - How do we share secrets (symmetric encryption) and public keys (asymmetric encryption) safely on the internet ?
- **Problem …**
  - Michel creates a pair of keys (private/public) and tells everyone that the public key he generated belongs to Alice
  - People send confidential stuff to Alice
  - Alice cannot read as she is missing the private key to decrypt …
  - Michel reads Alice's messages
- **Except if people have met in some private place and exchanged a key, they'll need help from a third party who can guarantee the other's identity.**
  - PKI is one technology to share and distribute public keys (asymmetric encryption)
  - Kerberos another technology to share and distribute shared secrets (symmetric encryption)

How to exchange Public Keys?

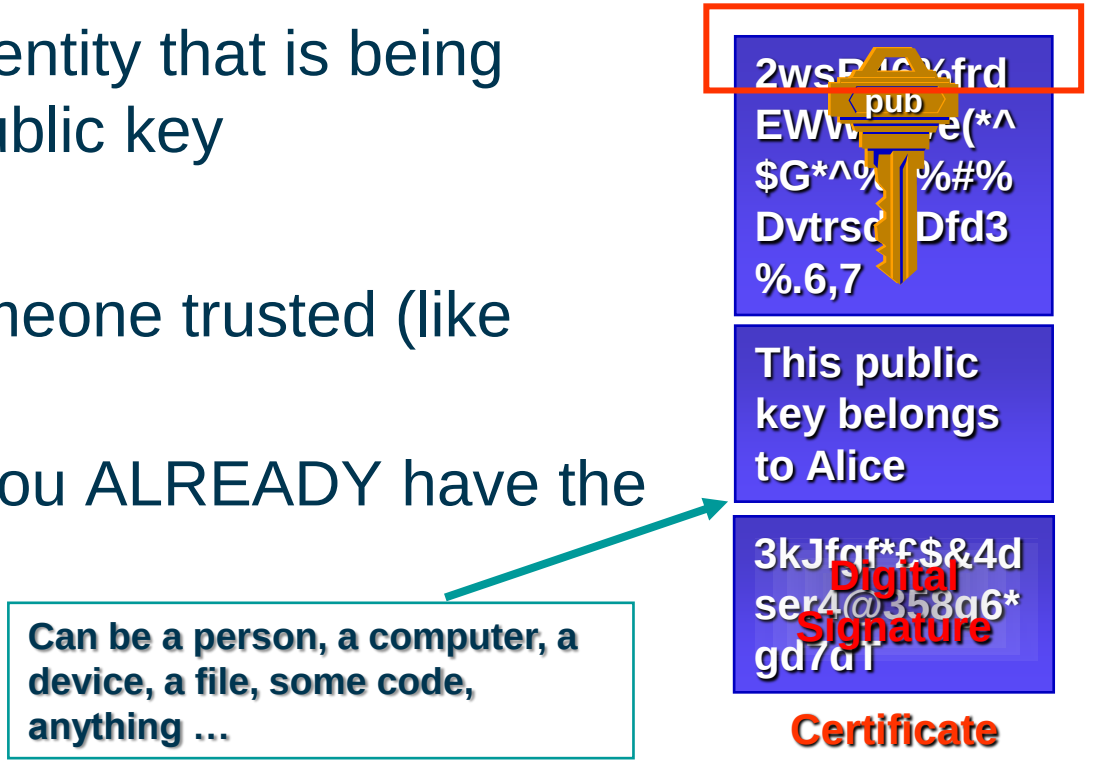
- **Two approaches:**
  - Before you use Alice's public key, call her or meet her and check that you have the right key
    - Have the public key sent to you in a USB stick or registered mail (if you trust registered mail)
    - You can use the telephone (if you trust the telephone)
  - Get someone you already trust to certify that the key really belongs to Alice
    - By checking for a trusted digital signature on the key
    - That's were certificates play a role

PKI = Public Key Infrastructure

- **"A technology to implement and manage E-Security" A. Nash, "PKI", RSA Press**
- **My definition of PKI**
  - "Public Key Infrastructure provides the technologies to enable practical distribution of public keys"
  - Using CERTIFICATES
- **PKI is a group of solutions for :**
  - Key generation
  - key distribution, certificate generation
  - Key revocation, validation
  - Managing trust

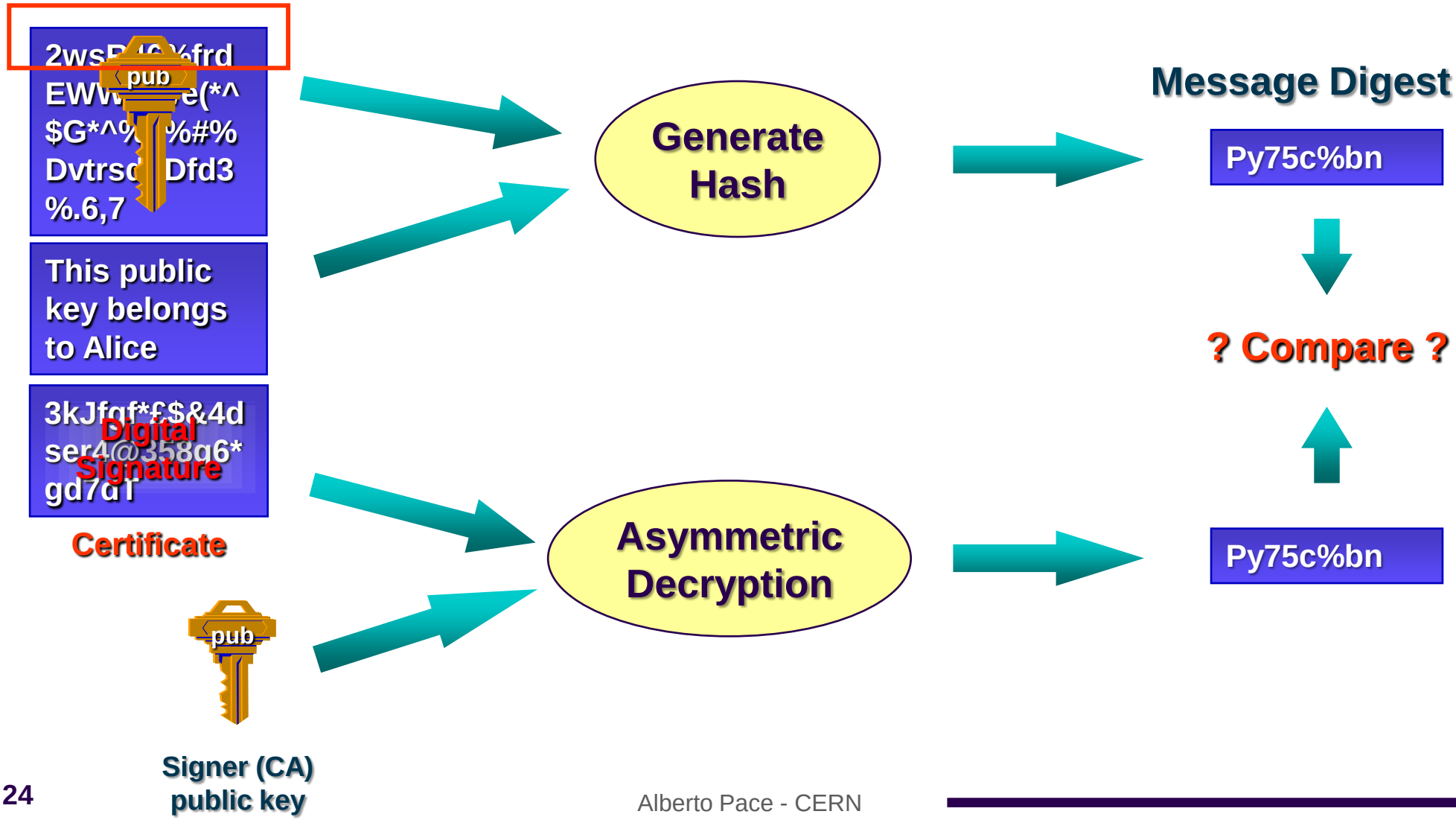
What is a Certificate ?

- **The simplest certificate just contains:**
  - A public key
  - Information about the entity that is being certified to own that public key
- **… and the whole is**
  - Digitally signed by someone trusted (like your friend or a CA)
  - Somebody for which you ALREADY have the public key

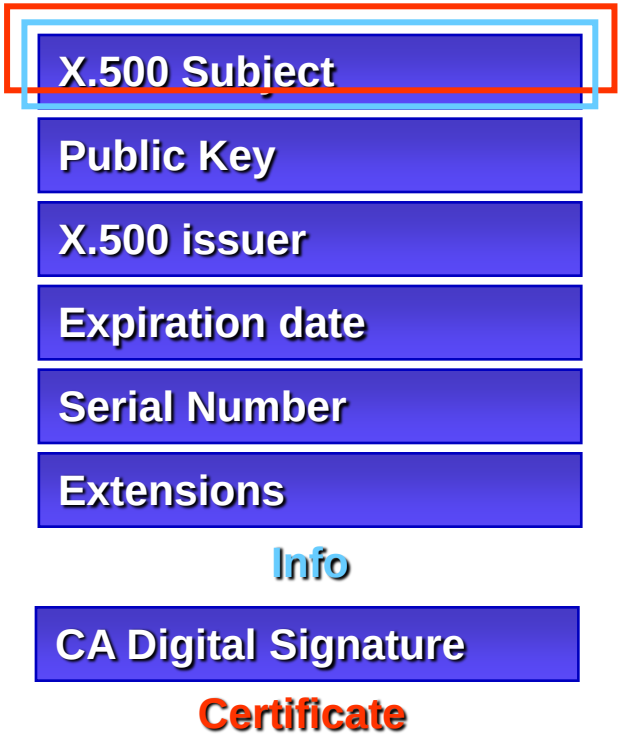


CERN School of Computing

Verifying a Certificate



X.509 Certificate (simplified)



Who is the owner, CN=Alice,O=CERN,C=CH

The public key or info about it

Who has signed, O=CERN,C=CH

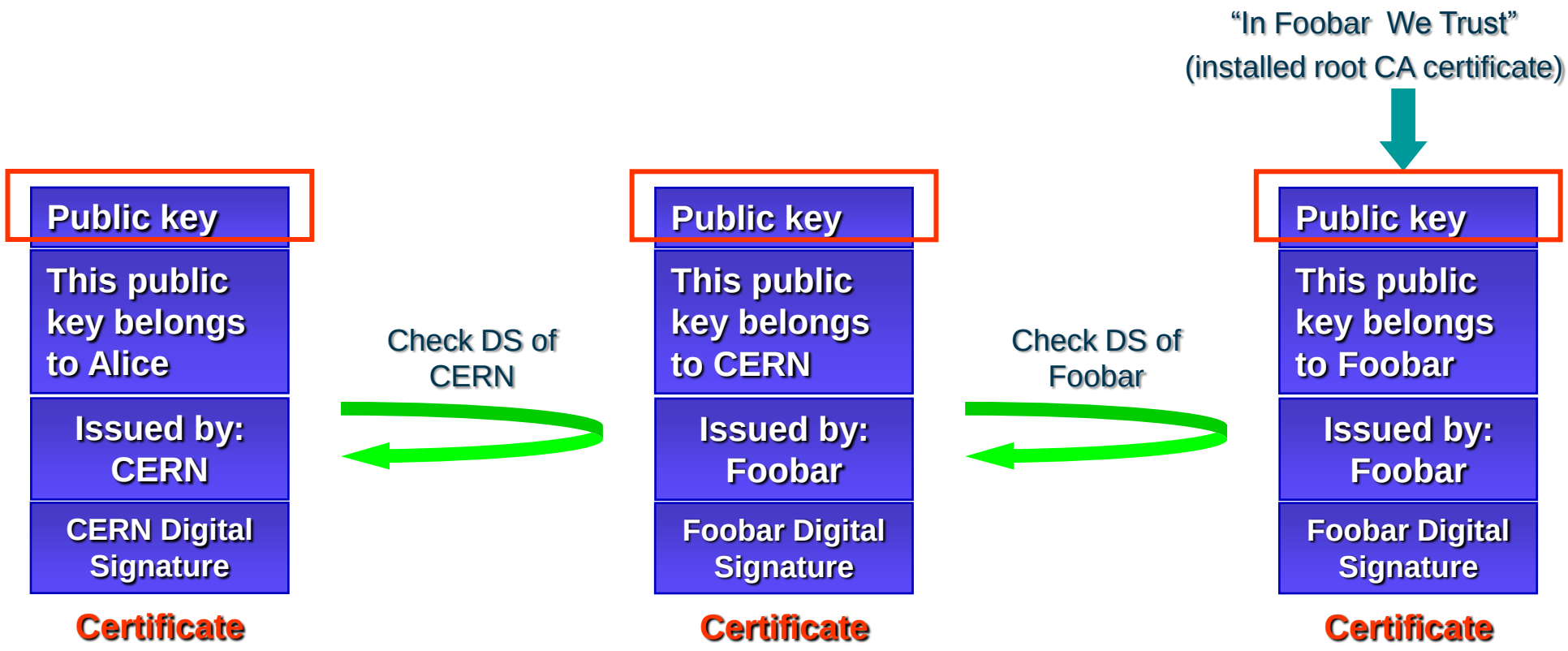
See later why expiration date is important

Additional arbitrary information

… of the issuer, of course

Certificate Validation

- **When checking the digital signature you may have to "walk the path" of all subordinate authorities until you reach the root**
  - Unless you explicitly trust a subordinate CA



Authentication with Certificates

- **Owning a Certificate of Alice does not mean that you are Alice**
  - Owning a Certificate does not imply you are authenticated
- **How would you verify that the person who comes to you pretending to be Alice and showing you a certificate of Alice is really Alice ?**
  - You have to challenge her !
  - Only the real Alice has the private key that goes in pair with the public key in the certificate.

Authentication with Certificates

- **Bob gets Alice's certificate**
- **He verifies its digital signature**
  - He can trust that the public key really belongs to Alice
  - But is it Alice standing if front of him, or is that Michel ?
- **Bob challenges Alice to encrypt for him a random phrase he generated ("I like green tables with flowers")**
- **Alice has (if she is the real Alice) the private key that matches the certificate, so she responds ("deRf35D^&#dvYr8^*$@dff")**
- **Bob decrypts this with the public key he has in the certificate (which he trusts) and if it matches the phrase he just generated for the challenge then it must really be Alice herself !**

Where should certificates be stored

- **Certificates can be stored anywhere**
  - Store them in a file or a “dumb” memory-only smartcard
  - You can publish them in your LDAP directory
- **No need to protect Certificates**
  - ... from being tampered as they are digitally signed
  - ... them from being read as they contain only public information
- **Private keys that match the public key are confidential**
  - Loosing the private key = Loosing the identity
- **Private keys should be stored in (at least) ...**
  - computers files, protected by pass phrases
  - OS protected storage
  - smartcards

CERN School of Computing

Certificates on Smartcards

- **A "bad" smartcard is only a dumb memory chip**
  - Containing the Certificate and the private key
  - Both readable: You must trust the machine reading your smartcard
  - Better than saving everything to a file
- **A "good" smartcard is more than a memory chip**
  - Contains the Certificate, readable
  - Contains the private key but not readable from outside. However it exposes a mechanism to challenge the knowledge of the private key by allowing the encryption of random strings using the private key
- **A "very good" smartcard**
  - May request the user to know a PIN code to execute any encryption request
  - (of course, now you have to protect the PIN code)
  - May support biometric recognition instead of the pin code

Certificate Revocation

- **(Private) keys get compromised, as a fact of life**
- **You or your CA issue a certificate revocation certificate**
  - Must be signed by the CA, of course
- **And you do everything you can to let the world know that you issued it. This is not easy**
  - Certificate Revocation Lists (CRL) are used
  - They require that the process of cert validation actively checks the CRL and keep it up-to-date
  - It is a non scalable process
  - Many people disable this function
- **This explains why**
  - Every certificate has an expiration date
  - short expiration policies are important

CERN School of Computing

Revoked certificates can be trusted

- **Alice creates a document on March 29th**
- **She signs it and sends it to Bob on April 8th**
- **On May 18th, she loses her private key and her certificate is revoked and published on the CRL**
- **Can Bob still trust the document as belonging to Alice ?**
  - YES
- **What if Bob would have received on June 29th the document dated March 29th, signed by Alice?**
  - NO
- **So …**
  - You can trust documents signed with revoked keys only if the date at which the document was signed is before the revocation date and it is certified by a trusted source (clearly not the revoked certificate entity)

CERN School of Computing

Traditional Human authentication versus Certificate authentication

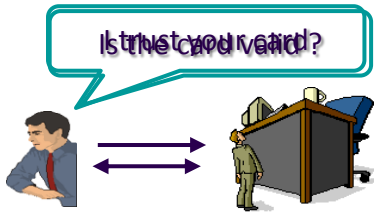
- High Security Entrance
- Immigration
- Authentication for payments

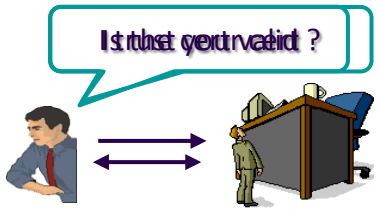
CERN School of Computing

A comparison with real life

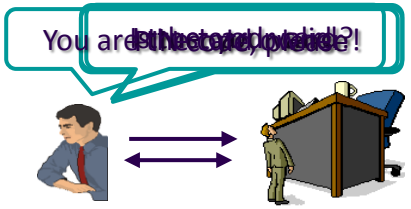
Motorway Toll (2.10 €)

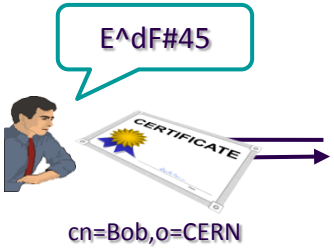
Shirt (20 €)

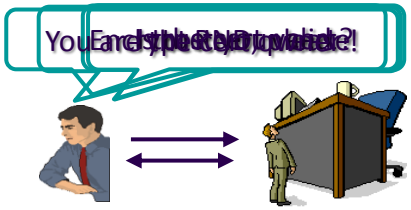




Notebook (1000 €)





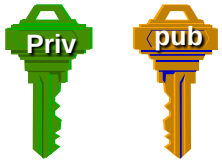


It is all about managing risks !

PKI Deployment

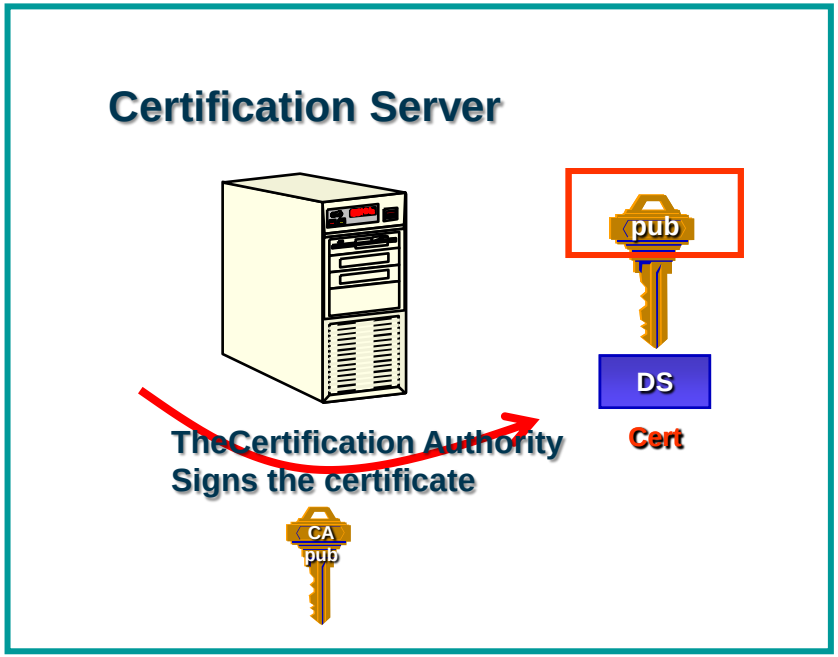
Certification Authority

User generates a key pair



Public key is submitted to CA for certification

Certificate is sent to the user



Certification Authority

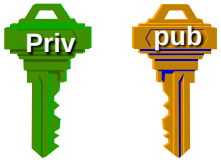
Certificate Authority Services

- **When deploying a Certificate Authority, you need to make an important decision:**
  - Use an external, well known CA
    - Your certificates will be universally recognised but you are dependent on the trustworthiness of the CA
    - You pay (a lot of $$)
  - Establish your own CA
    - Only partners who have explicitly trusted your CA recognise your certificates but you are in full control
- **You can also outsource CA services**
  - Not an economic viable option for large HEP labs

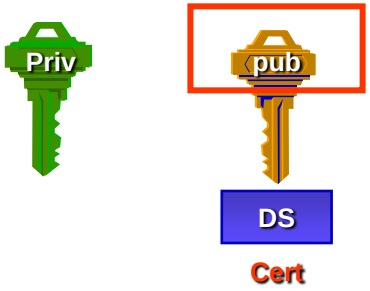
And there is more …

User request a certificate to CA

CA generates a key pair



Certification Server



On which grounds will you sign or not ?

Will you sign every requests ?

Identity Management Process

- **Before signing, you need to verify what you are signing**
- **You need to authenticate users by something other than certificates**
  - Otherwise Michel can get a valid certificate for Alice and her private key !
- **The strength of your verifications will define the class of the certificate you issue**

Certificate Classes

- **A Class 2 digital certificate is designed for people who publish software as individuals**
  - Provides assurance as to the identity of the publisher
- **A Class 3 digital certificate is designed for companies and other organizations that publish software**
  - Provides greater assurance about the identity of the publishing organization
  - Class 3 digital certificates are designed to represent the level of assurance provided today by retail channels for software
  - An applicant for a Class 3 digital certificate must also meet a minimum financial stability level

Social Problem

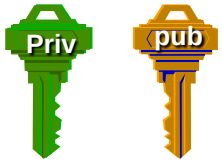
- **Real-life "certificates" are well understood**
  - What do you trust more: a national passport or a membership card of the video club rental ?
- **Digital certificates are a long way from public understanding**
  - Is Verisign Class 1 better or worse than Class 5 ?
  - What about BT Class 2 versus Thawte Class 3?

Storing Certificates and Keys

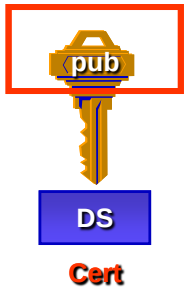
- **Certificates need to be stored so that anyone can obtain them**
  - This is not an issue. Certificates are “public”
- **Do we need to store private Keys for data recovery purposes ?**
  - Endless discussions on this topic
  - This weakens the system, but may be a necessity
- **This is a function of most certificate servers offer**
  - Those servers are also responsible for issuing, revoking, signing etc. of certs
- **But this requires the certificate server to generate the key pairs**

Example (no key recovery)

User generates a key pair



Public key is submitted to CA for certification



Certification Server

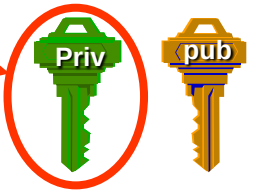
Certificate is sent to the user

Example (with key recovery)

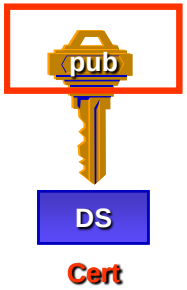
User request a certificate to CA

This model allows key recovery

CA generates a key pair



CA generates certificate



Certification Server

Private Key and Certificate are sent to the user

Current Strength Recommendations

- Your infrastructure should be ready to strengthen these at any time

	Minimum	Recommended
Symmetric Key	96 bits (avoid DES as it can do only 56, instead use AES-Rijndael or RC5)	256 bits (Rijndael, RC5 128bits, not DES)
Asymmetric Key	1024 (RSA)	4096 (RSA)
Hash: SHA/MD5	128 bits (not 64 bits)	256 bits or more
Cert Classes	Class 2	Class 3 at least

Is PKI relevant? Who uses PKI ?

- Web's HTTP and other protocols (SSL)
- VPN (PPTP, IPSec, L2TP…)
- Email (S/MIME, PGP, Exchange KMS)
- Files (PGP, W2K EFS, and many others)
- Web Services (WS-Security)
- Smartcards (Certificates and private key store)
- Executables (Java applets, .NET Assemblies, Drivers, Authenticode)
- Copyright protection (DRM)
- …

Part 3: An introduction to Kerberos

Alberto Pace

alberto.pace@cern.ch

CERN Data Management Group

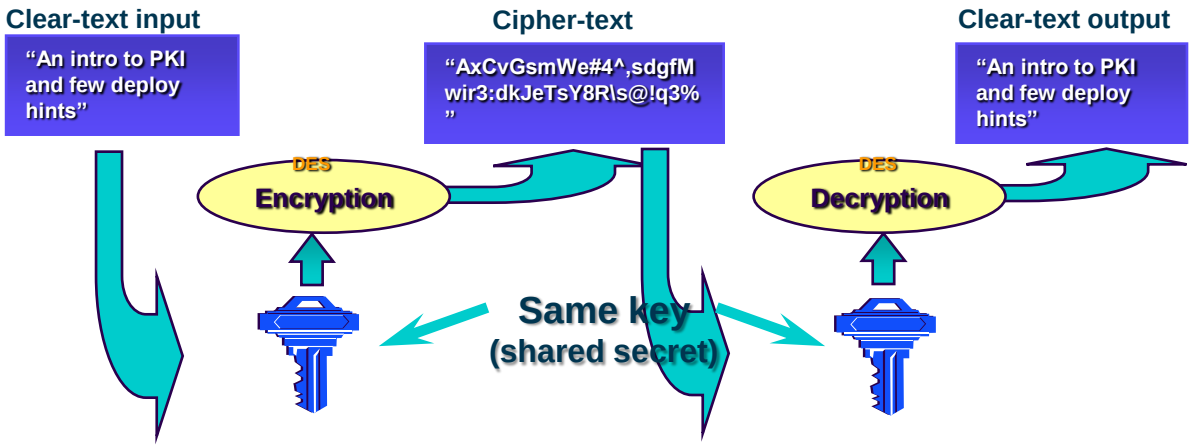
Kerberos gets its name from the mythological three headed dog that guards the entrance to Hell

An alternate technology to PKI

- **Identical goals of PKI**
- **Advantages:**
  - Simpler to manage, keys managed automatically, Users understand it better
  - Forwardable authentication easier to implement
- **Disadvantages**
  - Cross Domain Authentication and Domain Trusts more difficult to implement
  - Must be online

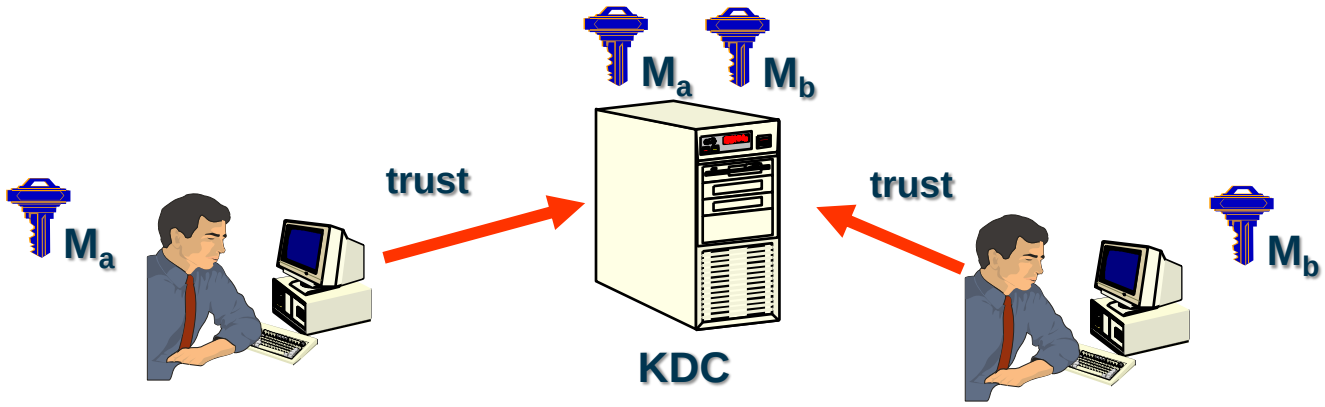
Kerberos Basics

- Kerberos is an authentication protocol based on conventional cryptography
- it relies on *symmetrical* cryptographic algorithms that use the same key for encryption as for decryption
  - Different from PKI !



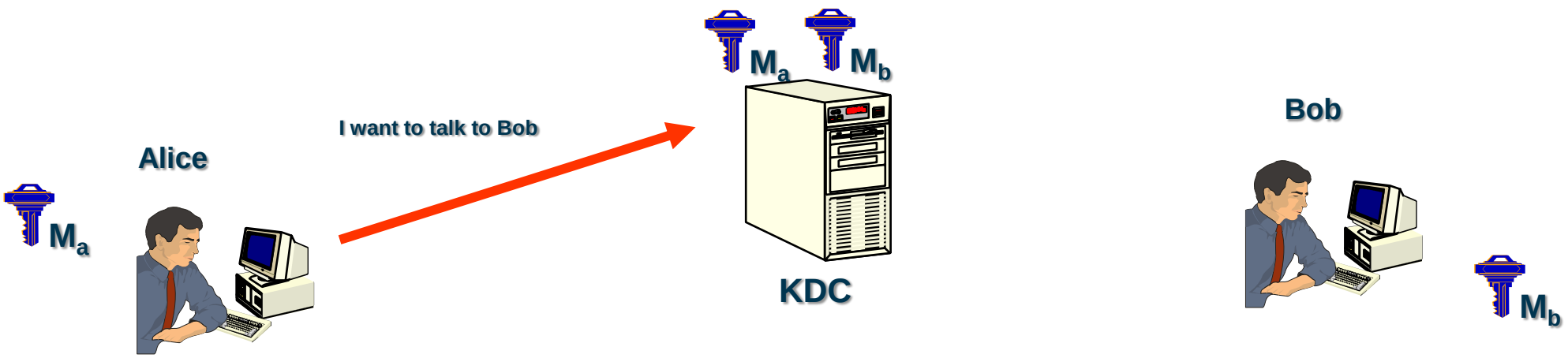
Basic principles

- There is a trusted authority known as the Key Distribution Center (KDC) which is the keeper of secrets.
- Every user shares a secret password with the KDC
- The secret master key is different for each user
  - Two users have no direct way of verifying each other's identity
- The job of the KDC is to distribute a unique session key to each pair of users (security principals) that want to establish a secure channel.
  - Using symmetric encryption
- Everybody trusts the KDC



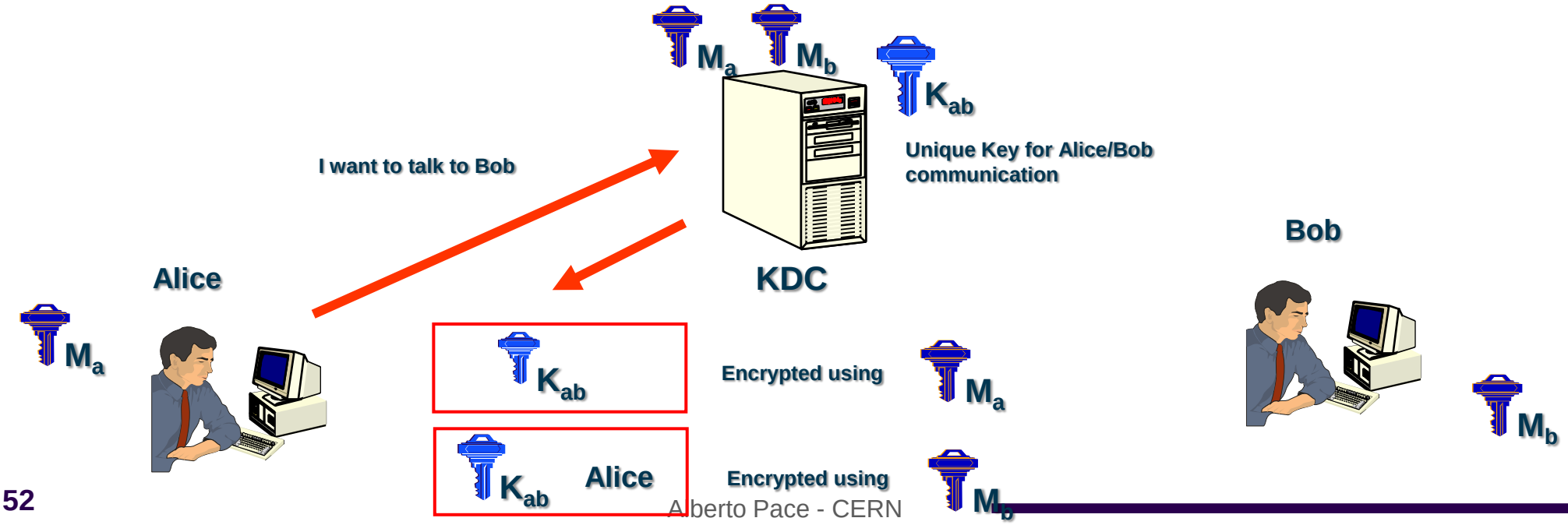
A (simplified) Kerberos session

- Alice wants to communicate with Bob
  - bob could be a server or a service
- Alice can communicate securely with the KDC, using symmetric encryption and the shared secret (Master Key)
- Alice tells the KDC that she wants to communicate with Bob (known to the KDC)



(simplified) Kerberos session 2

- The KDC generates a unique random key for Alice and Bob (K_{ab})
- Two copies of K_{ab} are sent back to Alice.
  - The first copy is sent encrypted using Alice's master key
  - The second copy of K_{ab} is sent with Alice's name encrypted with Bob's master key. This is known as the “kerberos ticket"

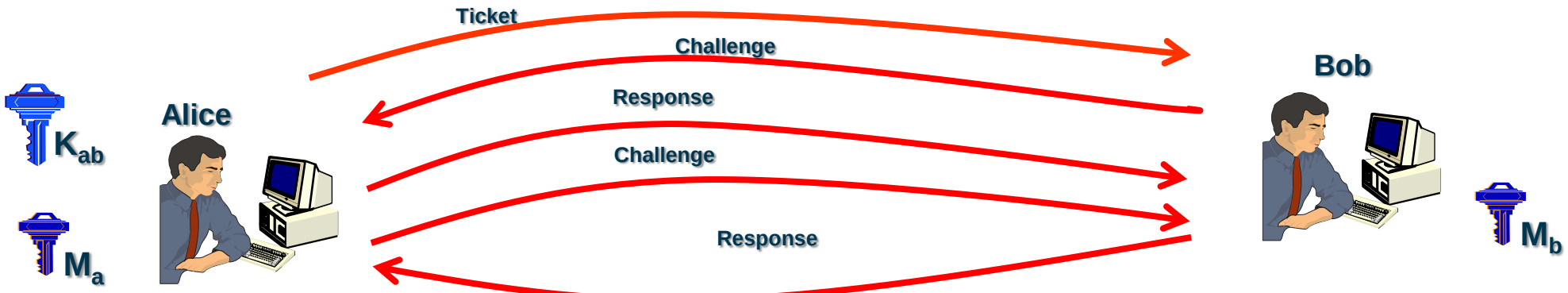


What is the ticket ?

- **The ticket is a message to Bob that only Bob can decrypt**
  - "This is your KDC. Alice wants to talk to you, and here's a session key that I've created for you and Alice to use. Only you, me and Alice know the value of K_{ab}, since I've encrypted it with your respective master keys. If your peer can prove knowledge of this key, then you can safely assume it is Alice."

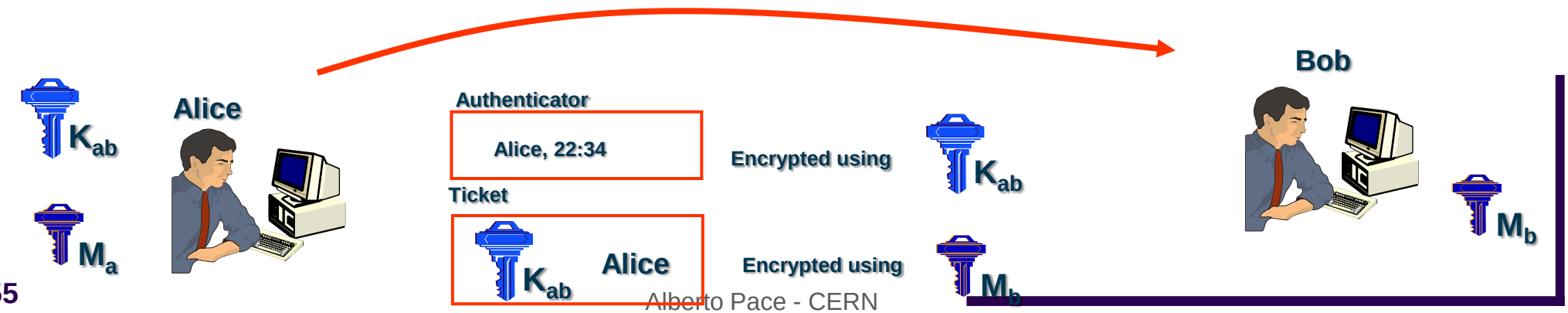
How authentication *could* be done

- Alice sends the ticket to Bob. Bob takes the ticket and decrypts K_{ab}
- Bob generates a "random phrase" (the challenge) that is sent back to Alice
- Alice encrypts the "random phrase" using K_{ab} and send the results to Bob
  - She proves that she knows K
- Bob has authenticated Alice
- The whole could be repeated to have Alice authenticating Bob
- But **Kerberos doesn't work that way !** It is smarter and anticipates the challenge by encrypting the "current time"



Kerberos authentication

- **Alice sends the ticket to Bob**
- **Alice must also proof that she knows K_{ab}**
  - She also sends her name and the current time, all encrypted with the session key K_{ab} (this is called the authenticator)
- **Bob takes the ticket, decrypts it, and pulls K_{ab} out. Then decrypts the authenticator using K_{ab}, and compares the name in the authenticator with the name in the ticket**
  - If the time is correct, this provides evidence that the authenticator was indeed encrypted with K_{ab}
  - Bob can also detect replays from attackers listening on the network where Alice, Bob, and the KDC are conversing
    - By rejecting authenticators using time already used
    - By rejecting authenticators using the wrong time

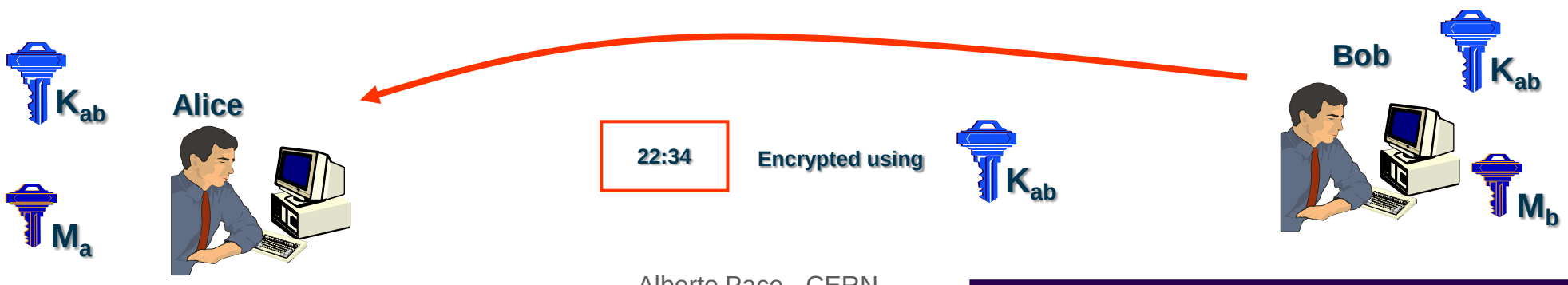


Kerberos authentication

- **It time is incorrect, bob reject the request**
  - with a hint of what his time is (Bob time isn't a secret)
- **If the time is correct …**
  - … it's probable that the authenticator came from Alice, but another person might have been watching network traffic and might now be replaying an earlier attempt. However, if Bob has recorded the times of authenticators received from Alice during the past “five minutes”, he can defeat replay attempts. If this authenticator yields a time later than the time of the last authenticator from Alice, then this message must be from Alice
  - This is why time synchronization is essential in kerberos and all KDC provides also time synchronization services
- **You can see this as a “challenge” on the knowledge of the shared secret (K_{ab}):**
  - “prove that you know K_{ab} by encrypting the current time for me”

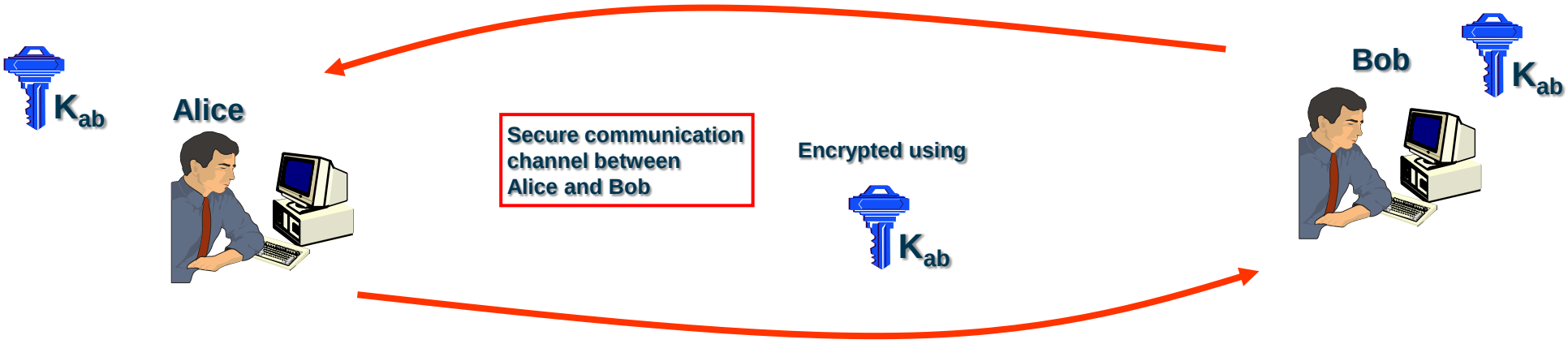
Mutual authentication

- **Alice has proved her identity to Bob**
- **Now Alice wants Bob to prove his identity as well**
  - she indicates this in her request via a flag.
- **After Bob has authenticated Alice, he takes the timestamp she sent, encrypts it with K_{ab}, and sends it back to Alice.**
- **Alice decrypts this and verifies that it's the timestamp she originally sent to Bob**
  - She has authenticated Bob because only Bob could have decrypted the Authenticator she sent
  - Bob sends just a piece of the information in order to demonstrate that he was able to decrypt the authenticator and manipulate the information inside. He chooses the time because that is the one piece of information that is sure to be unique in Alice's message to him



Kerberos Secure Communication

- Alice and Bob share now a unique secret K_{ab} that they use to communicate

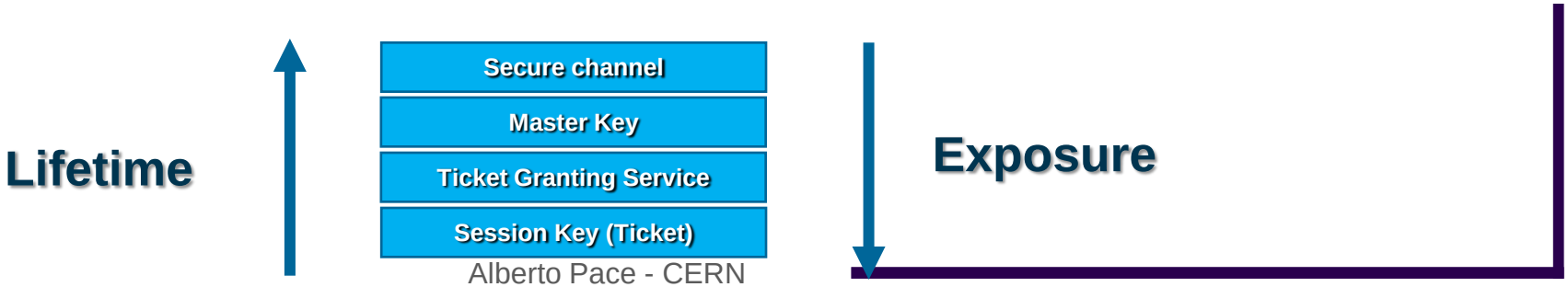


But real life is more complicated

- **Real Kerberos includes an extra step for additional security**
- **When Alice first logs in, she actually asks the KDC for what is called a "ticket granting ticket", or TGT.**
- **The TGT contains the session key (K_{ak}) to be used by Alice in her communications with the KDC throughout the day.**
  - This explains why when the TGT expires you have to renew it
- **So when Alice requests a ticket for Bob, she actually sends to the KDC her TGT plus an authenticator with her request.**
- **The KDC then sends back the Alice/Bob session key K_{ab} encrypted with K_{ak}**
  - as opposed to using Alice's master key as described earlier
  - Alice doesn't even need to remember her master key once she receives the TGT (unless she wants automatic TGT renewal).

Kerberos Key Hierarchy

- The session key (or short-term key). A session key is a secret key shared between two entities for authentication purposes. The session key is generated by the KDC. Since it is a critical part of the Kerberos authentication protocol, it is never sent in the clear over a communication channel: It is encrypted using the Ticket Granting Services key
- The Ticket Granting Services key (medium-term key). A secret key shared between each entities and the KDC to obtain session keys. It is never sent in the clear over a communication channel: It is encrypted using the master key.
- The master key (or long-term key). The master key is a secret key shared between each entity and the KDC. It must be known to both the entity and the KDC before the actual Kerberos protocol communication can take place. The master key is generated as part of the domain enrollment process and is derived from the creator's (user, machine, or service) password. The transport of the master key over a communication channel is secured using a secure channel.
- The secure channel. The secure channel is provided by the master key shared between the workstation you're working on and the KDC. In this case the master key is derived from the workstation's machine account password.



CERN School of Computing

Kerberos ticket in real life

	Field name	Description
	tkt-vno	Version number of the ticket format. In Kerberos v.5 it is 5.
	Realm	Name of the realm (domain) that issued the ticket. A KDC can issue tickets only for servers in its own realm, so this is also the name of the server's realm
	Sname	Name of the server.
	Flags	Ticket options
	Key	Session Key
	Crealm	Name of the client's realm (domain)
	Cname	Client's name
	Transited	Lists the Kerberos realms that took part in authenticating the client to whom the ticket was issued.
	Starttime	Time after which the ticket is valid.
	Endtime	Ticket's expiration time.
	renew-till	(Optional) Maximum endtime that may be set in a ticket with a RENEWABLE flag.
	Caddr	(Optional) One or more addresses from which the ticket can be used. If omitted, the ticket can be used from any address.
	Authorization-data	(Optional) Privilege attributes for the client. Kerberos does not interpret the contents of this field. Interpretation is left up to the service.

 : Fields encrypted using the session key of the recipient's TGT

PKI and Kerberos integration

Authentication Methods

- **Two technologies for authentication: Kerberos and X.509 Certificates (PKI)**
- **Both technologies have weak and strong points**
  - Distributed versus centralized management
  - Forwardable authentication
  - Offline authentication
- **Technology is different**
  - Asymmetric encryption with public/private key pairs versus symmetric encryption and shared secrets

Both technologies are here to stay

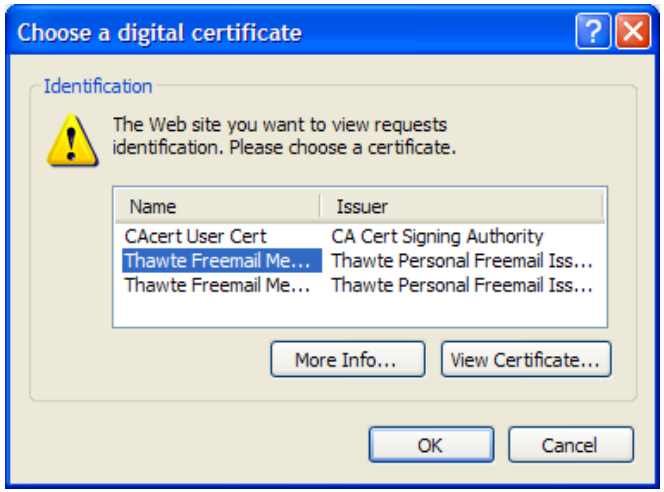
- Kerberos is used in Windows Domains and AFS
- PKI is used in all Grid related projects, with multiple certification authorities
- Multiple scenarios exist to integrate and interoperate the two technologies

PKI / Kerberos integration

- **A server in a Kerberos domain can authenticate users using certificates and map the "subject" of the certificate to any kerberos account (login name)**
  - "CN=Alberto Pace 8717;OU=GRID;O=CERN;C=CH" mapped to "pace"
- **The certificate-based authentication impersonates the corresponding kerberos account**
  - Requires the server to be "trusted" by the KDC

Example: Certificate Authentication mapped to Kerberos account

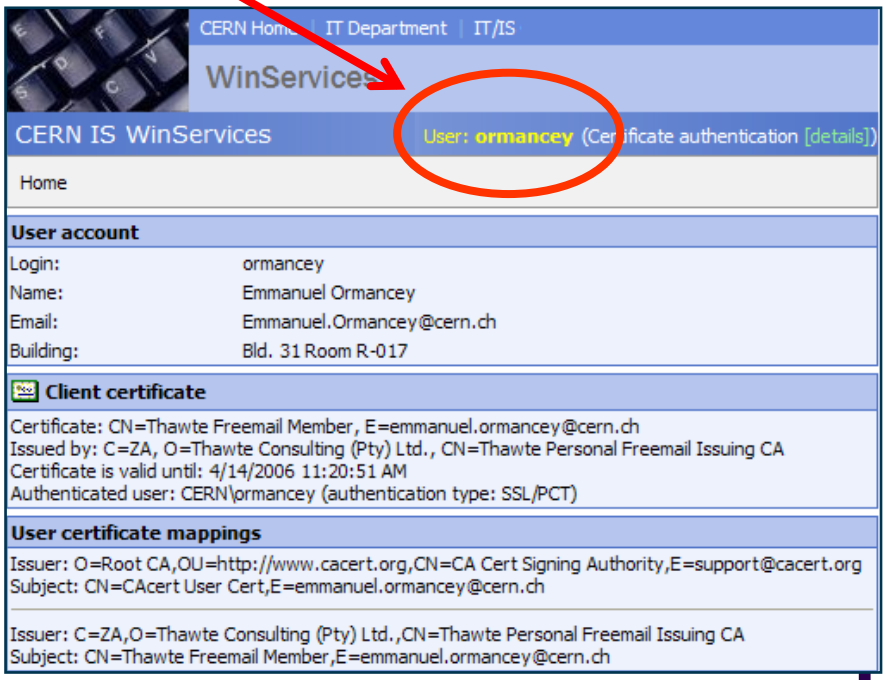
Connect to a website



The browser prompts to choose among the client certificates matching server requirement

→ Certificate authentication complete. →

The process in the web server impersonates the kerberos account



CERN School of Computing

Identity Management Architectures

- **Hand out kerberos passwords to users which they can use to obtain the certificate**

OR

- **Handout certificates to users that they can use to obtain their kerberos password**
  - You can handout smartcards
  - You can handout smartcards and prevent user to have knowledge of their kerberos password

Kerberos is the master, PKI the slave

PKI is the master, Kerberos the slave

Conclusion on PKI / Kerberos

- **Why both ?**
  - Provide a common authentication interface for all services, platform independent.
- **Careful thinking of the Master / slave architecture**
  - Both choices are secure but there are advantages and disadvantages for both cases

Identity Management

Alberto Pace
CERN, Information Technology Department
alberto.pace@cern.ch

CERN School of Computing

Computer Security

- **The present of computer security**
  - Bugs, Vulnerabilities, Known exploits, Patches
  - Desktop Management tools, anti-virus, anti-spam, firewalls, proxies, Demilitarized zones, Network access protection, …
- **This is no longer enough. Two additional aspects**
  - Social Engineering
    - "Please tell me your password"
    - Require corporate training plan, understand the human factor and ensure that personal motivation and productivity is preserved
  - Identity (and Access) Management

Discussed now

Definition

- **Identity Management (IM)**
  - Set of flows and information which are (legally) sufficient and allow to identify the persons who have access to an information system

More definitions

- **Identity and Access Management (IAM)**
- **Access Management**
  - For a given information system, the association of a right (use / read / modify / delete / …) and an entity (person, account, computer, group, …) which grants access to a given resource (file, computer, printer, room, information system, …), at a given time, from a given location
  - Access control and resources can be physical (specific location, door, room, …) or logical (password, certificate, biometric, token, …)

IAM Architecture

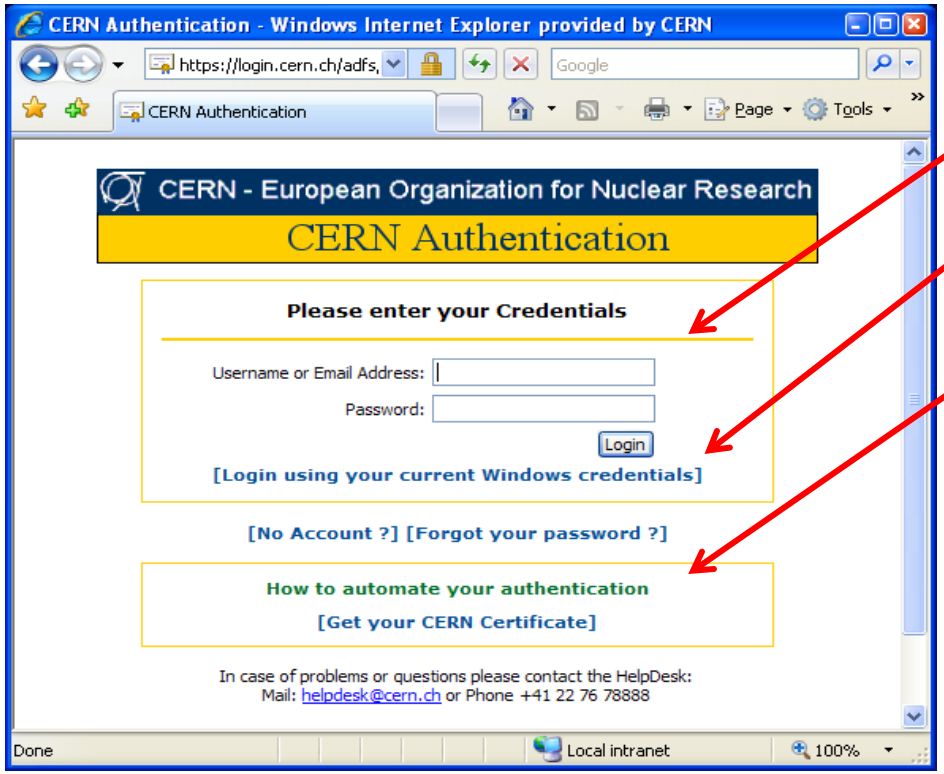
- **The AAA Rule. Three components, *independent***
- **Authentication**
  - Unequivocal identification of the person who is trying to connect.
  - Several technologies exist with various security levels (username / password, certificate, token, smartcard + pin code, biometry, …)
- **Authorization**
  - Verification that the connected user has the permission to access a given resource
  - On small system there is often the confusion between authorization and authentication
- **Accounting**
  - List of actions (who, when, what, where) that enables traceability of all changes and transactions rollback

More on IAM Architecture

- **Role Based Access Control (RBAC)**
  - Grant permissions (authorizations) to groups instead of person
  - Manage authorizations by defining membership to groups
- **Separations of functions**
  - granting permissions to groups (Role creation)
  - group membership management (Role assignment)
- **Be aware !**
  - RBAC should be a simplification
  - Keep the number of roles to a minimum

CERN School of Computing

Single Sign On Example



Username / Password

SSO using Windows Credentials

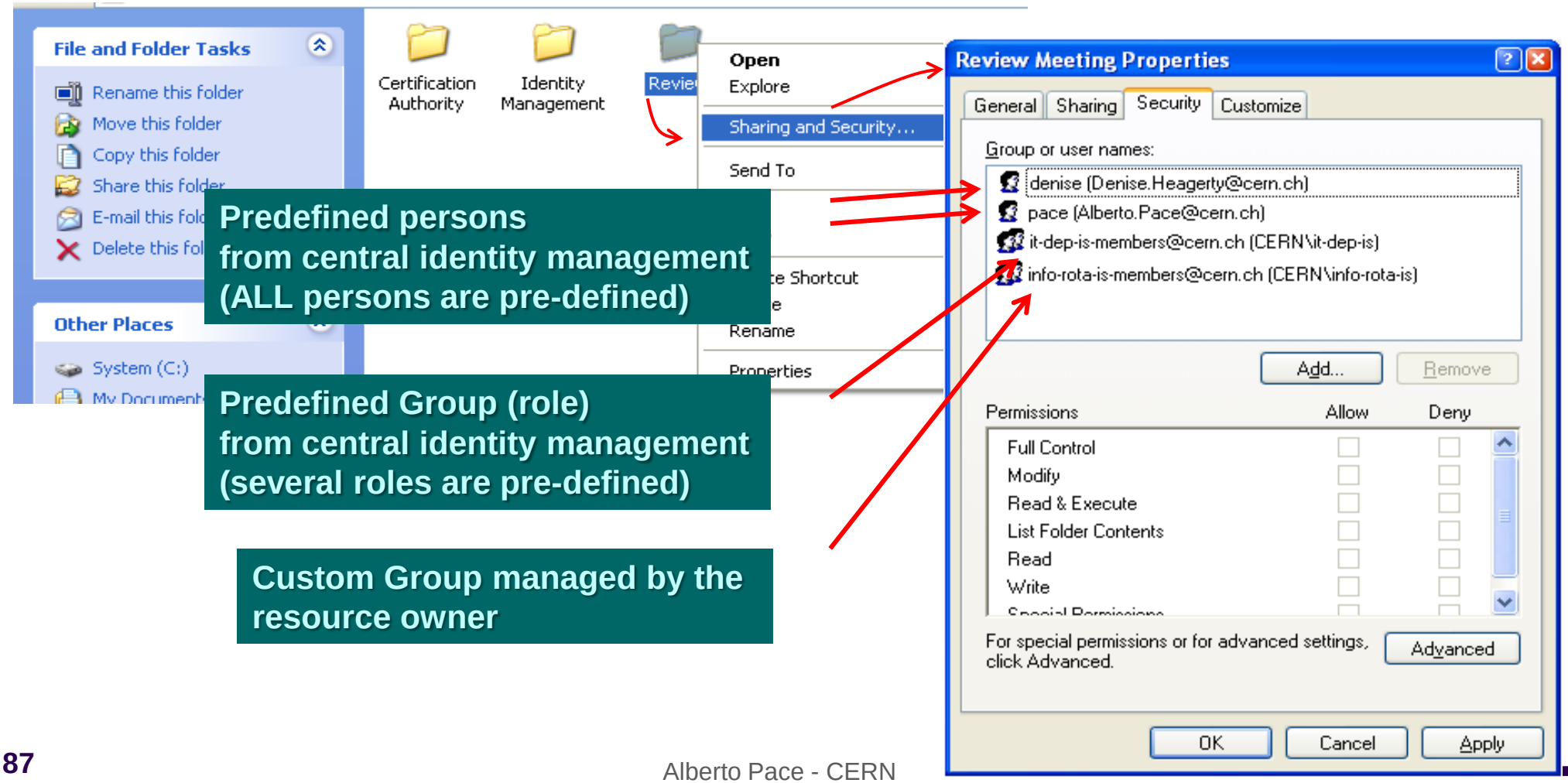
SSO using Grid Certificate

DEMO

- **Open a Windows hosted site:**
  - http://cern.ch/win
  - Click login, check user information
- **Open a Linux hosted site:**
  - http://shib.cern.ch
  - Check various pages
- **Go back to first site**
  - Click logout

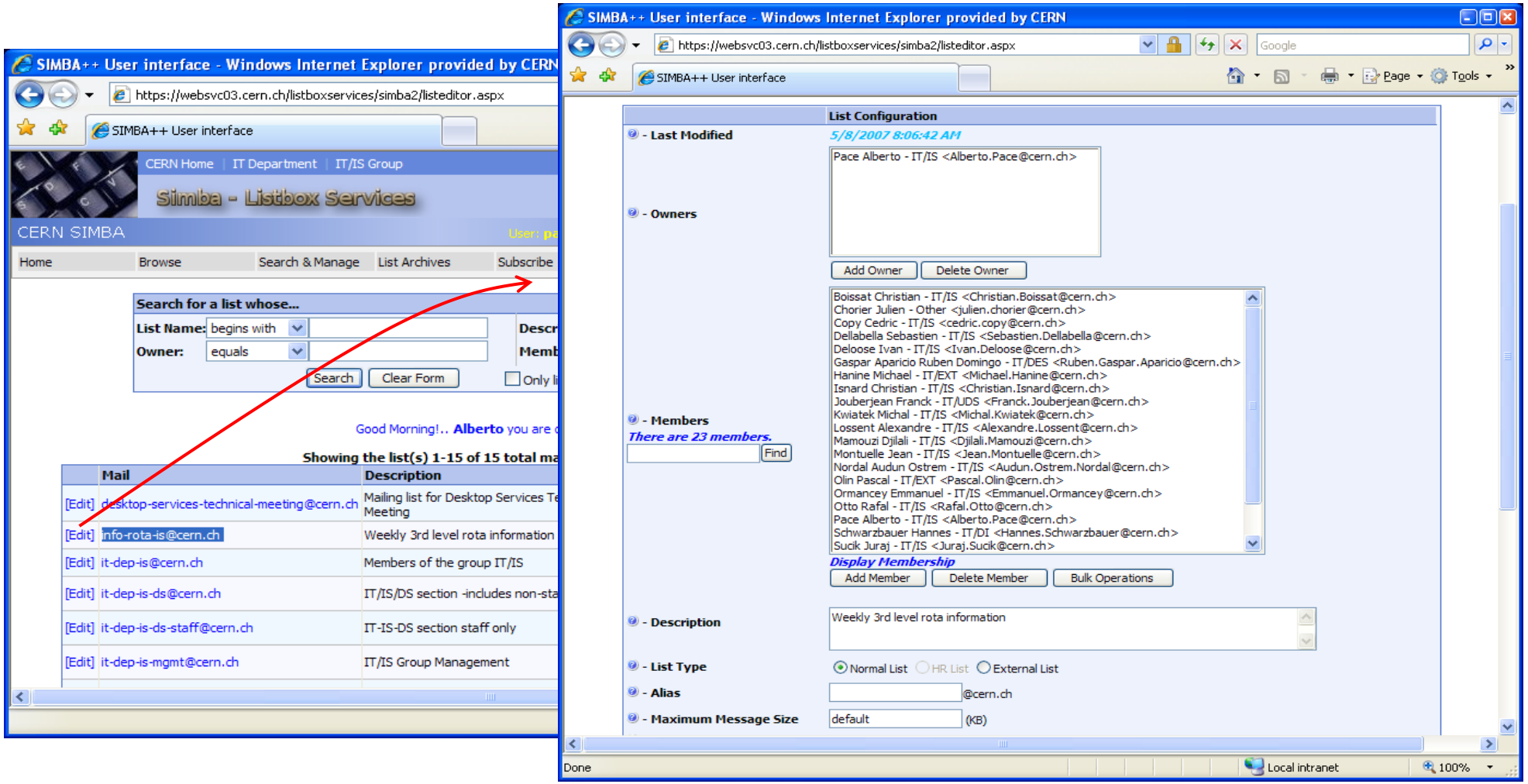
CERN School of Computing

Example



CERN School of Computing

Managing custom group example



Conclusion

- **Necessary to resist to pressure of having**
  - "Custom" solution for "special" users
  - Exception lists
- **Security in focus**
  - Complexity and security don't go together
- **Once identity management is in place …**
  - … you wonder why this was not enforced earlier

General Conclusion

- **Look for systems**
  - From well-know parties
  - With published algorithms
  - That have been hacked for a few years
  - That have been analysed mathematically
- **Do not "improve" algorithms yourself**
- **Apply security patches**
  - The technology is secure, but it is complex and leads to bugs in the various implementations
- **A managed infrastructure allows moving forward**
  - Trusted intranet applications, code signing, Antivirus, Secure E-mail, Secure Web, better spam fighting, anti flood mechanism, prevent DOS attacks, etc…

PKI References

- http://www.pkiforum.org/

- PKI: Implementing & Managing E-Security
  by Andrew Nash, Bill Duane, Derek Brink, Celia Joseph, Osborne, 2001
  http://www.amazon.com/exec/obidos/tg/detail/-/0072131233/ref=pd_sim_books_2/002-8363961-5776032?v=glance&s=books

CERN School of Computing

Additional info on Kerberos

- Miller, S., Neuman, C., Schiller, J., and J. Saltzer, "Section E.2.1: Kerberos Authentication and Authorization System," MIT Project Athena, Cambridge, MA, December 1987.
- Kohl, J., and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993.
- Linn, J., "The Kerberos Version 5 GSS-API Mechanism," RFC 1964, June 1996.
- Tung, B., Neuman, C., Wray, J., Medvinsky, A., Hur, M., and J. Trostle, "Public Key Cryptography for Initial Authentication in Kerberos," draft-ietf-cat-kerberos-pk-init.
- Neuman, C., Kohl, J. and T. Ts'o, "The Kerberos Network Authentication Service (V5)," draft-ieft-cat-kerberos-revisions-03, November 1998.
- Neuman, C., Kohl, J, and T. Ts'o, "The Kerberos Network Authentication Service (V5)," draft-ietf-cat-kerberos-revisions, November 1997.