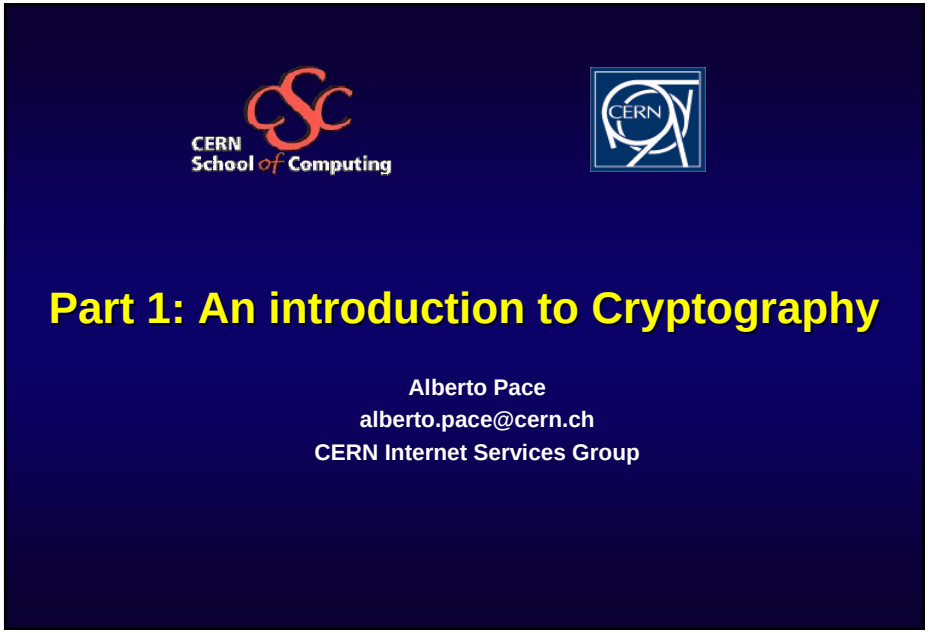
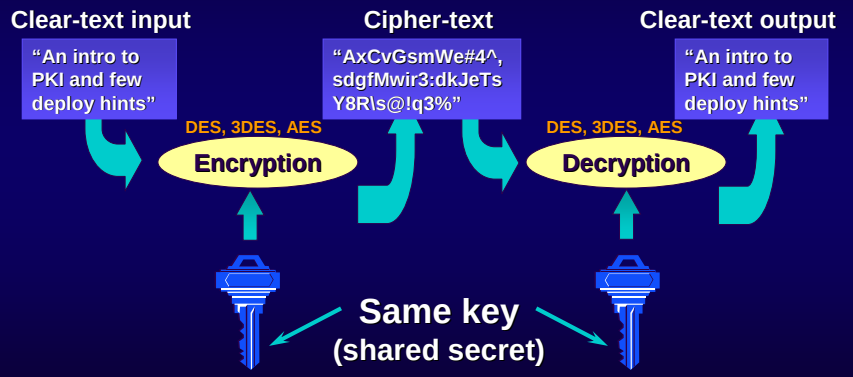

CERN
School of Computing
CERN
Computer Security
Alberto Pace
alberto.pace@cern.ch
CERN Internet Services Group


CERN
School of Computing
CERN
Part 1: An introduction to Cryptography
Alberto Pace
alberto.pace@cern.ch
CERN Internet Services Group

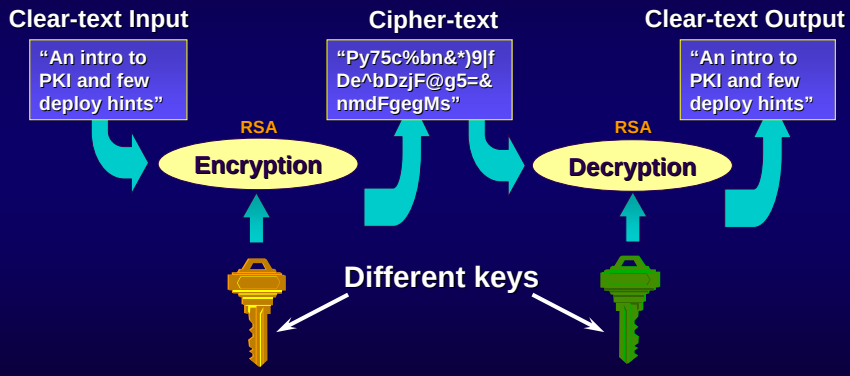
What does Cryptography solve?

- 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



Asymmetric Encryption

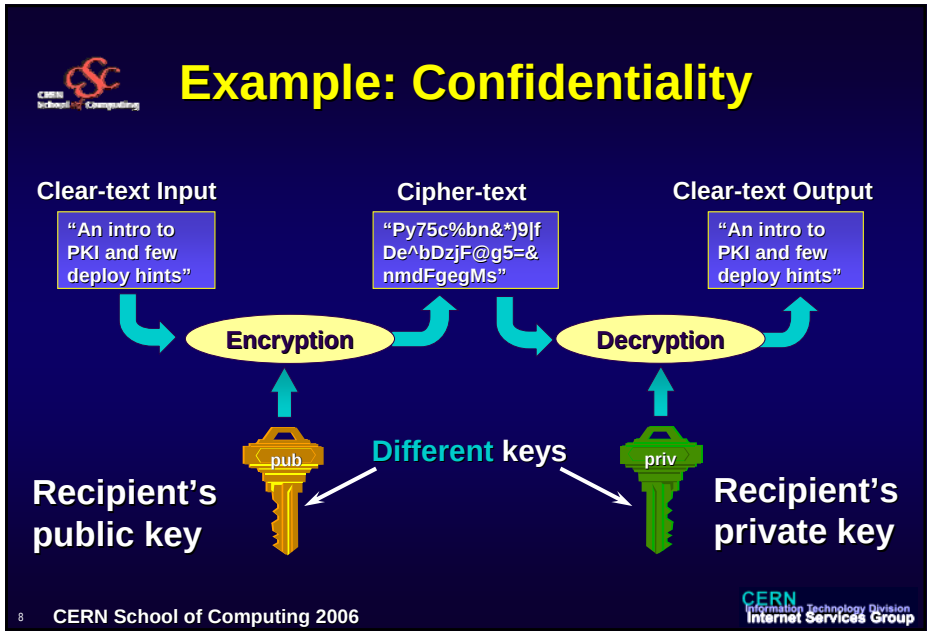


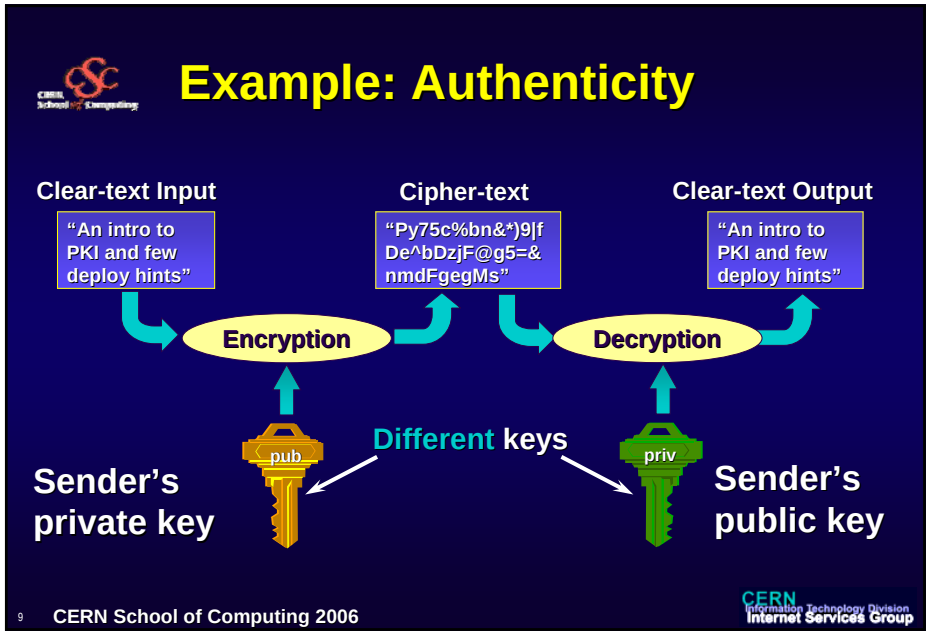
CERN Information Technology Division Internet Services Group

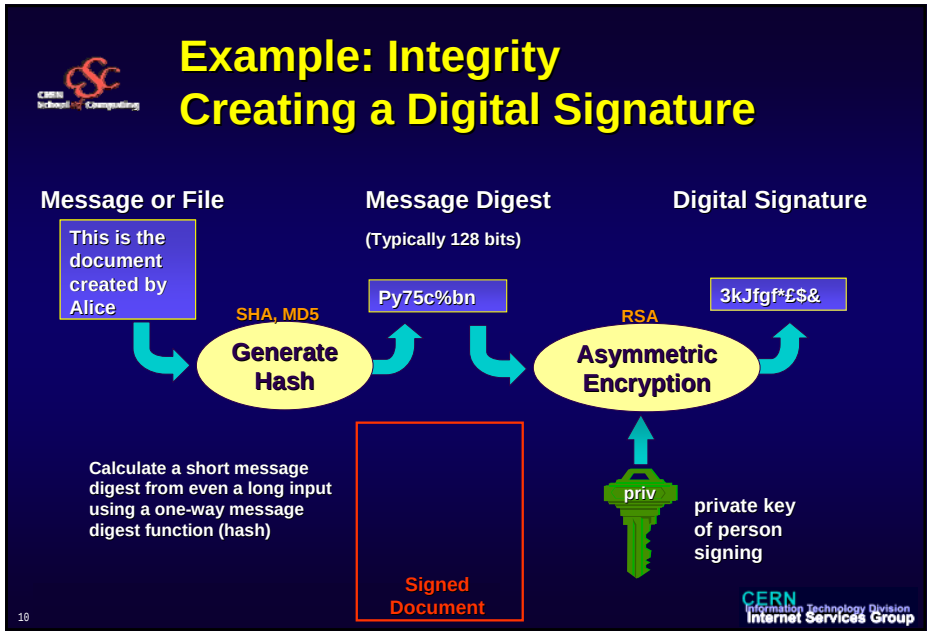
Asymmetric Encryption

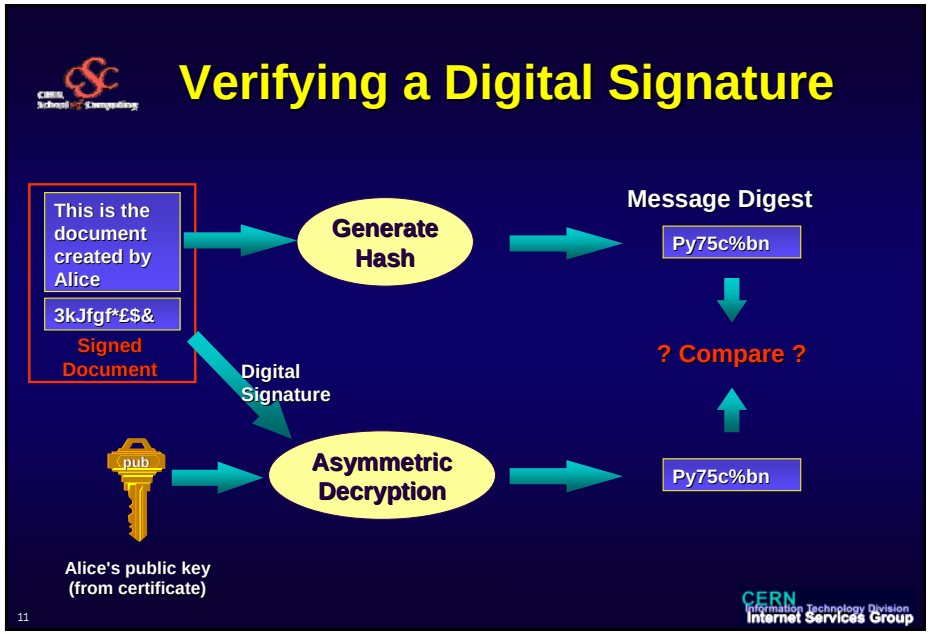
- Things to remember
  - The relation between the two keys is unknown and from one key you cannot gain knowledge of the other, even if you have access to clear-text and cipher-text
  - The two keys are interchangeable. All algorithms make no difference between public and private key. When a key pair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

CERN Information Technology Division Internet Services Group


Example: Confidentiality
Clear-text Input
"An intro to PKI and few deploy hints"
Cipher-text
"Py75c%bn&*)9|f De^bDzjF@g5=& nmdFgegMs"
Clear-text Output
"An intro to PKI and few deploy hints"
Encryption
Decryption
pub
priv
Different keys
Recipient's public key
Recipient's private key
8
CERN School of Computing 2006
CERN Information Technology Division Internet Services Group

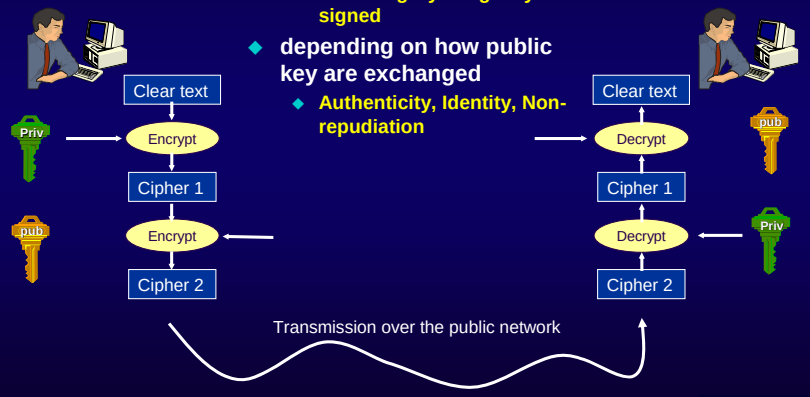

Example: Authenticity
Clear-text Input
"An intro to PKI and few deploy hints"
Cipher-text
"Py75c%bn&*)9|f De^bDzjF@g5=& nmdFgegMs"
Clear-text Output
"An intro to PKI and few deploy hints"
Encryption
Decryption
pub
priv
Different keys
Sender's private key
Sender's public key
9
CERN School of Computing 2006
CERN Information Technology Division Internet Services Group


Example: Integrity
Creating a Digital Signature
Message or File
This is the document created by Alice
Message Digest
(Typically 128 bits)
Py75c%bn
Digital Signature
3kJfgf*£$&
SHA, MD5
Generate Hash
RSA
Asymmetric Encryption
Calculate a short message digest from even a long input using a one-way message digest function (hash)
Signed Document
priv
private key of person signing
10
CERN
Information Technology Division
Internet Services Group


Verifying a Digital Signature
This is the document created by Alice
3kJfgf*£$&
Signed Document
Generate Hash
Message Digest
Py75c%bn
? Compare ?
Digital Signature
pub
Asymmetric Decryption
Py75c%bn
Alice's public key (from certificate)
11
CERN
Information Technology Division
Internet Services Group

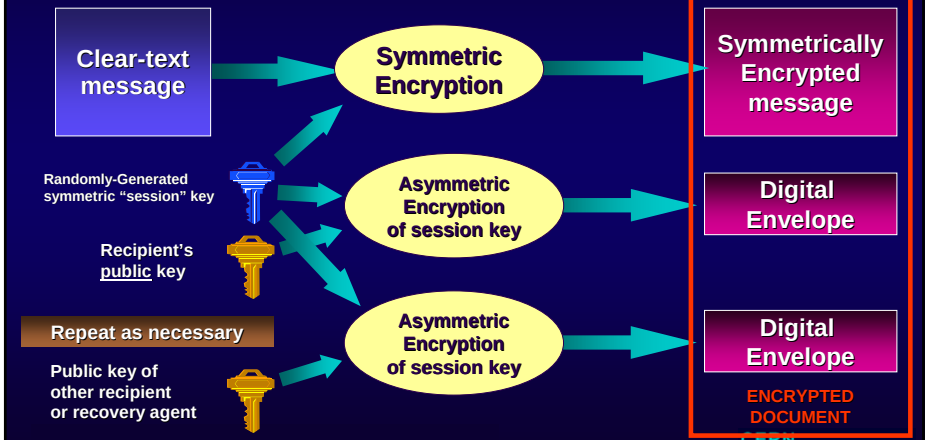
Example: SSL (simplified)

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



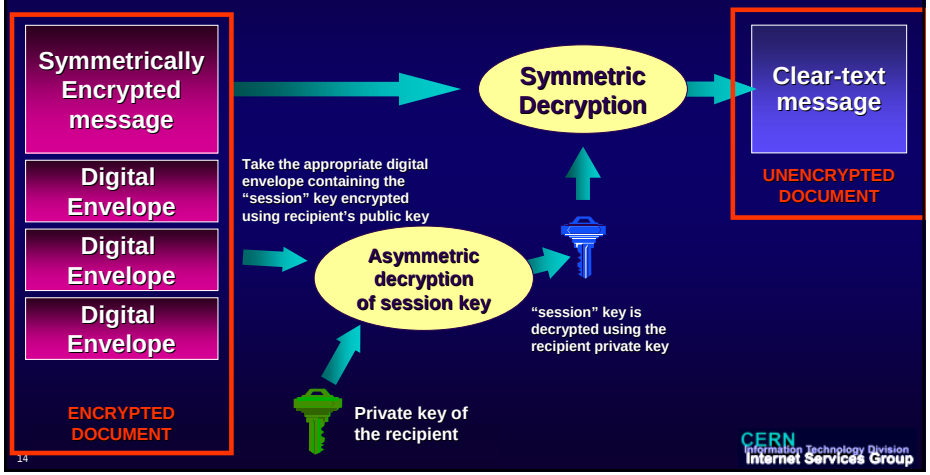
CERN Information Technology Division Internet Services Group

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



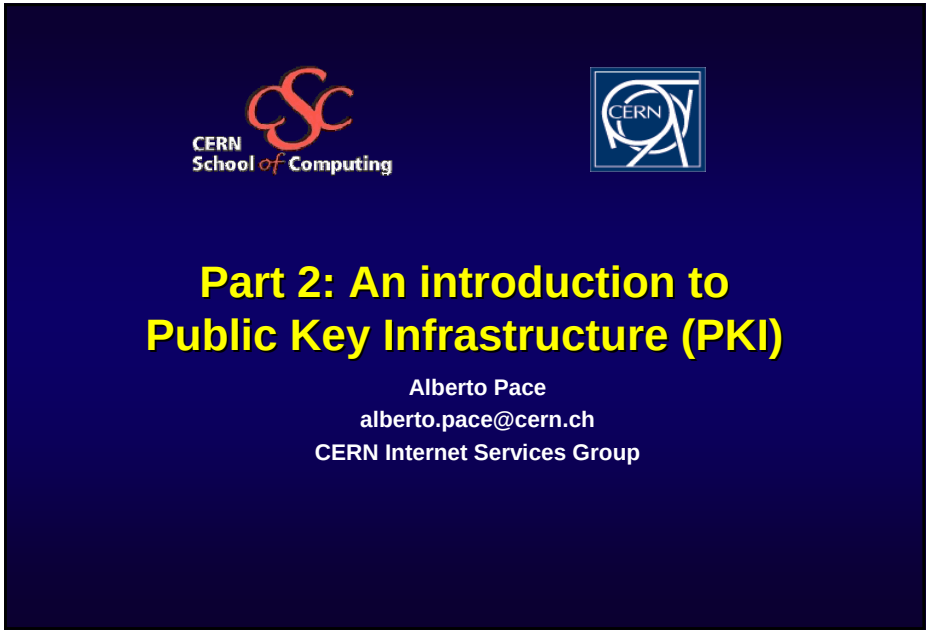
CERN Information Technology Division Internet Services Group

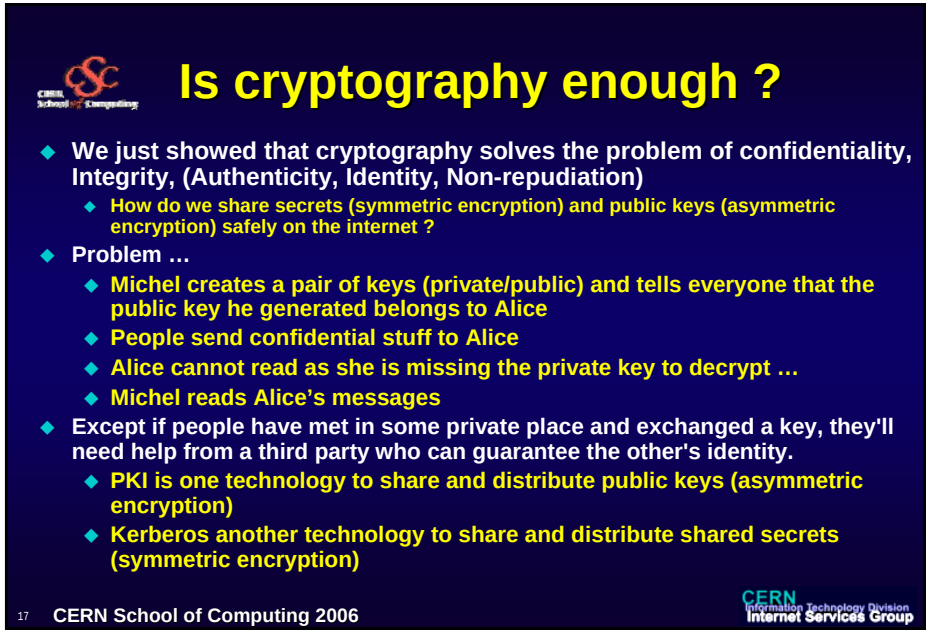
Real World: Hybrid Decryption



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


CERN
School of Computing
CERN
Part 2: An introduction to
Public Key Infrastructure (PKI)
Alberto Pace
alberto.pace@cern.ch
CERN Internet Services Group


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)
17
CERN School of Computing 2006
CERN
Information Technology Division
Internet Services Group

PKI = Public Key Infrastructure

- *"A technology to implement and manage E-Security"* A. Nash, "PKI", RSA Press
- PKI is a group of solutions for key distribution problems and other issues:
  - Key generation
  - Certificate generation, revocation, validation
  - Managing trust

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)
- …

My definition of PKI

- *"Public Key Infrastructure provides the technologies to enable practical distribution of public keys"*
  - **Using CERTIFICATES**

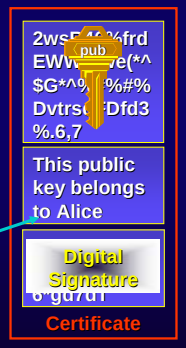
How to Verify a Public Key?

- **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 floppy using 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

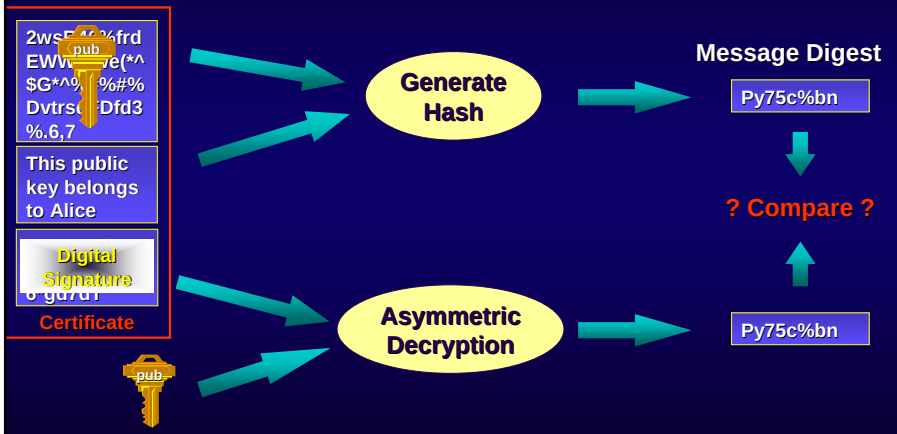
What is a Certificate ?

- 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

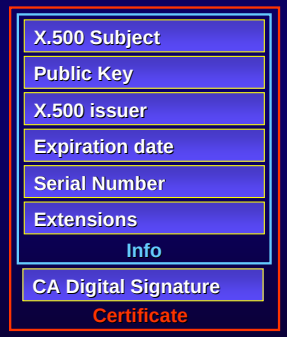
Can be a person, a computer, a device, a file, some code, anything …



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 is signing, O=CERN,C=CH

See later why expiration date is important

Additional arbitrary information

… of the issuer, of course

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
- Private keys should be protected
  - In computers files, protected by pass phrases
  - In OS protected storage
  - In smartcards

Certificates on Smartcards

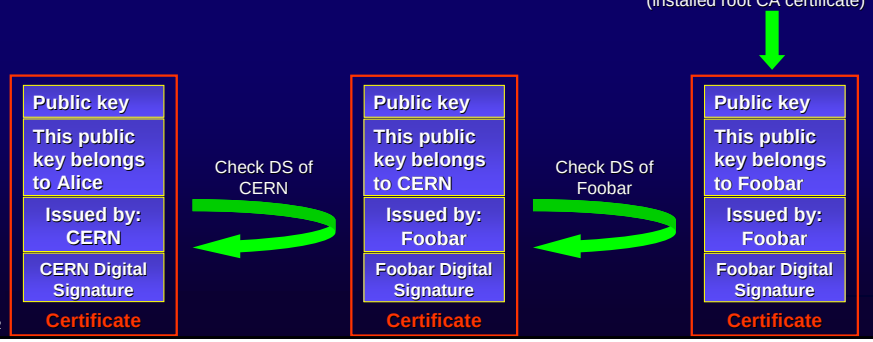
- A “bad” smartcard is only a dumb memory chip
  - Containing the Certificate and the private key
  - Both readable: You must trust the machine reading your smartcard
  - Better than using a floppy disk or saving everything to a file
- A “good” smartcard is more than a memory chip
  - Contains the Certificate, 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 and self-destruct

Handling Certificates

- Certificates are “safe to store”
  - No need to protect them too much, as they are digitally signed
  - Store anywhere, a file or a “dumb” memory-only smartcard
- Private keys that match the public key are strictly confidential
  - Loosing the private key = Loosing the identity
  - Must be very well protected
  - Use “Protected Storage” on your OS or a “smart” smartcard that will have crypto functionality on board

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



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

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)

Storing Certificates and Keys

- Certificates need to be stored so that interested users 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

Certificate is sent to the user

Certification Server

Example (with key recovery)

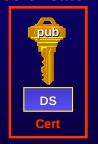
This model allows key recovery

CA generates a key pair

User request a certificate to CA

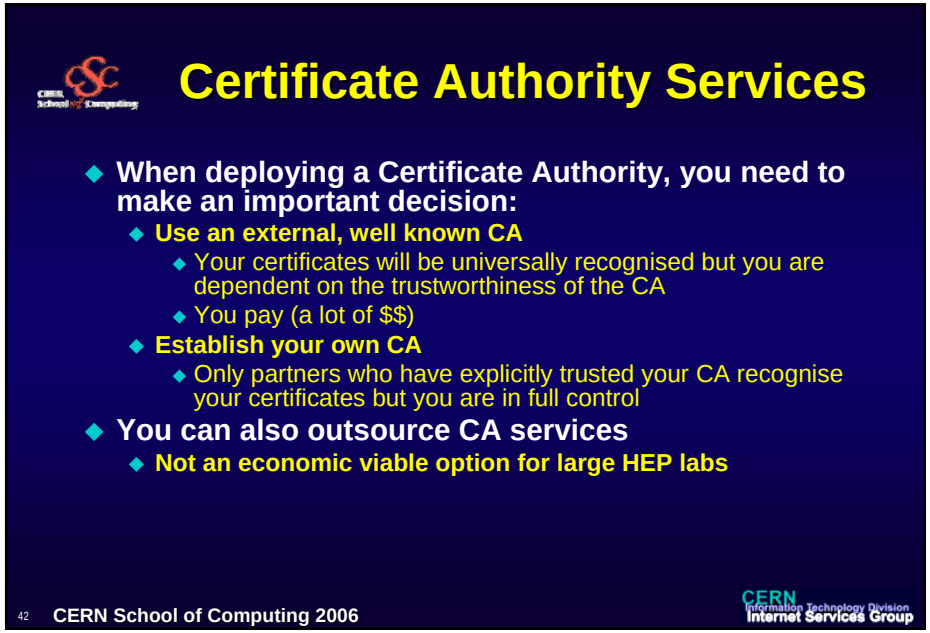
CA generates certificate

Certification Server



Private Key and Certificate are sent to the user


CERN
School of Computing
CERN
PKI Deployment


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
42
CERN School of Computing 2006
CERN
Information Technology Division
Internet Services Group

And there is more …

CERN Information Technology Division Internet Services Group

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

CERN Information Technology Division Internet Services Group

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?

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 based on ratings from Dun & Bradstreet Financial Services

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

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…

Part 3: An introduction to Kerberos

Alberto Pace
alberto.pace@cern.ch
CERN Internet Services Group

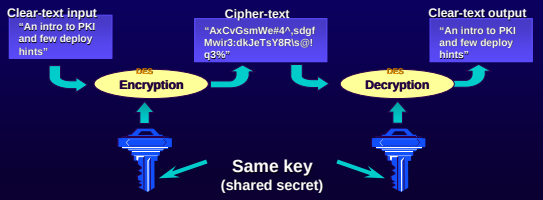
Kerberos gets its name from the mythological three headed dog that guards the entrance to Hell

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

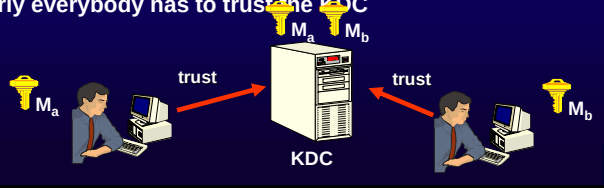
Kerberos Basics

- 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
  - technically the KDC doesn't know the password but rather a one way hash, which is used as the basis for a cryptographic "master key".
- The secret master key is different for each user
  - As two users don't know each other master key they have no direct way of verifying each other's identity
- The essence of Kerberos is key distribution. 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
- Clearly everybody has to trust the KDC

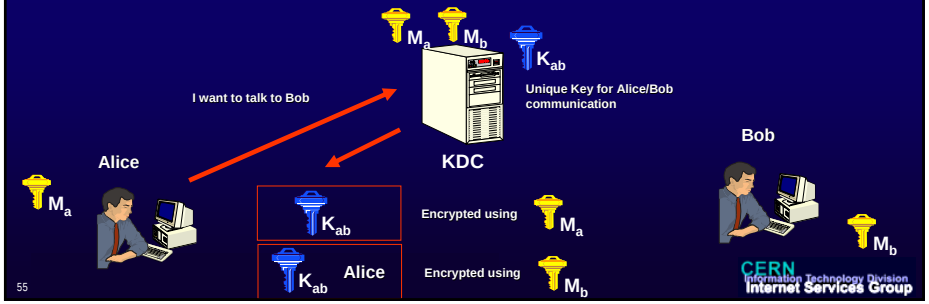


Breakthrough of a (simplified) Kerberos session

- 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 cryptographic key for Alice and Bob to use (call this K_{ab})
- He sends back two copies of K_{ab} back to Alice.
  - The first copy is for her to use, and is sent to her along with some other information in a data structure that is encrypted using Alice's master key.
  - The second copy of Kab is packaged along with Alice's name in a data structure encrypted with Bob's master key. This is known as a "ticket".

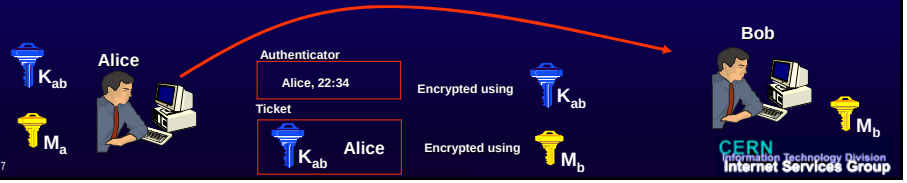


What is the ticket ?

- The ticket is effectively 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. Besides me, only you and Alice could possibly 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."

Kerberos authentication

- Alice must send the ticket to Bob
  - with proof that she knows K_{ab}
  - and she must do it in a way that allows Bob to detect replays from attackers listening on the network where Alice, Bob, and the KDC are conversing.
- The ticket is sent to Bob, with an authenticator (her name and the current time, all encrypted with the session key K_{ab})
- 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}

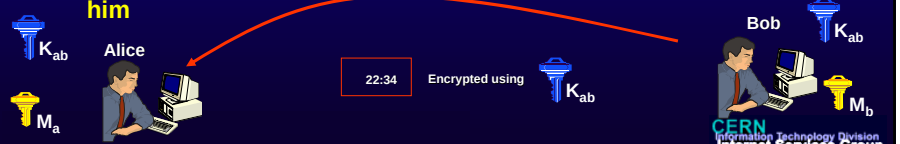


Kerberos authentication

- 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 Kab by encrypting the current time for me"

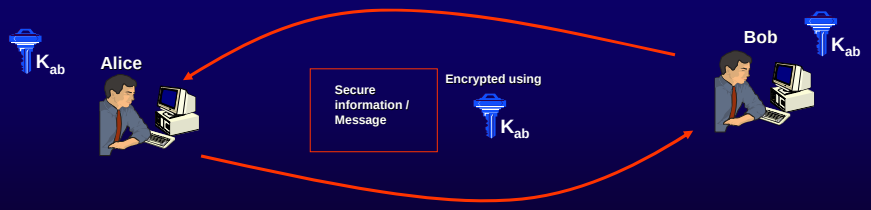
Mutual authentication

- Alice has proved her identity to Bob
- Now Alice wants Bob to prove his identity as well
  - she indicates this in her request to him 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

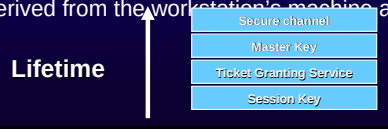


But life is more complicated

- 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.



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

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

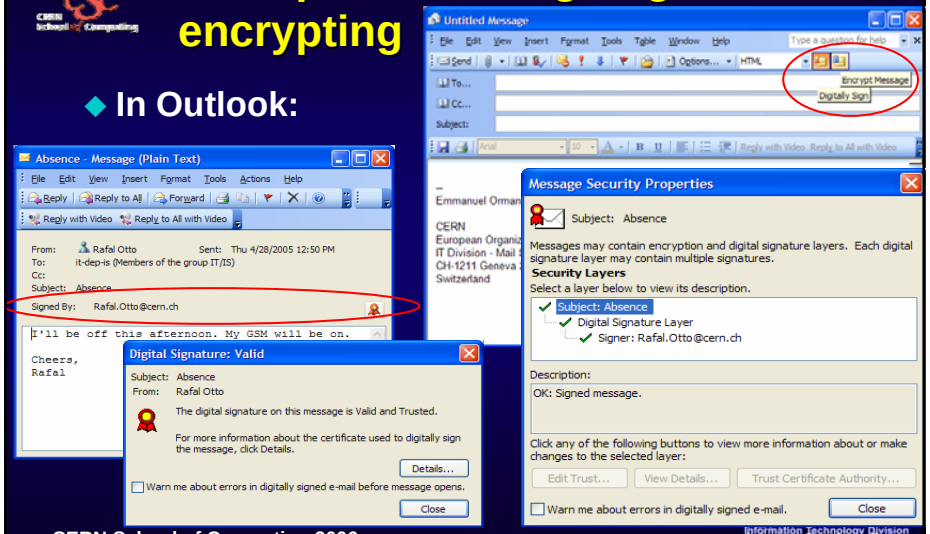
- Kerberos is used in Windows Domains and AFS
- PKI is used in all Grid related projects, with multiple certification authorities
- Multiple scenarios exist to integrate and interoperate the two technologies

Usage of Client Certificates

- Client authentication against a “service” (Example: a web server)
  - Proves your identity
- Digitally Sign documents and E-mail
  - Proves you wrote that document
- Encrypts information
  - Nobody else than your selected recipient can read the information

Example: Email signing and encrypting

- In Outlook:



Example: Web authentication using certificates (1)

- Certificate can be installed in any browser, on any platform.
- The web service offer the possibility to end-users to map the "subject" of their Certificate to their kerberos account (login name)
  - pace = "CN=Alberto Pace 8717;OU=GRID;O=CERN;C=CH"
  - pace = "E=Alberto.Pace@cern.ch;CN=Thawte Freemail Member"
- Authentication done automatically
  - The web browsers sends the client certificate to the web server
  - The web server verifies the digital signature and the validity of the certificate
  - The web server challenges the "client" system for the knowledge of the private key corresponding to the public key found in the certificate
  - If ok, the "subject" found in the certificate is authenticated. The Web server then can impersonate the kerberos account found in the PKI/Kerberos mapping table and proceeds with the user's credentials

Example: Web authentication using certificates (2)

- **Popup for selection if several certificates installed**
  - **multiple identity and roles are supported**
- **If no client certificate:**
  - **Optionally, downgrade smoothly to form-based authentication**
    - User enters kerberos username/password
    - Useful if using a public computer, but can be a security issue.
  - **Or force client certificate installation**
    - Requires the service provider to have an established Certification Authority
    - More secure but accessibility issue.

Web Authentication example

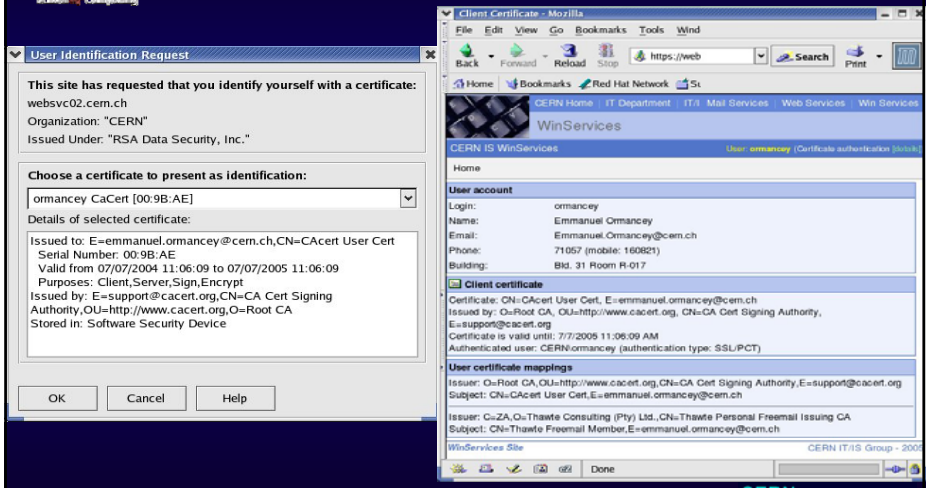
Opening a website

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

Certificate authentication complete.

Cancelled or no certificate installed

Technology not platform specific



CERN Information Technology Division Internet Services Group

PKI / Kerberos Integration example

- Setup a certification authority to create and sign X.509 certificates which are pre-mapped to kerberos accounts
- Publish a web interface to allow users to request, download and install Client certificates on their computer OR to map their existing (Grid / Thawte / CaCert / Other) certificate to their Kerberos account
  - Note: mapping should be possible only for certificates signed by certification authorities trusted by you
- Implement Certificate-based authentication on your servers

CERN Information Technology Division Internet Services Group

Example of current CERN plan

- For “managed” computers, the request, distribution and installation of Client certificates can be completely automated
  - For PCs member of a Windows domain, the CERN certificate can be pushed to the client as a domain policy
  - Its renewal can be handled automatically (allowing short validity periods)
  - Users do not need to understand, be aware, be informed. 100 % transparent.
- Similar automation levels exist for Linux and Mac OS systems

Architectural Comment (1)

- In this example, we have an interoperable Kerberos / PKI service in a master-slave situation
- Kerberos is the master, PKI is the slave
  - The Kerberos password is used to establish mapping between the Kerberos account and the PKI certificate
  - When possible, the Kerberos authentication triggers the client certificate installation
- This can be changed

Other possibilities

- If security needs to be strengthened
  - Store certificates elsewhere outside the computer (smart card) to better protect access to the private key during session authentication.
  - Smartcard protected by pin code
  - Disable form-based authentication based on password
- Consequence: No more passwords typed in
  - Passwords do not need to be known by users.
  - Passwords can be set to random string and can be reset very often, automatically.
- Consequences if Kerberos passwords not known by users
  - User must use certificate authentication as form based authentication is no longer possible.
  - less security problems but accessibility issue especially if using smartcards (public computers).

Architectural Comment (2)

- With smartcards or with passwords unknown to the users, we still have an interoperable Kerberos / PKI service in a master-slave situation
- PKI is the master, Kerberos is the slave
  - You distribute to users Certificates (smartcards) which are pre-mapped to Kerberos accounts

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

CERN Information Technology Division Internet Services Group

Identity Management

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

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**
  - **This includes**
    - All data on the persons
    - All workflows to Create/Read/Update/Delete records of persons, accounts, groups, organizational unit, …
    - All internal processes and procedures
    - All tools used for this purpose

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 can be physical (specific location, door, room, …) or logical (password, certificate, biometric, token, …)
  - Resources can also be physical (room, a terminal, …) or logical (an application, a table in a database, a file, …)

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

Why Identity Management ?

- Legal Constraints
  - In many areas there is a legal obligation of traceability
  - Basel II (Global Banking financial regulations)
  - Sarbanes Oxley Act (SOX) in the US
  - 8th EU Privacy Directive + national laws in Europe
- Financial constraints
  - Offload IT experts from administrative tasks with little added value (user registration, password changes, granting permissions, ...)
- Technical opportunity
  - Simplification of procedures, increased opportunity
  - Centralized security policy possible

Aware of legal constraints

- Laws are different in each country
- Laws depend on the type of institute
  - Public funded, Government, Privately owned, International Organization, …
- Laws depend on the sector of activity
  - Archiving, traceability, retention of log files and evidences
- Not easy to find the good compromise between security / accounting / traceability and respect of privacy / personal life

Implementing IM / IAM

- Overall strategy
  - Be realistic. Base the project on “short” iterations (4 - 8 weeks) with clear objectives and concrete results at each iteration
  - Understand the perimeter of the project.
    - Services included / excluded
    - One single project cannot fix all existing and cumulated projects
  - Understand the stakeholders
    - Who is affected
    - Who pays
    - Ensure to have management support
  - Inventory, simplify, streamline and document all administrative procedures
- It is an heavy project, there are many parameters

More IAM Architecture components (1/5)

- (web) application for person and account registration
  - **Used by the administration to create identities**
  - **Approval, workflow and information validation depends on the type of data**
    - Requiring a workflow or validation/approval by the administration. Examples: Name, passport no, date of birth
    - Available in self service to end-user:
      Examples: password change, preferred language, …

IAM Architecture

Identity Management (Administration)

HR Database

IAM Architecture components (2/5)

- Process and workflow well defined
  - What are the “administrative” requirements to be “authorized” to use service “xyz”
  - “administrative” means that you have all information in the IAM database
  - You can define rules and process to follow. You can implement a workflow.
- If you can’t answer this question, you can’t automate
  - Putting an administrative person to “manually handle” the answer to that question won’t solve the problem in large organizations

IAM Architecture

Identity Management (Administration)

HR Database

Accounts

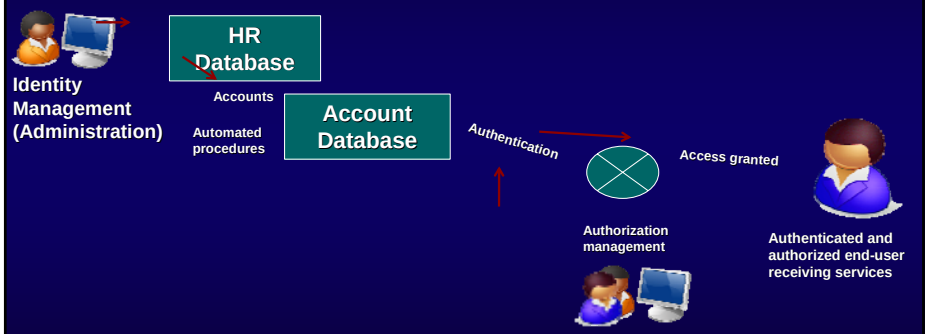
Automated procedures

Account Database

More IAM Architecture components (3/5)

- Service-specific interfaces to manage authorization
  - This is typically *platform and service dependent*
  - Allows assignment of permissions to groups or accounts or persons
  - Authorization can be made once to a specific group and managed using group membership

IAM Architecture

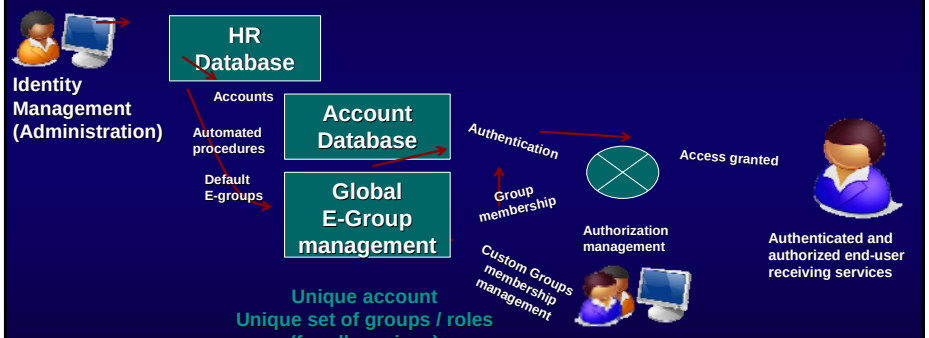


CERN School of Computing 2006
CERN Information Technology Division Internet Services Group

More IAM Architecture components (4/5)

- (web) application to manage group memberships
  - Indirect way to manage authorizations
  - Must foresee groups with manually managed memberships and groups with membership generated from arbitrary SQL queries in the IAM database
  - Must foresee nesting of groups

IAM Architecture

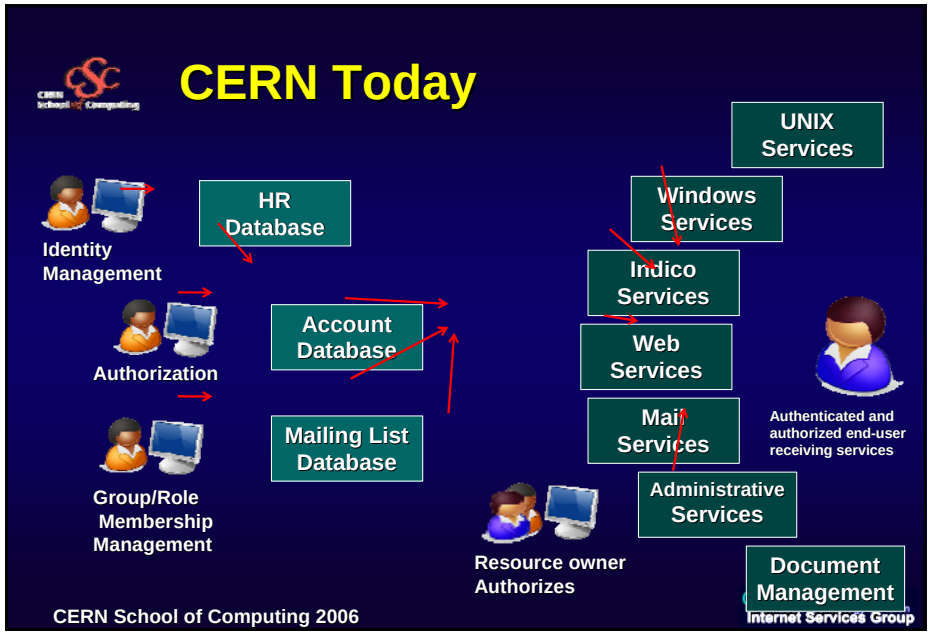


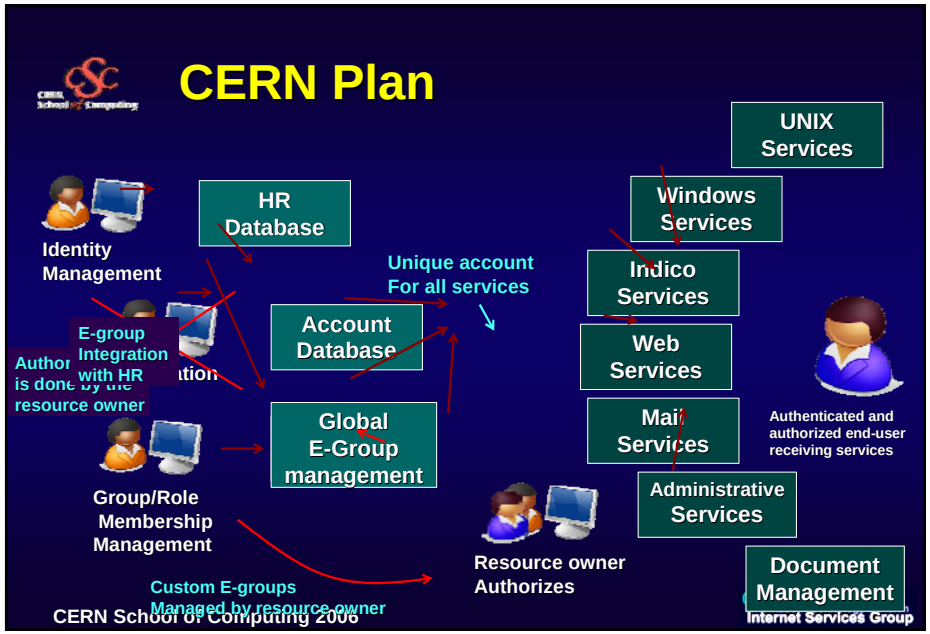
More IAM Architecture components (5/5)

- Single-Sign-On (SSO) services
  - aware of group memberships
  - Authentication portal for web-based applications
  - Kerberos services for Windows and/or AFS users
  - Certification authority for grid users
- Directories, LDAP, …
- A well thought communication plan to inform all users

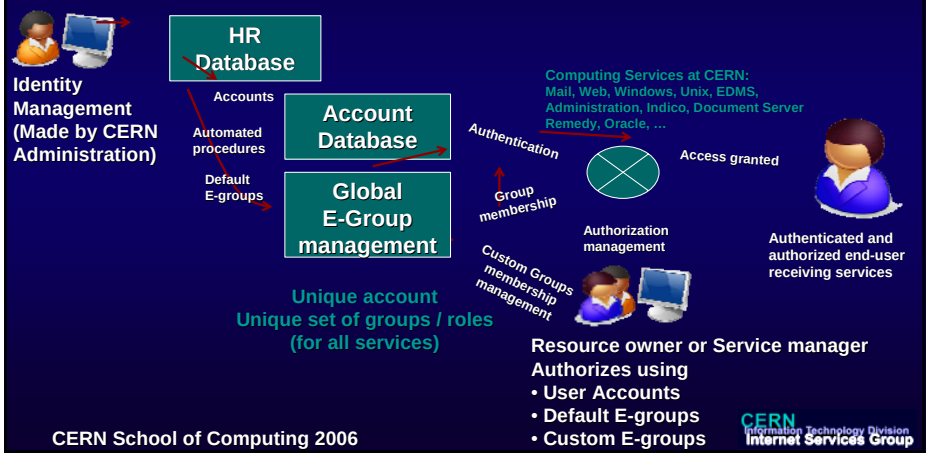
Experience at CERN

- CERN has an HR database with many records (persons)
- 23 possible status
  - Staff, fellow, student, associate, enterprise, external, ...
- Heavy rules and procedures to create accounts
  - Multiple accounts across multiple services
    - Mail, Web, Windows, Unix, EDMS, Administration, Indico, Document Server, Remedy, Oracle, …
  - Multiple accounts per person
  - Being migrated towards a unique identity management system with one unique account for all services


CERN Today
UNIX Services
HR Database
Windows Services
Identity Management
Indico Services
Account Database
Web Services
Authorization
Mail Services
Authenticated and authorized end-user receiving services
Mailing List Database
Group/Role Membership Management
Administrative Services
Resource owner Authorizes
Document Management
CERN School of Computing 2006
Internet Services Group


CERN Plan
UNIX Services
HR Database
Windows Services
Identity Management
Unique account For all services
Indico Services
E-group Integration with HR
Account Database
Web Services
Author is done by the resource owner
Mail Services
Authenticated and authorized end-user receiving services
Global E-Group management
Group/Role Membership Management
Administrative Services
Resource owner Authorizes
Custom E-groups Managed by resource owner
Document Management
CERN School of Computing 2006
Internet Services Group

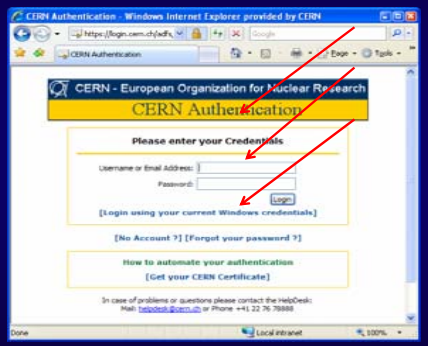
CERN Plan



CERN Plan summary

- Central account management
- Only one account across services
  - synchronize UNIX and Windows accounts
- Use Roles/Groups for defining access control to resources
  - No more: “close Windows Account, keep Mail account, block UNIX account”
  - But: “block Windows access, allow Mail access, block AIS access”.

Single Sign On Example

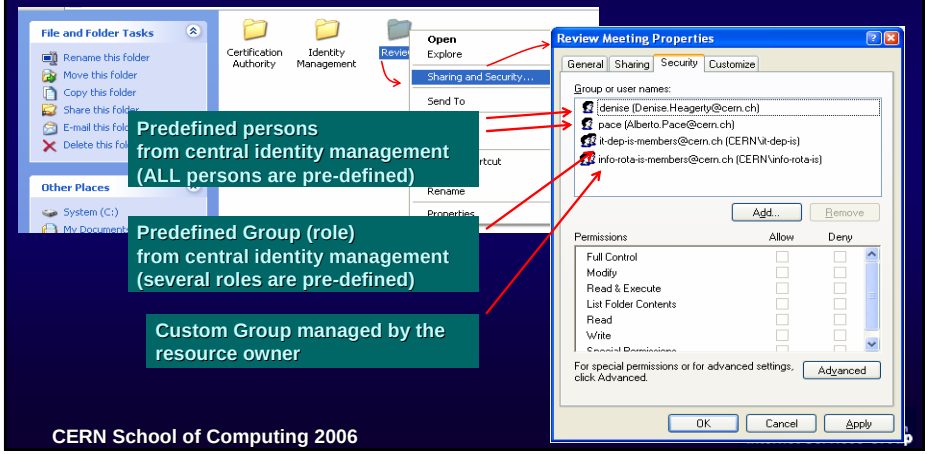


- Username / Password
- SSO using Windows Credentials
- SSO using Grid Certificate

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

Example

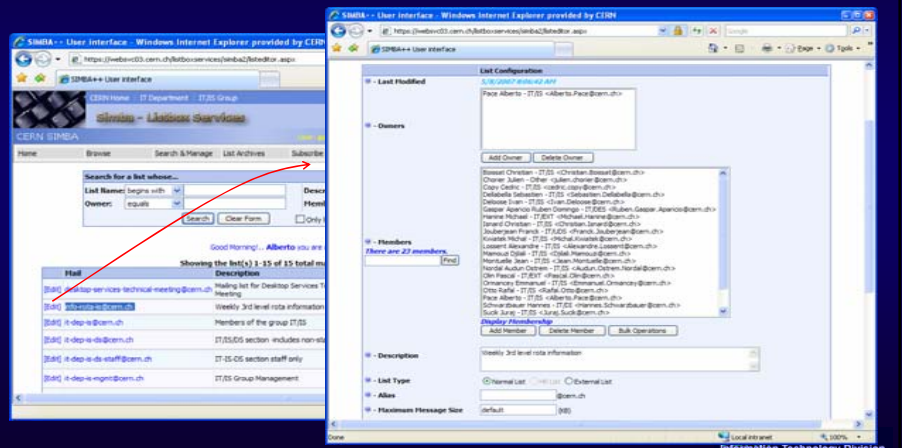


Predefined persons
from central identity management
(ALL persons are pre-defined)

Predefined Group (role)
from central identity management
(several roles are pre-defined)

Custom Group managed by the resource owner

Managing custom group example



Errors to avoid

- Legal
- Organizational Factors
  - Lack of management support, of project management / leadership
  - No clear and up to date communication
    - Inform user of constraints and benefits
  - RBAC with too many roles
- Technical
  - Incorrect estimation of quality of existing data
  - Implement an exception on each new demand
  - Lost mastering of technical solutions

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

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

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.