An introduction to Public Key Infrastructure (PKI)

Alberto Pace
CERN Internet Services Group

alberto.pace@cern.ch, http://cern.ch/alberto.pace

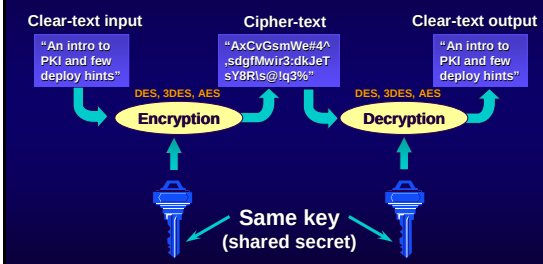
Agenda

- Quick Summary on Cryptography
- Fundamentals of PKI
- PKI Deployment

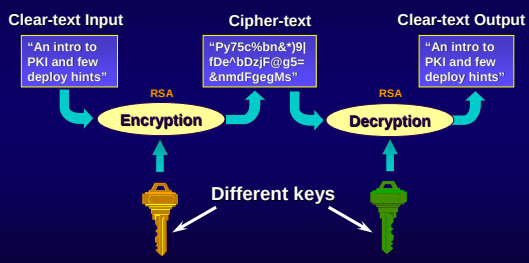
What does Cryptography solve?

- 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

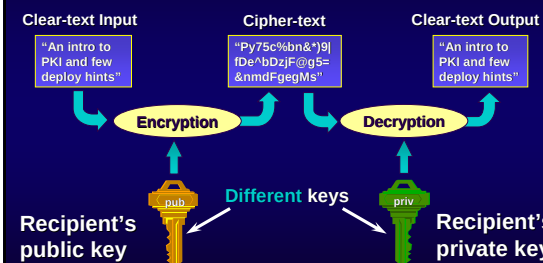
Symmetric Encryption

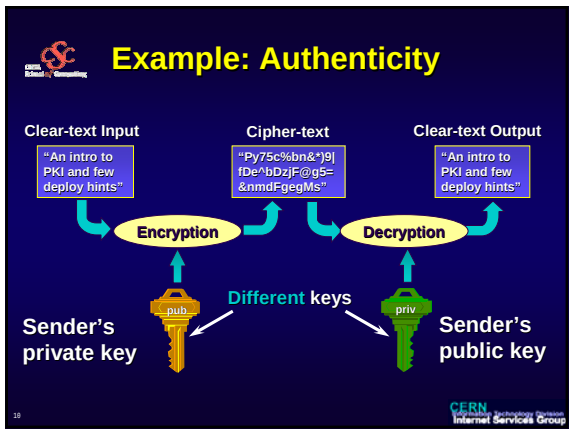


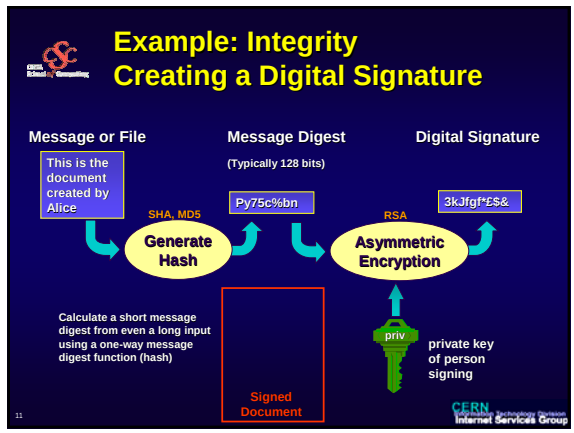
Asymmetric Encryption

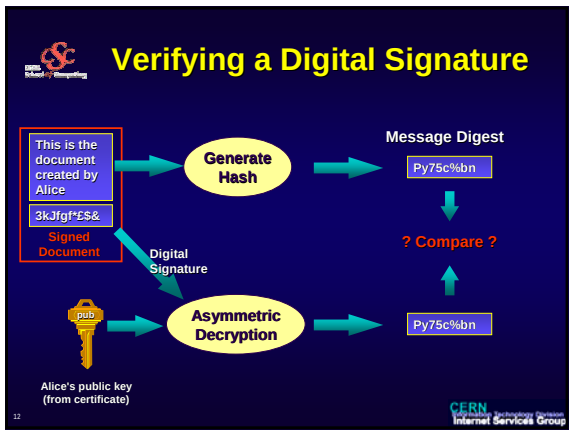


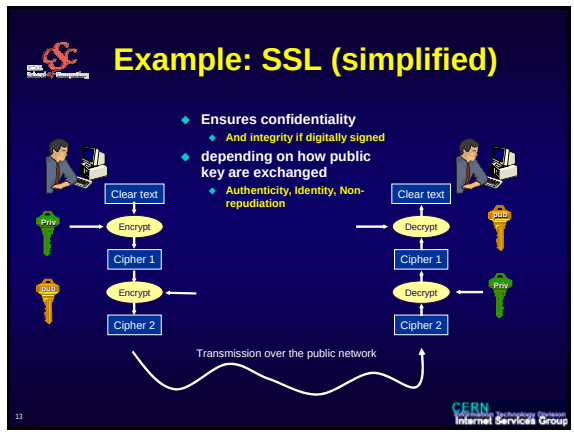
Example: Confidentiality

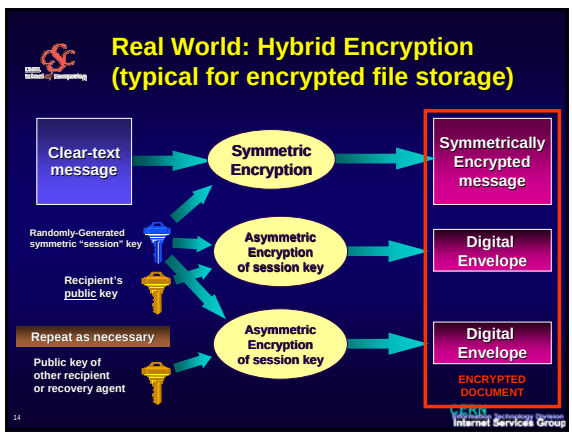


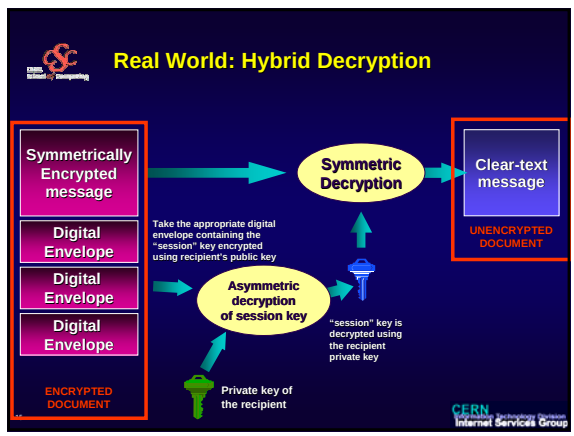

Example: Authenticity
Clear-text Input
"An intro to PKI and few deploy hints"
Cipher-text
"Py75c%bn&*)9|fDe^bDzjF@g5=&nmdFgegMs"
Clear-text Output
"An intro to PKI and few deploy hints"
Encryption
Decryption
Different keys
pub
priv
Sender's private key
Sender's public key
CERN Internet Services Group


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
CERN 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)
CERN Internet Services Group


Example: SSL (simplified)
Ensures confidentiality
And integrity if digitally signed
depending on how public key are exchanged
Authenticity, Identity, Non-repudiation
Clear text
Encrypt
Cipher 1
Encrypt
Cipher 2
Decrypt
Cipher 1
Decrypt
Cipher 2
Transmission over the public network
CERN Internet Services Group


Real World: Hybrid Encryption
(typical for encrypted file storage)
Clear-text message
Symmetric Encryption
Symmetrically Encrypted message
Randomly-Generated symmetric "session" key
Asymmetric Encryption of session key
Digital Envelope
Recipient's public key
Repeat as necessary
Asymmetric Encryption of session key
Digital Envelope
Public key of other recipient or recovery agent
ENCRYPTED DOCUMENT
CERN Internet Services Group


Real World: Hybrid Decryption
Symmetrically Encrypted message
Digital Envelope
Digital Envelope
Digital Envelope
ENCRYPTED DOCUMENT
Take the appropriate digital envelope containing the "session" key encrypted using recipient's public key
Asymmetric decryption of session key
Private key of the recipient
"session" key is decrypted using the recipient private key
Symmetric Decryption
Clear-text message
UNENCRYPTED DOCUMENT
CERN Internet Services Group

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

Agenda

- Quick Summary on Cryptography
- Fundamentals of PKI
- PKI Deployment

Is cryptography enough ?

- How do we share secrets (symmetric encryption) and public keys (asymmetric encryption) safely on the internet ?
  - We just showed that cryptography solves the problem of confidentiality, Integrity, (Authenticity, Identity, Non-repudiation)
- But…
  - 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)

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 this stuff?

- 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)
- Bad ID Smartcards (Certificates and private key store)
- Good ID Smartcards (Certificates and Challenge/Response)
- Executables (Java applets, .NET Assemblies, Drivers, Authenticode)
- Copyright protection (DRM)
- …

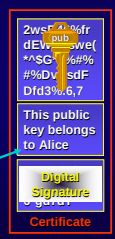
My definition of PKI

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

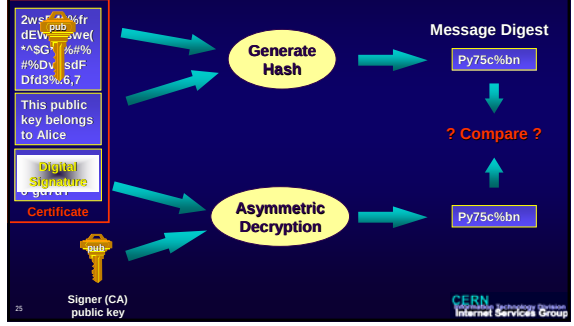
What is a Certificate ?

- 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)

X.500 Subject	Who is the owner, CN=Alice,O=CERN,C=CH
Public Key	The public key or info about it
X.500 issuer	Who is signing, O=CERN,C=CH
Expiration date	See later why expiration date is important
Serial Number	
Extensions	Additional arbitrary information
Info	
CA Digital Signature	… of the issuer, of course
Certificate	

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 !

Authentication with 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

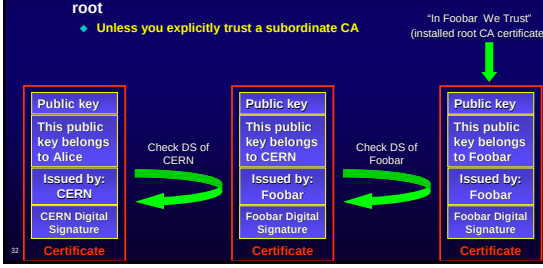
Increased cost

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

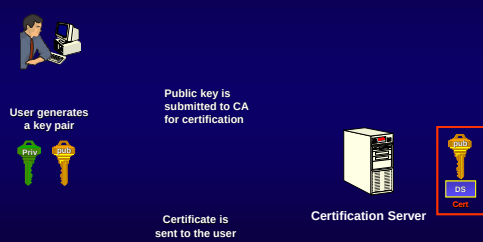
Revoked certificates can be trusted

- 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)

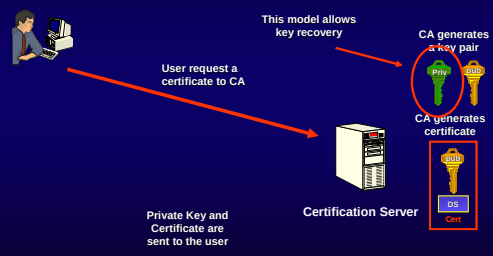
Storing Certificates and Keys

- 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)



Example (with key recovery)



Agenda

- Quick Summary on Cryptography
- Fundamentals of PKI
- PKI Deployment

Certificate Authority Services

- When deploying a Certificate Authority, you need to make an important decision:
  - Use a well known CA
    - Your certs will be universally recognised but you are dependent on the trustworthiness of the CA
    - You pay lot of $$
  - Establish your own CA
    - No one except your explicitly nominated partners or clients will recognise your certs 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 …

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

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…

Appendix: Kerberos

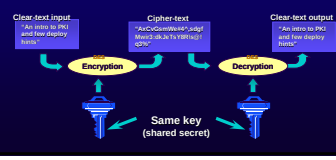
Kerberos gets its name from the mythological three headed dog that guards the entrance to Hell

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

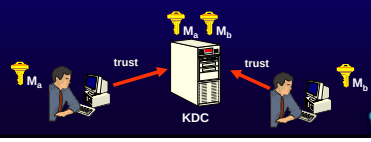
Kerberos Basics

- 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

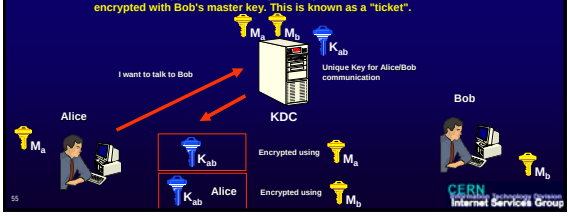


Breakthrough of a (simplified) Kerberos session

- 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".

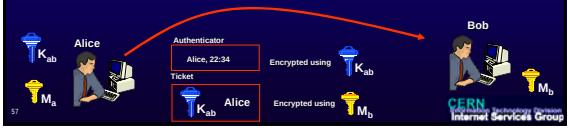


What is the ticket ?

- 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 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 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

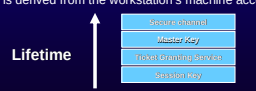
- Alice and Bob share now a unique secret K_{ab} that they use to communicate

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.



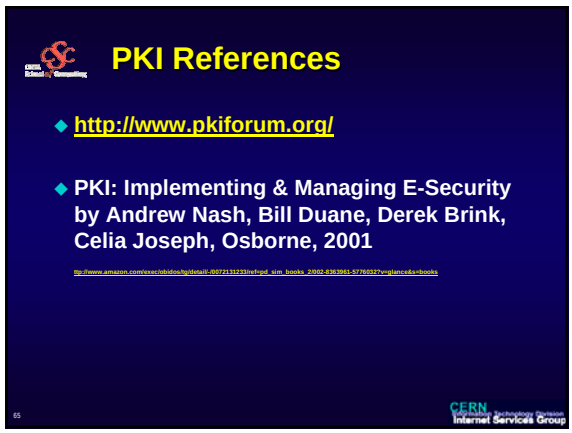
Kerberos ticket in real life

	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

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.


PKI References
http://www.pkiforum.org/
PKI: Implementing & Managing E-Security by Andrew Nash, Bill Duane, Derek Brink, Celia Joseph, Osborne, 2001
65
CERN
Internet Services Group