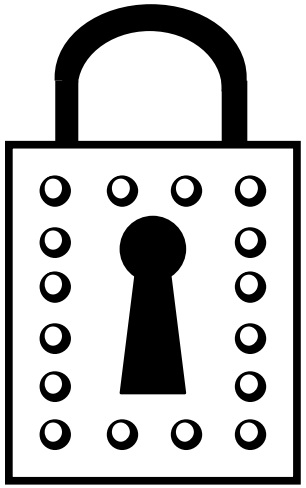
# Symmetric and Asymmetric Ciphers



# License

# 1. Symmetric Ciphers

# Symmetric ciphers

- A symmetric cipher is one in which you can:
  - Encrypt a plaintext message with a secret key
  - Decrypt a ciphertext message with the *same* secret key
- Encrypting a message twice produces the original message
- There are good symmetric ciphers that:
  - Are fast, that is, require an acceptable amount of CPU time to encrypt/decrypt
  - Are strong, that is, take thousands of years to crack
  - Are useful for encrypting files on your computer (you are the only person who needs to know the secret key)

# Limitations of symmetric ciphers

- Symmetric ciphers have a significant limitation:
  - Makes them unsuitable for communication with people in remote locations
- How can you negotiate a secret key with the remote party?
  - A telephone call containing the secret key might be intercepted
  - Same for postal mail or email messages that contain a secret key
  - Sending the secret key through a courier means you have to trust the courier. Could he be bribed?
- You have to keep track of multiple secret keys:
  - One secret key for messages to Alice
  - A different secret key for messages to Bob. And so on…
  - In general, a group of N people needs $O(N^2)$ secret keys
- *Asymmetric* ciphers addresses this limitation

# 2. Asymmetric Ciphers

# Asymmetric ciphers and public key cryptography

- Symmetric ciphers were used for thousands of years
  - In the 1970s, Ralph Merkle developed an asymmetric cipher
  - See www.merkle.com for his PhD thesis (of historical interest)
  - Since then, several better asymmetric ciphers have been developed
- An asymmetric cipher uses two keys:
  - A message encrypted with key1 can be decrypted only with key2
  - A message encrypted with key2 can be decrypted only with key1
- Knowing one key does not help you guess the other key
- *Public key cryptography* is another name for asymmetric ciphers
  - One key is called the *public key*
  - The other is called the *private key*

# Uses for public key cryptography

- I put my public key on, say, my business card or website
- Use 1:
  - To securely send me a message, encrypt it with my public key
  - Only I can decrypt the message (with my private key)
- Use 2:
  - I make some (compiled) software available from my website
  - I make a checksum of the software files and encrypt the checksum with my private key (this is called *signing*)
  - You download the software and the signed checksum
  - To verify that the software comes from me (rather than a hacker):
    - You calculate a checksum for the software you downloaded
    - You use my public key to decode my checksum, and compare it to your checksum
    - If the checksums match then you know the software is genuine

# Combining symmetric and asymmetric ciphers

- A serious limitation of asymmetric ciphers:
  - They can use 100–1000 times more CPU time than symmetric ciphers
- Solution: use a two-step approach for remote communication:
  - Use an asymmetric cipher to securely communicate a private key for use with a symmetric cipher
  - Then switch over to using the symmetric cipher with the agreed-upon private key
- SSL uses a (more elaborate) two-step approach:
  - Initial handshaking is slow due to use of an asymmetric cipher
  - Then communication gets much faster due to use of a symmetric cipher

# Securely storing the private key

- I can advertise my public key widely. However…
- I *must* keep my private key private
  - Otherwise, somebody else could pretend to be me
- Advice for storing your private key:
  - Store it in a file on your computer
  - Ensure nobody else can read the file:
    - Example: UNIX file permissions
    - Always lock your computer when you leave your office
    - Be careful who has access to backup disks of your computer
  - Also, encrypt the private key using a *pass phrase* (password) known only to you as the encryption key

# A limitation of asymmetric ciphers

- Anyone can create a public-private key pair
  - You just need to run a software utility to create the key pair
  - There are proprietary and open-source utilities, such as OpenSSL
- This creates a problem:
  - A public key enables you to securely communicate with whoever has the corresponding private key
  - The person with the private key *claims* to be, say, Amazon.com
  - How can you be sure?
- The solution is called a *certificate authority* (CA)

# 3. Certificate Authority (CA)

# Driving license authority

- Scenario: you want a driving license to use as a form of identification
  - You go to the *driving license authority* (DLA) building
  - You must prove your identity to the DLA
    - You might use your passport, recent utility bills, and so on
  - The DLA gives you a (difficult-to-forge) driving license document
    - Laminated card containing your photograph, age, height, eye color
    - Start and end validity dates
    - Also contains the DLA logo
- A driving license works as a form of identification because:
  - People can verify that details on the driving license match you
  - Lots of people trust the DLA (can't be bribed to give out fake ids)
- A *certificate authority* (CA) serves a role similar to a DLA

# Certificate Authority

- Scenario: you want people to have faith in your public-private key pair, so you can use it as a form of identification
  - You create a public-private key pair yourself
    - This is called a *certificate signing request* (CSR)
  - You go to a *certificate authority* (CA) building
  - You must prove your identity to the CA:
    - Passport, driving license, recent utility bills, and so on
  - The CA gives you an *X509 certificate* (a particular standard)
    - In other words, the CA *signs* your CSR
  - The certificate specifies:
    - Your public key
    - Your details (name, website address, …)
    - Name of the CA
    - A checksum for the certificate, signed by the CA's private key
    - Start and end validity dates

The purpose of a certificate is to securely associate your details with your public key

# Certificate Authority (cont')

- An X509 certificate works as a form of identification because:
  - Lots of software use a library to recognize (check) X509 certificates
  - Software is bundled with copies of public keys for popular CAs
    → the software can verify the signed checksum on the certificate
  - Lots of people trust the CA (can't be bribed to give out fake certificates)
- Example: online shopping with your web browser:
  - When you click on the *Pay Now* button, your browser visits a web page starting with https:// (the "s" denotes a secure web page)
  - The web browser downloads the website's X509 certificate
  - The web browser checks:
    - Has the certificate been signed by a CA trusted by the web browser?
    - Is the name of the website contained in the certificate's details?
  - If the checks are okay then you know the website is not an imposter

# Practical details of CAs

- How do you know you can you trust a CA?
  - You just have to trust them
  - Just like you have to trust the driving license authority
  - A CA “self signs” its own X509 certificate
- Can anyone set themselves up as a CA?
  - Yes, but it might take a lot of effort to convince web browser companies to bundle your public key in their products
  - Users can import extra CA certificates into their browser (but this is “too much bother” for most users)
- It is common for an organization to have its own *internal* CA:
  - Saves money: don’t have to pay an external CA for certificates
  - The internal CA may not be trusted outside of the organization, but that is not a problem for internally-deployed applications

# 4. Summary

# Summary

- Symmetric ciphers are fast, but have some limitations:
  - Difficult to *securely* agree on a secret key with a remote party
  - You need a separate secret key for each person you communicate with
- Asymmetric ciphers are slow but:
  - Enable secure communication without prior agreement of a secret key
  - Make it possible to electronically “sign” a document to prove it came from you
- SSL:
  - Uses an asymmetric cipher initially to agree on a secret key
  - Then switches over to a symmetric cipher (for speed)

# Summary (cont')

- X509 is a standard for a security certificate
  - A form of identification, just like a driving license or passport
  - The certificate contains a *public key*, so people can send you messages securely
  - You keep the corresponding *private key* a secret
  - The certificate is signed by a certificate authority
- A certificate authority (CA) is a trusted organization that can sign your X509 certificate
  - There are well known CAs that are trusted worldwide
  - An organization can have its own CA for internally-deployed applications