




# Public Key Infrastructure

What can it do for you?



# What is PKI?

- Centrally-managed cryptography, for:
  - Encryption
  - Authentication
- Automatic negotiation
- Native support in most modern Operating Systems
- Allows *untrustworthy, unrelated* (foreign) parties to communicate securely and to authenticate each other's identity



# What is PKI?

- One globally-trusted Certificate Authority
  - Every participant in a PKI trusts the same CA
  - CA must have strong security and policies
- Participants request certificates, CA issues and tracks certificates
  - Process called “Enrolment”



# Certificates

- Certificates are a standardized way of encapsulating Asymmetric Keys
  - ANSI X.509
    - Multiple formats exist (BER, DER, PEM, PKCS, etc.)
- Additional information is stored in a certificate along with the key (identity, authorization, etc.)



# Secret Key Cryptography

- A.K.A. Symmetric Cryptography
- Common Examples:
  - 3DES, AES (Rijndael), Blowfish, RC4, CAST5, IDEA
- One key used to encrypt & decrypt.
  - Both parties must have this key.
- Problems:
  - How to transfer key securely?
  - How to enable one-way (restricted) sharing?
- Postal/Padlock Analogy



# Public Key Cryptography

- A.K.A. Asymmetric Cryptography
- Examples: RSA, DSA
- Two keys: private & public
  - Public key encrypts, private key decrypts
  - Private key signs, public key verifies
  - Mathematically related, by a “hard problem”
- Does not (necessarily) replace Secret-Key
  - Makes secret-key systems more secure
- Postal/Padlock Analogy



# Postal Analogy

- An analogy which can be used to understand the advantages of an asymmetric system is to imagine two people, Alice and Bob, sending a secret message through the public mail. In this example, Alice has the secret message and wants to send it to Bob, after which Bob sends a secret reply.
- With a symmetric key system, Alice first puts the secret message in a box, and then locks the box using a padlock to which she has a key. She then sends the box to Bob through regular mail. When Bob receives the box, he uses an identical copy of Alice's key (which he has somehow obtained previously, maybe by a face-to-face meeting) to open the box, and reads the message. Bob can then use the same padlock to send his secret reply.
- In an asymmetric key system, Bob and Alice have separate padlocks. First, Alice asks Bob to send his open padlock to her through regular mail, keeping his key to himself. When Alice receives it she uses it to lock a box containing her message, and sends the locked box to Bob. Bob can then unlock the box with his key and read the message from Alice. To reply, Bob must similarly get Alice's open padlock to lock the box before sending it back to her.
- The critical advantage in an asymmetric key system is that Bob and Alice never need send a copy of their keys to each other. This substantially reduces the chance that a third party (perhaps, in the example, a corrupt postal worker) will copy a key while it is in transit, allowing said third party to spy on all future messages sent between Alice and Bob. In addition, if Bob were to be careless and allow someone else to copy his key, Alice's messages to Bob would be compromised, but Alice's messages to other people would remain secret, since the other people would be providing different padlocks for Alice to use.
- From: Wikipedia, [http://en.wikipedia.org/wiki/Asymmetric\\_key\\_algorithm](http://en.wikipedia.org/wiki/Asymmetric_key_algorithm)



# Two linked keys

- Not all asymmetric key algorithms operate in precisely this fashion.
- The most common have the property that Alice and Bob each own two keys, one for encryption and one for decryption.
- In a secure asymmetric key encryption scheme, the decryption key should not be deducible from the encryption key.
- This is known as public-key encryption, since the encryption key can be published without compromising the security of encrypted messages.
- In the analogy above, Bob might publish instructions on how to make a lock ("public key"), but the lock is such that it is impossible (so far as is known) to deduce from these instructions how to make a key which will open that lock ("private key").
- Those wishing to send messages to Bob use the public key to encrypt the message; Bob uses his private key to decrypt it.
- From: Wikipedia, [http://en.wikipedia.org/wiki/Asymmetric\\_key\\_algorithm](http://en.wikipedia.org/wiki/Asymmetric_key_algorithm)





# OpenSSL

- Originally named SSLey  
  - originally designed to implement SSLv2 in Perl
- Apache-style license
- Implements SSL v2/v3
  - Secure Sockets Layer (from Netscape)
- Implements TLS v1
  - Transport Layer Security (replaces SSL)
- Full-strength general purpose cryptography library



# OpenCA

- Full Certificate Authority system
- Three major parts
  - user interface
    - Six (6) default interfaces included!
  - OpenSSL backend
    - To perform the cryptography functions
  - Database
    - To keep track of the CSRs, Certs, CRRs & CRLs
- OpenSSL provides all the crypto necessary to do CA certificate signing function; OpenCA provides the management functions.



# Cacert.org

- Free and Open Certificate Authority
- Still one central entity to trust
- Uses Web-of-Trust mechanism to delegate trust decisions
  - Similar to PGP keys
  - Modeled after Thawte's WoT
    - Thawte offers free keys for personal e-mail use only
- Root certificate not yet included in Internet Explorer



# S/MIME

- Uses public and private keys to secure internet e-mail
  - Can encrypt,
    - protect contents from unauthorized viewing
  - Can sign,
    - Protect contents from unauthorized modification
  - Or both at the same time
    - Requires use of all four (4) keys:
      - Sender's public and private keys
      - Recipient's public and private keys



# SSH

- Public-key crypto is used to protect the negotiation phase only
  - switches to symmetric-key for actual data encryption
- SSH keys are public and private keys, not X.509 certificates
  - Certificates can be used, but must be processed into correct format first
- SSH can authenticate you based only on your public/private key pair
  - i.e. no password is used at all!
- SSH server needs your public key, you need your private key
  - Opposite of SSL: here the server is authenticating you, not you authenticating the server.



# HTTP-over-SSL (HTTPS)

- SSL uses public-key cryptography to securely negotiate the symmetric key actually used to encrypt data
- HTTPS is, simply, the HTTP protocol over an SSL-encrypted TCP channel.
- SSL uses the “Subject” of a certificate to ensure website identity - Subject must exactly match the website name (hostname).
- SSL also checks the certificate chain up to a trusted CA (built in to the browser).



# SmartCards

## ■ Microsoft

- Work-in-progress, unless you use a commercially-supported PKI product
- <http://wiki.cacert.org/wiki/DomainController>

## ■ Linux/\*BSD/etc

- Generally requires OpenSC in OSS environments
- Integrates with PAM
  - <http://www.opensc-project.org/projects.html>
  - <http://wiki.cacert.org/wiki/PamAuthentication>
- M.U.S.C.L.E.
  - *Movement for the Use of Smart Cards in a Linux Environment*
  - <http://www.muscocard.com/>



# Sample CA use

- `openssl genrsa -out ca.key 2048`
- `openssl req -new -x509 -days 365 -key ca.key -out ca.crt`
- Produce CSR with any tool
  - e.g. web server, e-mail client, openssl
- `openssl x509 -req -in certreq.txt -out cert.txt -CAcreateserial -CA ca.crt -CAkey ca.key`





# Links

- <http://www.openssl.org/>
- <http://www.openca.org/>
- <http://www.cacert.org/>
- <http://www.newpki.org/>
- <http://www.rsasecurity.com/rsalabs/node.asp?id=2152> (Crypto FAQ)
- <http://smartsign.sf.net>
- [http://en.wikipedia.org/wiki/Asymmetric\\_key\\_algorithm](http://en.wikipedia.org/wiki/Asymmetric_key_algorithm)
- <http://ospkibook.sourceforge.net/>
- [http://en.wikipedia.org/wiki/Certificate\\_authority](http://en.wikipedia.org/wiki/Certificate_authority)
- <http://www.opensc-project.org/>