Internet Engineering

241-461

Robert Elz

kre@munnari.OZ.AU
kre@fivedots.coe.psu.ac.th

http://fivedots.coe.psu.ac.th/~kre

Public Key Encryption

- Can do the same thing (as boxes)
  - Using encryption algorithms (mathematics)
- Have algorithm with two keys
  - Either key used to encrypt data
  - The other key must be used to decrypt
- Give one key away to everyone
    - The Public Key
- Keep the other key secret
    - The Private Key
- Terminology
  - data encrypted by private key
      - Signed
  - Anyone can decrypt it using the public key
        - So no secrecy from this encryption
  - When they decrypt
      - They know for certain who encrypted it
  - That is the same as a signature on a document

Use of Public/Private Keys

- Confidentiality
  - Encrypt using recipient's public key
- Integrity
  - Data not visible
      - Hard to modify
      - Easy to replace completely
          - public key known to all
- Authentication
  - Add signature
          - Encrypt with sender's private key
      - guarantees identity of sender
          - Only person who has that key
      - Also guarantees integrity
          - if data replaced, signature lost
- Non-repudiation
  - Only sender could have signed it
      - Even recipient does not know sender's key
  - So, if signed
      - Must have been sent by the sender

The remaining problem

- Who was the sender?
  - Data signed
    - So we know owner of private key sent it
    - But who is that?
  - How do we know who owns the private key
    - We have public key
    - Public key says
      - Aj Robert's Public key
    - How do you know that is correct?
- We need proof of identity somehow

Certificates

- Organisation (or person) creates public/private keys
- Takes public key to trusted organisation
  - Perhaps government department
  - Or another trustworthy organisation
    - Call this organisation Certificate Authority
- Proves identity to Certificate Authority
  - and proves ownership of public key
    - by using private key to encrypt some data
- Certificate Authority verifies all data
  - Then signs the public key
    - Plus owner's identity information
  - That is,
    - encrypts it all with CA's private key
- This is known as a certificate
  - Organisation takes away the signed certificate
    - Gives copies of this away instead of just public key

Certificates (2)

- CA's public key known to all
  - Widely distributed
  - If fake version appears
    - someone will recognise it is incorrect
    - advise everyone to be careful
- When anyone gets a certificate
  - They decrypt using CA's public key
  - If valid
    - That is: If decryption succeeds
  - Now know identity of owner
  - And owner's public key
- When we have a certificate
  - We can send secret message to its owner
  - We are certain who that is
    - Only that owner can read our message
    - We encrypt using their public key
  - We can authenticate their signature
    - We decrypt using their public key

Public Key Encryption

- Three common algorithms
  - RSA
    - Rivest Shamir Adleman
  - Diffie-Hellman
  - Elliptic Curve
- RSA most common
  - very slow to operate with private key
  - average speed operating with public key
  - slow to create key pair
- D-H
  - slow to encrypt/decrypt data with either key
  - very slow to create key pair
- Major Problem
  - Algorithms are very slow
    - We can encrypt only small amounts of data
      - Too much takes too long

Another Digression

- We have
  - Shared Key encryption
    - fast but needs shared secret
  - Public Key Encryption
    - slow but no shared secrets
- There is also
  - Cryptographic Hash Function
- Hash function
  - Take data (perhaps much data)
    - generate much smaller result
    - where result depends upon input
      - different input, different output
      - though many differnt inputs produce same output
  - Hash functions have many uses

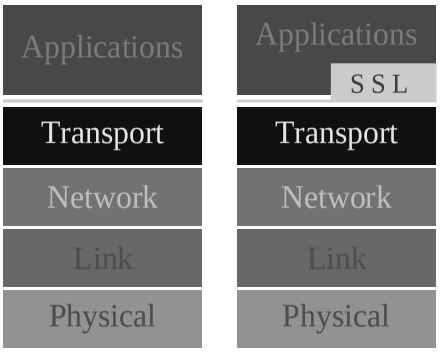
Cryptographic Hash

- Hash function with extra properties
  - Very difficult to find input
    - to generate a desired hash result
  - Very difficuly to modity input
    - and leave hash result unchanged
- Digital Signatures
  - Encrypt data using private key (very slow)
    - Too costly if data is large
    - Any data not included
      - easily changed by attacker
- Solution
  - Hash all data using crypto hash (very fast)
  - Encrypt hash result using private key (small result)
  - Attach encrypted hash result as digital signature
- Verification
  - Detach digital signature (decrypt and save)
  - Hash data - Compare result with that from signature
  - If the same: Signature verified, data unaltered

Back to the Web

- Secure Web pages
  - Page accompanied by a certificate
  - Certificate guarantees identity of owner
    - Page itself can be signed
- To send secure data
  - Generate key for a symmetric key protocol
  - Encrypt that using Public Key encryption
    - Using public key from certificate
  - Encrypt other data using symmetric key protocol
- Owner of certificate
    - The organisation we want to send to
  - Has private key
  - Can decrypt the key for symmetric protocol
  - Can then decrypt the other data
- Others
  - Do not have the private key
  - Cannot decrypt the message

Secure Sockets Layer



- Traditional Internet layers
- Secure Sockets Layer

SSL

  - Extends Sockets Interface
  - Implemented in Applications
    - As library routines
    - Available to any application

Implements security as just discussed

Network Security

- SSL is implemented at transport layer
    - or just above it
  - Also known as Transport Layer Security
- Can also have security at network layer
    - IP Security

IPsec

  - Node to Node security
- IPsec Protects all communications between two nodes
  - All transport protocols
    - Including ICMP
  - Hides transport headers
    - Port numbers protected
  - But:
    - Requires implementation in network stack
    - Key distribution is harder
      - Used much less often

Security Mechanisms

- IPsec
  - Network Level Security
  - Node to Node
- TLS
  - Transport Level Security
  - Process to Process
- For E-Mail
  - Which is appropriate?
  - Is either appropriate?
- NO
  - Want to know
    - message is from who it says
    - message is unmodified
    - message is private (perhaps)
  - Want user to user security

E-Mail Security

- IPsec for e-mail
  - Protects mail between MTA & next MTA
    - or MTA to/from MUA
  - Content of mail might be anything
    - No guarantees at all
  - Content is what is important
- TLS (SSL)
  - Almost the same as IPsec
    - But from MTA to MTA
    - rather than MTA's host to MTA's host
- Neither knows about people
  - e-mail is all about people
  - Is From: header address
    - really the authorising entity

Security Model for e-mail

- Most server security
  - client application must verify server
  - Make sure server client connected to
    - is the server it claims to be
  - Done using certificate
    - Owner of certificate has Private Key
    - Can prove ownership
    - Certificate contains identity
  - Client identity largely irrelevant
    - Server serves everyone
- Useless for e-mail
- For e-mail
  - client identity most important
    - Actual human user identity
  - Server largely irrelevant

Security Model for e-mail (2)

- Proof of identity
  - Certificates handle this
  - But "real" certificates cost a lot
    - perhaps 100,000 THB
    - every year or two
- How many people in the room have a certificate?
  - How many will ever have one?
- Costly part of certificate
  - The validation
  - Signature of Certificate Authority
    - They must investigate first
- Self-signed certificates
  - I say: "Yes, it really is me"
- Can we trust that?

Trust Model for e-mail

- No, self-signed certificate not much better
  - than the From: header address
- Sender creates both
  - Either might be fake
- But
  - Certificate can be widely published
  - Claimed to belong to me
  - If fake
    - hope someone will notice
    - expose the forgery
  - Better than nothing
- Better still
  - If I know you
  - You tell me that is your certificate
    - I can sign it for you
  - You tell other friends
    - They also sign your certificate

Trust Model for e-mail (2)

- Recipient receives mail
  - Signed by the sender
- Recipient looks up certificate
  - Sender's certificate
- Obtains Public Key
  - Can verify signature
- Now we know
  - Message was signed by certificate owner
- But, who is that?
- If sender's certificate is signed by me
  - Then I know it is correct
  - I checked before I signed it
- If sender's certificate is signed by you
  - And I trust you
  - Then I can also believe certificate
    - To same extent I trust you

Trust Model for e-mail (3)

- Can extend this
  - Certificate claims to be owner by A
    - Signed by B
  - B's certificate
    - Proves B is who he says
    - Allows us to validate signature
      - on A's certificate
    - Signed by C
      - C promises B is really B
      - And this is B's certificate
  - C's certificate
    - Same as B's Signed by D ...
- Eventually some certificate is signed by me
  - That one I certainly trust
  - Allows all certificates in chain
    - to be trusted
      - Including A's eventually
- We now know mail is really from A

E-Mail security encoding

- SMIME
  - Secure MIME
- PGP
  - Pretty Good Privacy
    - (Pretty here means "Fairly" or "Quite")
- Message is signed
  - Signature appended
  - Private encapsulation method
    - PGP pre-dates use of MIME
- Signature covers hash of message content
  - stops cutting signature from one message
  - pasting it into another
- Message is optionally encrypted
  - Encrypted using recipient's public key
  - Only recipient can decrypt it