Public Key / RSA

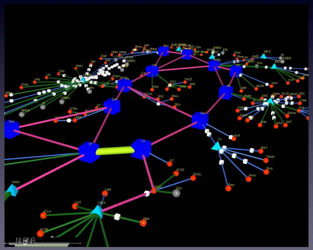
CMSC 426/626 - Fall 2014

Outline

- Public Key Fundamentals
- Signatures
- Certificates
- The RSA Algorithm

Why Public Key?

- The problems of symmetric key distribution…



The vBNS Network (Jeff Brown, National Laboratory for Applied Network Research)

Requirements

- Easy to generate public and private keys.
- Given the public key of a recipient, easy to generate an encrypted message.
- Using the private key, it is easy for the recipient to decrypt a message.
- An adversary who knows a public key can not determine the corresponding private key.
- An adversary who knows a public key and a message encrypted with that key can not recover the plaintext message.

In Formulas…

- Easy for party B to generate PU_B and PR_B.
- Given PU_B it is easy to compute $C = \mathrm{E}(PU_B, M)$.
- Easy for recipient B to compute $M = \mathrm{D}(PR_B, C)$.
- Given PU_B, it is infeasible to determine PR_B.
- Given PU_B and $C = \mathrm{E}(PU_B, M)$, it is infeasible to determine M.

Diffie and Hellman

- Whitfield Diffie and Martin Hellman wrote down these requirements in 1976.
- Three mathematicians at GCHQ (Ellis, Cocks, Williamson) invented such an algorithm prior to 1976, but it was classified.

(from telegraph.co.uk)

Public Key Systems

- *RSA* - Ron Rivest, Adi Shamir, Leonard Adleman in 1977 (also Clifford Cocks in 1973).
- *Diffie-Hellman* - Whitfiled Diffie and Martin Hellman (also Malcom Williamson in 1974).
- *Digital Signature Standard* - NIST FIPS PUB 186 in 1991; revised in 1993. Signatures only.
- *Elliptic Curve Cryptography* - not really a new PKC system, but a different approach to constructing them.

What is it good for?

Confidentiality

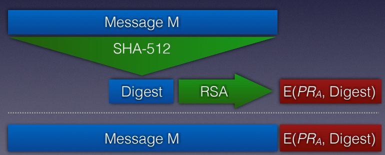
- Alice encrypts message for Bob using PU_B.
- Bob receives the message and decrypts with PR_B.
- Must have PR_B to decrypt; presumably only Bob can do this.

Authentication / Integrity

- Alice encrypts message for Bob using her own private key PR_A.
- Bob decrypts using PU_A (remember this is *public*).
- Must have PR_A to create the message; presumably only Alice can do this.

Digital Signatures

- Alice wants to send a message *M* to Bob, ensuring integrity and authenticity.
- Alice has public and private keys (PU_A, PR_A) and has published her public key in a directory.



Alice sends *M* along with the encrypted Digest.

Certificates

- Remember the "directory" from the previous slide? How do you know Alice's public key really belongs to Alice?
- What if Eve posts a public key and email address and *says* it belongs to Alice?
- How can we fix this?

- A certificate binds a user's identity (e.g. URL or email address) to their public key.

Version
Serial Number
Signature Algorithm ID
CA Name
Validity Period
Subject Name
Subject Public Key Info

- The public key, along with identifying information, is signed by a *Certification Authority* (CA).

ISO X.509

- *The* standard for certificates; published as RFC 2459.

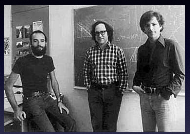
```
Certificate  ::=  SEQUENCE  {
       tbsCertificate       TBSCertificate,
       signatureAlgorithm   AlgorithmIdentifier,
       signatureValue       BIT STRING  }

  TBSCertificate  ::=  SEQUENCE  {
       version         [0]  EXPLICIT Version DEFAULT v1,
       serialNumber         CertificateSerialNumber,
       signature            AlgorithmIdentifier,
       issuer               Name,
       validity             Validity,
       subject              Name,
       subjectPublicKeyInfo SubjectPublicKeyInfo,
       issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                            -- If present, version shall be v2 or v3
       subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                            -- If present, version shall be v2 or v3
       extensions      [3]  EXPLICIT Extensions OPTIONAL
                            -- If present, version shall be v3
       }
```

- One last point about certificates: you still need to deliver CA public keys to the end users in a trusted manner.
- The CAs' signing keys are the root of trust for a public key infrastructure (think HTTPS).

RSA

- Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman in 1977.
- Also Clifford Cocks at GCHQ in 1973.
- RSA can be used for encryption or digital signatures.



Rivest, Shamir, Adleman (courtesy of AMS)

Cliff Cocks (from http://www.geekscontactee.com)

Private Information

p	A large prime number
q	A large prime number
d	The *decryption exponent*

Public Information

N	The product of p and q
e	The *encryption exponent*

Encryption

$\boxed{C = M^e \bmod N}$

Decryption

$\boxed{M = C^d \bmod N}$

A message M is a number between 0 and N - 1.

- For p and q, "large" means "at least 512 bits," but 1024 bits is now typical.
- Knowing only N and e, it is infeasible to find d.
- Knowing p, q, and e, it is easy to find d.
- d and e satisfy a mathematical relation:

$$d \cdot e = 1 \bmod \Phi(N)$$

- For our purposes, it is good enough to say that $\Phi(N) = (p - 1) \cdot (q - 1)$.

Some Examples

- Exercise: $p = 3$, $q = 11$, $e = 7$, $M = 5$. Encrypt M and then decrypt using RSA.
- A larger example in Python.

Security of RSA

- The security of RSA is based on the difficulty of integer factorization - it is infeasible to factor N.
- Current record for factorization: 768 bit RSA composite, completed in December 2009.
- It is believed that finding $\Phi(N)$ is no easier than factoring N.

Homework is posted on the website.