Lecture 24 (April 22, 2004)

Outline

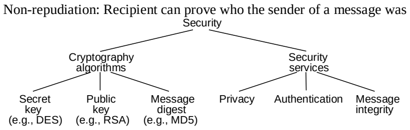
Network Security

Why do we care about Security?

- "Toto… I have a feeling we're not in Kansas anymore." Dorothy, *The Wizard of Oz*
- "The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable." *The Art of War*, Sun Tzu
- There **are** bad guys out there who can easily take advantage of you.
- Reference: *Cryptography and Network Security, Principles and Practice*, William Stallings, Prentice Hall

Overview

- Security services in networks
  - Privacy: preventing unauthorized release of information
  - Authentication: verifying identity of the remote participant
  - Integrity: making sure message has not been altered
  - Non-repudiation: Recipient can prove who the sender of a message was



- Cryptography functions – building blocks for security
  - Privacy/Authentication
    - Secret key (e.g., Data Encryption Standard (DES))
    - Public key (e.g., Rivest, Shamir and Adleman (RSA))
  - Integrity
    - Message digest/hash (e.g., Message Digest version 5 (MD5))

Issues in Security

- Threat models
  - How are bad guys trying to do bad things to you?
- Key distribution
  - How do folks get their keys?
- Implementation and verification
  - How can we be sure systems are secure?

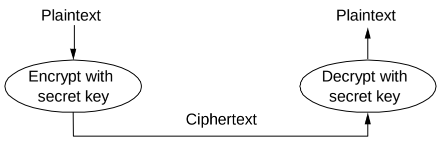
Crypto 101

- Cryptographic algorithms determine how to generate encoded text (ciphertext) from plaintext using keys (string of bits)
  - Can only be decrypted by key holders
- Algorithms
  - Published and stable
  - Keys must be kept secret
  - Keys cannot be deduced
  - Large keys make breaking code VERY hard
  - Computational efficiency

Basic Crypto Systems

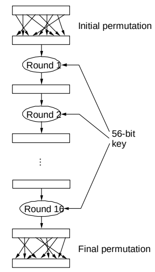
- Symmetric Key (or Secret Key)
  - Example: DES
  - Same key for encryption and decryption
- Public Key
  - Example: RSA
  - Private key (known only to the principal), public key is publicly known
- Message digests
  - Example: MD5

Secret Key (DES)

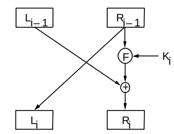


• Approach: Make algorithm so complicated that none of the original structure of plaintext exists in ciphertext

- Encrypt 64 bit blocks of plaintext with 64-bit key (56-bits + 8-bit parity)
- 16 rounds

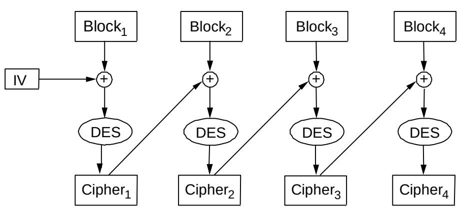


- Each Round



- L,R = 32 bit halves of 64 bit block
- K = 48 bits of 64 bit key
- F = combiner function
- + = XOR

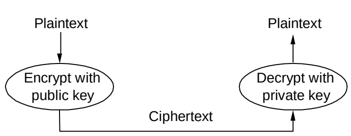
- Encryption steps are the same as decryption
- Repeat for larger messages (cipher block chaining)
  - IV = initialization vector = random number generated by sender



Establishing Secret Key

- Diffie Hellman Key establishment protocol
- p is a prime, g is the generator for p
- A chooses a random number a
- B chooses a random number b
- A sends to B: $g^a \bmod p$
- B sends to A: $g^b \bmod p$
- Secret key is: $g^{ab} \bmod p$

Public Key (RSA)



- One of the coolest algorithms ever!
- Encryption
  - *ciphertext* = $c = m^e \bmod n$ ($\langle e, n\rangle$ = *public key*)
- Decryption
  - *Message* = $m = c^d \bmod n$ ($\langle d, n\rangle$ = *private key*)
- M < n
  - Larger messages treated as concatenation of multiple n sized blocks

RSA contd.

- Choose two large prime numbers p and q (each 256 bits)
- Multiply p and q together to get n
- Choose the encryption key e, such that e and $(p - 1) \times (q - 1)$ are relatively prime.
- Two numbers are relatively prime if they have no common factor greater than one
- Compute decryption key d such that

$$d = e^{-1} \bmod ((p - 1) \times (q - 1))$$

- Construct public key as (e, n)
- Construct public key as (d, n)
- Discard (do not disclose) original primes p and q

RSA contd.

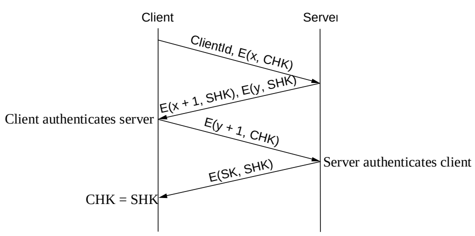
- See example in book for applying RSA
  - Many others as well
- Remember – always encrypt with *public* key and decrypt with *private* key
- Security based on premise that factoring is hard
  - The bigger the key the harder it is to factor
  - The bigger the key is more computationally expensive it is to encrypt/decrypt

Message Digest

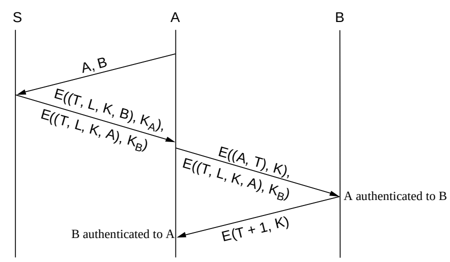
- Cryptographic checksum
  - a fixed length sequence of bits which is used to protect the receiver from accidental changes to the message; a cryptographic checksum protects the receiver from malicious changes to the message.
- One-way function
  - given a cryptographic checksum for a message, it is virtually impossible to figure out what message produced that checksum; it is not computationally feasible to find two messages that hash to the same cryptographic checksum.
- Relevance
  - if you are given a checksum for a message and you are able to compute exactly the same checksum for that message, then it is highly likely this message produced the checksum you were given.

Authentication Protocols

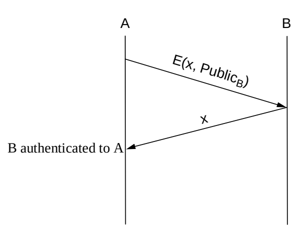
- Three-way handshake (uses secret key - eg. password)
  - E(m,k) = encrypt message *m* with key *k;* C/SHK = client/server handshake key; x, y = random numbers; SK = session key



- Trusted third party (Kerberos)
  - A and B share secret keys (K_A, K_B) with trusted third party S
  - A,B =ID's; T = timestamp; L = lifetime, K = session key



- Public key authentication (using eg. RSA)



Message Integrity Protocols

- Digital signature using RSA
  - special case of a message integrity where the code can only have been generated by one participant
  - compute signature with private key and verify with public key
- Keyed MD5 (uses MD5 and RSA)
  - sender: m + MD5(m + k) + E(k, *private*) where k = random number
  - receiver
    - recovers random key using the sender's public key
    - applies MD5 to the concatenation of this random key message
- MD5 with RSA signature
  - sender: m + E(MD5(m), *private*)
  - receiver
    - decrypts signature with sender's public key
    - compares result with MD5 checksum sent with message

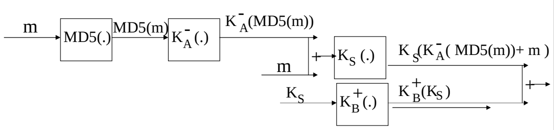
Key Distribution – a first step

- How can we be sure a key belongs to the entity that purports to own it?
- Solution = certificates
  - special type of digitally signed document:
    "I certify that the public key in this document belongs to the entity named in this document, signed X."
  - X is the name of the entity doing the certification
  - Only useful to the entity which knows the public key for X
  - Certificates themselves do not solve key distribution problem but they are a first step
- Certified Authority (CA)
  - administrative entity that issues certificates
  - useful only to someone that already holds the CA's public key.

Key Distribution (cont)

- Chain of Trust
  - if *X* certifies that a certain public key belongs to *Y*, and *Y* certifies that another public key belongs to *Z*, then there exists a chain of certificates from *X* to *Z*
  - someone that wants to verify *Z*'s public key has to know *X*'s public key and follow the chain
  - X.509 is a standard for certificates
- Certificate Revocation List
  - Means for removing certificates
  - Periodically updated by CA

Secure Email (PGP)



M: Message
K^+: public key
K^-: private key
K_S: secret key

Message from A to B