CS 640 Introduction to Computer Networks

Lecture23

Today's lecture

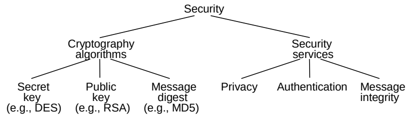
- Network security
  - Encryption Algorithms
  - Authentication Protocols
  - Message Integrity Protocols

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



- Cryptography algorithms – 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))

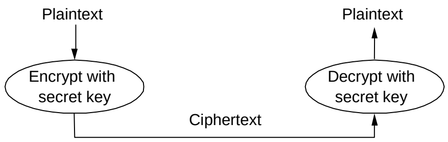
Issues in Security

- 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?
- Non-goal: details of crypto algorithms
  - We are not going to focus on proving anything about crypto algorithms
    - See CS642

Crypto 101

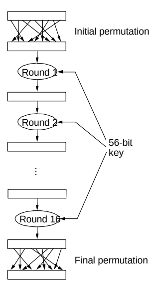
- Cryptographic algorithms determine how to generate encoded text (ciphertext) from plaintext using keys (string of bits)
  - Can only be decrypted by key holders
- Algorithms
  - Published and stable
  - Keys must be kept secret
  - Keys cannot be deduced
  - Large keys make breaking code VERY hard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

Secret Key (DES)

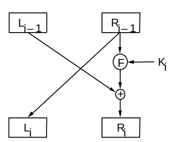


- Approach: Make algorithm so complicated that none of the original structure of plaintext exists in ciphertext

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

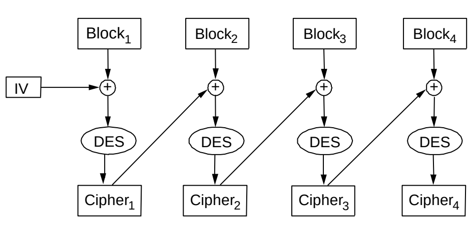


- Each Round

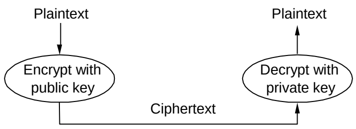


- L,R = 32 bit halves of 64 bit block
- K = 48 bits of 64 bit key
- F = combiner function
- + = XOR

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



Public Key (RSA)



- One of the coolest algorithms ever!
- Encryption
  - *ciphertext* $= c = m^e \mod n$ *($<e, n>$ = public key)*
- Decryption
  - *Message* $= m = c^d \mod n$ *($<d, n>$ = 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

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

- 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
- Usage
  - for privacy encrypt with recipient's *public* key and he decrypts with *private* key
  - for authentication encrypt with your *private* key and the recipient decrypts with your *public* 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.

CS 640

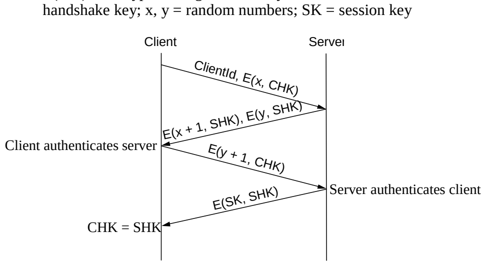
Today's lecture

- Network security
  - Encryption Algorithms
  - Authentication Protocols
  - Message Integrity Protocols

CS 640

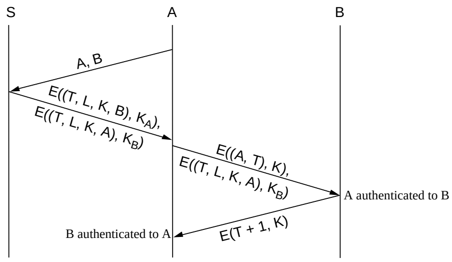
Authentication Protocols

- 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

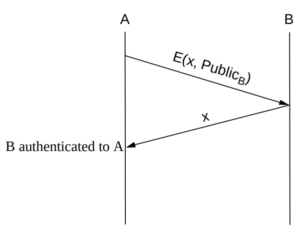


CS 640

- 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