Network Security

David Wetherall, with some slides from Radia Perlman's security lectures.
djw@cs.washington.edu

What do we mean by "Security"?

- Networks are shared:
  - Want to secure communication between legitimate participants from others with (passive or active) access to the network
- "Secure communication" can mean many things:
  - Other party (eavesdropper) can't read your messages [privacy*]
  - Or alter your messages in any way [integrity*]
  - Or impersonate someone you communicate with [authentication*]
  - Or exploit bugs to break into a system [access control]
  - Or deny service by consuming too many resources [availability]
  - Or learn who is communicating with whom [anonymity]
  (* = classic security threats that I'll talk about today)
- A threat model captures the attacks we seek to defend against.

Approaches at 10,000 ft

- Physical security
  - Tackle the problem of sharing directly with isolation
- "Security through obscurity"
  - Hope no-one will find out what you're doing!
- Throw math at the problem
  - Modern cryptography [*]

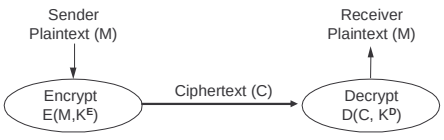
Cryptographic Schemes

- Secret key encryption
  - Secret is known by two parties
  - Block ciphers (3DES, AES, IDEA)
  - Stream ciphers (RC4 from WEP)
- Public key encryption
  - Users key is divided into private portion and public portion
  - RSA, Diffie-Hellman
- Hashes (or digests)
  - Take a variable length (long) message and turn it into a short (fixed length) identifier
  - MD5, SHA-1

Warning -- Difficulties in Practice

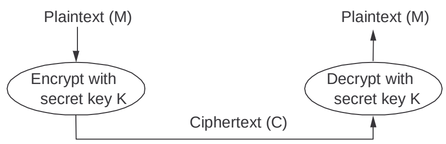
- It's a negative goal -- one flaw (typically due to invalid assumptions) can undermine the rest of the system:
  - Typically it's not that "your key was too short" or "math" …
  - Poor design that puts good crypto together the wrong way, e.g., WEP
  - Poor implementation of a good design, e.g., randomness, bugs
  - Poor usability of a good implementation, e.g., phishing
- It's really about cost-effectiveness, not absolutes
  - What's the value of undermining security?
    - consider WEP vs NSA
  - What's the least expensive way to do so?
    - build a supercomputer, break into a building, bribe/trick someone?

Encryption for Privacy



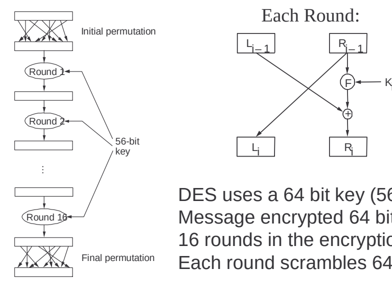
- Cryptographer chooses functions E, D and keys K^E, K^D
  - Mathematical (or at least computational) basis
- Cryptanalyst try to "break" the system
  - Depends on threat and what is known: E and D, M and C?
  - Brute force attack is computationally infeasible (not impossible)

Using Secret Key schemes (3DES, AES)



- Single key K (symmetric) is shared between parties
  - Often chosen randomly, but must be communicated
  - Schemes aim for diffusion (small changes in M cause large changes in C) and confusion (statistical relationship is complex) [Shannon]
  - Pretty much an art as far as I know.

Example: DES (Block Cipher)



DES uses a 64 bit key (56 + 8)
Message encrypted 64 bits at a time
16 rounds in the encryption
Each round scrambles 64 bits

DES Avalanche

```
Input:    ...............................................................*   1
Permuted: .......................................*........................   1
Round 1:  .......*........................................................   1
Round 2:  .*..*...*.....*........................*........................   5
Round 3:  .*..*.*.**..*.*.*.*...**....**.*..*...*.....*..................  18
Round 4:  ..*.*****.*.*****.*.*.....*.....*..*.*.**..*.*.*.*...**.....**  28
Round 5:  *...**..*.*...*.*.*.*...*.***..*..*.*****.*.*****.*.*......*....  29
Round 6:  ...*..**.....*.*..**.*.**...*..**...**..*.*...*.*.*.*...*.***..*  26
Round 7:  *****...***....**...*..*.*..*......*..**.....*.*..**.*.**...*..*
Round 8:  *.*.*.*.**.....*.*.*...**.*...*******...***....**...*..*.*..*...
Round 9:  ***.*.***...**.*.****.....**.*..*.*.*.*.**.....*.*.*...**.*...**
Round 10: *.*..*.*.**.*..*.**.***.**.*...****.*.***...**.*.****.....**.*..
Round 11: ..******......*..******....*....*.*..*.*.**.*..*.**.***.**.*...*
Round 12: *..***....*...*.*.*.***...****....******......*..******....*....
Round 13: **..*....*..******...*........*.*..***....*...*.*.*.***...****..
Round 14: *.**.*....*.*....**.*...*..**.****..*....*..******...*........*.
Round 15: **.*....*.*.*...*.**.*..*.*.**.**.**.*....*.*....**.*...*..**.**
Round 16: .*..*.*..*..*.**....**..*..*..****.*....*.*.*...*.**.*..*.*.**.*
Output:   ..*..**.*.*...*....***..***.**.*...*..*..*.*.*.**.*....*.*.*.**.
```

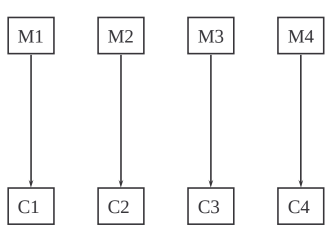
Source: *Willem de Graaf*, http://www-groups.dcs.st-and.ac.uk/~wdg/slides/node150.html

Encrypting large messages

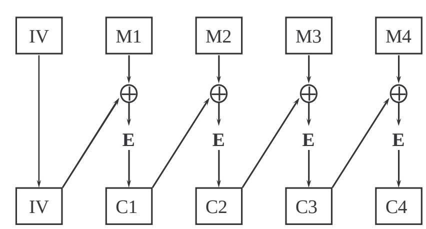
- What would happen if we encrypted each 64 bit block independently?
  - This is DES with ECB (Electronic Code Book) mode [over]
  - Repeated plaintext causes repeated ciphertext [privacy]
  - Also, very easy to move blocks and change plaintext [integrity]

- Instead, can chain blocks together in a variety of fashions to enhance privacy [but not integrity!]

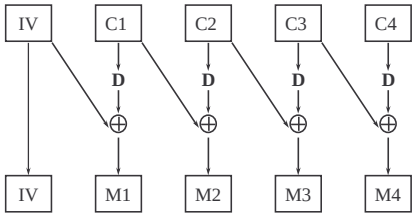
DES with ECB



CBC (Cipher Block Chaining)



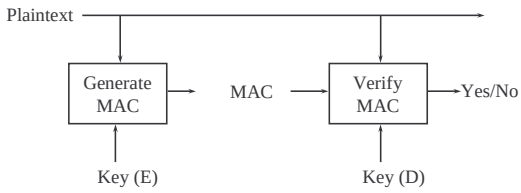
CBC Decryption



Creating a stream cipher

- Stream cipher:
  - Take an endless stream of random bits and encrypt by XORing this stream with the plaintext; XOR again to decrypt
  - Stream is a one-time pad; it can't be reused
- How do you get the stream?
  - One method is repeated encryption with a block cipher
  - DES in Output FeedBack (OFB) mode:
    - Start with IV (transmitted in the clear) and key
    - pad_1=e(IV, key), pad_2=e(pad_1, key), … , pad_i=e(pad_{i-1},key), …
- Stream can be generated in advance. Stream can't be reused or:
  - C1 xor C2 = M1 xor stream xor M2 xor stream = M1 xor M2

Encryption for Integrity (only!)



MAC is a Message Authentication Code (also MIC, I=integrity)

Using Secret Key for Integrity

- Example MAC = CBC "residue", the last cipher block of the CBC mode of DES encryption of M under K.
- Looks like a "cryptographic checksum". Why can't we use a regular checksum?

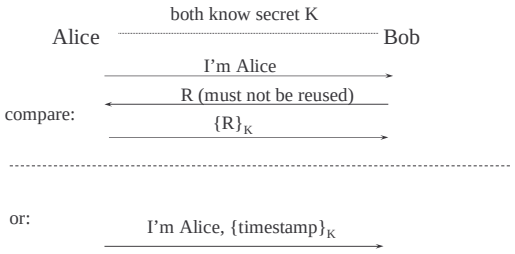
Privacy and Integrity Together

- Privacy does not guarantee integrity:
  - Any bag of bits decrypts to something; a human can see if the result has the right form, but we want a computer check
  - Depending on the scheme and knowledge, there can be a variety of attacks that alter messages
- Bottom line: to get both, encrypt with one key for integrity and another key for privacy (double encryption).

Authentication vs Integrity

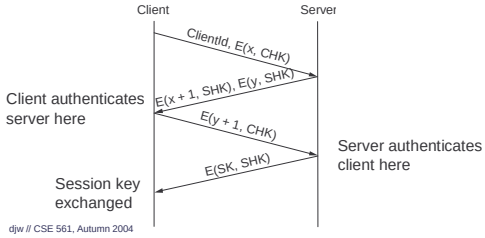
- Conventionally, authentication refers to verifying the identity of the sender or other party at a point in time, while integrity refers to checking the validity of messages. Authentication "factors":
  - What you know (password)
  - What you have (physical token)
  - What you are (biometrics)
- It gets messy when you involve people! Here, we consider only authenticating computers over a network with keys.
- The whole point of using crypto: Checking whether someone else knows a secret without giving away the secret. (Very cool that you can do this!)

Authentication with Secret Key



A more complete protocol

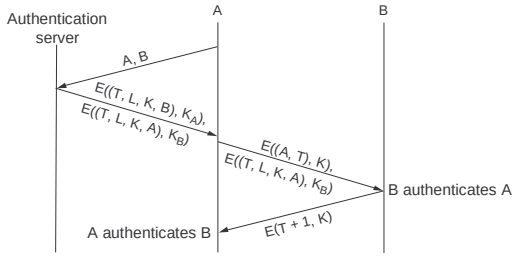
- Three-way handshake for mutual authentication
  - Client and server share secrets, e.g., login password



Problem: how to set up keys?

- Need one key for each pair of hosts, O(N^2)
- Instead, have each host share a secret with a trusted party, O(N). That party provide new secrets between a pair as needed.
  - The party is a Key Distribution Center (KDC)

Example: Kerberos



Cryptographic hashes

- Takes arbitrary sized input, generates fixed size output, e.g., 128 bits for MD5. No key necessary.
- Properties:
  - one-way (computationally infeasible to find input for a particular hash value)
  - collision-resistant (can't find two inputs that yield same hash)
  - output should look "random"
- How can this possibly work?
  - Universe of input messages is << $2^{\text{hash-length}}$
  - Design is an art as far as I know

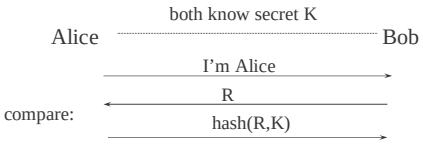
Creating Encryption with Hash

- Create one-time pad with repeated hashing:
  - pad_1=hash(K,IV); must send IV in the clear
  - pad_2=hash(K,pad_1)
  - pad_i=hash(I,pad_{i-1})
- Now just use as a stream cipher

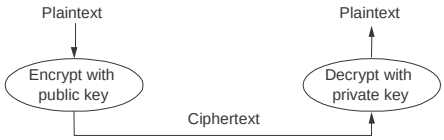
MACs with hashes for integrity

- Combine message with key and digest that.Could do:
  - MAC = hash(key | message)
- But some hashes have the property that they can be "continued" where they are left off, e.g., MD5
- So instead use the HMAC construction:
  - MAC = hash(K xor O | hash(K xor I | message))
  - I and O are constants and there is double hashing

Authentication with Hash



Privacy with Public Keys (e.g., RSA)

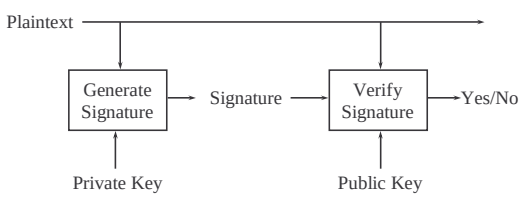


- Public and private key are "mathematical inverses"
- Public key can be published; private is a secret
  - Anyone can encrypt, only holder of the private key can decrypt

Overview of RSA

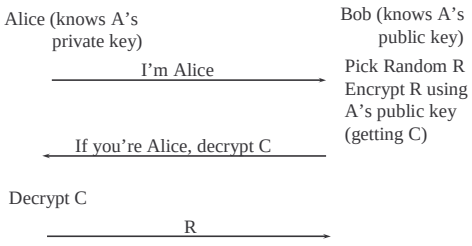
- Based on modular exponentiation. Named after its inventors: Rivest, Shamir, and Adelman.
- Choose modulus n, public exponent e, and related private exponent d. Public key is (e, n), private is (d, n)
  - n = pq, p and q are large primes [needs primality test]
  - e and d are such that e * d = 1 mod (p-1)(q-1) [use Euclid for d]
  - Difficulty is that d is hard to compute without factoring n
- Finally:
  - encryption: ciphertext = plaintext^{e} mod n
  - decryption: plaintext = ciphertext^{d} mod n
  - Show that they match using Euler's rule, $x^{phi(n)}$ mod n = 1

Public Key Integrity (Digital Signatures)



We reverse the roles of the keys; now only holder of the private key could have sent the message.

Public Key Authentication



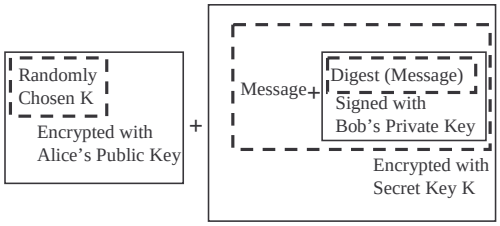
Problem: how to set up keys

- Hosts have to know each other's public keys. They get these from a trusted intermediary, the Certification Authority (CA)
  - CA signs that X's public key is Y; this is a certificate
  - Note CA can be offline while KDC must be online
  - But must also have revocation lists
  - Also called public key infrastructure (PKI)
- Example: Verisign and trust anchors in browsers; X.509 format certificates

Problem: public key is slow

- Encryption is maybe 1000X slower than secret key.
- So we can combine both of them to speed the system up
- Privacy:
  - Secret key encrypt message with random key K for privacy; public key encrypt and send K (much shorter than message)
- Integrity:
  - Generate MAC using secret key (or hash); public key sign MAC
- Authentication:
  - Use public key and mint a session key for secret key usage

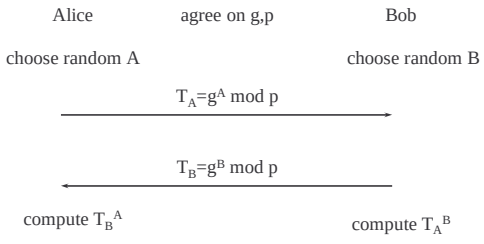
Signed and Encrypted Message



Diffie-Hellman Key Exchange

- Allows two individuals to agree on a secret key, even though they can only communicate in public!
- Alice chooses a private number and from that calculates a public number. Bob does the same.
- Each can use the other's public number and their own private number to compute the same secret
- An eavesdropper can't reproduce it
- Useful for "perfect forward secrecy" or escrow foilage
  - All messages plus long-term host secrets don't reveal DH secret.

Diffie-Hellman



Both sides compute $g^{AB} \bmod p$ as secret key. Attacker Can't do so without recovering X from $g^X \bmod p$ (hard).

Example Systems

- Secure Shell (ssh)
  - For secure logins
- Secure Sockets (SSL) and Secure HTTP (HTTPS)
  - For secure Web transactions
- IP Security (IPSEC)
  - Framework for encrypting/authenticating IP packets
- WEP
  - "Wired equivalent privacy" for wireless networks

ssh

- (this is woefully incomplete)
- Client authenticates the server using the server's public key. Bootstraps by letting the user accept a public key from the server and warning if it changes.
- Server authenticates the client using public key or password, as configured.
- Session keys are set up and encryption options are negotiated

SSL/TLS and HTTPS

- Secure transport layers targeted at Web transactions
  - SSL/TLS inserted between TCP and HTTP to make secure HTTP
- Extra handshake phase to authenticate and exchange shared session parameters
  - Such as secret keys used for encryption
  - Client might authenticate Web server but not vice-versa
    - Certificate Authority embedded in Web browser
- Performance optimization
  - Refer to shared state with session id
  - Can use same parameters across connections
    - Client sends session id, allowing server to skip handshake

IPSEC

- Framework for encrypted and authenticated IP packets
  - Choice of algorithms not specified
- Uses new protocol headers inside IPv4 packets
  - Authentication header
    - For message integrity and origin authenticity
    - Optionally "anti-replay" protection (via sequence number)
  - Encapsulating Security Payload
    - Adds encryption for privacy
- Depends on key distribution (IKE)
  - Sets up security associations
- Example use: secure tunnels between corporate offices

WEP

- All nodes share a key K
- Packets are sent encrypted with a stream cipher based on K and an IV (which is sent in the clear)
- A "MAC" (that is really a CRC) is added to packets for integrity
- WEP is seriously broken due to design and implementation errors.