Transport Security
Part I

Lecture 12, Computer Networks (198:552)
Fall 2019
Why security?

- Malicious people share your network
  - People who want to snoop, corrupt, destroy, pretend, steal, …

- Problem made more severe as Internet becomes more commercialized

- Active and passive attacks
Key aspects of network security

**Confidentiality:** only sender, intended receiver should “understand” message contents
- sender encrypts message
- receiver decrypts message

**Integrity:** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

**Authentication:** sender, receiver want to confirm identity of each other

**Non-repudiation:** Once someone sends a message, or conducts a transaction, she can’t later deny the contents of that message
Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob and Alice want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages
Who might Bob and Alice be?

- Real humans 😊
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
What can bad actors do?

A lot!

• **eavesdrop**: intercept messages
• actively **insert** messages into connection
• **impersonation**: can fake (spoof) source address in packet (or any field in packet)
• **hijacking**: “take over” ongoing connection by removing sender or receiver, inserting itself in place
• **denial of service**: prevent service from being used by others (e.g., by overloading resources)
Confidentiality

Cryptography: preventing adversaries from reading private messages
m plaintext message
$c = K_A(m), K_A(m)$ ciphertext, encrypted with key $K_A$
m’ = $K_B(c), K_B(c)$ decrypted plaintext with key $K_B$
Want: $m = K_B(K_A(m))$
Want: $K_A(m)$ to be uncorrelated with $m$

En/decryption algorithms are also called ciphers.
Cryptography: Algorithms and Keys

• Cryptography requires both an en-/decryption algorithm and keys
  • Key is a string known only to Alice and Bob, which controls how algorithm works

• Algorithm should be public and known to all
  • Inspires trust that the algorithm works

• Keys
  • Should be long enough to prevent easy breaking of the encryption
  • Should be short enough to keep algorithm efficient
  • Typical key lengths: 56-bit, 128-bit, 256-bit, 512-bit
Symmetric key cryptography

Symmetric keys: Bob and Alice share same (symmetric) key: $S$

Main techniques of symmetric key cryptography:
Substitution and Permutation

Q: how do Bob and Alice agree on key value?
How to agree on a shared secret key?

• In reality: two parties may meet in person or communicate “out of band” to exchange shared key

• But communicating parties may never meet in person
  • Example: An online retailer and customer
  • Much more common for a network 😊

• What if the shared secret is stolen?
  • All secret communications can now be decrypted and are visible
  • Including earlier ones that were encrypted using that secret

• How to communicate without necessitating key exchange?

Public key cryptography
Public Key Cryptography

- Sender and receiver do not share secret key
- **public** encryption key known to all
- **private** decryption key known only to the receiver
Public key cryptography (eg: RSA)

plaintext message, m

encryption algorithm

ciphertext

K^+(m)

decryption algorithm

plaintext message

m = K^-_B(K^+_B(m))

Bob’s public key

Bob’s private key
Diffie Hellman Merkle key exchange

• Alice and Bob agree on a modulus $p$ and base $g$

• Alice chooses secret $a$, sends $A = g^a \mod p$

• Bob chooses secret $b$, sends $B = g^b \mod p$

• Alice computes $B^a \mod p$
• Bob computes $A^b \mod p$

• Is the common key computed by Alice and Bob the same?

• In what sense is D-H-M key exchange secure?
Public vs. Symmetric key crypto

- Public key crypto
  - Expensive to encrypt using just modular exponentiation operations
  - No need to exchange keys

- Symmetric key crypto
  - Encryption and decryption are fast
  - But need to solve the key exchange problem
Crypto in practice: session keys

Use public key crypto or key exchange to agree on a symmetric session key

Use symmetric key to protect the rest of the session efficiently
Integrity

Did messages get across without tampering?
Message digests

Can we ensure that a receiver can detect message tampering?

**Idea:** fixed-length, easy-to-compute digital “fingerprint” of a message

- apply hash function $H$ to $m$, get fixed size message digest, $H(m)$.

Cryptographic hash function properties:
- Easy to calculate
- Produces fixed-size msg digest (fingerprint)
- Hard to reverse: given msg digest $x$,
  - computationally infeasible to find $m$ such that $x = H(m)$
  - Or another $m'$ such that $H(m) = H(m')$
Using message digests for integrity

- Verifies message integrity
- Requires a secret shared key
- No encryption
Message digest algorithms

• You’ll see the term “MAC” or Message Authentication Codes
  • I find it confusing (medium access); I will avoid using it.

• MD5 hash function widely used (RFC 1321)
  • computes 128-bit message digest in 4-step process.
  • arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x

• SHA-1 is also used
  • US standard [NIST, FIPS PUB 180-1]
  • 160-bit message digest
Digital signatures

Cryptographic technique analogous to handwritten signatures:

• sender (Bob) digitally signs document, establishing he is document owner/creator.

• verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
simple digital signature for message m:

- Bob signs m by encrypting with his private key $K_B^-$, creating “signed” message, $K_B^-(m)$
Authentication

How do I know you are who you say you are?
Authentication using public key crypto

Idea: Use **nonce** and public key cryptography

“I am Alice”

Bob computes

\[ K_A^+(K_A^-(R)) = R \]

and knows only Alice could have the private key, that encrypted \( R \) such that

\[ K_A^+(K_A^-(R)) = R \]
Security hole -- if you ask for public keys!

*man (or woman) in the middle attack:* Trudy poses as Alice (to Bob) and as Bob (to Alice)

\[
m = K_A^-(K_A^+(m))
\]

Trudy gets

\[
sends m to Alice\]

encrypted with Alice’s public key

\[
K_T^+(m)
\]

Send me your public key

\[
K_T^-(R)
\]

R

\[
K_A^-(R)
\]
Certification authorities

• **certification authority (CA):** binds public key to particular entity, E.
• E (person, router) registers its public key with CA.
  • E provides “proof of identity” to CA.
  • CA creates certificate binding E to its public key.
  • certificate containing E’s public key digitally signed by CA – CA says “this is E’s public key”
Certification authorities

- When Alice wants Bob’s public key:
  - gets Bob’s certificate (from Bob or elsewhere).
  - apply CA’s public key to Bob’s certificate, get Bob’s public key