Network Security: Threats and goals

Tuomas Aura

T-110.5240 Network security
Aalto University, Nov-Dec 2011

Outline

1. Network security
2. Basic network threats: sniffing and spoofing
3. Role of cryptography
4. Security and the network protocol stack
5. First security protocols: replay and freshness

Network security

What is network security

- Network security protects against intentional bad things done to communication
  - Protect messages (data on wire) and communication infrastructure
- Network security goals:
  - Confidentiality — no sniffing
  - Authentication and integrity — no spoofing of data or signaling, no man-in-the-middle attacks
  - Access control — no unauthorized use of network resources
  - Availability — no denial of service by preventing communication
  - Privacy — no traffic analysis or location tracking

Authentication and integrity

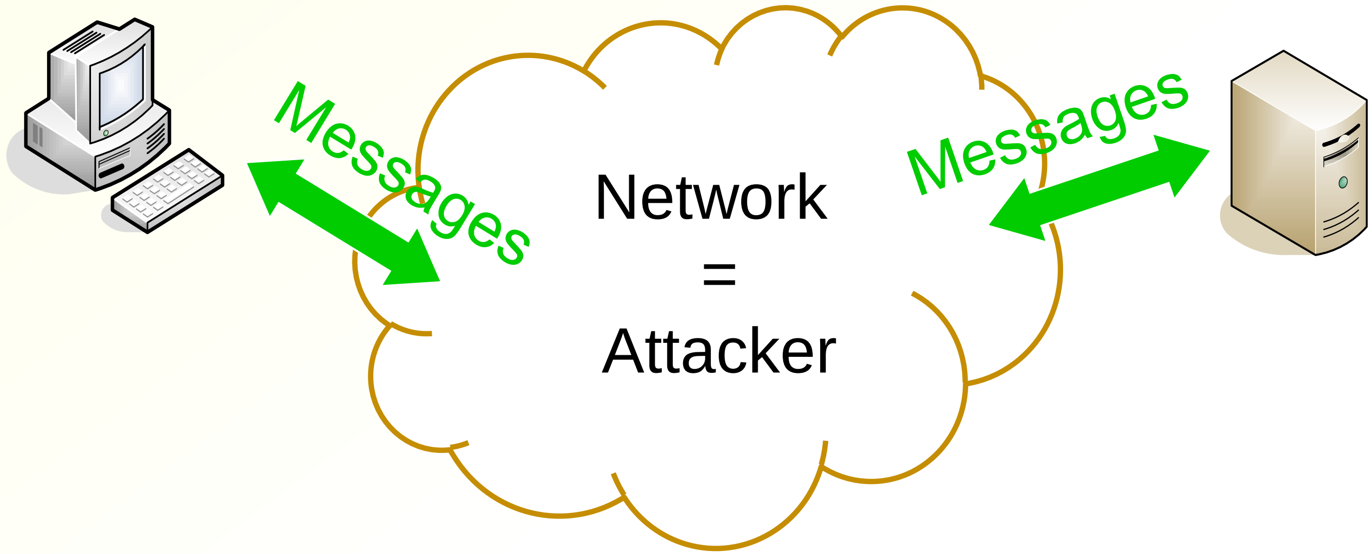
- Peer-entity authentication = verify that presence and identity of a person, device or service at the time; e.g. car key
- Data origin authentication = verify the source of data
- Data integrity = verify that the data was received in the original form, without malicious modifications
- In practice, data origin authentication and integrity check always go together
- Authentication (usually) requires an entity name or identifier

Who is the attacker?

- We partition the world into good and bad entities
  - Honest parties vs. attackers
  - Good ones follow specification, bad ones do not
  - Different partitions lead to different perspectives on the security of the same system
- Typical attackers:
  - Curious or dishonest individuals — for personal gain
  - Hackers, crackers, script kiddies — for challenge and reputation
  - Political activists — for political pressure
  - Companies — for business intelligence and marketing
  - Security agencies — NSA, FAPSI, GCHQ, DGSE, etc.
  - Military SIGINT — strategic and tactical intelligence, cyber-war
  - Organized criminals — for money
- Often, not all types of attackers matter
  - E.g. would you care if NSA/university/mom read your email?

Basic network threats: sniffing and spoofing

Traditional network-security threat model



- End are nodes trusted, the network is unreliable
- End nodes send messages to the network and receive messages from it
- Network will deliver some messages but it can read, delete, modify and replay them
- Metaphors: unreliable postman, notice board, rubbish basket

Example: email

- Alice sends an email that is addressed to Bob
- The attacker may read, delete and edit the email. It may copy the email, or cut and paste pieces from one email to another. It may write a new email
- Secrets and message integrity need protection

Basic network security threats

- Traditional major threats:
  - Sniffing = attacker listens to network traffic
  - Spoofing = attacker sends unauthentic messages
  - Data modification (man in the middle) = attacker intercepts and modifies data
- Corresponding security requirements:
  - Data confidentiality
  - Data-origin authentication and data integrity

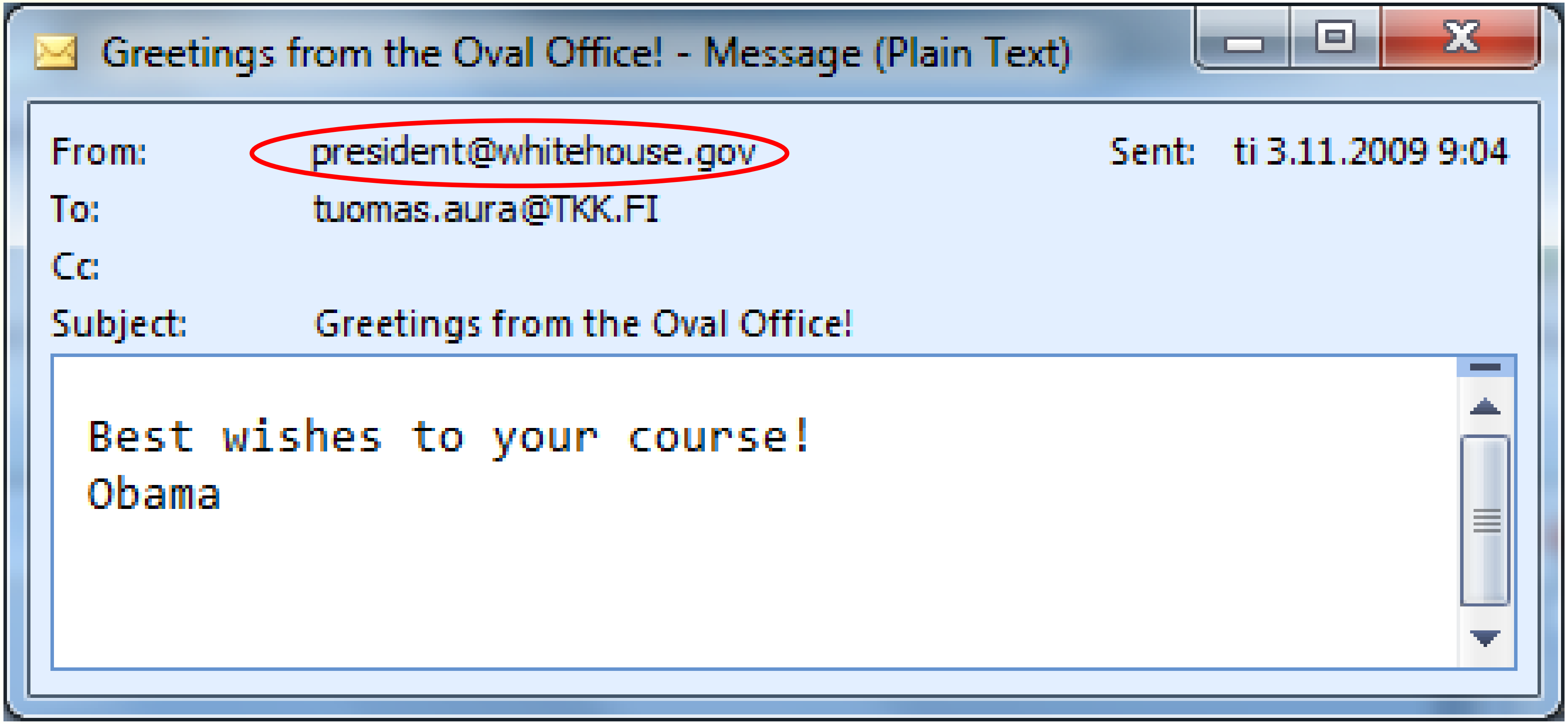
Sniffing

- Sniffing = eavesdropping = spying = unauthorized listening = monitoring
- Sniffers:
  - Packets are often broadcast on a local link
    → all local nodes can listen
  - Sniffers listen to packets on the network and pick out interesting details, e.g. passwords
  - Hackers install sniffer software on compromised hosts; tools are available for download
  - Wireless Ethernet is most vulnerable but tools exist on sniffing all types of networks
- Network admins and spies can monitor packets on routers, firewalls and proxies
  - Router security may become a serious issues

Spoofing

- Spoofing = sending unauthentic messages
  = using false sender address or identifier
- In the Internet, it is easy to send messages that appear to come from someone else
  - A modified version of the application or protocol stack is easy to write
- Examples:
  - Email spoofing: false From field
  - IP spoofing: false source IP address
  - DNS spoofing: false DNS responses
  - Mobile-IP BU spoofing: false location information

Example: email spoofing



Example: email spoofing

- SMTP does nothing to authenticate the sender

```
C:>telnet smtp.kolumbus.fi 25
220 emh05.mail.saunalahti.fi ESMTP Postfix
ehlo nowhere.net
250-emh05.mail.saunalahti.fi
250-PIPELINING
250-SIZE 280000000
250-8BITMIME
mail from: president@whitehouse.gov
250 2.1.0 Ok
rcpt to: tuomas.aura@tkk.fi
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
From: president@whitehouse.gov
To: tuomas.aura@tkk.fi
Subject: Greetings from the Oval Office!

Best wishes to your course!
Obama
.
250 2.0.0 Ok: queued as 9935A27D8C
```

Example: IP spoofing

- Attacker sends IP packets with false source address
  - Anyone can write software to do this with raw sockets
- The destination node usually believes what it sees in the source address field
- Attacker may be anywhere on the Internet
- Spoofing a connection is more difficult:
  - Attacker must sniff replies from B in order to continue the conversation

  ➔ Attacker must be on the route between A and B, or control a router on that path

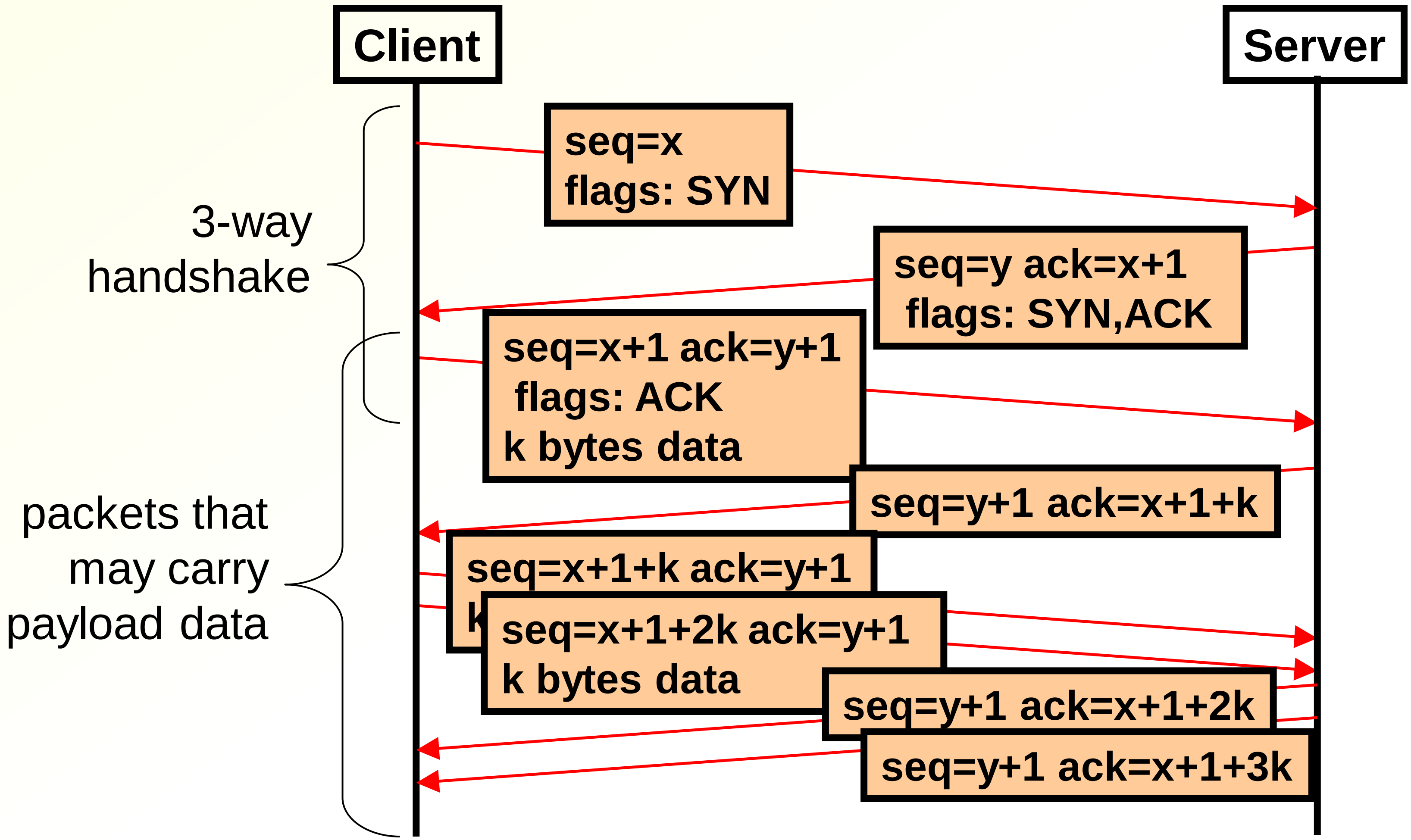
TCP sequence numbers and IP spoofing

- TCP sequence numbers are initialized to random values during the connection handshake
- Acknowledgment number in the third packet must be sequence number of the second packet + 1
- Sequence numbers are incremented for each byte sent. Packets must arrive in order
- Receiver rejects packets with incorrect sequence numbers and waits for the correct ones

→ TCP packets are difficult to spoof because the attacker must sniff or guess the sequence number

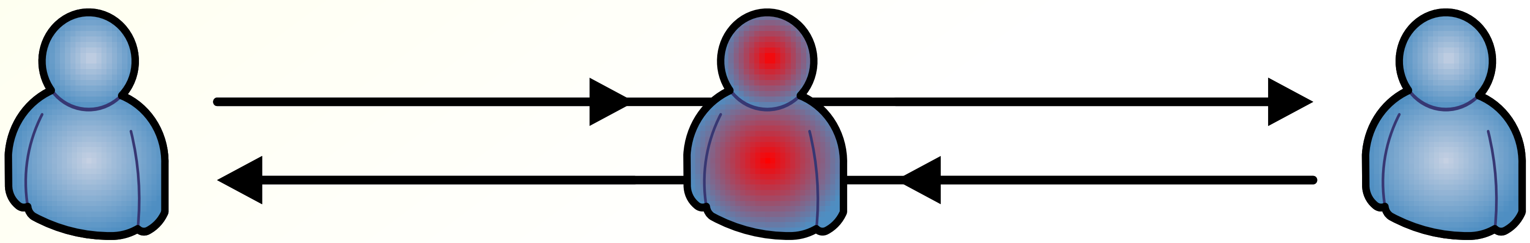
- Not secure in the traditional network security threat model, but limits attack quite well
- The first packet (SYN) is easy to spoof

TCP handshake



Man in the middle (MitM)

- In the man-in-the-middle attack, the attacker is between the honest endpoints



- Attacker can intercept and modify data
→ combines sniffing and spoofing
- On the Internet, a MitM attacker must
  - be at the local network of one of the end points
  - be at a link or router on the route between them, or
  - change routing to redirect the packets via its own location
- Note: Just forwarding data between two endpoints (like a piece of wire) is not an attack. What does the attacker gain?

Other network threats

- What other threats and security requirements are there on open networks?
- Other threats:
  - Unauthorized resource use (vs. access control)
  - Integrity of signalling and communications metadata
  - Denial of service (DoS) (vs. availability)
  - Traffic analysis, location tracking
  - Lack of privacy
  - Software security
- Not captured well by the traditional network-security model

Role of cryptography

Cryptographic primitives

- Symmetric (shared-key) encryption for data confidentiality
  - Block and stream ciphers, e.g. AES-CBC, RC4
- Cryptographic hash function
  - E.g. SHA-1, SHA256
- Message authentication code (MAC) for data authentication and integrity
  - E.g. HMAC-SHA-1
- Public-key (or asymmetric) encryption
  - E.g. RSA
- Public-key signatures
  - E.g. RSA, DSA
- Diffie-Hellman key exchange
- Random number generation

Crypto Wars – some history

- Until ‘70s, encryption was military technology
  - In ‘70s and ‘80s, limited commercial applications
  - American export restrictions and active discouragement prevented wide commercial and private use
- Reasons to ban strong encryption:
  - Intelligence agencies (e.g. NSA) cannot spy on encrypted international communications
  - Criminals, terrorists and immoral people use encryption
- In ‘90s: PGP, SSL, SSH and other commercial and open-source cryptography became widely available
  - Activists argued that cryptography was a tool for freedom
  - Researchers argued that weak crypto is like no crypto
- Most export restrictions were lifted in 2000

Network security mechanisms

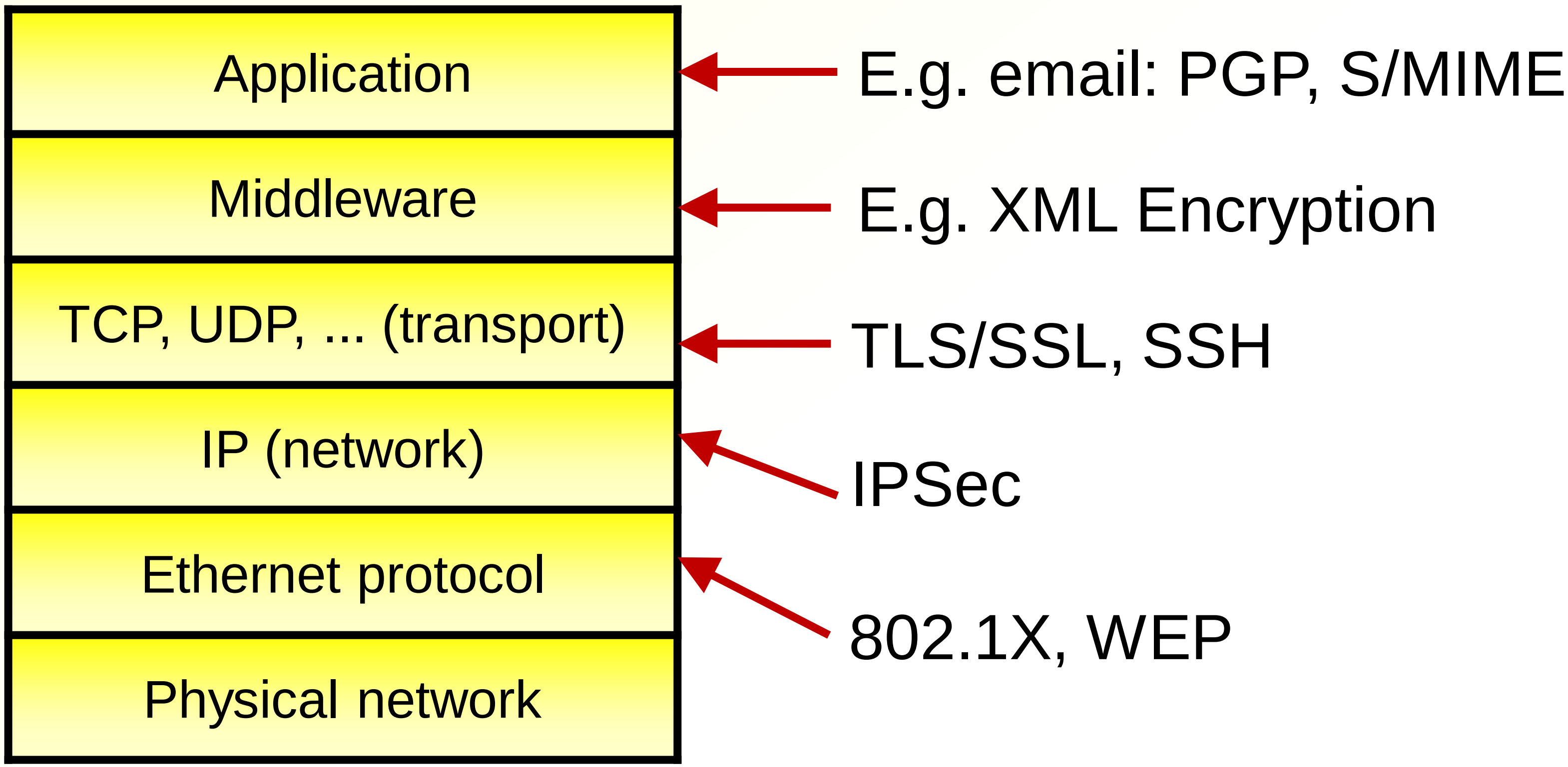
- Cryptography is the main building block for security protocols, but not the only security mechanism
- Strong cryptography:
  - Encryption → confidentiality
  - Cryptographic authentication → authentication and integrity
- Non-cryptographic security mechanisms:
  - Perimeter defense (firewalls)
  - Routing-based semi-secure solutions
  - Over-provisioning
  - Preventing attacks at source
  - Proxies and pseudonyms
  - Intrusion detection
- Non-technical solutions:
  security is also a social, legal and business problem
  (but that is not the topic of this course)

Security vs. cryptography

- However:
  "Whoever thinks his problem can be solved using cryptography, doesn't understand the problem and doesn't understand cryptography." —
  attributed to Roger Needham and Butler Lampson

Security and the network protocol stack

Protocol Stack and Security



- Security solutions exist for every protocol layer
- Layers have different security and performance trade-offs, trust relations and endpoint identifiers

Which layer security?

- Security mechanisms exist for all protocol layers
  - Which layer is right for encryption and authentication?
  - Which layer PDUs should a firewall filter or an IDS monitor?
- Reasons to implement cryptographic security in lower layers:
  - Security provided by physical, link or network layer is a service to all higher layers
  - Lower-layer security protects *all* higher-layer data
  - Security in lower layers is transparent to higher layers. No changes to applications needed
  - Lower-layer security protects the lower layer, too
- Reason to implement security in higher layers:
  - Security implemented in the application or middleware will fit exactly to the application requirements
  - Lower-layer identifiers may not be meaningful to higher layers
- Actually, we may need independent security in multiple network-stack layers

End-to-end security

- Security should be implemented between the endpoints of communication. All intermediaries are part of the untrusted network
- End-to-end security only depends on the end nodes
  - Hop-by-hop (link-layer) security assumes all routers are trusted and secure
- End-to-end security protocols are independent of the network technology at intermediate links
  - Link-layer security is different for each link type
- Confidentiality and authentication are usually user or application requirements
  - Network or link layer only cannot know application-level requirements
- But link and network layer infrastructure and signalling need protection, too

Endpoint names

- Authentication and integrity depend on names (identifiers)
- Each protocol layer has its own way of naming endpoints:
  - Ethernet (MAC) addresses in the link layer (e.g. 00-B0-D0-05-04-7E)
  - IP address in the network layer (e.g. 157.58.56.101)
  - TCP port number + IP address
  - DNS or NetBIOS name in the higher layers (e.g. vipunen.tkk.fi)
  - URI in web pages and services (e.g. http://www.example.org/myservice)

Using identifiers

- How are names and other identifiers allocated?
  - Authority, random allocation, ...
- What is the scope of the identifiers and are they unique?
- How does one find the owner of a name?
  - Data delivery, routing
  - Resolving name in one protocol layer to the name space of the layer below
- How to convince others that this is your name?
  - Authentication, authorization, name ownership
- Secure naming is a difficult problem and often leads to vulnerabilities

First security protocols: replay and freshness

The first broken protocol

- Meet Alice and Bob!

  $A \rightarrow B$: M, $S_A(M)$

  E.g., S_A(“Attack now!”)
- What is wrong with this protocol?

Replay and freshness

- Replay problem:

  A → B: M, S_A(M)   // S_A("Attack now!")
- Authentication is usually not enough in network security! Need to also check freshness of the message
- "Fresh" may mean that the message was sent recently, or that has not been received before (exact definition depends on application)
- Freshness mechanisms:
  - Timestamp
  - Nonce
  - Sequence number

Timestamps

- Checking freshness with A's timestamp:
  $A \rightarrow B$: T_A, M, $S_A(T_A, M)$
  E.g. S_A("2010-11-03 14:15 GMT", "Attack now!")
- Timestamp implementations:
  - Sender's clock value and time zone (validity ends after fixed period)
  - Validity period start and end times (or start and length)
  - Validity period end time
- Q: What potential problems remain?
  - Timestamps require clocks at the signer and receiver, and secure clock synchronization
  - Secure fine-grained synchronization is hard to achieve; loose synchronization (accuracy from minutes to days) is easier
  - Also, fast replays possible: $S_A(T_A$, "Transfer £10.")

Nonces

- What if there are no synchronized clocks?
- Checking freshness with B's nonce:

  A → B: "Hello, I'd like to send you a message."

  B → A: N_B

  A → B: N_B, M, $S_A(N_B, M)$
- Alice's nonce is a bit string selected by Alice, which is never reused and (usually) must be unpredictable
- Nonce implementations:
  - 128-bit random number (unlikely to repeat and hard to guess)
  - timestamp concatenated with a random number (protects against errors in RNG initialization and/or clock
  - hash of a timestamp and random number
- Problematic nonces: sequence number, deterministic PRNG output, timestamp
- Nonces require extra messages and are not well suited for asynchronous or broadcast communication

Sequence numbers

- What if there are no synchronized clocks and nonces do not fit into the protocol design?
- Sequence numbers in authenticated messages allow the recipient to detect message deletion, reordering and replay

  $A \rightarrow B$: seq, M, S_A(seq, M)

  E.g. S_A(44581, “Transfer 30€ to account 1006443.”)
- Dangerous, but can sometimes ensure that messages are not processes out of order or twice
- Good combination: timestamp from a loosely synchronized clock and sequence number

Unambiguous message encoding

- Many security protocols protect opaque upper-layer data
- Messages with many data fields must have unambiguous decoding and meaning:
  - E.g. “Send £100 to account 7322323.”
    vs. “100”,“7322323”
    vs. “1007”,“322323”
    vs. “£100 a/c 7322323”
- Some encodings:
  - Concatenation of fixed-length bit fields
  - Self-delimiting encodings, such as ASN.1 DER and other type-length-value (TLV) formats

Protocol engineering

- Network is a distributed system with many participants
- Computer networking is about protocols
  - Protocol = distributed algorithm
  - Algorithm = stepwise instructions to achieve something
- Security is just one requirement for network protocols
  - Cost, complexity, performance, deployability, time to market etc. may override perfect security
- Like the design of cryptographic algorithms, security engineering requires experienced experts and peer scrutiny
  - Reuse well-understood solutions; avoid designing your own
- The most difficult part is understanding the problem
  - Must understand both security and the application domain
  - When the problem is understood, potential solutions often become obvious

Puzzle of the day

- What should be the order of signing, compression and encryption?

Related reading

- William Stallings, Network security essentials: applications and standards, 3rd ed.: chapter 1
- William Stallings, Cryptography and Network Security, 4th ed.: chapter 1
- Dieter Gollmann, Computer Security, 2nd ed.: chapter 13
- Ross Anderson, Security Engineering, 2nd ed.: chapter 6

Exercises

- Design a more spoofing-resistant acknowledgement scheme to replace TCP sequence numbers. Hint: use random numbers (and maybe hashes) to ensure that acknowledgements can only be sent by someone who has really seen the packets
- Which applications of hash functions in network protocols require strong collision resistance? Which do not?
- Why is link-layer security needed e.g. in WLAN or cellular networks, or is it?
- To what extent are the identifiers in each protocol layer of the TCP/IP unique? Does one layer in the protocol stack know the identifiers of other layers?
- How do the properties of these identifiers differ:
  IP address, DNS name, email address, person’s name, national identity number (HETU)