Crypto protocols

ACS R209: Computer Security –
Principles and Foundations
Ross Anderson

Security Protocols

• Security protocols are the intellectual core of security engineering
• They are where cryptography and system mechanisms meet
• They allow trust to be taken from where it exists to where it’s needed
• But they are much older then computers…
Real-world protocol

• Ordering wine in a restaurant
  – Sommelier presents wine list to host
  – Host chooses wine; sommelier fetches it
  – Host samples wine; then it’s served to guests

• Security properties?

Real-world protocol

• Ordering wine in a restaurant
  – Sommelier presents wine list to host
  – Host chooses wine; sommelier fetches it
  – Host samples wine; then it's served to guests

• Security properties
  – Confidentiality – of price from guests
  – Integrity – can't substitute a cheaper wine
  – Non-repudiation – host can't falsely complain
Car unlocking protocols

- Principals are the engine controller E and the car key transponder T
- Static (T → E: KT)
- Non-interactive
  \[ T \rightarrow E: T, \{T,N\}_KT \]
- Interactive
  \[ E \rightarrow T: N \]
  \[ T \rightarrow E: \{T,N\}_KT \]
- N is a 'nonce' for 'number used once’. It can be a serial number, a random number or a timestamp

Two-factor authentication

\[ S \rightarrow U: N \]
\[ U \rightarrow P: N, PIN \]
\[ P \rightarrow U: \{N, PIN\}_KP \]
Key management protocols

- Suppose Alice and Bob each share a key with Sam, and want to communicate?
  - Alice calls Sam and asks for a key for Bob
  - Sam sends Alice a key encrypted in a blob only she can read, and the same key also encrypted in another blob only Bob can read
  - Alice calls Bob and sends him the second blob
- How can they check the protocol’s fresh?

Needham-Schroder

- 1978: uses ‘nonces’ rather than timestamps
  \[A \rightarrow S: A, B, NA\]
  \[S \rightarrow A: \{N_A, B, K_{AB}, \{K_{AB}, A\}_{KBS}\}_{KAS}\]
  \[A \rightarrow B: \{K_{AB}, A\}_{KBS}\]
  \[B \rightarrow A: \{NB\}_{KAB}\]
  \[A \rightarrow B: \{NB - 1\}_{KAB}\]
- The bug, and the controversy…
Identify Friend or Foe (IFF)

- Basic idea: fighter challenges bomber
  \[ F \rightarrow B: N \]
  \[ B \rightarrow F: \{N\}_K \]

- What can go wrong?

Identify Friend or Foe (IFF)

- Basic idea: fighter challenges bomber
  \[ F \rightarrow B: N \]
  \[ B \rightarrow F: \{N\}_K \]

- What if the bomber reflects the challenge back at the fighter's wingman?
  \[ F \rightarrow B: N \]
  \[ B \rightarrow F: N \]
  \[ F \rightarrow B: \{N\}_K \]
  \[ B \rightarrow F: \{N\}_K \]
A normal EMV transaction

1. Card details; digital signature
2. PIN entered by customer
3. PIN entered by customer; transaction description
4. PIN OK (yes/no); authorization cryptogram
5. Online transaction authorization (optional)

result

ISSUER

merchant

customer

PIN
The ‘No-PIN’ attack (2010)

- In theory: might block at terminal, acquirer, issuer
- In practice: may have to be the issuer (as with terminal tampering, acquirer incentives are poor)
- Barclays introduced a fix July 2010; removed Dec 2010 (too many false positives?); banks asked for student thesis to be taken down from web instead
- Real problem: EMV spec now far too complex
- With 100+ vendors, 20,000 banks, millions of merchants … everyone passes the buck (or tries to sell ECC…)

Fixing the ‘No PIN’ attack
Hardware Security Modules

API Attacks

- A typical HSM has 50–500 API calls!
- We found that evil combinations of API calls, or API calls with wicked data, can very often break the security policy
- E.g. HSM transaction defined by VISA for EMV for encrypted messaging between a bank and a chip card
- Send key from HSM to card or other HSM as \{text | key\} – where text is variable-length
- Attack – a bank programmer can encrypt \{text | 00\}, \{text | 01\}, etc to get first byte of key, and so on
- API vulnerabilities can turn up in multiple products, so are important to find – but are still hard to find formally
Public Key Crypto Revision

- Digital signatures: computed using a private signing key on hashed data
- Can be verified with corresponding public verification key
- Can't work out signing key from verification key
- Typical algorithms: DSA, elliptic curve DSA
- We'll write $\text{sig}_A\{X\}$ for the hashed data $X$ signed using A’s private signing key

Public Key Crypto Revision (2)

- Public key encryption lets you encrypt data using a user's public encryption key
- She can decrypt it using her private decryption key
- Typical algorithms Diffie-Hellman, RSA
- We’ll write $\{X\}_A$
- Big problem: knowing whose key it is!
PKC Revision – Diffie-Hellman

• Diffie-Hellman: underlying metaphor is that Anthony sends a box with a message to Brutus
• But the messenger’s loyal to Caesar, so Anthony puts a padlock on it
• Brutus adds his own padlock and sends it back to Anthony
• Anthony removes his padlock and sends it to Brutus who can now unlock it
• Is this secure?

PKC Revision – Diffie-Hellman (2)

• Electronic implementation:
  A → B: \( M^r_A \)
  B → A: \( M^r_A B \)
  A → B: \( M^r_B \)
• But encoding messages as group elements can be tiresome so instead Diffie-Hellman goes:
  A → B: \( g^r_A \)
  B → A: \( g^r_B \)
  A → B: \( \{M\} g^r_A B \)
Public-key Needham-Schroeder

• Proposed in 1978:
  A → B: \{NA, A\}_{KB}
  B → A: \{NA, NB\}_{KA}
  A → B: \{NB\}_{KB}
• The idea is that they then use NA⊕NB as a shared key
• Is this OK?

Public-key Needham-Schroeder (2)

• Attack found eighteen years later, in 1996:
  A → C: \{NA, A\}_{KC}
  C → B: \{NA, A\}_{KB}
  B → C: \{NA, NB\}_{KA}
  C → A: \{NA, NB\}_{KA}
  A → C: \{NB\}_{KC}
  C → B: \{NB\}_{KB}
• Fix: explicitness. Put all names in all messages
Public Key Certification

- One way of linking public keys to principals is for the sysadmin to physically install them on machines (common with SSH, IPSEC)
- Another is to set up keys, then exchange a short string out of band to check you’re speaking to the right principal (STU-II, Bluetooth simple pairing)
- Another is certificates. Sam signs Alice’s public key (and/or signature verification key)
  \[ CA = \text{sig}_S\{T_S,L,A,K_A,V_A\} \]
- But this is still far from idiot-proof…

The Denning-Sacco Protocol

- In 1982, Denning and Sacco pointed out the revocation problem with Needham-Schroder and argued that public key should be used instead
  \[
  A \rightarrow S: A, B \\
  S \rightarrow A: CA, CB \\
  A \rightarrow B: CA, CB, \{\text{sig}_A\{T_A, K_{AB}\}\}_K 
  \]
- What’s wrong?
The Denning-Sacco Protocol (2)

- Twelve years later, Abadi and Needham noticed that Bob can now masquerade as Alice to anyone in the world!
  - $A \rightarrow S: A, B$
  - $S \rightarrow A: CA, CB$
  - $A \rightarrow B: CA, CB, \{\text{sig}_A\{T_A, K_{AB}\}\}_{KB}$
  - $B \rightarrow S: B, C$
  - $S \rightarrow B: CB, CC$
  - $B \rightarrow C: CA, CC, \{\text{sig}_A\{T_A, K_{AB}\}\}_{KC}$

Public Key Protocol Problems

- It's also very easy to set up keys with the wrong people – man-in-the-middle attacks get more pervasive. Assumptions are slippery to pin down
- Technical stuff too – if the math is exposed, an attacker may use it against you!
- So data being encrypted (or signed) must be suitably packaged
- Many other traps, some extremely obscure…
TLS

- Formerly SSL, became TLS after many bugs fixed:
  
  \[
  \begin{align*}
  & C \rightarrow S: C, C#, N_C & \text{'client hello'} \\
  & S \rightarrow C: S, S#, N_S CS & \text{'server hello'} \\
  & C \rightarrow S: \{k_0\}_K S & \text{‘}k_0\text{ = pre-master secret'} \\
  & C \rightarrow S: \{\text{finished, } MAC_{K_1}(\text{everything to date})\} \\
  & S \rightarrow C: \{\text{finished, } MAC_{K_2}(\text{everything to date})\} \\
  & K_1, K_2 \text{ hashed from ‘master secret’ } K_1 = h(k_0, N_C, N_S)
  \end{align*}
  \]

- Formally verified to ‘work’ but still often used inappropriately (more later…)

TLS (2)

- Why doesn’t TLS stop phishing?
  - Noticing an ‘absent’ padlock is hard
  - Understanding URLs is hard
  - Websites train users in bad practice
  - …

- In short, TLS as used in e-commerce dumps compliance costs on users, who can’t cope

- There are solid uses for it though
Chosen protocol attack

- Suppose that we had a protocol for users to sign hashes of payment messages (such a protocol was proposed in 1990s):
  \[ C \to M: \text{order} \]
  \[ M \to C: X \quad [ = \text{hash(order, amount, date, …)}] \]
  \[ C \to M: \text{sig}_K\{X\} \]

- How might this be attacked?

Chosen protocol attack (2)

The Mafia demands you sign a random challenge to prove your age for porn sites!