Security protocols and their verification

Mark Ryan
University of Birmingham
Contents

1. Authentication protocols (*this lecture*)
2. Electronic voting protocols
3. Fair exchange protocols
4. Digital cash protocols
Alice and Bob want to communicate with each other over an insecure medium, e.g., the internet.

But first, they need to

- Authenticate each other: they want to be sure they are talking to each other, not to an imposter
- Agree a secret key: they want to encrypt their conversation, for privacy and integrity.

Alice and Bob might be humans, but they might also be: a web browser and a web server; a mobile phone and the network operator; etc.
Security protocols

- **Alice and Bob** will engage in an exchange of messages (= a protocol) which will result in each of them
  - Having authenticated each other
  - Having established a shared secret key which they can use to encrypt their subsequent conversation (a session key).
- The protocol may involve a trusted third party (TTP).
Security protocols

A security protocol (or cryptographic protocol) is a protocol that performs a security-related function and applies cryptographic methods.

Most cryptographic protocols incorporate at least some of these aspects:

- Use of cryptographic functions
- Entity authentication
- Construction of shared secret(s), useful for establishing a secret key
- Secured application-level data transport
- Non-repudiation guarantees
Authentication

- **Authentication between humans**
  - How can your bank know that an instruction to pay money to X from your account is actually coming from you?

- **Authentication between devices**
  - How can the mobile phone network know that instructions apparently coming from your phone are actually from your phone?

- **Authentication between programs**
  - How can your browser establish that the server at the other end is really "hooook" and not a look-alike web site designed to defraud you?
Protocols

• **A protocol** is an agreed way, or convention, for two or more parties to achieve a goal. It fixes the number and format of the messages between the parties. These rules are known to all of the participants.

• **Example:** Wide Mouth Frog
  
  – It assumes that A and B have access to a trusted third party S, and that they each share a symmetric encryption key with S:
    
    • The key between A and S: \( K_{AS} \)
    • The key between B and S: \( K_{BS} \)
Wide Mouth Frog

\[ A, \{T_A, K_{AB}, B\}_{K_{AS}} \]

\[ \{T_S, K_{AB}, A\}_{K_{BS}} \]
Wide Moth Frog - remarks

• Good things
  – Thanks to the encryption, an evesdropper cannot get access to the secret key $K_{AB}$ invented by Alice.
  – Thanks to the timestamps, the server and Bob can detect if messages they are receiving are old.

• Bad things
  – The protocol relies on timestamps to ensure freshness of messages. This assumes a global clock.
  – $K_{AB}$ is completely determined by A. This assumes A is competent, e.g. to generate random numbers.
  – In spite of this, S gets to know the value of $K_{AB}$. 
Otway-Rees

A

\[ M, A, B, \{ M, A, B, N_A \}_{K_{AS}} \]

\[ M, \{ N_A, K_{AB} \}_{K_{AS}} \]

B

\[ M, A, B, \{ M, A, B, N_A \}_{K_{AS}}, \{ M, A, B, N_B \}_{K_{BS}} \]

S

\[ M, \{ N_A, K_{AB} \}_{K_{AS}}, \{ N_B, K_{AB} \}_{K_{BS}} \]

Checks message
Otway-Rees -- remarks

• M is a session identifier.

• Good things
  – S generates the session key, instead of one of the participants A,B
  – S still has to deal only with one of the participants

• Bad things
  – S has quite a lot of work to do: decrypting twice, checking the message format, inventing key, encrypting twice. This makes it vulnerable to Denial of Service attacks.
  – An intruder could interfere with the protocol so that A and B get different keys!
Attack resulting in different keys
Assumptions and rules of protocol design

- The “Dolev-Yao model”: the attacker
  - Can read messages
  - Can block messages
  - Can generate fake messages
  - Can read/generate encrypted messages if s/he has the key
  - Knows the protocol being used. (This may not be obvious if A and B are humans, but it is obvious if A is a web server and B a browser.)

- The attacker cannot read/generate encrypted messages if s/he doesn't have the key. (Thus, we assume the crypto is unbreakable.)
Assumptions and rules contd.

- When we use a *trusted third party* (TTP), we should avoid overloading it:
  - It should not have to remember state (e.g., remember nonces, keep lists of used keys, etc.)
  - Preferably, it should deal with only one of the participants

- Otherwise, we make it vulnerable to denial of service attacks.
The third protocol introduced in this lecture is the Needham-Schroeder public key authentication protocol, introduced in 1978.

- The protocol consists of three parts.
  - In two of the parts, A and B are each obtaining the other one's public key from a server S.
  - In the other part, they exchange nonces which are intended to be used to set up a secret session key.

- A serious flaw in the protocol's third part was found by Gavin Lowe in 1995.
Needham-Schroeder PK

A, B

$\{K_B, B\}_{S^{-1}}$

$\{N_A, A\}_{K_B}$

B, A

$\{K_A, A\}_{S^{-1}}$

$\{N_A, N_B\}_{K_A}$

$\{N_B, A\}_{K_B}$
Needham-Schroeder PK, in essence

\[ \{ N_A, A \}_{K_B} \]

\[ \{ N_A, N_B \}_{K_A} \]

\[ \{ N_B, A \}_{K_B} \]
Needham-Schroeder PK attack

\[
\begin{align*}
\{N_A, A\}_{K_I} & \quad \text{from A to I} \\
\{N_A, N_B\}_{K_A} & \quad \text{from I to B} \\
\{N_B\}_{K_I} & \quad \text{from B to I} \\
\{N_A, N_B\}_{K_A} & \quad \text{from I to A} \\
\{N_A, A\}_{K_B} & \quad \text{from A to B}
\end{align*}
\]
NS PK attack

• The attack:
  – A has engaged in a session with I, & is aware of that.
  – I has engaged in a session with B, but B thinks he is talking to A. B is duped.
  – Example: A=you, I=dodgy retailer, B=your bank

• The solution is very simple:
  – Replace the message

\[ \{ N_A, N_B \}_K_A \]

with

\[ \{ N_A, N_B, B \}_K_A \]
Conclusions

- Security protocols are short, but very hard to get right.
  - This makes them ideal for formal analysis
  - One has to be clear about what the goals of a security protocol are.
  - And about the “attacker model”

- Next lectures:
  - Fair exchange protocols
  - Electronic voting protocols
  - Digital cash protocols