Cryptographic protocols: design and analysis

David Wagner
University of California, Berkeley

Notation

A, B, S: legitimate parties. (Short for: Alice, Bob, server.)

$A \rightarrow B : x$ means the message x sent by party A to party B.
(Note that B can't tell the difference between $A \rightarrow B : x$ and $M \rightarrow B : x$, so messages should include the sender's name if the recipient needs to know it.)

$1. A \rightarrow B : x$ means the protocol designer intended this message to be the first in a series of several.
(But there is no guarantee that A, the network, or the adversary will behave as intended.)

Notation

k is a key; k^{-1} is its inverse.
For symmetric cryptosystems, $k = k^{-1}$; for public-key cryptosystems, k is the public key and k^{-1} the corresponding private key.

$\{x\}_k$ means x encrypted under k.

$[x]_{k^{-1}}$ means x signed under k^{-1}.
(Conventionally one writes $\{x\}_{k^{-1}}$, although this notation is probably a poor choice.)

T_A is a timestamp chosen by A.
N_A is an unpredictable random nonce (a "challenge") chosen by A.

Who's awake?

What does the following notation mean?

$$\begin{array}{ll} \texttt{1.} & A \rightarrow B: \quad \{A, [K_{AB}, A, B]_{K_A^{-1}}\}_{K_B} \\ \texttt{2.} & B \rightarrow A: \quad \{\text{message}\}_{K_{AB}} \end{array}$$

Warmup

Establishing a secure channel with a challenge-response protocol:

$$
\begin{array}{lll}
1. & A \to B: & A \\
2. & B \to A: & N_B \\
3. & A \to B: & [N_B]_{K_A^{-1}} \\
4. & A \to B: & \{\text{message}\}_{K_B} \\
5. & A \to B: & \{\text{message}'\}_{K_B} \\
\ldots & &
\end{array}
$$

Can you spot the flaw?

Denning-Sacco #1

Key exchange with the aid of an online certification server S.

$$
\begin{array}{lll}
\texttt{1.} & A \to S: & A, B \\
\texttt{2.} & S \to A: & \text{cert}_A, \text{cert}_B \\
\texttt{3.} & A \to B: & \text{cert}_A, \text{cert}_B, \{[K_{AB}, T_A]_{K_A^{-1}}\}_{K_B}
\end{array}
$$

Can you spot the flaw?

Breaking Denning-Sacco #1

Look closely:

$$3. \quad A \to B : \quad \mathsf{cert}_A, \mathsf{cert}_B, \{[K_{AB}, T_A]_{K_A^{-1}}\}_{K_B}$$

The key K_{AB} isn't bound to the names of the endpoints A, B.

Therefore, B can extract the quantity $[K_{AB}, T_A]_{K_A^{-1}}$ and use it to spoof A in a new connection to C, like this:

$$3'. \quad B \to C : \quad \mathsf{cert}_A, \mathsf{cert}_C, \{[K_{AB}, T_A]_{K_A^{-1}}\}_{K_C}$$

As a result, C mistakenly concludes he is speaking with A.

Moral: Be explicit. Bind all names, and all other relevant context, to every message.

Credits: Abadi and Needham.

Early SSL

Key exchange with mutual authentication:

$$\begin{array}{lll} 1. & A \rightarrow B: & \{K_{AB}\}_{K_B} \\ 2. & B \rightarrow A: & \{N_B\}_{K_{AB}} \\ 3. & A \rightarrow B: & \{\text{cert}_A, [N_B]_{K_A^{-1}}\}_{K_{AB}} \end{array}$$

Can you spot the flaw?

Breaking early SSL

B can spoof A in a new connection with C.

$$\begin{array}{lll} 1'. & B \to C: & \{K_{BC}\}_{K_C} \\ 2'. & C \to A: & \{N_C\}_{K_{BC}} \end{array}$$

When C challenges B with nonce N_C, Bob sends $N_B = N_C$ back to A and uses her as an oracle.

$$\begin{array}{lll} 1. & A \to B: & \{K_{AB}\}_{K_B} \\ 2. & B \to A: & \{N_C\}_{K_{AB}} \\ 3. & A \to B: & \{\text{cert}_A, [N_C]_{K_A^{-1}}\}_{K_{AB}} \end{array}$$

A will sign *anything*, so B extracts $[N_C]_{K_A^{-1}}$ and he's in:

$$3'. \quad B \to C: \quad \{\text{cert}_A, [N_C]_{K_A^{-1}}\}_{K_{AB}}$$

Fix: replace $[N_B]_{K_A^{-1}}$ with $[A, B, N_A, N_B]_{K_A^{-1}}$.

Credits: Abadi and Needham.

Early SSL lessons

Moral: Don’t let yourself be used as a signing oracle. Add your own randomness—and bind names—before signing.

GSM challenge-response

A is cellphone handset, B is a base station.

$$\begin{array}{ll} 1. & B \to A: \quad N_B \\ 2. & A \to B: \quad [N_B]_{K_A^{-1}}, \{\text{data}\}_k \end{array}$$

where $k = f(K_A, N_B)$ is the voice privacy key.

Can you spot the weakness?

X.509 standard #1

Sending a signed, encrypted message to B:

$$1. \quad A \to B : \quad A, [T_A, B, \{\text{message}\}_{K_B}]_{K_A^{-1}}$$

Can you spot the flaw?

Breaking X.509 standard #1

$$1.\quad A \to B: \quad A, [T_A, B, \{\text{message}\}_{K_B}]_{K_A^{-1}}$$

There's no reason to believe the sender was ever aware of the contents of the message.

Example: Proving yourself by sending a password. Attacker C intercepts (1) above, extracts $\{\text{password}\}_{K_B}$, and sends

$$1.\quad C \to B: \quad C, [T_C, B, \{\text{password}\}_{K_B}]_{K_C^{-1}}$$

Now C is in, without needing to know the password.

Example: Secure auctions. This attack provides an easy way for C to send in a copy of A's bid under his own name, without needing to know what A's bid was.

Moral: sign before encrypting.

Credits: Abadi and Needham.

TMN

Pop quiz. Watch carefully.

A, B establish a shared key k_B using the help of a fast server S:

$$\begin{array}{lll} 1. & A \to S: & \{k_A\}_{K_S} \\ 2. & B \to S: & \{k_B\}_{K_S} \\ 3. & S \to A: & k_A \oplus k_B \end{array}$$

A recovers k_B as $k_A \oplus (k_A \oplus k_B)$.

Can you spot the flaw?

Breaking TMN

Let's play spot the oracle!

The attack: C, D conspire to recover k_B:

$$
\begin{array}{lll}
1'. & C \to S: & \{k_B\}_{K_S} \\
2'. & D \to S: & \{k_D\}_{K_S} \\
3'. & S \to C: & k_B \oplus k_D
\end{array}
$$

Now C, D can recover k_B.

Credits: Simmons.

Goss railway protocol

A and B establish an authenticated shared key $k_{A,B} = r_A \oplus r_B$:

$$\begin{array}{lll} 1. & A \to B: & A, \{r_A\}_{K_B} \\ 2. & B \to A: & B, \{r_B\}_{K_A} \end{array}$$

Do you see the subtle weakness?

Attacks on Goss

In class, I thought there were bugs in the following slide, which describes an attack on the Goss railway protocol.

Upon more reflection, I now think I was wrong; I think my written description was completely correct, though I didn't show it to you in class. (On the other hand, I could just be thoroughly confused. This just underscores my point that these things are not at all easy to get right.)

Triangle attacks on Goss

If session keys sometimes leak, the system breaks.

C can recover r_A from $\{r_A\}_{K_B}$ by opening a session to B and replaying A's encrypted contribution to the key:

$$\begin{array}{lll} 1. & C \rightarrow B: & C, \{r_A\}_{K_B} \\ 2. & B \rightarrow C: & B, \{r'_B\}_{K_C} \end{array}$$

If C can learn $k_{B,C}$, he can compute $r_A = k_{B,C} \oplus r'_B$.

Play the same games with A to learn r_B; you get $k_{A,B}$.

Credits: Burmester.

Implementing protocols

Explicitness is powerful (and cheap).

The mathematical notation

$$\begin{array}{ll} \texttt{1.} & B \to A: \quad N_B \\ \texttt{2.} & A \to B: \quad \{N_B, K_{A,B}\}_{K_A} \end{array}$$

should probably be implemented in practice as

$\texttt{1.}\ B \to A:$ Message 1 from B to A of GSM protocol v1.0 is a challenge N_B.
$\texttt{2.}\ A \to B:$ $\{$Message 2 from A to B of GSM protocol v1.0 is a response to the challenge N_B; and A asserts that the fresh session key $K_{A,B}$ is good for communication between A and B on the session where N_B was seen.$\}_{K_A}$

(Can you see why each of the elements above are there?)

Implementing protocols

Any value received as cleartext should be treated as untrustworthy: you may use it as a **hint** for performance, but don't depend on it for security.

Minimize state; each message should be self-explanatory.

Hash everything. Each message should include the (signed?) hash of all previous messages in the interaction. This makes cut-and-paste attacks harder.

Measure twice, cut once.