Design Principles for Cryptographic Protocols

Sergey Berezin and Andy Myers

Outline

- Principles for creating secure protocols
  (Neither necessary nor sufficient)
- Interesting attacks
- Ways to reason about protocols

N.B. Material in this presentation is a mix of 5 sources. If you'd like to sort out which source said what, talk to us after the presentation.

8.5 Steps to the Perfect Protocol

1. Be explicit (but not redundant)
2. Know what you're doing
   - Know who you trust
3. Sign before encrypting (usually)
4. Encrypt correctly, minimally
5. Assume nothing (especially about freshness)
6. Don't be an oracle
7. Always count your bits
8. KISS

Be Explicit

Otway-Rees:

1. $A \Rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$
2. $B \Rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$
3. $S \Rightarrow B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
4. $B \Rightarrow A : M, \{N_A, K_{AB}\}_{K_{AS}}$

There should be no ambiguity about:

- what a message means,
- what step a message is from, or
- what run of the protocol a message is from

A (Bad) Fix

1. $A \Rightarrow B : M, A, B, \{This\ is\ message\ 1!, N_A, M, A, B\}_{K_{AS}}$

...

4. $B \Rightarrow A : M, \{This\ is\ message\ 4!, N_A, K_{AB}\}_{K_{AS}}$

Denning-Sacco

(Based on an assymetric cryptosystem)

1. $A \Rightarrow S : A, B$
2. $S \Rightarrow A : CA, CB$
3. $A \Rightarrow B : CA, CB, \{\{K_{AB}, T_A\}_{K_A^{-1}}\}_{K_B}$

But now B can spoof A:

4. $B(A) \Rightarrow C : CA, CC, \{\{K_{AB}, T_A\}_{K_A^{-1}}\}_{K_C}$

If a message should be interpreted relative to a principal, then that principal's name should be in the message.

Denning-Sacco Fix

1. $A \Rightarrow S : A, B$
2. $S \Rightarrow A : CA, CB$
3. $A \Rightarrow B : CA, CB, \{\{B, K_{AB}, T_A\}_{K_A^{-1}}\}_{K_B}$

Redundancy?

General attack on a public key cryptosystem:
"Decrypt" random bit strings with public key until a reasonable message appears. Then claim that the owner of the public key said this.

- Assumes that signing a document consists of encrypting the document with a private key
- Would be much harder if hash of document is encrypted
- Realistic? Maybe for a short, unredundant message.

Can combat this by adding *redundancy* to all signed messages.

Redundancy??

Mao and Boyd point out:
Redundancy $\Rightarrow$ Successful attacks

Redundancy gives hackers a known plaintext attack, or at least more data for cryptanalysis.

Otway-Rees's first message:
$A \Rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$

Know what needs to be secret and what just needs to be protected for integrity. Don't encrypt known quantities!

Know what you're doing

Wide-mouthed Frog Protocol:
1. $A \Rightarrow S : A, \{K_{AB}, T_A, B\}_{K_{AS}}$
2. $S \Rightarrow B : \{T_S, A, K_{AB}\}_{K_{AB}}$

Hidden assumption: That A is capable of generating a good session key.

All aspects of a protocol should be thoroughly and exactly specified:

- Each principal's actions at each step
- Be clear about why encryption is being done
- Explicitly state properties of crypto algorithms

Needham-Schroeder '87

1. $A \Rightarrow B : A$
2. $B \Rightarrow A : \{A, N_B\}_{K_{BS}}$

$\ldots$

5. $A \Rightarrow B : \{K_{AB}, A, N_B\}_{K_{BS}}$

$\ldots$

Hidden assumption: A hacker can't strip off the first block of a message and append an arbitrary string.

5'. $C(A) \Rightarrow B : X, \{A, N_B\}_{K_{BS}}$

Trust

Scenario: An evil hacker becomes able to manipulate the clocks of an organization which uses Kerberos.
Result: Major vulnerability. Old tickets can be reused.

Hidden assumption: The time service is trusted.

Be explicit about which parties are trusted and what each party is trusted to do.

N.B. The process of deciding who or what is trustworthy isn't a protocol design issue.

Assume Nothing

CCITT X.509 protocol:

$$A \Rightarrow B : A, \{T_A, N_A, B, X_A, \{Y_A\}_{K_B}\}_{K_A^{-1}}$$

T_A is a timestamp, N_A is a nonce, X_A and Y_A are user data.

An evil hacker could send:

$$C \Rightarrow B : C, \{T_C, N_C, B, X_C, \{Y_A\}_{K_B}\}_{K_C^{-1}}$$

Don't assume that a principal who signs an encrypted message has any inkling about its meaning. This isn't true for a message that is signed and then encrypted.

Old secrets can hurt you

Goss's Diffie-Hellman-based key exchange protocol:
Assymetric key cryptosystem.
Globals: generator, g, of subgroup Z_p^*
Private key for A: x_A
Public key for A: g^{x_A}
Key for A and B: $g^{x_A r_B} \oplus g^{x_B r_A}$
where A picks r_A and B picks r_B.

Key exchange:
1. $A \Rightarrow B : g^{x_A}, g^{r_A} \pmod{p}$
2. $B \Rightarrow A : g^{x_B}, g^{r_B} \pmod{p}$

Now A has $r_A, g^{x_B}, x_A, g^{r_B}$
And B has $r_B, g^{x_A}, x_B, g^{r_A}$
So both can compute the shared key.

Old secrets can hurt 2

An attack (by Alice) on Goss's protocol:
1. $A \Rightarrow B : g^{x_A}, z(g^{r_A}) \pmod p$
2. $B \Rightarrow A : g^{x_B}, g^{r_B} \pmod p$

Bob can compute the key, $g^{x_A r_B} \oplus z^{x_B}$
Alice can only compute the first part, $g^{x_A r_B}$
So, if Bob lets that session key leak after some period of time, Alice can find z^{x_B}.

Do not assume the secrecy of anyone else's secrets.

RSA

Alice encrypts and then signs a message, M, to Bob:

$$(M^{e_B} \pmod{n_B})^{d_A} \pmod{n_A}$$

Bob wants to make it appear that Alice signed a different message, M'. So he finds x such that:

$$(M')^x = M \pmod{n_B}$$

And so,

$$(M')^{xe_B} = M^{e_B} \pmod{n_B}$$

Finally, Bob registers a new public key/modulus pair, (xe_B, n_B). Now Bob can claim that Alice actually signed M'.

Not realistic because new certificate is newer than message that Alice signed.

Encrypting Encrypted Messages

A secure coin flip protocol:

1. $A \Rightarrow B : \{M_1\}_{K_A}, \{M_2\}_{K_A}$
2. $B \Rightarrow A : \{\{M_i\}_{K_A}\}_{K_B}$
3. $A \Rightarrow B : \{M_i\}_{K_B}$
4. $B \Rightarrow A : M_i$
5. $A \Rightarrow B : K_A$
6. $B \Rightarrow A : K_B$

Bob, being a devious fellow, will find two keys, B_1 and B_2, such that:

$$\{\{M_1\}_{K_A}\}_{K_{B_1}} = \{\{M_2\}_{K_A}\}_{K_{B_2}}$$

and

$$\{\{M_1\}_{K_A}\}_{K_{B_2}} = \{\{M_2\}_{K_A}\}_{K_{B_1}}$$

Bob can now win the coin flip every time.

Moral

Anderson and Needham say not to assume that a message has a particular form unless you can verify it.

Syverson says that even if you can verify that a message has a particular form, you can't be sure that's its *only* form.

Freshness

Needham-Schroeder:

1. $A \Rightarrow S : A, B, N_A$
2. $S \Rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \Rightarrow B : \{K_{AB}, A\}_{K_{BS}}$
4. $B \Rightarrow A : \{N_B\}_{K_{AB}}$
5. $A \Rightarrow B : \{N_B + 1\}_{K_{AB}}$

Notice that an *unbounded* amount of time can pass between messages 2 and 3.

Don't assume that a key is fresh just because it's been used recently.

Fix: Either have S add a timestamp, or else let B pass a nonce to S.

About timestamps...

The largest tolerable difference between local time and a timestamp has to be larger than the largest difference between any two clocks.

N.B. Clock synchronization becomes a security issue.

Timeliness

A basic protocol to find out what time it is:

1. $A \Rightarrow S : A, N_A$
2. $S \Rightarrow A : \{T_S, N_A\}_{K_{AS}}$

If N_A is predictable, an evil party could make Alice set her clock back:
1. $C(A) \Rightarrow S : A, N_A$
2. $S \Rightarrow C(A) : \{T_S, N_A\}_{K_{AS}}$
...
3. $C(S) \Rightarrow A : \{T_S, N_A\}_{K_{AS}}$

Predictable values can be used to ensure timeliness, but only if they are effectively protected.

Fix: Encrypt (“randomize”) N_A.

Nonces

Otway-Rees (again):

1. $A \Rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$
2. $B \Rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$
3. $S \Rightarrow B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
4. $B \Rightarrow A : M, \{N_A, K_{AB}\}_{K_{AS}}$

In the above protocol, nonces N_A and N_B are being used both to *guarantee timeliness* and *as references* to Alice and Bob.

Be explicit about a nonce's use and meaning.