Foundations of Computer Security

Lecture 56: Cryptographic Protocols

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin

Thought Experiment

Consider the following scenario:

- Your friend Ivan lives in a repressive country where the police spy on everything and open all the mail.
- You need to send a valuable object to Ivan.
- You have a strongbox with a hasp big enough for several locks, but no lock to which Ivan also has a key.

How can you get the item to Ivan securely?

A Possible Answer

You might take the following sequence of steps:

1. Put the item into the box, attach your lock to the hasp, and mail the box to Ivan.
2. Ivan adds his own lock and mails the box back to you.
3. You remove your lock and mail the box back to him. He now removes his lock and opens the box.

The procedure just described could be regarded as a *protocol*—a structured dialog intended to accomplish some communication-related goal.

What's This Got to do with Computing?

What goal: To send some content confidentially in the context of a hostile or untrustworthy environment, when the two parties don't already share a secret/key.

You could implement the "same" protocol to send a message confidentially across the Internet. Here,

- the *valuable thing* is the contents of a secret message;
- the *locks* are applications of some cryptographic algorithm with appropriate *cryptographic keys.*

But for this to work in the computing world there's a particular feature that the ciphers have to satisfy. *Can you see what it is?*

What is the Property?

Imagine that instead of putting another lock on the hasp, Ivan puts your lockbox inside another locked box. The protocol no longer works; you can't reach inside his box to take off your lock in step 3.

On-line, you'd have to be able to "reach inside" his encryption to undo yours. One way this would be true is if the ciphers *commute*.

$$\{\{M\}_{k_1}\}_{k_2} = \{\{M\}_{k_2}\}_{k_1}$$

Most encryption algorithms don't have this property. But one that does is: exclusive or (XOR) your message with a randomly generated string (key) of the same length.

So Here's the Protocol

Let K_a be a random string generated by A, and K_b be a random string generated by B, both K_a and K_b of the same length as M.

1. $A \rightarrow B : M \oplus K_a$
2. $B \rightarrow A : (M \oplus K_a) \oplus K_b$
3. $A \rightarrow B : ((M \oplus K_a) \oplus K_b) \oplus K_a)$

In step 3, the two applications of K_a "cancel out," leaving $(M \oplus K_b)$, which B can easily decrypt with his own key K_b.

Whoops!

This is effectively using the one-time pad, so should be very strong. *Right?*

Even though the one-time pad is a theoretically unbreakable cipher, there's a good reason it's called "one-time." Our protocol is fundamentally flawed. *Can you see why?*

1. $A \rightarrow B : M \oplus K_a$
2. $B \rightarrow A : (M \oplus K_a) \oplus K_b$
3. $A \rightarrow B : ((M \oplus K_a) \oplus K_b) \oplus K_a)$

An evesdropper who stores the three messages can XOR combinations of them to extract any of M, K_a, and K_b. *Verify this for yourself.*

Lessons

- Cryptographic protocols accomplish security-related functions via a structured exchange of messages.
- They are very important to security on the Internet.
- They are difficult to design and easy to get wrong in subtle ways.

Next lecture: Cryptographic Protocols II