Lecture 19 - Authenticated Key Exchange and the SSL Protocol

Boaz Barak

December 1, 2005

Key exchange Suppose we have following situation: Alice wants to buy something from the well known website Bob.com

Since they will exchange private information (Alice's credit card, address etc.) they want to use encryption. However, they do not share a key between them.

Using a key exchange protocol. It seems that we already learned a protocol to do that: Alice and Bob can run a *key exchange protocol.* One such protocol is the Diffie-Hellman protocol, but they can also run the following RSA-based protocol:

$A \leftarrow B$ Bob chooses a pair of RSA keys (e, d) and sends e to Alice.

$A \rightarrow B$ Alice chooses a key $k \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends $\mathsf{E}_e(k)$ to Bob.

$A \leftrightarrows B$ Bob and Alice can now can now continue their interaction with the shared secret key k.

Insecurity of basic key exchange protocol: This protocol is secure for a *passive / eavesdropping* adversary, but it is not secure against an *active* adversary. Indeed, a man-in-the-middle Charlie can play Bob to Alice and Alice to Bob. That is, Charlie will receive (e, d) from Bob but will not pass this on to Alice. Rather he will choose his own RSA pair (e', d') and send e' to Alice. Alice will then send $\mathsf{E}_{e'}(k)$ to Charlie. Charlie can decrypt to find k and then send $\mathsf{E}_e(k)$ to Bob.[1] From now on Charlie will be able to listen in to all of Alice and Bob's communication.

Obvious fix. This attack is inherent since if Bob and Alice don't know anything about each other then of course Charlie can impersonate them to one another. However, we are in a setting where Bob is a well known web site, and hence we can assume that Alice already has Bob's public key. This prevents this attack but it is not clear that it is secure.

Example: SSL protocol. The SSL protocol is the most widely used protocol for such transactions (this is the protocol used to access encrypted web sites, and is the standard for all transactions involving credit card etc.). However, in V3.0, the heart of the protocol was the following interaction:

- Client sends $\mathsf{E}_e(k)$ to the server where $\mathsf{E}_e(\cdot)$ is padded RSA according to standard PKCS #1 V1.5 (a scheme believed to be semantically secure).
- Server validates decryption is according to standard, otherwise sending `invalid decryption`, and if so, uses k as the key.

[1] He can also choose his own key k' and send $\mathsf{E}_e(k')$ to Bob.

The padding scheme is the following: if $\{f_e\}$ is the RSA trapdoor permutation collection then to encrypt x choose r to be a random string (of length at least 8 bytes) conditioned on not having any zero byte, and let $x' = 0 \circ 2 \circ r \circ 0 \circ x$. Define the function $PKCS(x')$ to output 1 iff x' is of this form. For a random x', the probability that $PKCS(x') = 1$ is about 2^{-16}.

In a surprising paper, Bleichenbacher proved that the function $PKCS(\cdot)$ is some kind of a *hard core* of RSA.[2] That is, he showed that if you have an oracle that given y outputs 1 iff $PKCS(f_e^{-1}(y)) = 1$, then you can use it to invert the one-way permutation $f_e(\cdot)$ using not too many queries. It follows that SSL protocol is insecure, since an attacker can open as many sessions with the server as it likes, essentially using the server as this oracle. (Note that no matter what happens later in the protocol, once the attacker received this error message, she got the response she needed, even if the server will abort later.)

Reflection: In retrospect, it should have been clear that it is a bad idea to use a scheme that is only CPA secure and not a CCA secure scheme. In fact, if the designers of SSL had tried to *prove* security of their protocol, they would have seen that CCA (or a close variant) is an essential condition for such a proof to go through.

The actual SSL protocol. See Lindell's notes for the description of the actual SSL protocol.

Some attacks on the SSL protocol:

- Goldberg-Wagner attack on the pseudorandom generation.
- Version-rollback attack,
- Protocol changing attack.
- Variations/extensions of the Bleichenbacher attack.

[2]Actually, there were several previous results about very related hard-core functions for RSA, but people always thought about these results as establishing theoretical security and not practical insecurity.