CSE 5852: Lecture 24

Prof. Benjamin Fuller
Scribe: Chao Shang

November 30, 2016

1 Review of Last Class

The last class introduced collision-resistance hash function are sufficient to build a signature scheme. We introduced Merkle trees, which is a hash based data structure that is a generalization of the hash list. Then we learned Chain-based and Tree-based signatures.

Today we will introduce one topic: how individuals talk 'securely' on the Internet? We will cover the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols, how they can be applied to a web application, and the requirements necessary to create a secure link between a server and a client machine. The SSL and TLS offers security for the HTTP protocol.

2 SSL and TLS

2.1 Background

TLS (Transport Layer Security) and SSL (Secure Sockets Layer) are protocols that provide data encryption and authentication between applications and servers. TLS and its predecessor, SSL, both frequently referred to as "SSL," are cryptographic protocols that provide communications security over a computer network.

Netscape developed the original SSL protocols. The current version of SSL is version 3.0, released by Netscape in 1999. TLS 1.0 was first defined in RFC 2246 in January 1999 as an upgrade of SSL Version 3.0. TLS 1.2 is the current version and TLS 1.3 is currently in development.

2.2 SSL/TLS Handshake

First, let's see what is the SSH/TLS handshake briefly. The Simplified SSL Handshake as shown in Figure 1 is:

1. Client and server negotiate on cipher selection.
2. Cooperatively establish session keys.

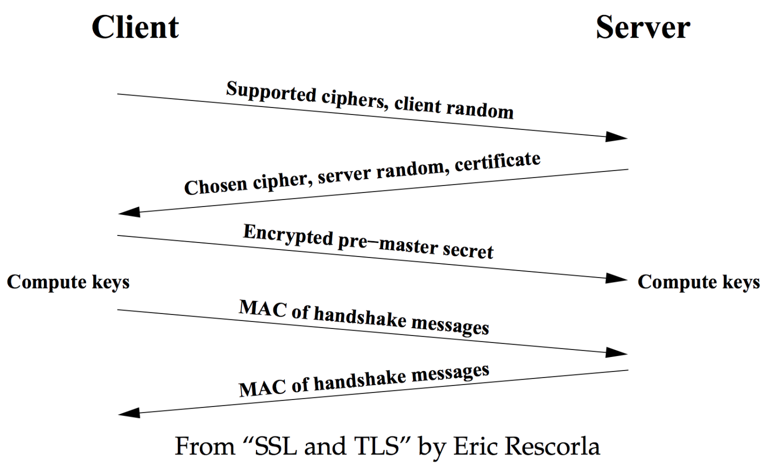
Client
Server
Supported ciphers, client random
Chosen cipher, server random, certificate
Encrypted pre–master secret
Compute keys
Compute keys
MAC of handshake messages
MAC of handshake messages
From "SSL and TLS" by Eric Rescorla

Figure 1: Overview of SSL Handshake

3. Use session keys for secure communication.

In this process, we can find several problems as shown in Figure 2.

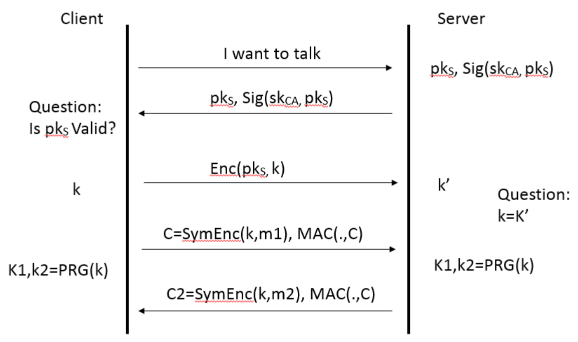

Figure 2: SSL Handshake

For the first problem, the client must acquire a digital certificate. This can be obtained from a Certification Authority (CA). For the second problem, server want to know if k is equal to k'. We never said that MACs are guaranteed to be secure if the two values are not the same. So the last two messages have unclear security unless we are assuming a stronger property than standard MAC security. An alternative would be for the client to sign the encrypted pre-master secret but this would be vulnerable to a man in the middle attack as the server knows nothing about the client identity.

What is Certification Authority?

A certificate authority or certification authority (CA) is an entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate.

CAs play a critical role in how the Internet operates and how transparent, trusted transactions can take place online. CAs issue millions of Digital Certificates each year, and these certificates are used to protect information, encrypt billions of transactions, and enable secure communication. But we need to know we assume certificate authority (CA) is a trusted entity.

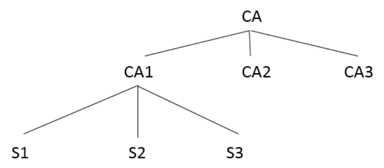

Figure 3: CA

In addition, we security is based on the privacy of server pk_S. This means that if this value is captured by an attacker at a later time then all previous conversations (observed by the attacker) of the server will be decryptable. The use of Diffie-Hellman cryptography allows for *forward security* which decouples long-term identity from short-term exchange of secrets.

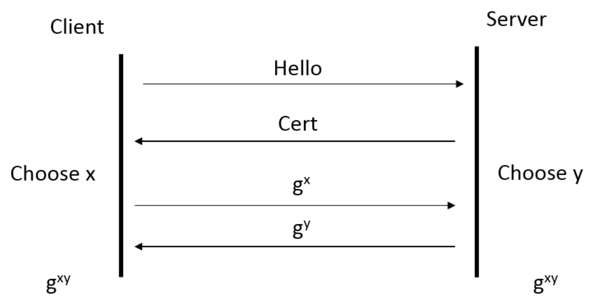

Figure 4: public-key cryptpgraphy to improve security

Let's see the details of the messages exchanged during the Client-authenticated TLS handshake. The steps are as follows:

1. The SSL or TLS client sends a "client hello" message that lists cryptographic information such as the SSL or TLS version and so on. The message also contains a random byte string that is used in subsequent computations.

2. The SSL or TLS server responds with a “server hello” message that contains the CipherSuite chosen by the server from the list provided by the client, the session ID, and another random byte string. The server also sends its digital certificate. If the server requires a digital certificate for client authentication, the server sends a “client certificate request” that includes a list of the types of certificates supported and the Distinguished Names of acceptable Certification Authorities (CAs).

3. The SSL or TLS server verifies the client’s certificate.

4. The SSL or TLS client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server’s public key.

5. If the SSL or TLS server sent a “client certificate request,” the client sends a random byte string encrypted with the client’s private key, together with the client’s digital certificate, or a “no digital certificate alert.” This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.

6. The SSL or TLS server verifies the client’s certificate.

7. The SSL or TLS client sends the server a “finished” message, which is MAC’d with the secret key, indicating that the client part of the handshake is complete.

8. The SSL or TLS server sends the client a “finished” message, which is MAC’d with the secret key, indicating that the server part of the handshake is complete.

9. For the duration of the SSL or TLS session, the server and client can now exchange messages that are symmetrically encrypted with the shared secret key.

3 Knowledge Points

In SSL/TLS Handshake, we have covered these points:

1. Digital Signiture

2. Public-key Encryption

3. Symmetric Encryption/ PRF

4. MAC

5. Pseudorandom Generator

6. Key exchange

Other things coverd in this semester:

1. Random Oracle
2. One-time pad
3. Universial Hash Function
4. Discrete Log
5. Factoring
6. Pseudorandom Function
7. GCD algorithm
8. Definitions: Encryption, semantic security, indistinguishable security, Integrity, and EU-CMA
9. polynomial, negligible