

CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture Notes 24 (extra)
April 23, 2013

Secure Shell (SSH)

Transport Layer Security (TLS)

Digital Rights Management and Trusted Computing Platform

Secure Shell (SSH)

Secure Shell (SSH)

SSH is a family of protocols that provide a secure encrypted channel connecting two networked computers over an insecure network.

It was initially designed to allow secure login between a user and a remote computer. This replaced the older `telnet`, `rlogin`, and `rsh` programs that provided unencrypted versions of this service and sent unencrypted passwords over the network.

The first version was designed by Tatu Ylönen at Helsinki University of Technology, Finland, and released as freeware in July 1995.

Open source and version forking

By December 1995, Ylönen created a startup company to market and develop SSH. The original version remained free, but many enhancements were only available in the commercial version.

Over the next five years, the licensing agreement became more and more restrictive as they worked to remove open source code (such as libgmp) from their code base.

Several vulnerabilities in SSH-1.5 were discovered, giving further motivation to create an open source version of SSH that could be more easily vetted for bugs and distributed more widely.

Starting from Ylönen's SSH-1.2.12, the last version released under an open source license, Björn Grönvall developed OSSH. OpenBSD developers then forked Grönvall's code and did further development to create the widely used OpenSSH.

SSH-2

By 2006, an improved but incompatible version 2 of the SSH protocol was adopted as a standard by the Internet Engineering Task Force (IETF).

Development of OpenSSH continues to this day, allowing more and more applications derive the benefits of secure encrypted communications.

SSH protocol outline

SSH is based on public key cryptography. Each machine has a public and private *host key*. Each user also has a public and private *user key*.

When logging onto a remote machine, the client first authenticates the remote host key using the public key for the host with that domain name or IP address found in the local `known_hosts` file.

Host key verification

If the host is not in the file or cannot authenticate the public key found there, one gets a prompt

```
The authenticity of host 'vm1.cs.yale.edu (128.36.229.150)' can't be established.  
RSA key fingerprint is c9:a5:be:55:af:ab:05:77:b4:30:62:ed:bd:be:50:43.  
Are you sure you want to continue connecting (yes/no)?
```

If you say yes, the public key of that host gets entered into the `known_hosts` file and used the next time.

User authentication

SSH supports several authentication methods. I'll describe the `publickey` method.

Here, the host checks that the user's public key is in its `authorized_users` file. If it is, it verifies that the user has the matching private key and accepts the authentication if it does.

Actual protocol

The actual protocol is much more complicated than this.

It include negotiations for which cipher to use, how to generate the shared session, what an encrypted packet looks like, and so forth.

Transport Layer Security (TLS)

TLS protocol

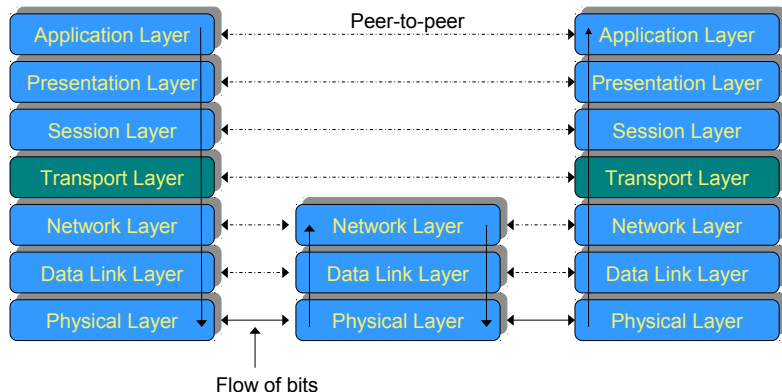
Transport Layer Security (TLS) is a protocol to secure and encrypt network traffic at the transport layer.

Like SSH, it provides authentication and encryption.

It is used to implement secure web traffic (`https:`) but applies much more generally.



ISO/OSI Model SSL: Security at Transport Layer



History

The TLS protocol has gone through several versions. It was originally called *Secure Socket Layer (SSL)*.

TLS 1.0 was first defined in 1999. It did not interoperate with the existing SSL 3.0 and so was renamed.

Key management

TLS uses X.509 certificates for its key management as described in lecture 16.

They allow the client to authenticate the server as follows:

1. The client obtains the server's certificate, generally from the server itself.
2. The client checks the validity of the certificate by verifying that it is properly signed by a trusted certificate authority.
3. The client then runs a simple authentication protocol whereby the server shows that it has the private key corresponding to its certificate.
4. Finally, client and server establish a shared symmetric key and use it to encrypt traffic between themselves.

The actual protocol

The actual protocol has all of the complications of the other practical protocols we've mentioned.

- ▶ It begins with a handshaking phase where client and server agree on the protocol level, cipher suite, and other parameters.
- ▶ Generally the server is authenticated by the client.
- ▶ The protocol allows for the server to require client authentication. However, this is not usually done for two reasons:
 1. Most clients lack certificates.
 2. Most servers use other means such as passwords to authenticate clients. This occurs after the encrypted connection has been established, so the passwords are not sent in the clear.

Preventing man-in-the-middle attacks

TLS is secure against man-in-the-middle attacks, even with only one-way authentication.

This is possible because the certificate give the client a reliable means of obtaining its public key.

At the end of the handshaking protocol, after the session key has been established, the client sends the server the hash of the complete transcript between client and server as seen by the client, secured with the server's public key.

The server checks that hash value against the hash of its view of the same handshake. If they don't agree, it indicates the presence of a man-in-the-middle or other network error and the protocol does not continue.

Digital Rights Management and Trusted Computing Platform

Control of information

Another attempted use of cryptography has been to control the use of information.

Digital Rights Management (DRM) is the term for a class of encryption schemes to disallow certain kinds of use of data, for example, copying and modifying.

Trusted Computing Platform (TCP) is a hardware architecture that requires authorization tokens to perform certain operations. These tokens are cryptographically produced using keys that are securely stored inside of a special crypto module.

A different paradigm

In most uses of cryptography studied so far, the owner of the secret keys also possesses and protects them.

With DRM and TCP, the party controlling the keys is not the same as the owner of the machine that uses them.

This means that the keys must be hidden from the owner of the device on which they are used.

While it is easy to prevent casual users from looking inside their box, protecting embedded secrets against sophisticated attacks has proved to be very difficult, and many DRM schemes have been broken soon after being introduced.