Contents

- Introduction
- Overview
- SSL/TLS
- SSH
- SSO
- Other Security Options
- PKI
- Conclusion
Public Key

- Everywhere
Public Key

Bob’s public key

Plain Text

Bob’s private key

Cipher Text

Encryption Algorithm (E.G : RSA)

Decryption Algorithm (E.G : RSA)

Plain Text

Encryption
Public Key

Alice’s private key

Encryption Algorithm (E.G : RSA)

Cipher Text

Decryption Algorithm (E.G : RSA)

Alice’s public key

Plain Text

Plain Text

Authentication
Public Key (Digital Signature)

**Signing**

1. Data
2. Hash Function
3. Hash: 101010101010101
4. Encrypt hash using signer’s private key: 101010101010101
5. Signature: 101010101010101

Attach to data

Digitally Signed Data
Public Key (Digital Signature)

**Verification**

Digitally Signed Data

Data

Hash function

101010101010101

hash

101010101010101

Signature

Decrypt using signer’s public key

101010101010101

hash

?
SSL (Secure Socket Layer)

- Introduction
- Overview
- Cryptographic primitives in SSL
- Handshake and cipher suites
- Certificates
- CA and Certificate enrolment
- Server configuration
Introduction

- Let’s talk
- Security Arch & Design
- Expectation
- Knowledge
- what to secure
- How
Overview

- End to end security in the networking space
Overview: (NM Apps. server)

- Network Management Apps
- Multiple communication path

To client Apps

SSO/IdP Or R. Auth Server

To NE Apps over SSH and more

To Secure over SSL:
- Native Data Mngt. Channel
- Fault/Event channel
- North Bound (OSS) Mngt. channel

Clustering Tech. for Redundancy
SSL 1.0, 2.0 and 3.0

The SSL protocol was originally developed by Netscape. Version 1.0 was never publicly released; version 2.0 was released in February 1995 but "contained a number of security flaws which ultimately led to the design of SSL version 3.0" (Rescorla 2001). SSL version 3.0 was released in 1996.
Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communications security over the Internet.[1] TLS and SSL encrypt the segments of network connections above the Transport Layer, using symmetric cryptography for privacy and a keyed message authentication code for message reliability.
NM Apps. Server and SSL

**TLS 1.0 (SSL 3.1)**
TLS 1.0 was first defined in [RFC 2246](https://tools.ietf.org/html/rfc2246) in January 1999 as an upgrade to SSL Version 3.0. As stated in the RFC, "the differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate." TLS 1.0 does include a means by which a TLS implementation can downgrade the connection to SSL 3.0.

**TLS 1.1 (SSL 3.2)**
TLS 1.1 was defined in [RFC 4346](https://tools.ietf.org/html/rfc4346) in April 2006. It is an update from TLS version 1.0. Significant differences in this version include:
- Added protection against [Cipher block chaining](http://en.wikipedia.org/wiki/Cipher_block_chaining) (CBC) attacks.
- The implicit [Initialization Vector](http://en.wikipedia.org/wiki/Initialization_vector) (IV) was replaced with an explicit IV.
- Change in handling of [padding errors](http://en.wikipedia.org/wiki/Padding_error)
- Support for [IANA](http://www.iana.org) registration of parameters.
TLS 1.2 (SSL 3.3)
TLS 1.2 was defined in RFC 5246 in August 2008. It is based on the earlier TLS 1.1 specification. Major differences include:
The **MD5-SHA-1** combination in the pseudorandom function (PRF) was replaced with **SHA-256**, with an option to use cipher-suite specified PRFs.
The **MD5-SHA-1** combination in the Finished message hash was replaced with **SHA-256**, with an option to use cipher-suite specific hash algorithms.
TLS 1.2 (SSL 3.3)
TLS 1.2 was defined in RFC 5246 in August 2008. It is based on the earlier TLS 1.1 specification. Major differences include:
The MD5-SHA-1 combination in the pseudorandom function (PRF) was replaced with SHA-256, with an option to use cipher-suite specified PRFs. The MD5-SHA-1 combination in the Finished message hash was replaced with SHA-256, with an option to use cipher-suite specific hash algorithms. The MD5-SHA-1 combination in the digitally-signed element was replaced with a single hash negotiated during handshake, defaults to SHA-1.
TLS 1.2 (SSL 3.3)

TLS 1.2 was defined in RFC 5246 in August 2008. It is based on the earlier TLS 1.1 specification. Major differences include:

The MD5-SHA-1 combination in the pseudorandom function (PRF) was replaced with SHA-256, with an option to use cipher-suite specified PRFs.

The MD5-SHA-1 combination in the Finished message hash was replaced with SHA-256, with an option to use cipher-suite specific hash algorithms.

The MD5-SHA-1 combination in the digitally-signed element was replaced with a single hash negotiated during handshake, defaults to SHA-1.

Enhancement in the client's and server's ability to specify which hash and signature algorithms they will accept.
NM Apps. Server and SSL

**TLS 1.2 (SSL 3.3)**

TLS 1.2 was defined in [RFC 5246](https://tools.ietf.org/html/rfc5246) in August 2008. It is based on the earlier TLS 1.1 specification. Major differences include:

*The MD5-SHA-1 combination in the pseudorandom function (PRF) was replaced with SHA-256, with an option to use cipher-suite specified PRFs.*

*The MD5-SHA-1 combination in the Finished message hash was replaced with SHA-256, with an option to use cipher-suite specific hash algorithms.*

*The MD5-SHA-1 combination in the digitally-signed element was replaced with a single hash negotiated during handshake, defaults to SHA-1.*

*Enhancement in the client's and server's ability to specify which hash and signature algorithms they will accept.*

*Expansion of support for authenticated encryption ciphers, used mainly for Galois/Counter Mode (GCM) and CCM mode of Advanced Encryption Standard encryption.*
NM Apps. Server and SSL

TLS 1.2 (SSL 3.3)
TLS 1.2 was defined in RFC 5246 in August 2008. It is based on the earlier TLS 1.1 specification. Major differences include:
The MD5-SHA-1 combination in the pseudorandom function (PRF) was replaced with SHA-256, with an option to use cipher-suite specified PRFs.
The MD5-SHA-1 combination in the Finished message hash was replaced with SHA-256, with an option to use cipher-suite specific hash algorithms.
The MD5-SHA-1 combination in the digitally-signed element was replaced with a single hash negotiated during handshake, defaults to SHA-1.
Enhancement in the client's and server's ability to specify which hash and signature algorithms they will accept.
Expansion of support for authenticated encryption ciphers, used mainly for Galois/Counter Mode (GCM) and CCM mode of Advanced Encryption Standard encryption.

TLS Extensions definition and Advanced Encryption Standard CipherSuites were added.
Apps. Server: Why SSL/TLS?

- Provides encryption
- Provides Identification

Hiding what is sent from one computer (1) to another (2)

Making sure the computer you are speaking to is the one you trust
Apps. Server: Why SSL/TLS?

Encryption

- Client and server agree on how to encrypt
- Server send certificate
- Client ask server to start encrypting
- Server to send a start encryption
- All Data are now encrypted
Apps. Server: Why SSL/TLS?

- Client and server agree on how to encrypt

Hello

Pub Key: RSA
Cipher: AES
Hash: HMAC-MD5

Random num: 612316256.. To compute the master secret key

Version: 3.3
Client and server agree on how to encrypt

Server sends certificate to client

Serial number: 1244124
Issuer: Entrust, Verisign, self etc..
Valid: From-To
Public Key:
Subject:
Site, company, other details
Apps. Server: Why SSL/TLS?

- Client and server agree on how to encrypt

1. Client Key exchange
2. Change Cipher Spec.
3. Done!

Both Server & Client compute a master secret code

Let’s Encrypt

Start encrypting
Apps. Server: Why SSL/TLS?

Identification

Making sure the computer you are speaking to is the one you trust
Apps. Server: Why SSL/TLS?

Identification

Who to trust?

a) Company ask. CA for a certificate
b) CA create certificate and sign it
c) Certificate is installed (browser or elsewhere)
d) Browser trusts valid certs (correctly signed)
SSL/TLS: Messages

SSL Messages

1. Client hello
2. Server hello
3. Certificate *optional*
4. Certificate request *optional*
5. Server key exchange *optional*
6. Server hello done

7. Certificate *optional*
8. Client key exchange
9. Certificate verify *optional*
10. Change cipher spec
11. Finished

12. Change cipher spec
13. Finished
14. Encrypted data
15. Close messages
X.509 Certificates

Certificate Information

This certificate is intended for the following purpose(s):
- Ensures the identity of a remote computer
- Proves your identity to a remote computer

*Refer to the certification authority’s statement for details.

Issued to: pclick.internal.yahoo.com

Issued by: DigiCert High Assurance CA-3

Valid from 05/12/2010 to 09/12/2012

Learn more about certificates
**X.509 Certificates**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>V3</td>
</tr>
<tr>
<td>Serial number</td>
<td>08 62 d9 dd 38 58 b4 41 70 bc...</td>
</tr>
<tr>
<td>Signature algorithm</td>
<td>sha1RSA</td>
</tr>
<tr>
<td>Signature hash algorithm</td>
<td>sha1</td>
</tr>
<tr>
<td>Issuer</td>
<td>DigiCert High Assurance CA-3,...</td>
</tr>
<tr>
<td>Valid from</td>
<td>December-05-10 7:00:00 PM</td>
</tr>
<tr>
<td>Valid to</td>
<td>December-09-12 6:59:59 PM</td>
</tr>
<tr>
<td>Subject</td>
<td>CLICK internal yahoo.com.Yah</td>
</tr>
</tbody>
</table>

Learn more about [certificate details](#)
JSSE (Java™ Secure Socket Extension)

http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html#SSLOverview
JSSE (Java™ Secure Socket Extension)

- Java Keytool:
- keytool usage?
- Open a terminal!
SSH (Secure Shell)

- Introduction
- Overview
- SSL/TLS
- SSH
- SSO
- Other Security Options
- PKI
- Conclusion
SSh (Secure Shell)

- **Secure Shell** or **SSH** is a network protocol that allows data to be exchanged using a secure channel between two networked devices. The two major versions of the protocol are referred to as **SSH1** or **SSH-1** and **SSH2** or **SSH-2**.

- Used primarily on **Linux** and **Unix** based systems to access shell accounts, SSH was designed as a replacement for **Telnet** and other insecure remote shells.
SSH (History and Dev.)

Version 1.x

- In 1995, Tatu Ylönen, a researcher at Helsinki University of Technology, Finland, designed the first version of the protocol (now called SSH-1) prompted by a password-sniffing attack at his university network. The goal of SSH was to replace the earlier rlogin, TELNET and rsh protocols, which did not provide strong authentication or guarantee confidentiality.
SSH (History and Dev.)

Version 2.x

- "Secsh" was the official Internet Engineering Task Force's (IETF) name for the IETF working group responsible for version 2 of the SSH protocol. In 2006, a revised version of the protocol, **SSH-2**, was adopted as a standard. This version is incompatible with SSH-1.

- SSH-2 features both security and feature improvements over SSH-1.
SSH (History and Dev.)

Version 2.x

- Better security, for example, comes through **Diffie-Hellman key exchange** and strong **integrity** checking via **message authentication codes**. New features of SSH-2 include the ability to run any number of **shell** sessions over a single SSH connection.

- Risk ➔
All Versions

- **Vulnerabilities**
- In November 2008, a vulnerability was discovered for all versions of SSH, which allowed recovery of up to 32 bits of plaintext from a block of ciphertext that was encrypted using what was then the standard default encryption mode, **CBC**.
SSH (Standard)

- **RFC 4250**, The Secure Shell (SSH) Protocol Assigned Numbers
- **RFC 4251**, The Secure Shell (SSH) Protocol Architecture
- **RFC 4252**, The Secure Shell (SSH) Authentication Protocol
- **RFC 4253**, The Secure Shell (SSH) Transport Layer Protocol
- **RFC 4254**, The Secure Shell (SSH) Connection Protocol etc..
SSH (Can be used for ..)

- SSH is a protocol that can be used for many applications across many platforms including
  - **Unix**,  
  - **Microsoft Windows**,  
  - **Apple's Mac OS X**,  
  - and **Linux** etc..  
- Anything else?
SSH (Can be used for ..)
SSH (Libraries..)

- Does not come with default JDK
- Open source
- OpenSSH (C )
- Ganymed SSH (Java)
- Commercial
- Maverick SSH (java)
- SSH Comm. SSH (java and C)
- And More
SSH (Libraries..)

- Challenges when using Open source libs?
- Pros and Cons!

- Your Opinion! 😊
SSO (Single Sign On)

- Introduction
- Overview
- SSL/TLS
- SSH
- SSO
- Other Security Options
- PKI
- Conclusion
Single Sign On (Why, How)

SSO

User Management
Application & Portal

Multiple User Apps but login once

Users

1

2

SSL

AES
Single Sign On (How)

SSO

Single sign-on (SSO) is a property of access control of multiple related, but independent software systems. With this property a user logs in once and gains access to all systems without being prompted to log in again at each of them. Single sign-off is the reverse property whereby a single action of signing out terminates access to multiple software systems.

As different applications and resources support different authentication mechanisms, single sign-on has to internally translate to and store different credentials compared to what is used for initial authentication.
Single Sign On (Mechanism)

- Tokens

Digital Certificates

Protocol
SAML or Proprietary
Contents

- Introduction
- Overview
- SSL/TLS
- SSH
- SSO
- Other Security Options
- PKI
- Conclusion
SSH (Can be used for ..)

SNMP

Other equipment
Other Sec. Options

- End to end security in the networking space

- AES
- SSH
- NE
- User Management
  - Application
  - Portal

1. OSS
2. User Management
   - Application
   - Portal
3. Radius, Tacacs+, LDAP
4. IdP/RAS
5. Service Provider
   - Network Manager

SSL -> AES

SSH -> NE
Other Sec. Options

- End to end security in the networking space

- AES
- SSH
- NE
- IdP/RAS
- SSL
- Radius, Tacacs+, LDAP
- AAA
- SSH → NE
- AES
Other Sec. Options (AAA)

- Remote Authentication Dial In User Service (RADIUS) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA) management for computers to connect and use a network service. RADIUS was developed by Livingston Enterprises, Inc., in 1991 as an access server authentication and accounting protocol and later brought into the Internet Engineering Task Force (IETF)
Other Sec. Options (AAA)

- In computer networking, TACACS+ (Terminal Access Controller Access-Control System Plus) is a Cisco Systems proprietary protocol which provides access control for routers, network access servers and other networked computing devices via one or more centralized servers. TACACS+ provides separate authentication, authorization and accounting services.
Contents

- Introduction
- Overview
- SSL/TLS
- SSH
- SSO
- Other Security Options
- PKI
- Conclusion
PKI?

- Trust store
- Key store
- Cert. enrolment
- Valid for # years!
- Revocation
- Cert. Expiry
- Key backup
- Conclusion