Advanced Network Security
SSL/TLS

Thomas Martin

t.martin@mmu.ac.uk

March 12, 2018

Outline

Introduction

Secure Sockets Layer (SSL) and **Transport Layer Security** (TLS) use asymmetric cryptography to allow to parties to authenticate and establish a session key to protect the rest of the session.

The two protocols are often grouped together, and they do serve the same purpose. However, they are incompatible. Most browsers and web servers support both protocols. Within each version, different implementations can support different sets of cryptographic algorithms (called *suites*). The protocols allow the strongest compatible algorithm to be negotiated.

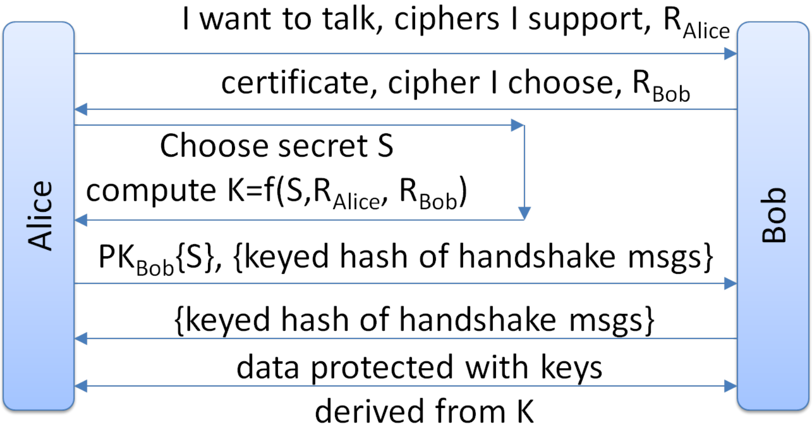
SSL/TLS operates in the application layer of the TCP model, but is used to tunnel other protocols: email, IM, VoIP, but mostly HTTP.

History

SSL version 2 was developed by Netscape in 1995. Microsoft released a competiting protocol called Private Communications Technology (PCT), to promote their Internet Explorer against the competition of Netscape Navigator. In response to the many security vulnerabilities in v2, Netscape updated to SSLv3.

In 1999, IETF produced another, incompatible protocol TLS version 1.0, intended as an upgrade to SSLv3. There are enough differences in the two to prevent compatibility. However, TLS 1.0 has an option to downgrade a connection to SSL v3. TLS also had a restriction to avoid cryptographic algorithms with commercial licenses or patents, which included RSA at the time.

Simplified Handshake


Alice
Bob
I want to talk, ciphers I support, R_{Alice}
certificate, cipher I choose, R_{Bob}
Choose secret S
compute $K=f(S,R_{Alice}, R_{Bob})$
$PK_{Bob}\{S\}$, {keyed hash of handshake msgs}
{keyed hash of handshake msgs}
data protected with keys
derived from K

Basic Protocol

Initially, the client (Alice) sends a request to start a connection to the server. She does not identify herself. She provides a list of cryptographic algorithms (symmetric, asymmetric, as well as hashing) and a random nonce.

The server (Bob) replies with a random nonce of his own, a selection from the ciphers Alice provided (usually the strongest), and his certificate.

Bob's certificate includes his public key. As previously mentioned, public keys remove the need for secrecy in key exchange. But the keys need to be authentic. Alice needs to be sure what she receives is in fact Bob's public key and not one provided by an impostor.

Certificate Verification

To ensure it is not replaced or modified, Bob's certificate includes his identity (domain name in this case) with the public key, all of which is signed. This moves the problem, but does not solve it, for Alice now needs to make sure she has the correct public key of whoever signed Bob's certificate.

Any server will typically provide a chain of such certificates, the last one signed by a Trusted Root Authority. In the settings of any browser (or other client that supports SSL/TLS) will be a list of such authorities. Alice is free to customize this as needed, but the majority of certificates are signed by one of Comodo, IdenTrust, DigiCert, GoDaddy, or GlobalSign[1].

[1] `https://w3techs.com/technologies/overview/ssl_certificate/all`

Basic Protocol

Once Alice is satisfied she does have Bob's public key, she generates a random number S (the **pre-master secret**), and uses Bob's key to encrypt it. The **master secret key** K is generated from S and the two nonces. Alice also sends a keyed hash of the master secret key, and the previous messages. This serves two purposes:

1. Proves to Bob that Alice knows the key K
2. Proves to Bob that the first two handshake messages were not tampered with

There are actually six keys generated. Keys for transmission are called *write* keys, and keys for receiving are called *read* keys. In each direction, there is a key for encryption, integrity and an IV.

Basic Protocol

Bob uses his private key to decrypt S, and can then calculate the rest of the keys. After verifying the integrity of the handshake messages, Bob needs to provide the same assurance to Alice. Obviously, the same keyed hash sent back would not prove anything. Instead, a special field is used to identify who is calculating and sending the keyed hash, "CLNT" for Alice and "SRVR" for Bob.[2] Also, as mentioned different keys are used for different directions.

Assuming the exchange completes, Alice has authenticated Bob. The protocol supports the optional authentication of Alice by Bob. As this requires that the client have a certificate, this is rarely done in practice (username and password being used instead).

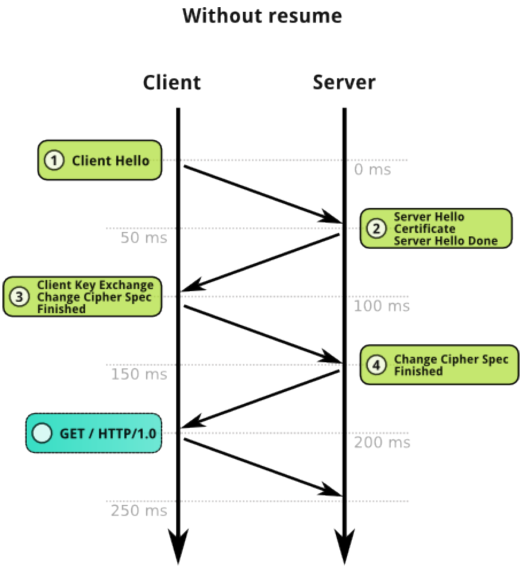
[2] Different strings are used in TLS. SSL v2 only had two session keys, one in each direction

Session Resumption

The SSL/TLS exchange is quite expensive in terms of computations. And because it was designed to work with HTTP, a client is likely to want to open several connections to the server simultaneously. A browser can display a page more quickly if it can download the many files (html, images, style sheets, scripts, etc.) in parallel rather than one at a time. Multiple connections are allowed in SSL/TLS that use nonces to ensure unique session keys, but do not require the public key operations.

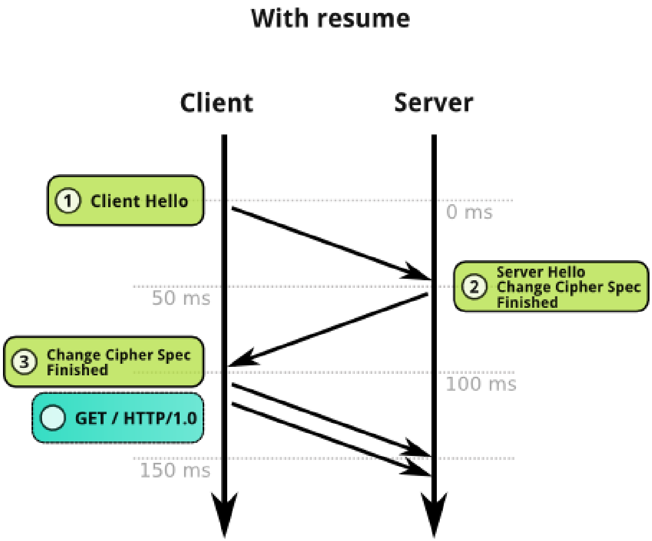
The mechanism for this is the same one used to allow a session to be resumed. In the initial exchange, Bob provides the client with a *session-id*. This can later be submitted by Alice to initiate a connection without going through the whole initial exchange.

Without Resume[3]



[3] https://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077

With Resume[4]



[4] `https://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077`

Protocol Details

The random nonces are 32 bytes long, and the practice is to use the first 4 bytes to be the Unix time when the message was generated. This guarantees that nonces generated for different sessions will be different (even though the probability of nonces of this size repeating is incredibly small).

Client authentication is supported in SSL/TLS, but very rare. The server can send a “certificate request” in message 2. It will also specify the X.500 names of the CAs it trusts and the types of keys it can handle. Since X.500 is seldom used, typically the common name is interpreted as being the domain name, and the rest is ignored. Alice will reply with her certificate, as well as her signature on a hash of the handshake messages.

Trusted CAs

Clients (browsers) come with a list of “trusted” CAs, determined by the developer, but typically editable by the user (or administrator at least). As usability is highly rated among browser developers, entities are in this list not necessarily because they have proved their trustworthiness, but because no obvious reason to not trust them has become publicly known.

Certificate errors are often encountered, but rarely due to an impersonation attack. They are more likely to be:

- Certificate is out of date
- Certificate is self-signed
- Mis-configuration of the web server

Cipher Suites

SSL/TLS defines different cipher suites, which include all cryptographic algorithms used in the protocol (symmetric/asymmetric encryption, key length, integrity checksum). In SSL v2 this was a 3 byte value, but was changed to two bytes in SSLv3/TLS. There are about 30 defined cipher suites, with a descriptive name as well as a byte value.

Cipher suites labelled with `EXPORT` were deliberately configured to be weak. This was because countries like USA had restrictions on the strength of encryption products that were exported.

SSL/TLS is also configured to support the negotiation of the compression method used on the transferred data. This previously only included NULL compression due to patents protecting most methods, but the `DEFLATE` method has since been added.

Cipher Suites[5]

```
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_MD5
SSL_CK_RC4_128_WITH_MD5
SSL_CK_DES_192_EDE3_CBC_WITH_MD5
TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_NULL_SHA
```

[5] http://msdn.microsoft.com/en-us/library/aa374757(VS.85).aspx

Attacks fixed in SSLv3

There are two important attacks on SSLv2 that were fixed in the upgrade:

Downgrade Attack: In version 2, there was no integrity protection on the initial handshake. Since this included the cipher suite negotiation, it would be possible for an attacker in the middle to modify the list (remove all but the weakest cipher Alice suggested). If an attacker attempted this in version 3, then the keyed hash calculated on the message would be different from what the other party sent.

Truncation Attack: Version 2 depended on TCP connection closing. But since there were no cryptographic protections, this could be easily spoofed, allowing an attacker to prematurely end the connection.

Padding Oracle Attack

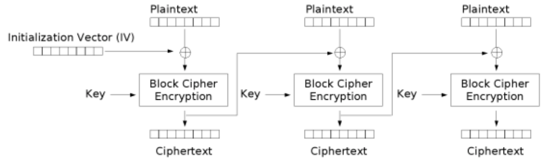
Padding Oracle Attacks on CBC were theorized in 2002 by Serge Vaudenay[6], but were not actually implemented and demonstrated until 2010[7].

Padding is necessary when text of variable length needs to be encrypted in blocks. Extra data needs to be added in a way that it can be unambiguously removed. TLS uses PKCS7 padding, which adds p bytes all with value p in order to make up the necessary block length. E.g.:
`d8 b8 0f d9` becomes `d8 b8 0f d9 04 04 04 04`

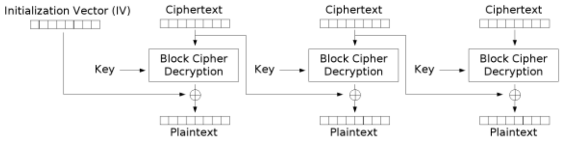
[6] `http://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf`

[7] `https://threatpost.com/en_us/blogs/new-crypto-attack-affects-millions-aspnet-apps-091310`

Cipher Block Chaining Mode[8]



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

[8] https://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Padding Oracle Attacks

If the server receives an malformed message in an SSL channel, there are two possible (and very similar) responses. If the decrypted message does not have proper padding at the end, it will be rejected and an error returned. If the decrypted message does not have the required integrity protection (keyed hash), it will be rejected and an error returned. An attacker is considered to have a *Padding Oracle* if he is able to tell the difference between the two for any message he submits.

Padding oracles are not that difficult to construct. Even if the server does not give specific information in the error message, the extra time taken to calculate the keyed hash can tell that the padding was correct. The size of the log file (how many characters were added) could also leak the type of error.

Padding Oracle Attacks

A padding oracle can give a single bit about decrypted text. This is limited on its own, but can be developed to decrypt of an entire block. The attacker submits two blocks, the first r with random bytes, the second y the block he wishes to decrypt. Suppose the block length is 8 bytes. The attacker submits to the oracle $r|y$, changing the last byte of r to all possible 256 values. One of these will be correctly padded. If we knew what padding was used, then we could work out the corresponding bytes of y.
If we suppose the last decrypted block ended in 0x0202, then we could modify r to convert this to end in 0x01 (need to XOR the last block of r with 0x03$= 11_2$). If submitting this to the oracle results in a positive response, we know the padding was 0x0202. Otherwise, we can do a similar test to check for 0x030303, 0x04040404, etc. Eventually we know the padding, and hence some of the plaintext.

Padding Oracle Attacks

This can be easily extended to decrypt the rest of the block y. Suppose the padding was 0x01, and the attacker has y_8. The attacker changes $r_8 \rightarrow r_8 \oplus$ 0x03, and tries all possible values of r_7. The one that the oracle says was padded correctly will directly determine $dec(y)_7$ because $r_7 \oplus dec(y)_7 =$ 0x02. This process can be repeated byte-by-byte until the whole block is decrypted. The only downside to this attack is that it cannot be used to decrypt the first block of any message. While the IV is known (ciphertext from previous message), it cannot be changed by the attacker.

The fix for this attack is to perform the verification of the keyed hash first, before removing the padding. The attacker will not be able modify blocks in a way that maintains the cryptographic integrity, so he will not get as far as being able to use the padding oracle.

B.E.A.S.T.

In 2011, researchers Duong and Rizzo released the Browser Exploit Against SSL/TLS (BEAST). This was a proof of concept, practical exploit against TLS 1.0, based on a CBC vulnerability discovered in 2004 by Gregory Bard[9]. TLS 1.1 was released in 2006 and 1.2 in 2008, and both provide protection against the attack. However, adoption was slow, and many sites still support the less secure 1.0 version[10].

BEAST exploits the fact that the mode of operation (CBC) uses a predictable IV (the ciphertext of the previous messages). BEAST was demonstrated on the OpenSSL package.

[9] http://eprint.iacr.org/2004/111
[10] https://www.ssllabs.com/ssl-pulse/

B.E.A.S.T.

Guessing if a ciphertext block is the encryption of a given plaintext block (without the key) is straightforward under CBC. Suppose an attacker observes a ciphertext C_i, and guesses that the plaintext P_i is x. He simply gets the user to encrypt $P_j = (x \oplus C_{i-1} \oplus C_{j-1})$. If he's right then encryption gives:

$$C_j = Encrypt(x \oplus C_{i-1} \oplus C_{j-1} \oplus C_{j-1}) = Encrypt(P_i \oplus C_{i-1}) = C_i$$

All this means is brute-force against the plaintext is possible, but just like brute-forcing the key it is impractical. What BEAST does to improve this is to inject known bytes into the plaintext so that the secret plaintext does not take up a whole block. If 7 of the 8 bytes are known and one unknown, it only takes 256 guesses. The boundary is moved and the next byte is attacked.

B.E.A.S.T.

BEAST manages to decrypt cookies from a secure SSL/TLS session, which allows impersonation. It requires malicious javascript, but not on the target site (it breaks the same-origin policy). It also requires the network traffic to be captured separately. Due to the number of queries, it takes half an hour to decrypt one cookie.

The countermeasure for BEAST is to force a random IV (unpredictable) for every message. TLS 1.1 and 1.2 both include this. Earlier versions have implemented a workaround to fix this, OpenSSL keeps the standard IV, but includes an empty message first to effectively create a random IV.

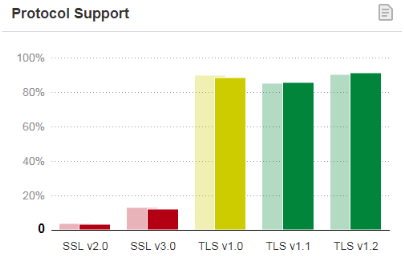
C.R.I.M.E.

The authors of BEAST also developed the Compression Ratio Information leak Made Easy (CRIME) attack[11]. The idea exploits SSL Compression, which is enabled by default in many webservers (notably Apache). SSL Compression uses a dictionary to replace long repeated strings. If the attacker can inject data into the SSL stream, they can try to guess different values of portions of the cookie. When they get it correct, it will result in a shorter encrypted message.

The recommended countermeasure is to disable SSL Compression (as well as the less well used SPDY protocol).

[11] http://isecpartners.com/blog/2012/9/14/details-on-the-crime-attack.html

SSL/TLS Survey[12]

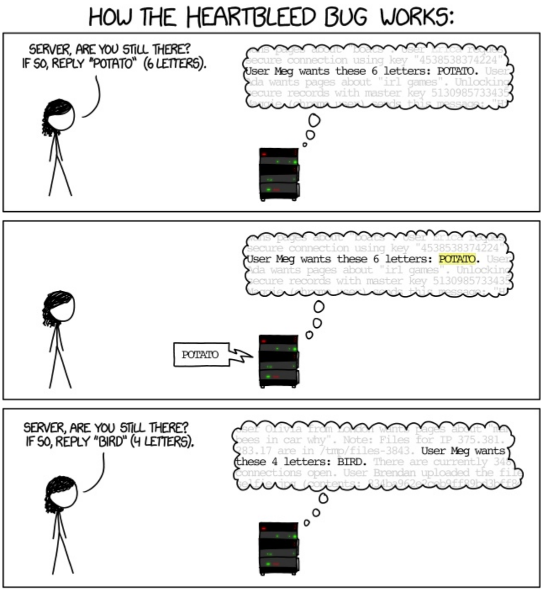


The survey grades sites security:

- Complete certificate chain
- Cipher suite/key strength
- Protocol support
- Strict Transport Security support
- Extended Validation Certificates
- Mitigation against various attacks
- Forward secrecy
- Use of RC4
- OCSP stapling

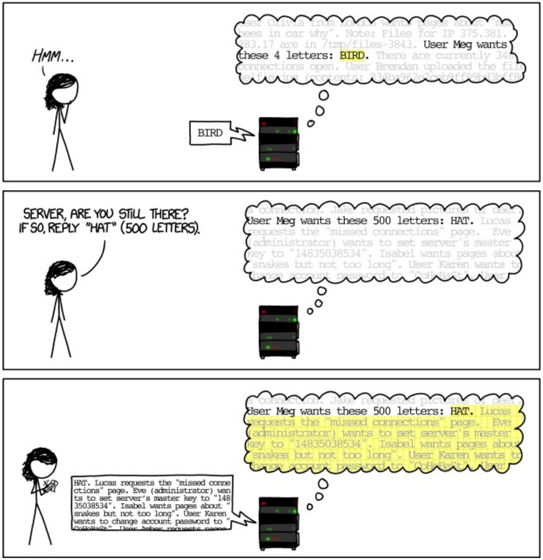
[12] https://www.trustworthyinternet.org/ssl-pulse/

Heartbleed Attack[13]


HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
ans pages about "boats". User Erica requests secure connection using key "4538538374224". User Meg wants these 6 letters: POTATO. User Ada wants pages about "irl games". Unlocking secure records with master key 5130985733435. Meggie (chrome user) sends this message: "H
POTATO
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Olivia from London wants pages about "bees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8

13 https://xkcd.com/1354/

Heartbleed Attack


HMM...
User Olivia from London wants pages about "man bees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff84
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages

Heartbleed Attack

Heartbleed[14] was a very serious attack that was brought to the public's attention (in a very professionally branded manner) in April 2014. It was due to an implementation error in the highly popular OpenSSL library, and affected an estimated 17% of webservers (as well as some VPNs).

TLS has a Heartbeat extension to make sure a connection is kept active (avoids expensive renegotiation). Due to improper input validation, an attacker can perform a buffer over-read (on server or client) and read more data than allowed. It was demonstrated that private keys were exposed[15].

[14] `http://heartbleed.com/`

[15] `https://blog.cloudflare.com/the-results-of-the-cloudflare-challenge/`

In Class Tasks

The vast majority of encrypted connections over the Internet are performed over SSL/TLS.

- What would happen if SSL/TLS could no longer be trusted? If it was assumed by all that the encrypted connections could be easily eavesdropped by anyone?
- Is the privacy of connections more or less important than the protection of end systems, e.g. from malware?
- How would you explain the importance and significance of SSL/TLS to someone who was not tech-savy (e.g. a younger or older friend/relative)?

PKI

Public key cryptography is a vital component of many systems. At a very basic level, there is a need for secure communication and pure symmetric systems are only efficient to a certain point. Where you just need confidentiality between a fixed number of entities it works. But for a scalable solution, one that can dynamically accommodate new participants, and one that can offer such services as verifiability to third parties and non-repudiation, asymmetric encryption is needed.

Any security system is going to need to use cryptography at some point. Authentication requires the transfer or credentials. If disclosed or modified, there will be problems. There needs to be a way to hide the details of the messages from eavesdroppers, and to make sure they are safe from malicious modification.

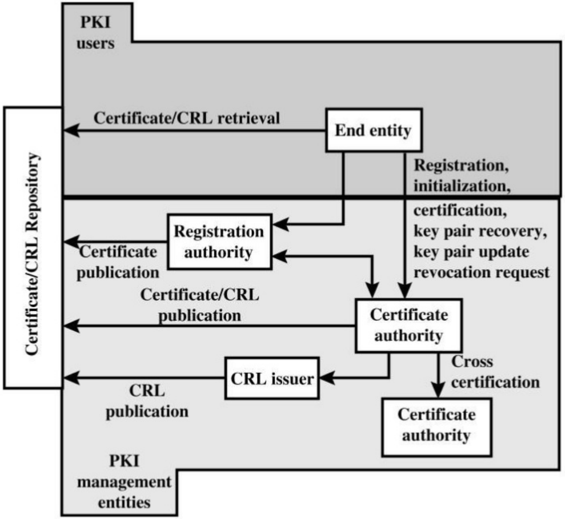
PKI

But PKIs can offer much more than just secure communications. They offer a way to bind an un-forgeable digital construct (a private key) with real person. If this is done in a reliable and universal way, then it simplifies the problems of managing identities over the network.

The IETF setup a Public Key Infrastructure X.509 (PKIX) working group to devise a certificate-based architecture for the Internet. PKI is defined as[16]:
The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

[16] `https://tools.ietf.org/html/rfc4949`

PKI Overview[17]


PKI users
Certificate/CRL retrieval
End entity
Registration, initialization, certification, key pair recovery, key pair update revocation request
Certificate/CRL Repository
Registration authority
Certificate publication
Certificate/CRL publication
Certificate authority
CRL issuer
Cross certification
Certificate authority
CRL publication
PKI management entities

[17] Stallings, Cryptography and Network Security, Pearson.

PKI Entities

- End entity: Generic term for anyone what would want to be identified in a PKI.
- Certification authority (CA): Issuer of certificates (may be combined with other functions, such as managing CRLs).
- Registration authority (RA): Responsible for registering end entities.
- CRL issuer
- Repository

PKI Processes

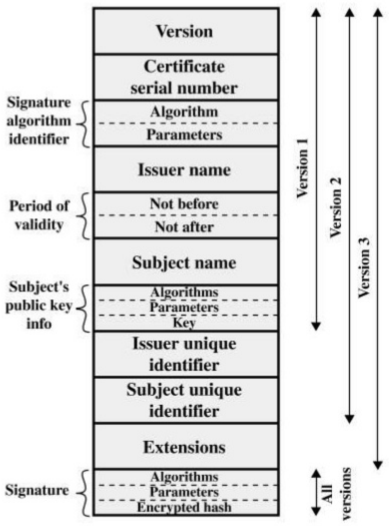
There are a number of functions necessary in the operation of a PKI:

- Registration: A user provides proof of who they are to the CA/RA.
- Initialization: All clients must have installed and secured the credentials of trusted entities (i.e. public keys of root CAs).
- Certification: The CA issues a certificate for a user's public key.
- Key pair recovery: If private keys are lost/destroyed, all data (past and future) encrypted with that key is lost. Having redundancy is important.
- Key pair update: Certificates have limited lifetimes and need to be updated.

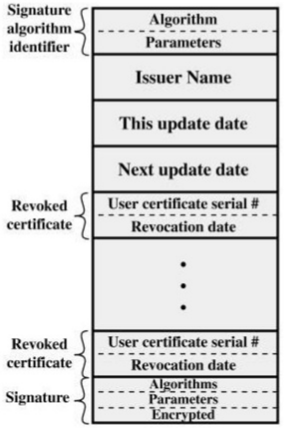
PKI Processes

- Revocation request: If a private key is disclosed, or the entity's name changes, the certificate must be revoked.
- Cross certification: If two CA's trust each other, they can sign each other's certificates. Their users will then be able to seamlessly accept certificates signed by the other authority.

Certificates[18]



(a) X.509 certificate



(b) Certificate revocation list

[18] Stallings, Cryptography and Network Security, Pearson.

Certificate Types

Different types of certificate represent different levels of validation:

- Domain validation (DV): purchaser demonstrates administration of the domain (e.g. can upload a file)
- Organization validation (OV): purchaser demonstrates administration of the domain and their organization exists as a legal entity
- Extended validation (EV): manual verification checks by a human of the existence of the organisation

Since the introduction of the Let's Encrypt service[19] offering free DV certificates, major sites are expected to have EV certificates.

[19] https://letsencrypt.org/

Implementing a PKI

What is the minimum needed to run a CA? OpenSSL commands can create a root certificate that can be used to create certificates:

```
> openssl req -new -newkey rsa:2048 -nodes -out ca\ca.csr
-keyout ca\ca.key
> openssl x509 -trustout -signkey ca\ca.key -days 365
-req -in ca\ca.csr -out ca\ca.pem
> openssl x509 -CA ca\ca.pem -CAkey ca\ca.key -CAserial
ca\ca.srl -req -in server\server.csr -days 365
-out server\server.pem
> openssl pkcs12 -export -in server\server.pem -inkey
server\server.key -out server\server.pfx -certfile ca\ca.pe
```

The two certificates need to be installed onto the webserver, and the CA's certificate needs to be installed on the client.

PKI Topology

There are a number of different possible topologies of the trust relationships in a PKI. If each user trusts a single CA, then there needs to be mechanisms in place to assist users in making trust decisions about entities in other realms. This could mean either restricting the PKI to a single root, or cross-signing of root CAs.

Another approach is the decentralized web-of-trust approach. PGP uses a system such as there where individuals sign each others public keys, and can decide how close someone needs to be in the "web" before they are trusted.

CA/Browser Forum

The PKI that governs SSL/TLS (as well as code-signing and other applications) has a collection of many trusted root-certificates. Each of these are granted full authority to sign certificates for any domain (as well as delegate this to intermediaries). The browser vendors are the ones who decide which root certificates to install the users browsers (users can over-ride this, but rarely do). As such, there is a need to ensure the CAs are behaving appropriate to warrant such trust.

The CA/Browser Forum[20] was created for all relevant parties to agree appropriate guidelines for issuing certificates.

[20] `https://cabforum.org`

CAB Forum Requirements - General

General Protections for the Network and Supporting Systems[21]:

- Maintain Root CA Systems in a High Security Zone and in an offline state or air-gapped from all other networks;
- Configure each network boundary control with rules that support only the services, protocols, ports, and communications that the CA has identified as necessary to its operations;
- Grant administration access to Certificate Systems only to persons acting in Trusted Roles and require their accountability for the Certificate System's security;
- Implement multi-factor authentication to each component of the Certificate System that supports multi-factor authentication;

[21] `https://cabforum.org/network-security/`

CAB Forum Requirements - Logging

Logging, Monitoring, & Alerting.

- Implement automated mechanisms under the control of CA or Delegated Third Party Trusted Roles to process logged system activity and alert personnel, using notices provided to multiple destinations, of possible Critical Security Events;
- Require Trusted Role personnel to follow up on alerts of possible Critical Security Events;
- Conduct a human review of application and system logs at least every 30 days and validate the integrity of logging processes and ensure that monitoring, logging, alerting, and log-integrity-verification functions are operating properly (the CA or Delegated Third Party MAY use an in-house or third-party audit log reduction and analysis tool);

CAB Forum Requirements - Vulnerability Detection

- Implement detection and prevention controls under the control of CA or Delegated Third Party Trusted Roles to protect Certificate Systems against viruses and malicious software;
- Document and follow a vulnerability correction process that addresses the identification, review, response, and remediation of vulnerabilities;
- Undergo a Penetration Test on the CA's and each Delegated Third Party's Certificate Systems on at least an annual basis and after infrastructure or application upgrades or modifications that the CA determines are significant;
- Record evidence that each Penetration Test was performed by a person or entity with the skills, tools, proficiency, code of ethics, and independence necessary to provide a reliable Penetration Test;

Revocation and OSCP

Where certificates need to be revoked, two simple approaches have been used in practice:

- CRL - Certificate Revocation Lists. The CA produces a signed list of all revoked certificates available on its website. A browser can check this when visiting a site to make sure the site has a valid certificate.
- OCSP - Online Certificate Status Protocol. CRLs can grow large in size, and downloading them can delay an initial connection to a secure site. Instead, a browser can query the CA for the status of a particular certificate.

Problems with PKI

Certificates can become invalid before their expiration date. There are different ways of making holders of certificates aware of this. The CA maintains a list of those certificates it has issued that have been revoked, or users engage in an Online Certificate Status Protocol (OCSP).

PKI was hailed as the solution to all security problems, but its growth has been comparatively slow. Criticisms include[22]:

- How is the trust placed in the CA warranted?
- An insecure client machine can undermine the claimed security (e.g. non-repudiation is not defensible)
- A verifier does not need a secret key, but that does not make them immune from attack. A false root certificate can cause fraudulent sites to appear legitimate

[22] `http://www.schneier.com/paper-pki.pdf`

Problems with PKI

- Names used in certificates may not be unique. It is not enough to use a format that gives enough information to uniquely identify someone, you need to rule out certificates that are "close-enough" that they are incorrectly accepted
- Much of the technical details are difficult for average users to grasp
- How can a CA identify an applicant? Personal information is often used, but this can be readily obtained from Credit Bureaus
- Have the CA's practices been designed with solid security reasons? Why are the certificate lifetimes/key lengths set to the values they are?
- PKI solves a very specific problem, and cannot be naively applied to solve any security problem (e.g. SSO)

2011 Comodo attacks

On March 15, 2011, Comodo (the third largest SSL Certificate Authority), announced[23] an affiliated Registration Authority had been compromised. An attacker requested, and is believed to have obtained certificates for 9 sites, including:
`mail.google.com`, `login.live.com`, `login.yahoo.com`, `login.skype.com` and `addons.mozilla.org`.

Due to the fact that the origin IP addresses were based in Iran, and the fact that the target was communications not financial sites, Comodo initially stated that this was a "state-driven attack". Subsequently, a lone hacker, going by the name of "ComodoHacker" claimed responsibility, releasing one of the related private keys.

[23] `http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`

Attacks on PKI

The compromise was quickly detected, the certificates were quickly revoked by CRL and OCSP. The window was too small for any serious attack to take place. The major Browser publishers were notified and updates quickly deployed. Mozilla made the following requests of Comodo:

1. Publish a full account of exactly what happened. (they published an incident report and a blog post.)
2. Monitor their OCSP logs for evidence of use of these certificates, or evidence that access to their OCSP responders is being blocked from any geographical locations. (there was no sign of use or blocking.)
3. Cancel all relationship with the RA concerned. (the RA was suspended.)
4. Change their practices to use intermediate certs rather than issuing directly off the root, and use a different one for each RA.

2011 DigiNotar attacks

On 28th August 2011, several Iranian users reported certificate problems. A man-in-the-middle attack had been attempted on Google sites using certificates issued by the Dutch DigiNotar CA[24]. The CA had been compromised in July, and despite detecting the breach, did not announce it. At least 531 certificates were issued in error.

Chrome users were initially the first to notice the problem, as Chrome has inbuilt features to check Google's own certificates. Google, Microsoft and Mozilla all removed DigiNotar from their lists of trusted root CAs, and the company went bankrupt within a month of the attacks becoming public.

DigiNotar was also one of the CAs for the Dutch Government's PKI. Once their certs were marked untrusted, access to government services was impacted, until DigiNotar was replaced.

[24] http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-man-in-middle.html

2011 Malaysian CA issued certs with weak keys

The Malaysian certificate authority Digicert issued at lest 22 certificates with 512-bit RSA keys[25]. As keys of this size had been broken as far back as 1999, their use is discontinued and CAs are required not to sign such requests. They also lacked an EKU extension specifying their intended usage (i.e. could be used for any purpose) and were issued without revocation information.

Due to the highly suspicious behavior, the Digicert intermediate certificate authority was removed from both Mozilla and Microsoft's trusted list.

[25] https://nakedsecurity.sophos.com/2011/11/03/another-certificate-authority-issues-dangerous-certficates/

2012 Trustwave man-in-the-middle certificate

In February 2012, CA Trustwave revealed that they had issued a digital certificate that enabled a private company (they did not reveal which one) to spy on SSL-protected connections[26]. This was achieved by using a subordinate root certificate that could sign fake certs for any domain on the Internet.

The announcement did not come about due to a breach. In fact, Trustwave assured all that the private key was securely contained within a Hardware Security Module. The purpose of the certificate was to enable a data-loss prevention system to monitor all out-going connections. The traditional approach of using custom root CAs installed in clients does not work when users bring their own devices inside the network.

[26] https://www.computerworld.com/s/article/9224082/Trustwave_admits_issuing_man_in_the_middle_digital_certificate_Mozilla_debates_punishment

2015 CNNIC Misused Certificates

In March 2015 it was revealed that unauthorized certificates were being issued for several Google-related domains[27]. The certificates were issued by the Egypt-based MCS Holdings, who were operating a man-in-the-middle proxy. The root CA that had granted this ability to MCS was the China Internet Network Information Center (CNNIC).

Google's response was to remove CNNIC root certificate from Chrome.

[27] `http://arstechnica.com/security/2015/04/google-chrome-will-banish-chinese-certificate-authority-for-breach-of-trust/`

2015 CNNIC Misused Certificates

Mozilla's response was:

We have concluded that CNNIC's behaviour in issuing an unconstrained intermediate certificate to a company with no documented PKI practices and with no oversight of how the private key was stored or controlled was an 'egregious practice' as per Mozilla's CA Certificate Enforcement Policy. Therefore, after public discussion and consideration of the scope and impact of a range of options, we have decided to update our code so that Mozilla products will no longer trust any certificate issued by CNNIC's roots with a notBefore date on or after 1st April 2015.

2017 Symantec vs. Google

In March 2017, Google announced[28] that was going to limit the trust its browser (Chrome) placed in certificates issued by Symantec following the mis-issuance of 30,000 certificates and the manner in which Symantec had reacted (Symantec disputed this number, saying it was only 127).

Google explained they would no longer recognise the extended validation status of Symantec certificates, and that they would start a staggered/gradual nullification of all certificates.

Later in the year, Symantec announced that they were celling their CA Business (which they had purchased from VeriSign) to DigiCert.

[28] https://arstechnica.com/information-technology/2017/03/google-takes-symantec-to-the-woodshed-for-mis-issuing-30000-https-certs/

2018 Trustico Discloses Private Keys

Trustico is a reseller of TLS certificates issued by Comodo and Symantec. They contacted DigiCert (who was now running Symantec's certificate division) requesting that 50,000 certificates be revoked. When asked for proof, the Trustico CEO emailed across 23,000 private keys of those certificates[29].

While this had the effect of immediately forcing DigiCert to revoke the certificates, the lax security caused a great deal of concern among their customers and the community. According to the CA Baseline Requirements, resellers are not allowed archive private keys. To make matters worse, the Trustico website was taken down shortly after due to a vulnerability that granted root access.

[29] `https://arstechnica.com/information-technology/2018/03/23000-https-certificates-axed-after-ceo-e-mails-private-keys/`

Problems with Revocation

There are a number of problems with CRL's and OCSP[30].

- CRL growing in size, delaying initial connection.
- Privacy - OCSP informs CA of all domains you visit.
- CA with poor/no service can cause browsers to fail to connect to secure sites.
- OCSP Hard Fail is more security, but Soft Fail more common.
- MitM can simply block OCSP requests.
- System was not designed with bad CAs in mind.

Revocation has become a seat-belt that snaps if it is ever in an accident. Google Chrome has stopped checking revocation status and Mozilla Firefox looks set to follow.

[30] `https://scotthelme.co.uk/revocation-is-broken/`

DANE

DANE is DNS-based Authentication of Named Entities[31]. It allows certificates to be bound to DNS names using DNSSEC.
"*DANE allows the owner of a domain to signal which site certificate can be used, which CAs can be used and which public keys can be used for a given host in a domain*"[32]
One of the problems with the existing PKI is that any authority can sign certificates for any domain. With DANE, the administrator of a domain name gets to certify which keys are used in that domain's TLS servers. A CA that has mis-issued SSL certificates would be quickly noticed. The downside is that it does require DNSSEC records, which is not yet in widespread use. Certificate revocation is difficult, and the model is self-signing which is worrying to some.

[31] https://tools.ietf.org/html/rfc6698
[32] Yngve Pettersen

Certificate Transparency

Google are known for making frequent updates to their Chrome browser. An new feature they have been working on will make detection of mis-behaving CAs simple, without having to make widespread changes to the infrastructure. Their Certificate Transparency would create public logs that record every certificate signed by a CA[33].

The goal is to make it impossible (or at least very difficult) for a certification authority to issue a certificate for a domain without it being visible to the owner of that domain. Google began requiring Certificate Transparency for EV certificates in 2015.

[33] `http://www.certificate-transparency.org/`

Certification Authority Authorization

Certification Authority Authorization is a DNS resource record that allows a DNS domain name holder to specify the Certification Authorities authorized to issue certificates for that domain. It has been released as an IETF RFC[34]. The publication of these records would allow public CAs to implement additional controls to reduce the risk of unintended certificate issuance. Benefits include[35]:

- Increases the reliability of a validated domain name
- Implementation can be automatic and low maintenance
- Allows high-value targets to identify themselves, which will help with global verification of high-value domains

This approach is similar to DANE, but does not require DNSSEC.

[34] `https://tools.ietf.org/html/draft-ietf-pkix-caa-11`

[35] Adoption: `https://www.ssllabs.com/ssl-pulse/`

Key Pinning

Google has been using Public Key Pinning in their Chrome browser for a few years now. It has been the way some of the recent CA breaches have come to light. Google has used their dual roles as browser developer and high-traffic webserver. They know which authorities they used to issue their SSL certificates. These are stored in a white-list in the browser. If at any time a user browses to a Google site which is not certified by one of the pinned CAs, an error will occur and the page will not be presented.

Google plans on offering this feature to several other large, high-security websites. Potential issues include corporate MitM proxies, parental controls and debugging tools. This have been addressed by allowing user-installed CA certificates to override the pins.

OCSP Stapling

OCSP Stapling[36] reduces the back-and-forth needed for OCSP by including the OCSP status message with the certificate that is sent to the browser. The website will only need to obtain this OCSP message from the CA once a week or day, which reduces the strain on the CA site. The message is only valid for a short period (unlike the months-long validity period of certificates).

They same problem of a MitM simply blocking OCSP would also apply to stapling. Which is why certificates now have the OCSP Must Staple flag. With this flag set, the website must include a OCSP stapled certificate or the browser will reject the connection.

[36] https://scotthelme.co.uk/revocation-is-broken/#ocspmuststaple

Responses

Other suggested countermeasures to the attacks on SSL/TLS include:

- **HTTP Strict Transport Security** (HSTS)[37] - which allows webservers to force browsers to only connect using HTTPS
- **Convergence**[38] - a Firefox addon that obtains trust from a consensus among multiple notaries, preventing any single notary from having the ability to compromise security
- **SSL Observatory**[39] - An EFF run project that gathers public data on all operating CAs

[37] https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet
[38] https://addons.mozilla.org/en-us/firefox/addon/convergence-extra/
[39] https://www.eff.org/observatory

Thank you

Relevant chapters:
Stallings, Cryptography and Network Security, Chapters 14 and 17.

Any Questions?