Key Establishment Protocols

CS 411/507 - Cryptography

Erkay Savaş & Atıl Utku Ay

Department of Computer Science and Engineering
Sabancı University

December 19, 2023

Key Management Problem

- Security of the keys
  - Even if the cryptographic algorithms & protocols are cryptographically ultra-secure, a possible compromise of secret keys or part of them will have grave consequences.
- How two (or more) parties will exchange (agree on) keys for secret communication if they are unable to meet.
- Main problem is to share secret information for symmetric cryptography.
  - Public key cryptography keys may be stored on public databases.

Key Agreement Protocols

- Protocols whereby a secret key is established by exchanging information between two or more parties.
- Each party derives the secret key from the exchanged information.
- Key exchange is best done using public key cryptography.
- Diffie-Hellman protocol establishes a key with transfer of two messages.
- However, DH does not provide <u>authentication</u>.
- Station-to-Station protocol is an authenticated version of DH protocol.

DH Key Exchange

- Large prime p.
- A generator $\alpha \bmod p$

<u>Alice</u>

1. Picks a random a
2. Computes $p_A = \alpha^a \bmod p$
3. Publishes p_A
4. Computes k_{AB}
   $R_{BA} = {p_B}^a \bmod p$
   $R_{BA} = \alpha^{ba} \bmod p$

<u>Bob</u>

1. Picks a random b
2. Computes $p_B = \alpha^b \bmod p$
3. Publishes p_B
4. Computes k_{BA}
   $R_{AB} = {p_A}^b \bmod p$
   $R_{AB} = \alpha^{ab} \bmod p$

$$R = R_{AB} = R_{BA} = \alpha^{ab} \bmod p$$

$$K = \texttt{KGF}(R) \text{ (key generation)}$$

The Intruder-in-the-Middle Attack

Alice		Eve		Bob
a		e		b
$p_A = \alpha^a$		$p_E = \alpha^e$		$p_B = \alpha^b$
p_A	$\dashrightarrow$	$p_A \quad p_B$	$\dashleftarrow$	p_B
p_E	$\dashleftarrow$	$p_E \quad p_E$	$\dashrightarrow$	p_E
${p_E}^a = \alpha^{ea}$				${p_E}^b = \alpha^{eb}$
		$K_1 \equiv {p_A}^e = \alpha^{ae} \quad K_2 \equiv {p_B}^e = \alpha^{be}$		

Station-to-Station Protocol 1/2

- Authenticated key agreement protocol.
- Alice and Bob use their private keys to sign the exchanged messages.
- Sign function : $\texttt{Sign}_{sk}()$
- Verify function : $\texttt{Verify}_{pk}()$
- Public keys are obtained from a public trusted database

Alice	Bob	
a	b	private keys
$p_A = \alpha^a \bmod p$	$p_B = \alpha^b \bmod p$	public keys

Station-to-Station Protocol 2/2

Alice

$R_A = \alpha^x \bmod p$

$T = \alpha^{yx} \bmod p$

$K = \mathtt{Hash}(T||param)$

Bob

$R_B = \alpha^y \bmod p$

$T = \alpha^{xy} \bmod p$

$K = \mathtt{Hash}(T||param)$

$E_K(\mathtt{Sign}_b(R_B, R_A))$

Alice

$D_K(\mathtt{Sign}_b(R_B, R_A)) =$

$\mathtt{Sign}_b(R_B, R_A)$

$\mathtt{Verify}_{p_B}(\mathtt{Sign}_b(R_B, R_A))$

$E_K(\mathtt{Sign}_a(R_B, R_A)))$

Bob

$D_K(\mathtt{Sign}_a R_B, R_A)) =$

$\mathtt{Sign}_a(R_B, R_A)$

$\mathtt{Verify}_{p_A}(\mathtt{Sign}_a(R_B, R_A))$

Forward Secrecy

- **Definition:**
  - A secure communication protocol is said to have forward secrecy if compromise of long-term keys does not compromise past session keys
  - Attacker can store all past communications
- Station-to-Station Protocol provides forward secrecy
  - Bob sends $E_K(\texttt{Sign}_b(R_B, R_A))$ to Alice
  - Assume that Bob's private key b is compromised at a later date.
  - This is not going to disclose the session key y chosen by Bob in a previous session of the station-to-station protocol

Conference Keying

- Multi-party DH key exchange

Conference Keying 1/2

- Problem statement
  - A group of t users wants to agree on a key
    - U_i where $i = 0, 1, \ldots, t-1$
  - each user contributes to the agreed key
- Setup
  - Large prime p, smaller prime q and a generator g in G_q.
- Partial key generation
  1. User U_i selects a random integer $r_i, 1 < r_i < q$,
  2. computes $z_i = g^{r_i} \mod p$
  3. sends z_i to $U_{i-1 \mod t}$ and $U_{i+1 \mod t}$

- Computation of key
  - Each user U_i, after receiving z_{i-1} and z_{i+1} computes

$$x_i = \left(\frac{z_{i+1}}{z_{i-1}}\right)^{r_i} \bmod p = g^{r_{i+1}r_i - r_{i-1}r_i} \bmod p$$

  - and broadcasts x_i
  - After receiving x_j for $1 \leq j \leq t$ and $j \neq i$,
    U_i computes
    $K = K_i = (z_{i-1})^{tr_i} x_i^{t-1} x_{i+1}^{t-2} \cdots x_{i+(t-3)}^{2} x_{i+(t-2)}^{1} \bmod p$

Example: Conference Keying 1/2

- Four users U_0, U_1, U_2 and U_3
- They select r_0, r_1, r_2 and r_3 at random
- They compute
  - $z_0 = g^{r_0} \bmod p$
  - $z_1 = g^{r_1} \bmod p$
  - $z_2 = g^{r_2} \bmod p$
  - $z_3 = g^{r_3} \bmod p$
- User U_i sends z_i to U_{i-1} and U_{i+1} for $i = 0, 1, 2, 3$

Example: Conference Keying 2/2

- Upon receiving, each user computes corresponding values
  - $U_0 : x_0 = g^{r_1 r_0 - r_3 r_0} \bmod p$
  - $U_1 : x_1 = g^{r_2 r_1 - r_0 r_1} \bmod p$
  - $U_2 : x_2 = g^{r_3 r_2 - r_1 r_2} \bmod p$
  - $U_3 : x_3 = g^{r_0 r_3 - r_2 r_3} \bmod p$

$$K_0 = (x_3)^{4r_0} {x_0}^3 {x_1}^2 {x_2}^1 \bmod p = g^{r_3 r_0 + r_1 r_0 + r_2 r_1 + r_3 r_2} \bmod p$$
$$K_1 = (x_0)^{4r_1} {x_1}^3 {x_2}^2 {x_3}^1 \bmod p = g^{r_1 r_0 + r_2 r_1 + r_3 r_2 + r_3 r_0} \bmod p$$
$$K_2 = (x_1)^{4r_2} {x_2}^3 {x_3}^2 {x_0}^1 \bmod p = g^{r_1 r_2 + r_3 r_2 + r_3 r_0 + r_0 r_1} \bmod p$$

Key Distribution

- Shortcoming of the key pre-distribution protocol:
  - Keys are predetermined and not easily changed.
  - Keys must be changed after certain time.
- Transport protocols
  - A class of key establishment protocols
  - Two approaches:
    1. One party to decide on a key and transmit it to other,
       - Alice employs a secure protocol to transmit the key
    2. A trusted authority, Trent, will act as a key server.
       - Alice requests a key from Trent that is good for a single session
       - Trent sends this key to both Alice and Bob via a secure channel.

Authentication using Symmetric Encryption

- There exists a Key Distribution Center (KDC)
  - Each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - Master keys are used to distribute these session keys in a secure way

Needham-Schroeder Protocol

- A three-party key distribution protocol
  - For session between A and B, mediated by a trusted KDC
  - KDC should be trusted since it knows the session key
- Protocol Overview
  1. A→KDC: $ID_A||ID_B||N_1$
  2. KDC→A: $E_{K_A}(K_S||ID_B||N_1||E_{K_B}(K_S||ID_A))$
  3. A→B: $E_{K_B}(K_S||ID_A)$
  4. B→A: $E_{K_S}(N_2)$
  5. A→B: $E_{K_S}(f(N_2))$
- Step 4 and Step 5 prevent a kind of a replay attack against replay of message 3 by an attacker

Needham-Schroeder Protocol

- Protocol Overview
  1. A→KDC: $ID_A||ID_B||N_1$
  2. KDC→A: $E_{K_A}(K_S||ID_B||N_1||E_{K_B}(K_S||ID_A))$
  3. A→B: $E_{K_B}(K_S||ID_A)$
  4. B→A: $E_{K_S}(N_2)$
  5. A→B: $E_{K_S}(f(N_2))$
- Protocol is still vulnerable to a replay attack
  - If an old session key has been compromised, then message 3 ($E_{K_B}(K_S||ID_A)$) can be resent to B by an attacker X impersonating A.
  - After that, Eve intercepts message 4 ($E_{K_S}(N_2)$) and sends a message 5 ($E_{K_S}(f(N_2))$) to B as if it is A.
  - Now, Eve can impersonate A for the future communications with the session key

Needham-Schroeder Protocol with Timestamp

- Protocol Overview
  1. A→KDC: $ID_A||ID_B||N_1$
  2. KDC→A: $E_{K_A}(K_S||ID_B||N_1||E_{K_B}(K_S||ID_A||T))$
  3. A→B: $E_{K_B}(K_S||ID_A||T)$
  4. B→A: $E_{K_S}(N_1)$
  5. A→B: $E_{K_S}(f(N_1))$
- A and B can understand replays by checking the timestamp in the message
  - even if attacker knows K_S, he cannot generate message 3 ($E_{K_B}(K_S||ID_A||T)$) with a fresh timestamp since he does not know K_B.

Kerberos

- Kerberos originated from a larger project in M.I.T., called Athena.
  - Athena was originally designed for connecting a huge network of workstations so that students can securely access their files from anywhere in the net.
  - Uses only secret (symmetric) key cryptography
- Kerberos provides security and authentication in key exchange (or establishment) between users in a network.
  - Users could be programs as well as individuals
  - Supports both entity authentication and key establishment.

Kerberos

- Based on a <u>client-server</u> architecture.
  - A client is either a user or a software that has some tasks to accomplish.
    - Sends an e-mail, print documents, mount devices, etc.
  - Servers are larger entities whose function is to provide services to the clients.
- The basic Kerberos model has the following participants
  1. Cliff: a client
  2. Simon: a server
  3. Trent: a trusted authority (a.k.a. authentication server)
  4. Grant: ticket-granting server.

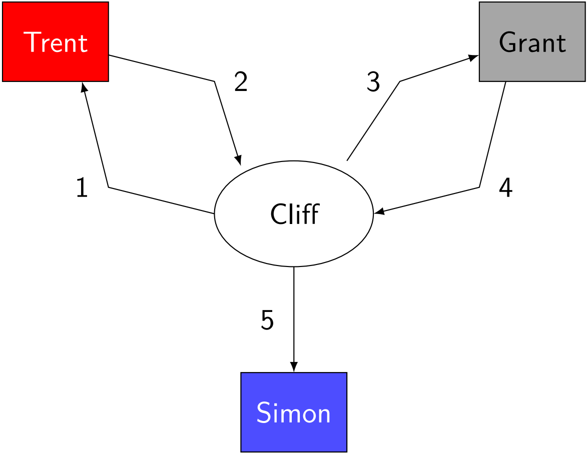
Kerberos: Roles

- Cliff requests a service from Simon,
  - But, they do not have any shared secret
  - Kerberos give them a secret information securely so that they can interact secretly.
- Trent
  - Authentication server
  - shares secret information (e.g., a password) with each user in the system
  - issues credentials to Cliff when he first logins.
    - with this credential Cliff can authenticate himself
  - Key Distribution Center (KDC).

Kerberos: Roles

- Grant
  - Ticket granting server
  - After Cliff logins to the system, Grant issues him a ticket to use any particular service
  - Cliff presents his credentials (issued by Trent) to Grant to get this new ticket
  - Ticket contains information, from which Cliff and Simon generates a shared key
- Simon
  - Service provider
  - Receives Cliff's ticket and fulfills Cliff's service request

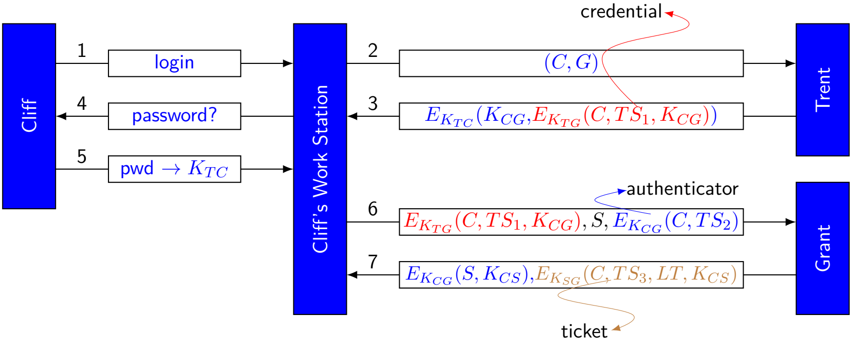
Overview of Kerberos Protocol



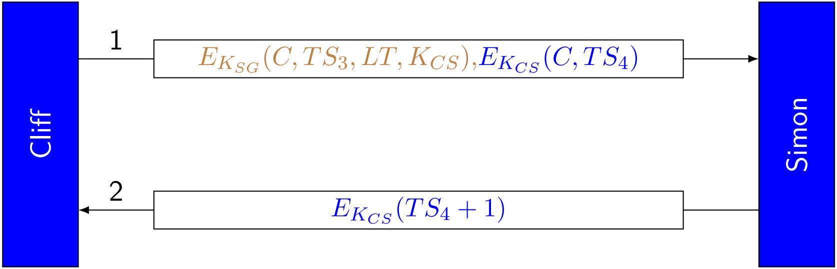
Kerberos: Notation

- K_{AB}: key that entity A and entity B share
  - K_{TG}: key Grant and Trent share
  - K_{TS}: key Simon and Trent share
- TS_i: timestamp
- LT: Validity period of the ticket (lifetime)
- $E_{K_{AB}}()$: encryption under key K_{AB}
- Credential: $E_{K_{TG}}(C, TS_1, K_{CG})$
- Authenticator: $E_{K_{CG}}(C, TS_2)$

Kerberos: Authentication



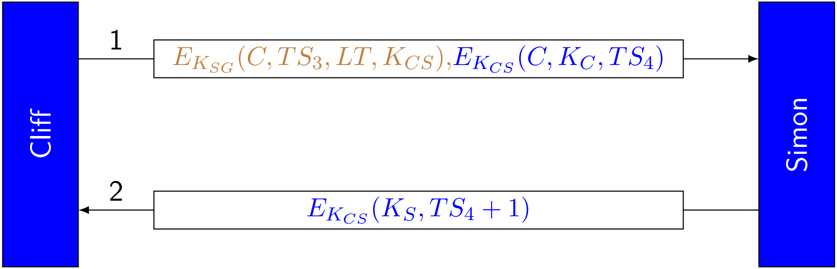
Kerberos: Use of Service



Security and Options in Kerberos

- Timestamps are used
  - Hosts must provide both secure and synchronized clocks.
- Security of Kerberos
  - If the shared key between Trent and Cliff are passwords, the protocol is no more secure than the strength of the password.
- Reuse of tickets
  - The lifetime of a ticket allows reuse of the ticket for multiple interactions with no additional interactions with Grant.
- Optional keys
  - K_C and K_S allow Cliff and Simon to combine these keys to derive another key, $F(K_C, K_S)$.

Optional Keys



Key Management in Public Key Cryptography

- In other words,
  - distribution of public keys
  - use of Public Key Cryptography to distribute secret keys
  - public/private key as a master key
- Basic question?
  - How can I make sure about the legitimacy of a public key?
  - How can I make sure that Bob's public key really belongs to Bob, not to Charlie?

Distribution of the Public Keys

- Public Announcement
  - Broadcast your public key to the public via forums, mailing lists, from personal website, whatsapp, social media, etc.
  - Major weakness is anyone can easily pretend as yourself
- Publicly available databases/directories
  - There exists a directory/database for name, public key pairs
  - Write controlled by a trusted administrator
  - If administered thoroughly, this method is good. However, a proper administration is difficult. Secure mechanisms for registration, update and delete is required.
- Centralized distribution
  - Users interact with public key authority (PKA) to obtain any desired public key securely requires real-time access to directory when keys are needed
  - Users should know public key of PKA
- Certificates

Public Key Infrastructure (PKI)

- PKI is an infrastructure that keeps track of public keys
- A framework consisting of
  - policies defining the rules under which the cryptographic systems operate, and
  - procedures for generating and publishing keys and certificates.
- All PKIs consists of
  - certification
    - binding a public key to an entity
  - validation
    - guarantees that certificates are valid

Certificates

- A certificate contains information signed by its publisher, who is commonly referred as the Certification Authority (CA).
- There are different types of certificates
  1. Identity certificates contains entity's identity information such as e-mail address, and a list of public keys for the entity.
  2. Credential certificates contain information describing access rights.
- Data in certificates is signed by the CA
  - If Alice knows the public key of the CA, she can extract with assurance Bob's identity and his public keys from his certificate issued by the CA.

Trust in PKI

- Alice might not trust Bob,
- She might trust the CA, publisher of Bob's certificate
- PKI consists of many CAs.
- A CA can certify another CA if the former is more trusted.
- Different levels of trust
  - Alice and Bob may have different CAs
  - Alice's CA may only trust Bob's CA to certify Bob and but not certify others.
- Trust relationships become very elaborate.
  - It may be difficult to determine how much Alice can trust a certificate she receives.

X.509 Certificates

- X.509 is an international standard by ITU-T
  - designed to provide authentication services on large computer networks.
  - Initially issued in 1988
  - X.509 specifies public-key certificate format.
  - X.509 certificates are used in Visa and Mastercard's SET standard, in S/MIME, IP Security, and SSL/TLS.

Version
Serial no.
Signature Algorithm Identifier
Issuer name
Validity period
Subject name
Subject's public key info
Issuer unique identifier
Subject unique identifier
Extension
Signature

- X.509 certificates contain fields describing trust policies.
- It is possible to designate that a public key is suitable for secure e-mail, but not suitable for e-commerce applications.

- Notation
  - `CA<<A>>` : the certificate of user `A` issued by certification authority `CA`.
  - $\mathtt{CA{<}{<}A{>}{>} = CA\{V,SN,AI,CA,T_A,A,Ap\}}$

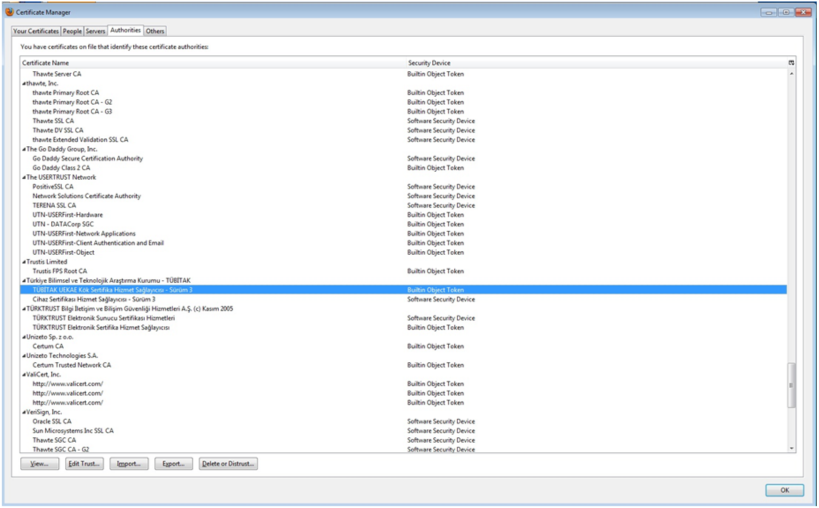
Sample Certificate

```
Certificate:
    Data:
        Version:  1 (0x0)
        Serial Number:  7829 (0x1e95)
        Signature Algorithm:  md5WithRSAEncryption
        Issuer:  C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/Email=server-certs@thawte.com
        Validity
            Not Before:  Jul 9 16:04:02 1998 GMT
            Not After :  Jul 9 16:04:02 1999 GMT
        Subject:  C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                 OU=FreeSoft, CN=www.freesoft.org/Email=baccala@freesoft.org
        Subject Public Key Info:
            Public Key Algorithm:  rsaEncryption
            RSA Public Key:  (1024 bit)
                Modulus (1024 bit):
                    00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                ...
                Exponent:
                    65537 (0x10001)
    Signature Algorithm:  md5WithRSAEncryption
        93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
        ...
```

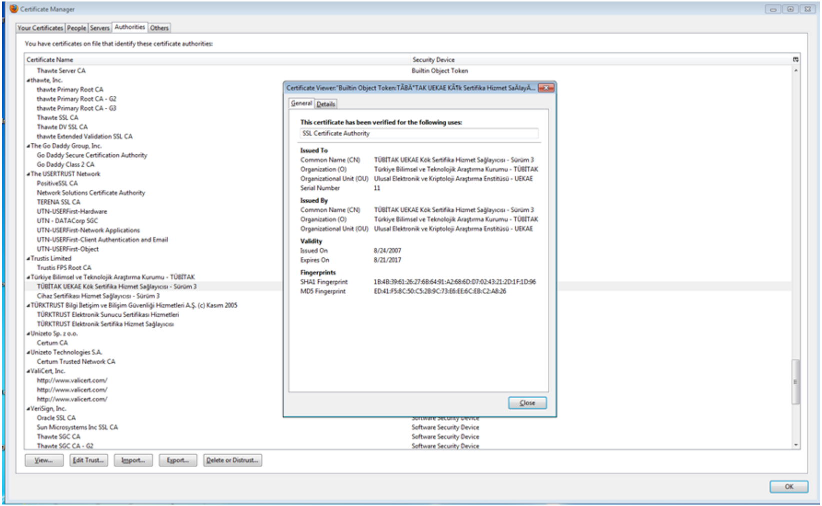
Self-Signed Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/Email=server-certs@thawte.com
        Validity
            Not Before: Aug 1 00:00:00 1996 GMT
            Not After : Dec 31 23:59:59 2020 GMT
        Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                 OU=Certification Services Division,
                 CN=Thawte Server CA/Email=server-certs@thawte.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                    ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
        ...
```

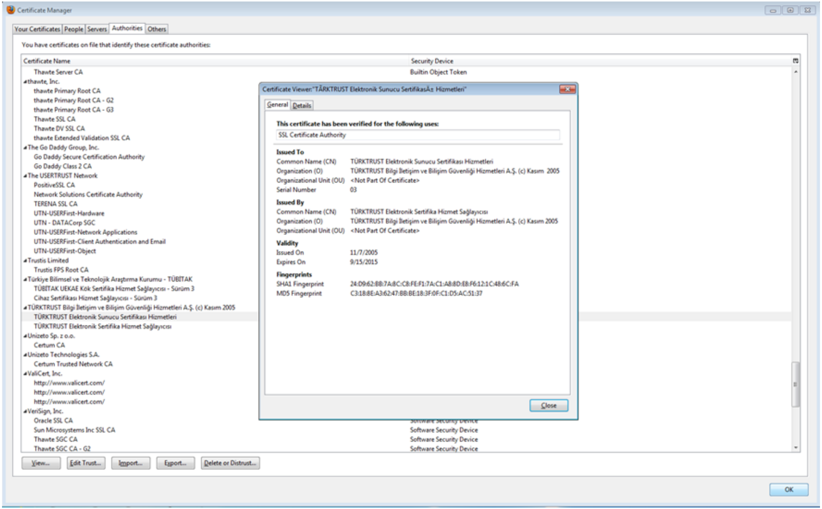
Certificates from My Computer



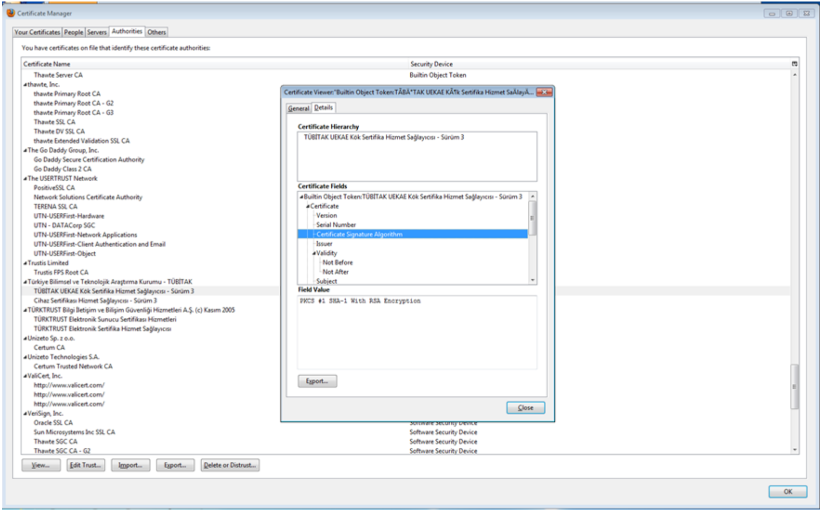
Certificates from My Computer



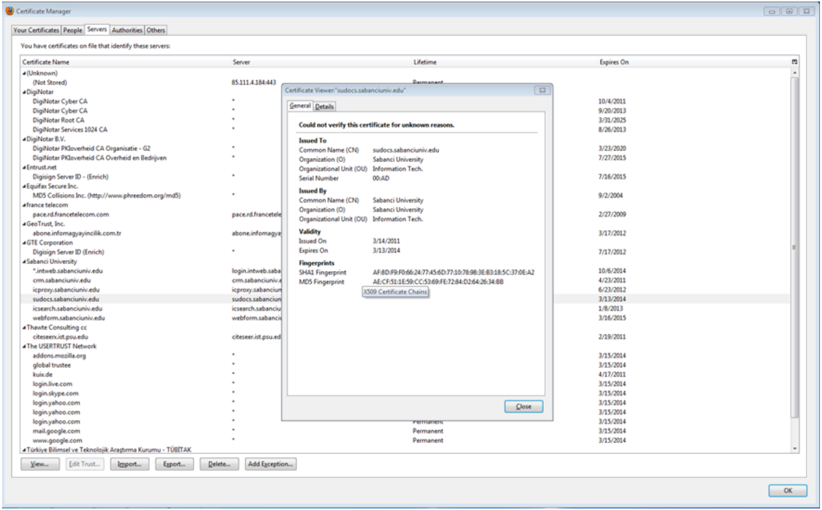
Certificates from My Computer



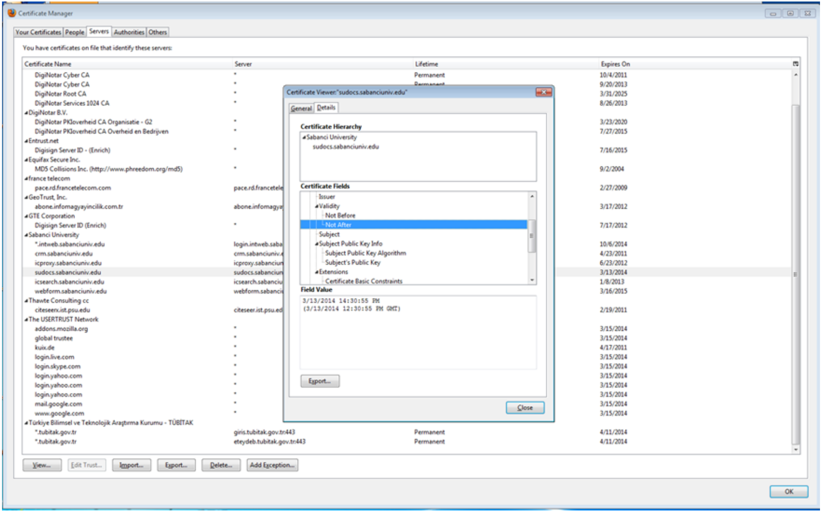
Certificates from My Computer



Certificates for SuDocs



Certificates for SuDocs



Chain of Certificates

Alice — $k_{pk,\texttt{CA1}}$, CA1<<Alice>>

Bob — $k_{pk,\texttt{CA2}}$, CA2<<Bob>>

Alice $\longleftarrow$ CA2<<Bob>> $\longleftarrow$ Bob

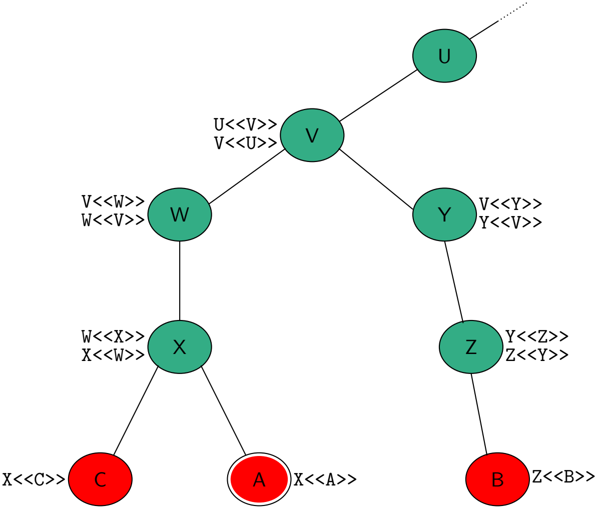
Alice — $k_{pk,\texttt{CA1}}$

CA2

Alice $\longrightarrow$ RQST(CA1<<CA2>>) $\longrightarrow$ CA2

Alice $\longleftarrow$ CA1<<CA2>> $\longleftarrow$ CA2

X509 Certificate Chains



X509 Certificate Chains

- A can acquire the following certificates from the directory to establish a certification path to B:
  - X<<W>>, W<<V>>, V<<Y>>, Y<<Z>>, Z<<B>>
- B does the same thing:
  - Z<<Y>>, Y<<V>>, V<<W>>, W<<X>>, X<<A>>

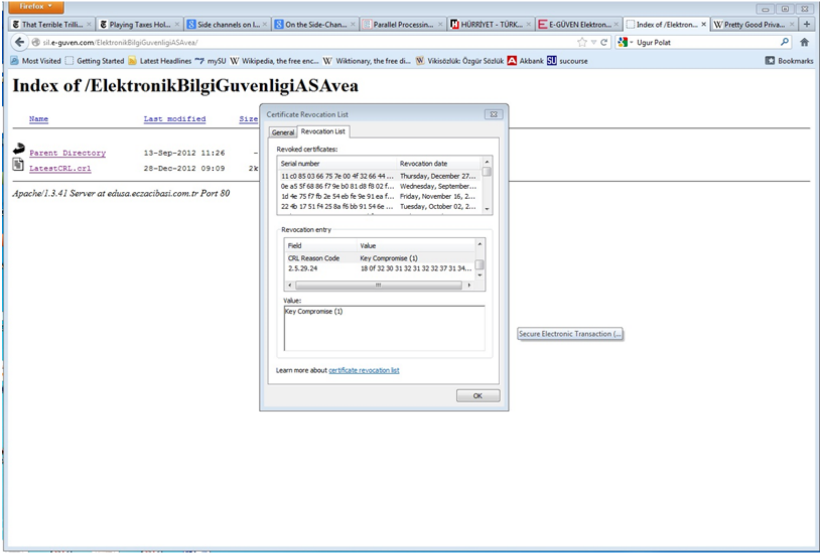
Revocation of Certificates

- Like credit cards, certificates expire.
- On occasion, it may be desirable to revoke (invalidate) a certificate before it expires.
- The reasons are
  1. The user's private key is (suspected to be) compromised.
  2. The user is no longer certified by this CA.
  3. The CA's certificate is (suspected to be) compromised.

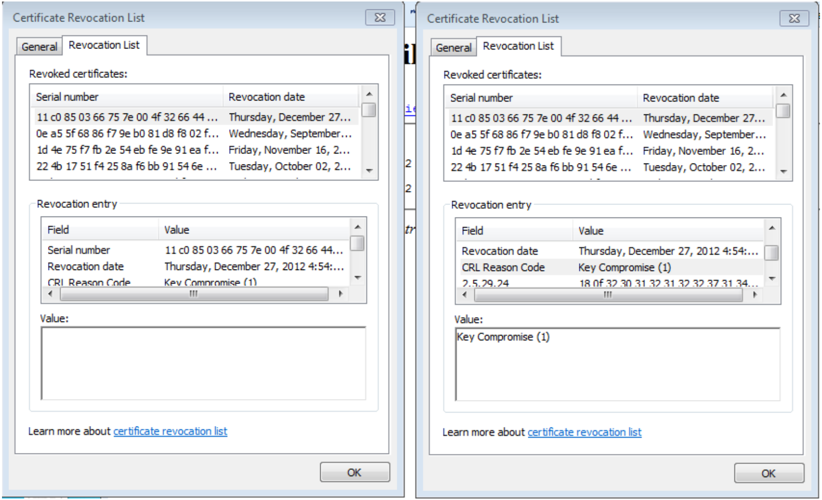
Revocation of Certificates

- Along with the certificates, certificate revocation list (CRL) is posted to the directory.
- CRL is signed by the issuer
- When a user receives a certificate, it also checks the CRL to see if the certificate is still valid.
- Example:
  - https://www.e-guven.com/
    - SIL (Sertifika Iptal Listeleri)
    - https://www.e-guven.com/bilgi-bankasi/sertifika-iptal-listeleri/
  - http://crl.verisign.com/Class3SoftwarePublishers.crl

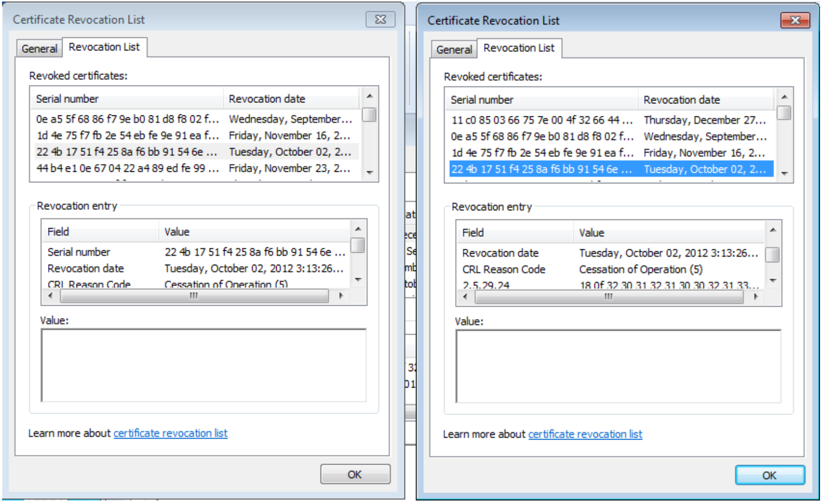
CRL


Index of /ElektronikBilgiGuvenligiASAvea
Name
Last modified
Size
Parent Directory
13-Sep-2012 11:26
LatestCRL.crl
28-Dec-2012 09:09
Apache/1.3.41 Server at edusa.eczacibasi.com.tr Port 80
Certificate Revocation List
General
Revocation List
Revoked certificates:
Serial number
Revocation date
Revocation entry
Field
Value
CRL Reason Code
Key Compromise (1)
Value:
Key Compromise (1)
Learn more about certificate revocation list
OK

CRL


Certificate Revocation List
General
Revocation List
Revoked certificates:
Serial number
Revocation date
11 c0 85 03 66 75 7e 00 4f 32 66 44 ...
Thursday, December 27...
0e a5 5f 68 86 f7 9e b0 81 d8 f8 02 f...
Wednesday, September...
1d 4e 75 f7 fb 2e 54 eb fe 9e 91 ea f...
Friday, November 16, 2...
22 4b 17 51 f4 25 8a f6 bb 91 54 6e ...
Tuesday, October 02, 2...
Revocation entry
Field
Value
Serial number
11 c0 85 03 66 75 7e 00 4f 32 66 44...
Revocation date
Thursday, December 27, 2012 4:54:...
CRL Reason Code
Key Compromise (1)
Value:
Learn more about certificate revocation list
OK
Certificate Revocation List
General
Revocation List
Revoked certificates:
Serial number
Revocation date
11 c0 85 03 66 75 7e 00 4f 32 66 44 ...
Thursday, December 27...
0e a5 5f 68 86 f7 9e b0 81 d8 f8 02 f...
Wednesday, September...
1d 4e 75 f7 fb 2e 54 eb fe 9e 91 ea f...
Friday, November 16, 2...
22 4b 17 51 f4 25 8a f6 bb 91 54 6e ...
Tuesday, October 02, 2...
Revocation entry
Field
Value
Revocation date
Thursday, December 27, 2012 4:54:...
CRL Reason Code
Key Compromise (1)
2.5.29.24
18 0f 32 30 31 32 31 32 32 37 31 34...
Value:
Key Compromise (1)
Learn more about certificate revocation list
OK

CRL



Certificate Revocation List
General
Revocation List
Revoked certificates:
Serial number
Revocation date
0e a5 5f 68 86 f7 9e b0 81 d8 f8 02 f...
Wednesday, September...
1d 4e 75 f7 fb 2e 54 eb fe 9e 91 ea f...
Friday, November 16, 2...
22 4b 17 51 f4 25 8a f6 bb 91 54 6e ...
Tuesday, October 02, 2...
44 b4 e1 0e 67 04 22 a4 89 ed fe 99 ...
Friday, November 23, 2...
Revocation entry
Field
Value
Serial number
22 4b 17 51 f4 25 8a f6 bb 91 54 6e ...
Revocation date
Tuesday, October 02, 2012 3:13:26...
CRL Reason Code
Cessation of Operation (5)
Value:
Learn more about certificate revocation list
OK
Certificate Revocation List
General
Revocation List
Revoked certificates:
Serial number
Revocation date
11 c0 85 03 66 75 7e 00 4f 32 66 44 ...
Thursday, December 27...
0e a5 5f 68 86 f7 9e b0 81 d8 f8 02 f...
Wednesday, September...
1d 4e 75 f7 fb 2e 54 eb fe 9e 91 ea f...
Friday, November 16, 2...
22 4b 17 51 f4 25 8a f6 bb 91 54 6e ...
Tuesday, October 02, 2...
Revocation entry
Field
Value
Revocation date
Tuesday, October 02, 2012 3:13:26...
CRL Reason Code
Cessation of Operation (5)
2.5.29.24
18 0f 32 30 31 32 31 30 30 32 31 33...
Value:
Learn more about certificate revocation list
OK

OCSP

- The Online Certificate Status Protocol (OCSP)
  - an Internet protocol used for obtaining the revocation status of an X.509 digital certificate.
  - created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs in a public key infrastructure (PKI)
- How It Works
  - Alice and Bob have public key certificates issued by Ivan, the (CA).
  - Alice wishes to perform a transaction with Bob and sends him her public key certificate.

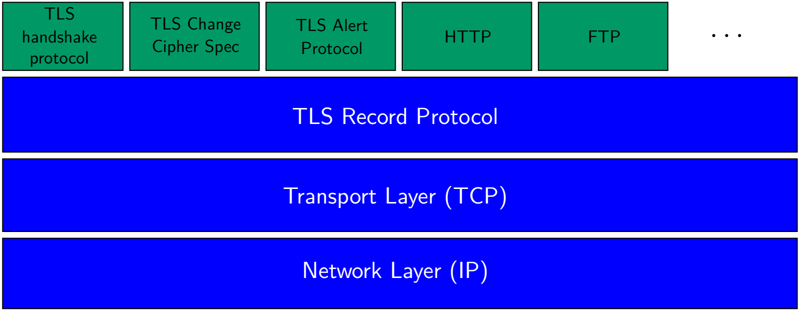
How OSCP Works

- Bob, creates an 'OCSP request' that contains Alice's certificate serial number and sends it to Ivan.
- Ivan's OCSP responder reads the certificate serial number from Bob's request. The OCSP responder uses the serial number to look up the revocation status of the certificate in a CA database that Ivan maintains.
- Ivan's OCSP responder confirms that Alice's certificate is still OK, and returns a signed, successful 'OCSP response' to Bob.
- Bob cryptographically verifies Ivan's signed response. Bob has stored Ivan's public key sometime before this transaction. Bob uses Ivan's public key to verify Ivan's response.
- Bob completes the transaction with Alice.

Secure Socket Layers 1/2

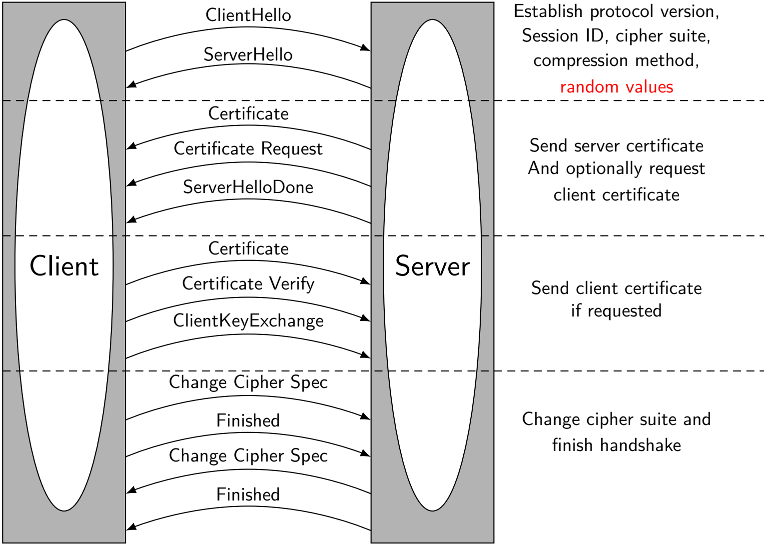
- Originally developed by Netscape in 1996
- SSL or TLS
- Negotiable encryption and authentication algorithms
- Most commonly used implementation: `www.openssl.org`
- Unencrypted communication for initial exchanges
- public-key cryptography to establish secret keys
- switch to secret-key cryptography
  - hybrid encryption scheme
- secure channel is established between a client and a server

Secure Socket Layers 2/2

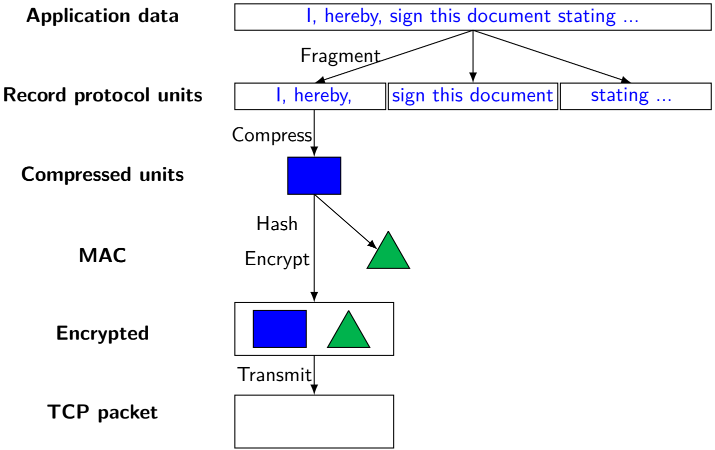


- TLS record protocol implements a secure channel
  - encrypting and authenticating messages in any connection-oriented protocol
- Other related protocols
  - establish and maintains a TLS session

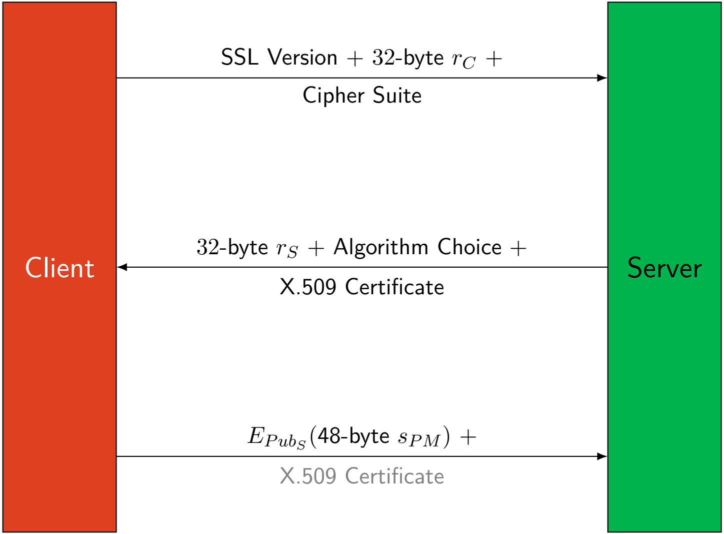
TLS Handshake Protocol



TLS Record Protocol



SSL 3.0 (TLS 1.0) (Key Exchange with RSA)



SSL 3.0 (TLS 1.0)

- Master secret key
  - $\texttt{MD5}(s_{PM}||\texttt{SHA1}(C||s_{PM}||r_C||r_S))||$
    $\texttt{MD5}(s_{PM}||\texttt{SHA1}(pad_0||s_{PM}||r_C||r_S))||$
    $\texttt{MD5}(s_{PM}||\texttt{SHA1}(pad_1||s_{PM}||r_C||r_S))$
  - 48 bytes
- Master secret key $\rightarrow$ key block
  - Six secrets from the key block
  - 2 keys for confidentiality (for each direction)
  - 2 keys for authentication (for each direction)
  - 2 initial values for CBC mode of the block cipher

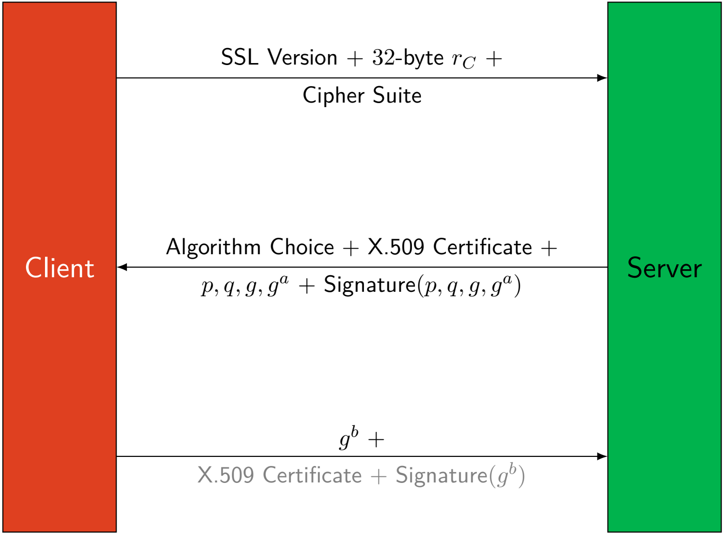
TLS 1.2

- $s = \mathrm{PRF}(s_{PM}, \text{“}\texttt{master secret}\text{''}, r_C + r_S)$ returns 48 B
- PRF stands for pseudo-random function and uses SHA256
  - It takes as input a secret, a seed, and an identifying label and produces an output of arbitrary length.
- $\mathrm{P\_SHA256}(s_{PM}, \text{“}\texttt{master secret}\text{''} + r_C + r_S)$
  - $\mathrm{P\_SHA256}$ is a keyed message authentication code (MAC).

$$\begin{aligned}\mathrm{P\_SHA256}(\texttt{secret}, \texttt{seed}) &= \mathrm{HMAC\_SHA256}(\texttt{secret}, \texttt{A(1)+seed}) + \\ &= \mathrm{HMAC\_SHA256}(\texttt{secret}, \texttt{A(2)+seed}) + \\ &= \mathrm{HMAC\_SHA256}(\texttt{secret}, \texttt{A(3)+seed}) + \ldots\end{aligned}$$

where `A(0) = seed` and $\texttt{A}(i) = \mathrm{HMAC\_SHA256}(\texttt{secret, A(i-1)})$.

TLS with Forward Secrecy (Ephemeral DH)



Key Exchange Techniques until TLS 1.2

- RSA
  - The server sends its RSA public key with X.509 certificate
  - The client generates a Pre-master Secret and encrypt using server's public key.
- Fixed DH
  - The server sends its DH public key with X.509 certificate
  - The client generates a DH key pair.
  - Pre-master secret is computed using the output of DH.
- Anonymous DH
  - No authentication mechanism is used for DH public keys.
  - Self-signed certificates are used.
- Ephemeral DH
  - The server generates a unique DH key pair for each session.
  - Ephemeral DH public key of the server is signed using the signature key of the server
  - Explained in the previous page.

Key Exchange in TLS 1.3

- The only allowed key exchange technique is Ephemeral DH.
  - Key exchange algorithm is removed from cipher suite.
- 0-RTT
  - Optional method to improve the performance of Handshake
  - Ephemeral Keys and X.509 certificate are send in Client Hello and Server Hello message. (There is no need for Phase 2 & 3)

Session vs Connection

- TLS session is association between client and server
  - Session created with Handshake protocol
  - Multiple connections can be associated with one session
- Session vs. Connection
  - Sessions are to avoid expensive negotiation for crypto parameters for each connection
- State information is stored after Handshake protocol (using Session ID)
  - Session: ID, certificate, compression, cipher spec, master secret, is-resumable
  - Connection: random values, encrypt keys, MAC secrets, IV, sequence numbers
- To initiate
  - A session: All phases of handshake are done.
  - A connection: Only phase 1 and 4 are done.

Key Exchange in TLS 1.3

- The only allowed key exchange technique is Ephemeral DH.
  - Forward Secrecy is a default feature.
- 1-RTT Handshake
  - Key exchange algorithm is removed from cipher suite.
  - Ephemeral Keys and X.509 certificate are send in Client Hello and Server Hello messages.
  - There is no need for Phase 2 & 3