



**Cryptography for EveryOne.
Learn from
Crypto Principle to Applied Cryptography
With Practical Example**

The Modern Cryptography CookBook

By Anish Nath

The Modern Cryptography CookBook

Cryptography is for EveryOne. Learn from Crypto Principle to Applied Cryptography With Practical Example

Anish

This book is for sale at <http://leanpub.com/crypto>

This version was published on 2018-09-02



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.



This work is licensed under a [Creative Commons Attribution 4.0 International License](#)

I would like to dedicate this book to my 8-year old son Arjun Nath , Grandfather Sri Rajeshwar Prasad; wife Manisha Prasad; mother Indu Sinha; and all my family members (my father Anil Kumar Sinha; chote papa Sunil Kumar Sinha; choti mummy: Poonam Sinha; and friends). Without them, this would not have been possible.

I would also thanks to Cisco where I work, I got most of the learning from there, and not least the opensource community.

Contents

Cryptography	1
Symmetric key Cryptography	4
PUBLIC KEY INFRASTRUCTURES	14
TLSv1.3	28

Cryptography

Cryptography goal is to achieve **Confidentiality, Integrity, Identification & Authentication, Non Repudiation**

- **Data integrity** services address the unauthorized or accidental modification of data. The goal is for the receiver of the data to verify that the data has not been altered.
- **Confidentiality** services restrict access to the content of sensitive data to only those individuals who are authorized to view the data. Confidentiality measures prevent the unauthorized disclosure of information to unauthorized individuals or processes.
- **Identification and authentication** services establish the validity of a transmission, message, and its originator. The goal is for the receiver of the data to determine its origin.
- **Non-repudiation** services prevent an individual from denying that previous actions had been performed. The goal is to ensure that the recipient of the data is assured of the sender's identity.

These four cryptography services is achieved through

1. Symmetric key cryptography
2. Secure Hash
3. Asymmetric (Public-key) cryptography
4. Digital Signatures

These **terms** are often used when we discuss cryptography:

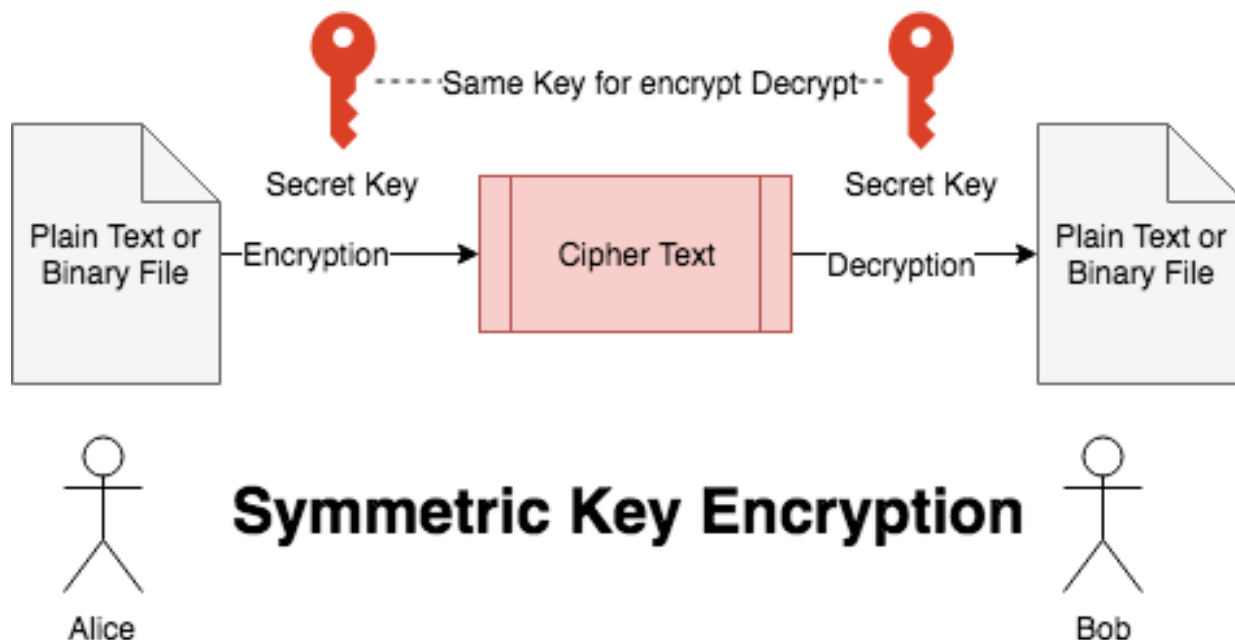
- **Encryption:** The process of converting data from plaintext to ciphertext. Also referred to as enciphering.
- **Decryption:** The process of converting data from ciphertext to plaintext. Also referred to as deciphering.
- **Key:** A parameter that controls the transformation of plaintext into ciphertext or vice versa. Determining the original plaintext data without the key is impossible. Keys can be both public and private. Also referred to as a cryptovariable.
- **Symmetric:** An encryption method whereby a single private key both encrypts and decrypts the data. Also referred to as private or secret key encryption.
- **Asymmetric:** An encryption method whereby a key pair, one private key and one public key, performs encryption and decryption. One key performs the encryption, whereas the other key performs the decryption. Also referred to as public key encryption.

- **Digital signature:** A method of providing sender authentication and message integrity. The message acts as an input to a hash function, and the sender's private key encrypts the hash value. The receiver can perform a hash computation on the received message to determine the validity of the message.
- **Hash:** A one-way function that reduces a message to a hash value. A comparison of the sender's hash value to the receiver's hash value determines message integrity. If the resultant hash values are different, then the message has been altered in some way, provided that both the sender and receiver used the same hash function.
- **Digital certificate:** An electronic document that identifies the certificate holder.
- **Plaintext:** A message in its original format. Also referred to as cleartext.
- **Ciphertext:** An altered form of a message that is unreadable without knowing the key and the encryption system used. Also referred to as a cryptogram.
- **Cryptosystem** The entire cryptographic process, including the algorithm, key, and key management functions. The security of a cryptosystem is measured by the size of the keyspace and available computational power.
- **Cryptanalysis:** The science of decrypting ciphertext without prior knowledge of the key or cryptosystem used. The purpose of cryptanalysis is to forge coded signals or messages that will be accepted as authentic signals or messages.
- **Key clustering:** Occurs when different encryption keys generate the same ciphertext from the same plaintext message.
- **Keyspace:** All the possible key values when using a particular algorithm or other security measure. A 40-bit key would have 240 possible values, whereas a 128-bit key would have 2128 possible values.
- **Collision:** An event that occurs when a hash function produces the same hash value on different messages.
- **Algorithm:** A mathematical function that encrypts and decrypts data. Also referred to as a cipher.
- **Cryptology** The science that studies encrypted communication and data.
- **Encoding:** The process of changing data into another form using code.
- **Decoding:** The process of changing an encoded message back into its original format.
- **Transposition:** The process of shuffling or reordering the plaintext to hide the original message. Also referred to as permutation. For example, AEEGMSS is a transposed version of MESSAGE.
- **Substitution:** The process of exchanging one byte in a message for another. For example, ABCCDEB is a substituted version of MESSAGE.
- **Confusion:** The process of changing a key value during each round of encryption. Confusion is often carried out by substitution. Confusion conceals a statistical connection between the plaintext and ciphertext. Claude Shannon first discussed confusion.
- **Diffusion:** The process of changing the location of the plaintext within the ciphertext. Diffusion is often carried out using transposition. Claude Shannon first introduced diffusion.
- **Avalanche effect:** The condition where any change in the key or plaintext, no matter how minor, will significantly change the ciphertext. Horst Feistel first introduced avalanche effect.

- **Work factor or work function:** The amount of time and resources that would be needed to break the encryption.
- **Trapdoor:** A secret mechanism that allows the implementation of the reverse function in a one-way function.
- **One-way function:** A mathematical function that can be more easily performed in one direction than in the other.

Symmetric key Cryptography

Symmetric key cryptography is a class of algorithms where Alice and Bob share a secret key. These algorithms are primarily used to achieve **confidentiality**.



symmetric Cryptography

Symmetric algorithms are ideally suited for **confidentiality**. To use a symmetric algorithm, Alice transforms a **plaintext** message to **ciphertext** using a symmetric algorithm (AES/Des/3DES..) and a **key**.

Alice transmits the **ciphertext** to **Bob**. Bob uses the **same key** to transform the **ciphertext** back into the **plaintext**.

Feature of Symmetric key

Symmetric algorithms	Integrity	Confidentiality	Identification & Authentication	Non Repudiation	Key Distribution
Encryption	No	Yes	No	No	No

Symmetric algorithms	Integrity	Confidentiality	Identification & Authentication	Non Repudiation	Key Distribution
Message Authentication Code	Yes	No	Yes	No	No
key Transport	No	No	No	No	Yes with TTP (Trusted Third Party)

In this chapter we will be discussing below topics related to Symmetric key Cryptography.

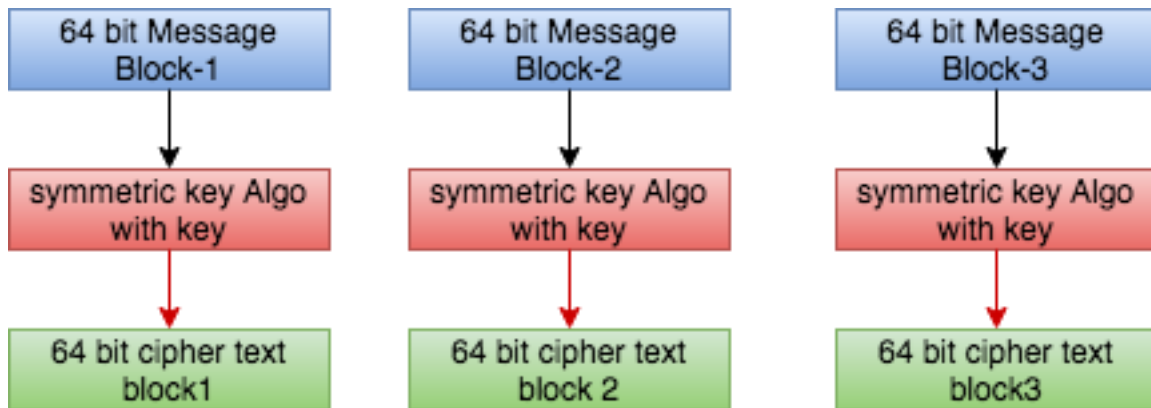
1. Modes of Operation
2. Authenticated Encrypted Modes
3. Initial Vector requirement
4. Some Examples Block Cipher Algorithms
 1. Data Encryption Standard
 2. Triple Data Encryption Standard (TDEA)
 3. SkipJack
 4. Advanced Encryption Standard
 5. RC4
 6. BlowFish
5. Hash based Symmetric key Algorithms

Block ciphers Modes of Operation

Block ciphers modes provides **encryption** but **not message integrity**

ECB:

The **Electronic Codebook (ECB)**, The message is divided into blocks, and each block is encrypted separately, this is the simplest encryption and decryption mode.



ECB Mode

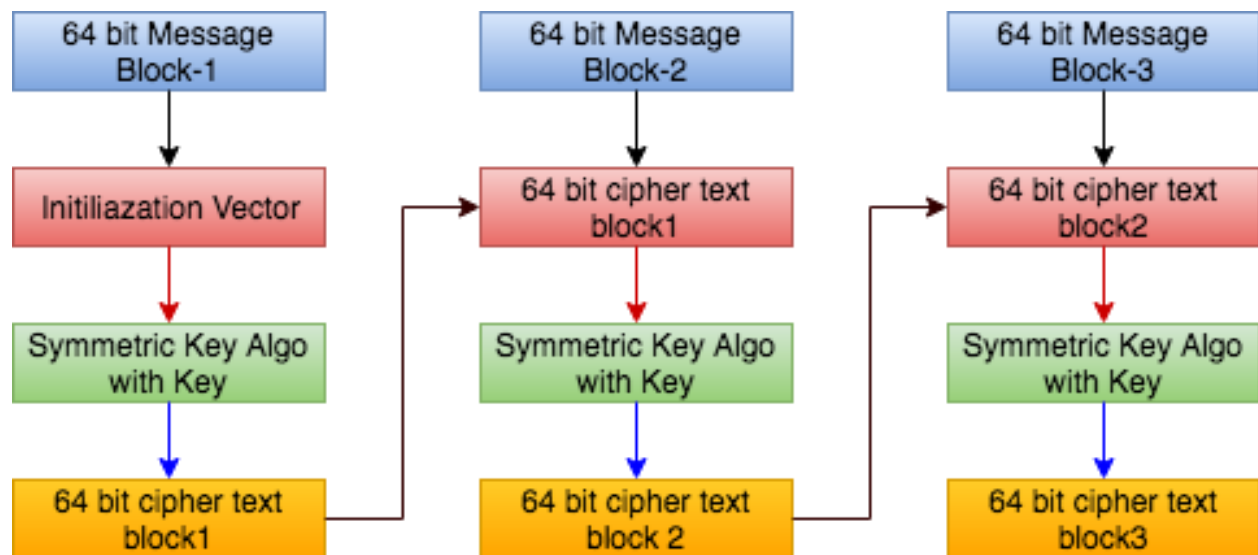
ecb

Example Ciphers

AES-128-ECB, AES-192-ECB, AES-256-ECB, BF-ECB, CAMELLIA-128-ECB, CAMELLIA-192-ECB, CAMELLIA-256-ECB, CAST5-ECB, DES-ECB, IDEA-ECB, RC2-ECB, RC5-ECB, SEED-ECB, AES-128-ECB, AES-192-ECB, AES-256-ECB, BF-ECB, CAMELLIA-128-ECB, CAMELLIA-192-ECB, CAMELLIA-256-ECB, CAST5-ECB, DES-ECB, IDEA-ECB, RC2-ECB, RC5-ECB, SEED-ECB

CBC

Cipher Block Chaining (CBC) mode is an IV-based encryption scheme, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.



CBC Mode

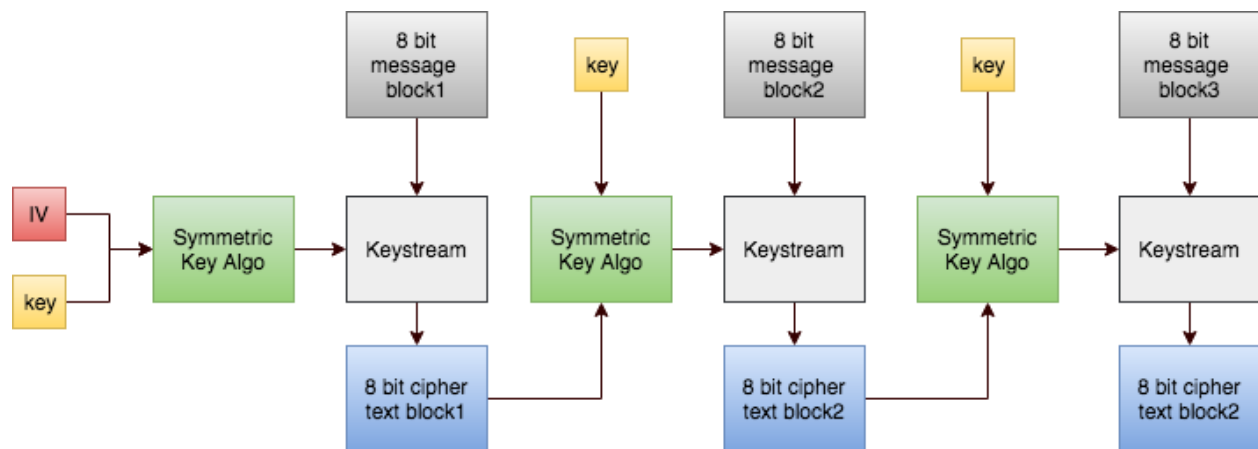
ecb

Example Ciphers

AES-128-CBC, AES-128-CBC-HMAC-SHA1, AES-128-CBC-HMAC-SHA256, AES-192-CBC, AES-256-CBC, AES-256-CBC-HMAC-SHA1, AES-256-CBC-HMAC-SHA256, AES128 => AES-128-CBC, AES192 => AES-192-CBC, AES256 => AES-256-CBC, BF => BF-CBC, BF-CBC, CAMELLIA-128-CBC, CAMELLIA-192-CBC, CAMELLIA-256-CBC, CAMELLIA128 => CAMELLIA-128-CBC, CAMELLIA192 => CAMELLIA-192-CBC, CAMELLIA256 => CAMELLIA-256-CBC, CAST => CAST5-CBC, CAST-cbc => CAST5-CBC, CAST5-CBC, DES => DES-CBC, DES-CBC, DES-EDE-CBC, DES-EDE3-CBC, DES3 => DES-EDE3-CBC, DESX => DESX-CBC, DESX-CBC, IDEA => IDEA-CBC, IDEA-CBC, RC2 => RC2-CBC, RC2-40-CBC, RC2-64-CBC, RC2-CBC, RC5 => RC5-CBC, RC5-CBC, SEED => SEED-CBC, SEED-CBC, AES-128-CBC, AES-128-CBC-HMAC-SHA1, AES-128-CBC-HMAC-SHA256, AES-192-CBC

CFB

Cipher Feedback (CFB) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input block



CFB Mode

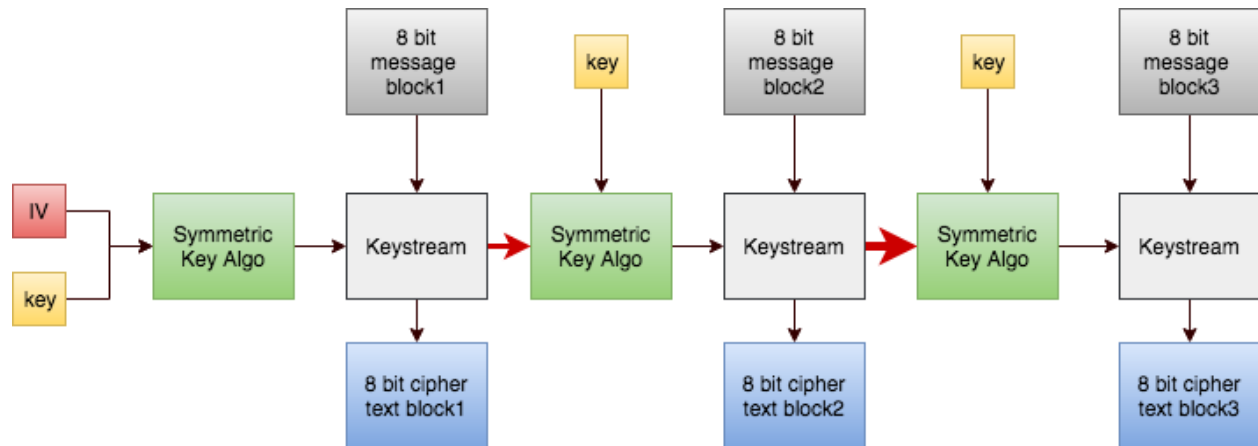
cfb

Example Ciphers

AES-128-CFB, AES-128-CFB1, AES-128-CFB8, AES-192-CFB, AES-192-CFB1, AES-192-CFB8, AES-256-CFB, AES-256-CFB1, AES-256-CFB8, BF-CFB, CAMELLIA-128-CFB, CAMELLIA-128-CFB1, CAMELLIA-128-CFB8, CAMELLIA-192-CFB, CAMELLIA-192-CFB1, CAMELLIA-192-CFB8, CAMELLIA-256-CFB, CAMELLIA-256-CFB1, CAMELLIA-256-CFB8, CAST5-CFB, DES-CFB, DES-CFB1, DES-CFB8, DES-EDE-CFB, DES-EDE3-CFB, DES-EDE3-CFB1, DES-EDE3-CFB8, IDEA-CFB, RC2-CFB, RC5-CFB, SEED-CFB

OFB

The **Output Feedback (OFB)** mode makes a block cipher into a synchronous stream cipher. It generates keystream blocks, which are then XORed with the plaintext blocks to get the ciphertext



OFB Mode

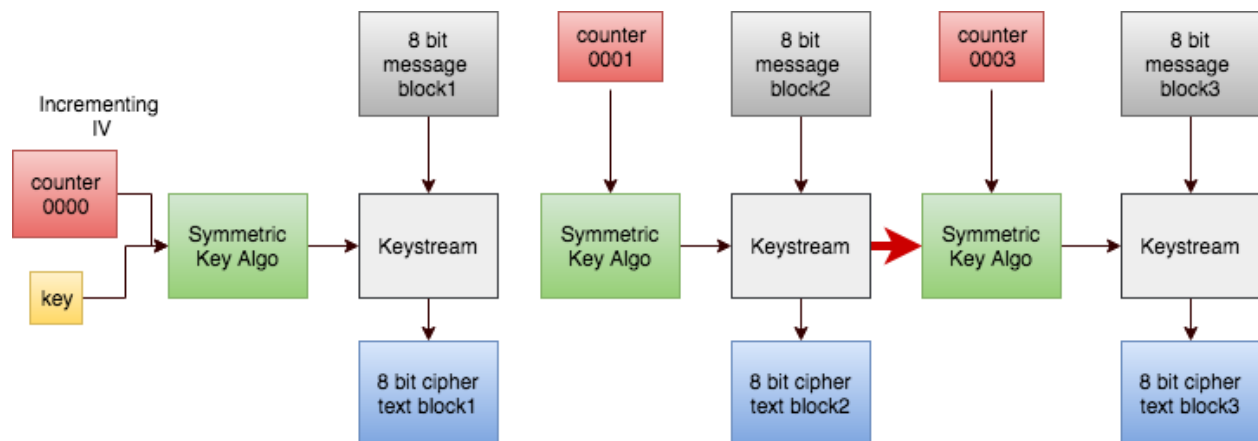
ofb

Example Ciphers

AES-128-OFB, AES-192-OFB, AES-256-OFB, BF-OFB, CAMELLIA-128-OFB, CAMELLIA-192-OFB, CAMELLIA-256-OFB, CAST5-OFB, DES-EDE-OFB, DES-EDE3-OFB, DES-OFB, IDEA-OFB, RC2-OFB, RC5-OFB, SEED-OFB, AES-128-OFB, AES-192-OFB, AES-256-OFB, BF-OFB, CAMELLIA-128-OFB, CAMELLIA-192-OFB, CAMELLIA-256-OFB, CAST5-OFB, DES-EDE-OFB, DES-EDE3-OFB, DES-OFB, IDEA-OFB, RC2-OFB, RC5-OFB, SEED-OFB

CTR

The CTR mode has similar characteristics to OFB, but also allows a random access property during decryption. CTR mode is well suited to operate on a multi-processor machine where blocks can be encrypted in parallel. Furthermore, it does not suffer from the short-cycle problem that can affect OFB.



CTR Mode

ctr

Example Ciphers

AES-128-CTR, AES-192-CTR, AES-256-CTR, AES-128-CTR, AES-192-CTR, AES-256-CTR

XTS

An IV-based encryption scheme, the mode works by applying a tweakable blockcipher (secure as a strong-PRP) to each n -bit chunk. For messages with lengths not divisible by n , the last two blocks are treated specially. The only allowed use of the mode is for encrypting data on a block-structured storage device. The narrow width of the underlying PRP and the poor treatment of fractional final blocks are problems. More efficient but less desirable than a (wide-block) PRP-secure blockcipher would be.

Example Ciphers

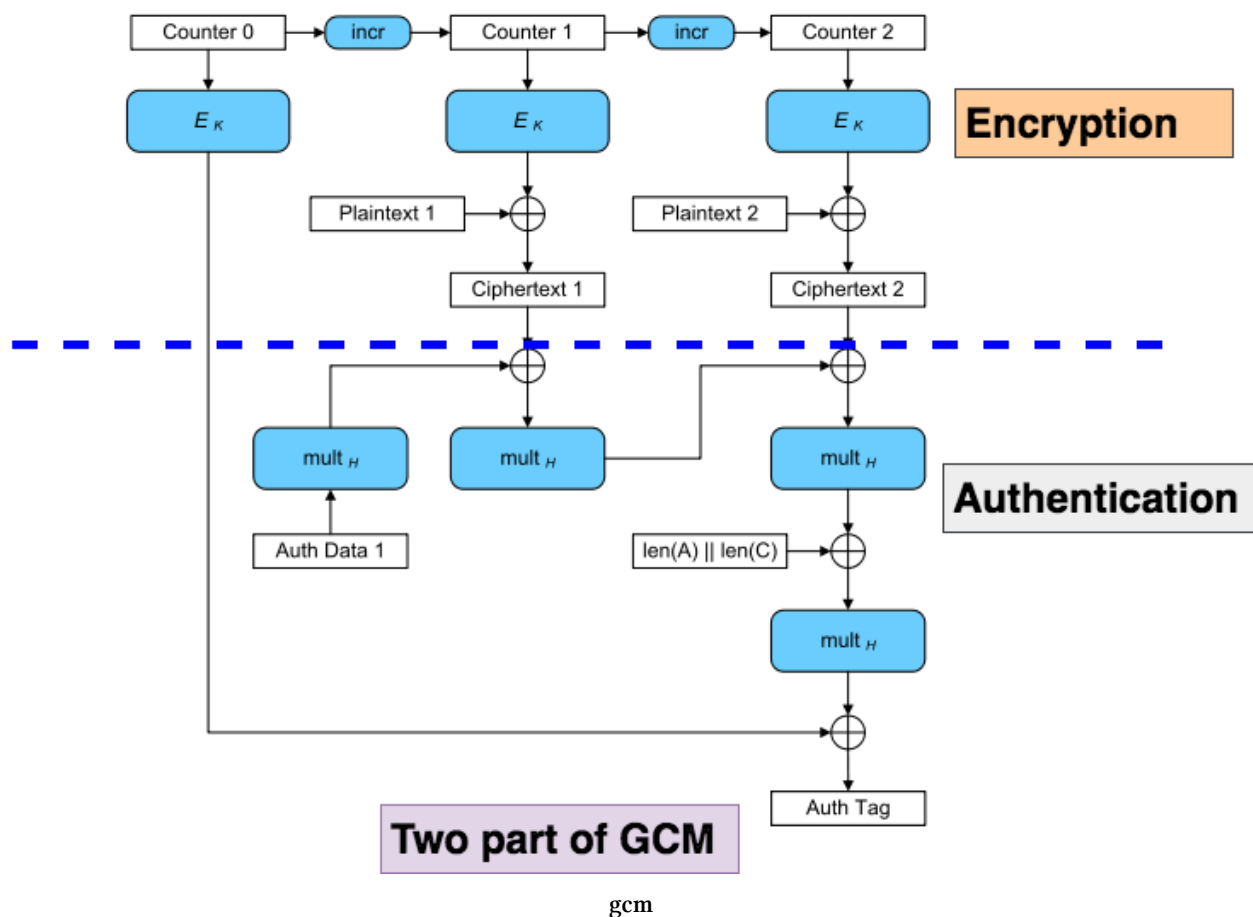
AES-128-XTS, AES-256-XTS, AES-128-XTS, AES-256-XTS

Authenticated encryption Modes

Authenticated encryption provides both **encryption** and **message integrity**

GCM

Galois/Counter Mode (GCM) A nonce-based AEAD scheme that combines CTR mode encryption and a GF(2128)-based universal hash function. Good efficiency characteristics for some implementation environments. Good provably-secure results assuming minimal tag truncation. Attacks and poor provable-security bounds in the presence of substantial tag truncation. Can be used as a nonce-based MAC, which is then called GMAC. Questionable choice to allow nonces other than 96-bits. Recommend restricting nonces to 96-bits and tags to at least 96 bits. Widely standardized and used.



Example Ciphers

id-aes128-GCM, id-aes192-GCM, id-aes256-GCM, id-aes128-GCM, id-aes192-GCM, id-aes256-GCM

CCM

CCM mode (Counter with CBC-MAC), A nonce-based AEAD scheme that combines CTR mode encryption and the raw CBC-MAC. Inherently serial, limiting speed in some contexts. Provably secure, with good bounds, assuming the underlying blockcipher is a good PRP. Ungainly construction that demonstrably does the job. Simpler to implement than GCM. Can be used as a nonce-based MAC. Widely standardized and used.

Example Ciphers

id-aes128-CCM,id-aes192-CCM,id-aes256-CCM,id-aes128-CCM,id-aes192-CCM,id-aes256-CCM

Initialization Vectors

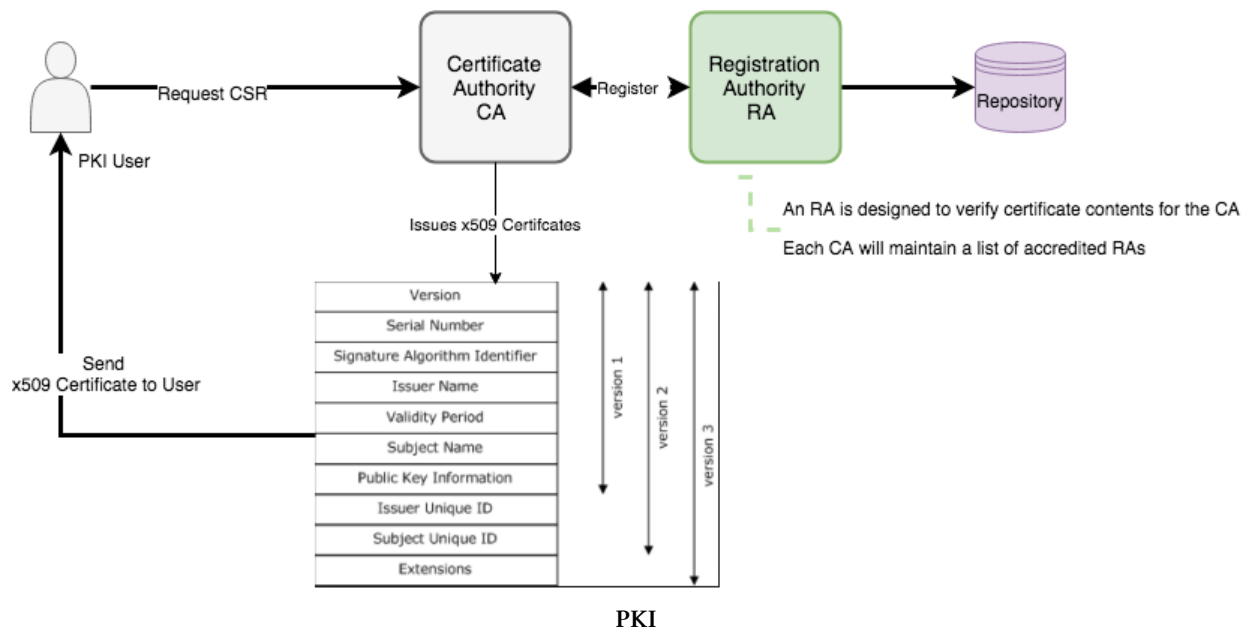
Generation of Initial Vectors - The CBC, CFB, and OFB modes require an initialization vector as input, in addition to the plaintext. An IV must be generated for each execution of the encryption operation, and the same IV is necessary for the corresponding execution of the decryption operation

- For the CBC and CFB modes, the IVs must be **unpredictable**

PUBLIC KEY INFRASTRUCTURES

A public key infrastructure (PKI) binds public keys to entities, enables other entities to verify public key bindings, and provides the services needed for ongoing management of keys in a distributed system

PKI COMPONENTS



Certification Authority (CA)

The CA issues a public key certificate for each identity, confirming that the identity has the appropriate credentials.

Main Functions of CA

- Verify the CSR request
- Issue the Certificate (Create and Sign them)
- Attach CRL for Certificate revocation
- Publish it's current (Expired Certificate) and CRL's

Registration Authority (RA)

An RA is designed to verify certificate contents for the CA. Each CA will maintain a list of accredited RAs; that is a list of RAs determined to be trustworthy

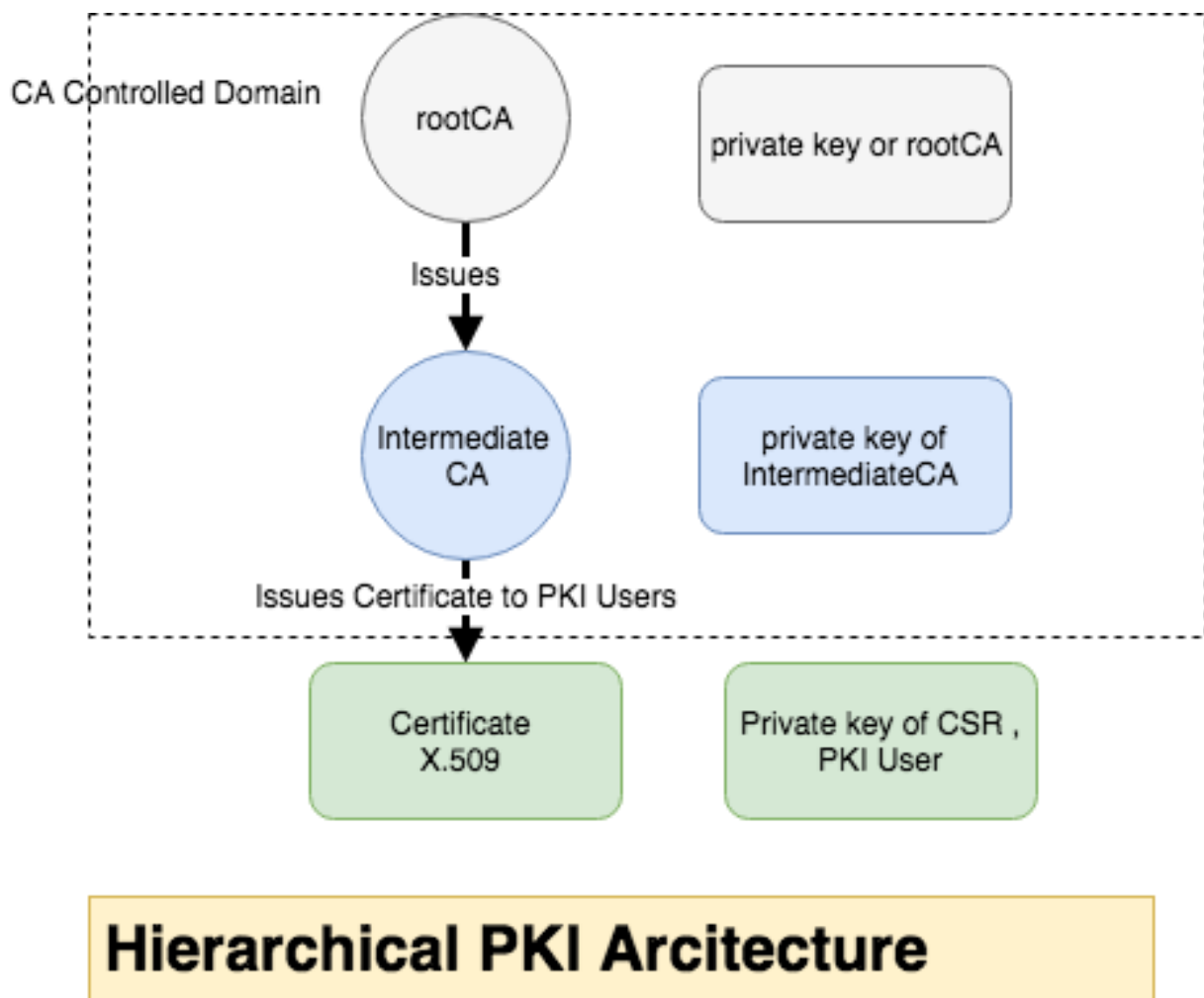
PKI Users

PKI Users are organizations or individuals that use the PKI, but do not issue certificates.

Main Functions of PKI user - Generate the Certificate Signing Request. - Maintain the Certificate obtain by the CA.

PKI Architecture

CAs may be linked in a number of ways. Most enterprises that deploy a PKI will choose either a **mesh** or a **hierarchical** architecture, This is an example of **hierarchical** Structure



PKI

Generating hierarchical CA structure

- **rootCA** will generate **self signed certificate** and **key** with longer validity

```
openssl genrsa -des3 -out rootCA.key 4096
openssl req -new -x509 -days 3650 -key rootCA.key -out rootCA.crt
```

- **intCA** will generate **CSR** and get it signed with **rootCA** and set Validity for Longer Year.

```
openssl genrsa -des3 -out intCA.key 4096
openssl req -new -key intCA.key -out intCA.csr
```

intCA Submitted the CSR information to rootCA to get it signed

```

Enter pass phrase for intCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:KA
Locality Name (eg, city) []:IN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IntermediateCA Issuers
Organizational Unit Name (eg, section) []:intermediateCA Issue of myOrg
Common Name (e.g. server FQDN or YOUR name) []:intCA
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

```

- Sign **intermediateCA** with rootCA private key, control the serial Number to **specific** pattern (Usability)

```

openssl x509 -req -days 365 -in intCA.csr -CA rootCA.crt -CAkey rootCA.key -set_serial 1111111 -out intCA.crt
Signature ok
subject=/C=AU/ST=KA/L=IN/O=IntermediateCA Issuers/OU=intermediateCA Issue of myOrg\
/CN=intCA
Getting CA Private Key
Enter pass phrase for rootCA.key:

```

- **PKI (End)Users** will create the CSR and send to CA to get it signed

```

openssl genrsa -des3 -out client.key 2048
openssl req -new -key client.key -out client.csr

```

End user submit the CSR information to get it signed by the CA, the CA used intermediate CA to sign the CSR

Enter pass phrase for client.key:
 You are about to be asked to enter information that will be incorporated
 into your certificate request.
 What you are about to enter is what is called a Distinguished Name or a DN.
 There are quite a few fields but you can leave some blank
 For some fields there will be a default value,
 If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:
 State or Province Name (full name) [Some-State]:My Org
 Locality Name (eg, city) []:IN
 Organization Name (eg, company) [Internet Widgits Pty Ltd]:8gwifi
 Organizational Unit Name (eg, section) []:Crypto
 Common Name (e.g. server FQDN or YOUR name) []:8gwifi.org
 Email Address []:

Please enter the following 'extra' attributes
 to be sent with your certificate request
 A challenge password []:
 An optional company name []:

- Sign with Intermediate CA, set the expiry date to 1 or 2 year Max, and generate a serial number for this

```
openssl x509 -req -in client.csr -days 530 -CA intCA.crt -CAkey intCA.key -CAcreat\
eserial -out client.crt
```

The CSR getting signed

Signature ok
 subject=/C=AU/ST=My Org/L=IN/O=8gwifi/OU=Crypto/CN=8gwifi.org
 Getting CA Private Key
 Enter pass phrase for intCA.key:

- View the Client Certificate Information , The issuer is IntermediateCA

```
openssl x509 -text -in client.crt
.....
Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=AU, ST=KA, L=IN, O=IntermediateCA Issuers, OU=intermediateCA Issue o\
f myOrg, CN=intCA
```

- **Verify Certificate**

Verify Client certificate with **Full CA chain**

```
openssl verify -verbose -CAfile <(cat intCA.crt rootCA.crt) client.crt
client.crt: OK
```

Verify **intermediateCA** belongs to **rootCA** chain

```
openssl verify -verbose -CAfile rootCA.crt intCA.crt
intCA.crt: OK
```

PKI Data Structure

Two Basic data structures used in PKI - **X.509** Public Key Certificates - **CRL** Certificate Revocation List - **Attribute** Certificates

X.509

There are **ten** common fields, **six** mandatory and **four** optional.

The mandatory fields are: **serial number**, **signature algorithm identifier**, **certificate issuer name**, **certificate validity period**, **public key Info** , and the **subject name**

X.509 Attributes	Description
Version	v2 or v3
Serial Number	CA Assigned Serial Number to the Certificate
Signature	Indicate which digital Signature Algorithms ex: SHA-256 with RSA Encryption
Issuer	Contains x.500 DN
Validity	Certificate Expiry Dates
Subject	Contains the DN of the Holder of private Key Corresponding to the public key in the certificate

X.509 Attributes	Description
Subject public key Information	Optional Parameter, Algorithm Identifiers
Issuer UniqueID and Subject UniqueID	ID of the Issues only in v2 and v3
Extensions	Optional only in v3 Certificate version
subjectType	Indicate whether a subject is CA or ENtity
Names and identity information	c=US; o=8gwifi; ou=Crypto; cn=8gwifi.org
Key Attributes	Specifies relevant attributes whether it used for key transport, or be used to verify a digital signature
Policy Information	Policies related to Certificate
Certificate Extensions	extension identifier, a criticality flag, and extension value

Reader Note *what is self Signed Certificate*

In self signed certificate the issuer and subject are same :)

Issuer: C=AU, ST=KA, L=IN, O=rootCA Issuers, OU=rootCA, CN=rootCA

Subject: C=AU, ST=KA, L=IN, O=rootCA Issuers, OU=rootCA, CN=rootCA

CRL

The CRL contains the following fields:

CRL Fields	Description
Version	Optional Version, Default 2
Signature	algorithm identifier for the digital signature algorithm used by the CRL issuer to sign the CRL
Issuer	X.500 DN of the CRL issuer
This update	Issue date of the CRL
Next update	Next CRL Issue date
Revoked Certificates	Structured List of Revoked Certificates
CRL Extensions	Additional Information About the CRL

Pem format of CRL

```
-----BEGIN X509 CRL-----
-----END X509 CRL-----
```

Implement CERTIFICATION REVOCATION LIST

- Make a directory for a CRL:


```
mkdir -p /etc/pki/crl
```

- Create an **index file, the CRL Database** with the following command:

```
touch /etc/pki/crl/index.txt
```

- Create a file for the CRL number. This file should contain the text 00 only.

```
echo 00 > /etc/pki/crl/crl_number
```

- **crl_openssl.conf**: create and write the following contents into a crl_openssl.conf file.

```
cat <<EOF > crl_openssl.conf
# OpenSSL configuration for CRL generation
#
#####
[ ca ]
default_ca          = CA_default          # The default ca section

#####

[ CA_default ]
database = /etc/pki/crl/index.txt
crlnumber = /etc/pki/crl_number

default_days        = 365                  # how long to certify for
default_crl_days= 30                      # how long before next CRL
default_md          = default             # use public key default MD
preserve            = no                  # keep passed DN ordering

#####

[ crl_ext ]
# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.
# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always,issuer:always
EOF
```

- **Generate CRL file**

```
openssl ca -gencrl -keyfile intCA.key -cert intCA.crt -out intCA.crl -config crl_\
openssl.conf
```

- **View CRL file**

Added with required CRL field, at this time no certificate is Invoked

```
openssl crl -text -noout -in intCA.crl
```

Certificate Revocation List (CRL):

Version 2 (0x1)

Signature Algorithm: sha256WithRSAEncryption

Issuer: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=intermediateCA

Last Update: Aug 8 06:54:55 2018 GMT

Next Update: Sep 7 06:54:55 2018 GMT

CRL extensions:

X509v3 CRL Number:

0

No Revoked Certificates.

Signature Algorithm: sha256WithRSAEncryption

if the CRL is DER Encoded then issue the below command

```
openssl crl -inform DER -text -noout -in mycrl.crl
```

- **Revoke the Certificate**

```
openssl ca -revoke client.crt -keyfile intCA.key -cert intCA.crt -config crl_openssl.conf
```

Using configuration from crl_openssl.conf

Enter pass phrase for intCA.key:

Adding Entry with serial number AC12C39820C69327 to DB for /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=8gwifi.org

Revoking Certificate AC12C39820C69327.

Data Base Updated

- **Again revoking the same client certificate will through an error **Already Revoked****

```
openssl ca -revoke client.crt -keyfile intCA.key -cert intCA.crt -config crl_openssl.conf
```

Using configuration from crl_openssl.conf

Enter pass phrase for intCA.key:

ERROR:Already revoked, serial number AC12C39820C69327

- **Regenerate the CRL list**

```
openssl ca -gencrl -keyfile intCA.key -cert intCA.crt -out intCA.crl -config crl_o\
penssl.conf
Using configuration from crl_openssl.conf
Enter pass phrase for intCA.key:
```

- **View CRL file**

One revoked certificate is Added in the CRL entry

```
openssl crl -text -noout -in intCA.crl
Certificate Revocation List (CRL):
    Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=intermediateCA
    Last Update: Aug  8 07:08:08 2018 GMT
    Next Update: Sep  7 07:08:08 2018 GMT
    CRL extensions:
        X509v3 CRL Number:
            2
Revoked Certificates:
    Serial Number: AC12C39820C69327
    Revocation Date: Aug  8 07:01:07 2018 GMT
```

Adding CRL distribution point

- Edit the file `crl_openssl.conf` and point out the **PEM** and **DER**

```
crlDistributionPoints=@crl_section
[crl_section]
URI.1 = https://8gwifi.org/intCA.crl
URI.2 = https://8gwifi.org/intCA.der
```

- CA role , once CA sign the file it will use `-extfile crl_openssl.conf` to locate the **crlDistributionPoints**

```
openssl x509 -req -in client.csr -days 530 -CA intCA.crt -CAkey intCA.key -CAcreat\
eserial -out client.crt -extfile crl_openssl.conf
```

- Once certificate is Signed, the x.509 certificate will have **X509v3** extensions that contains CRL Distribution Points

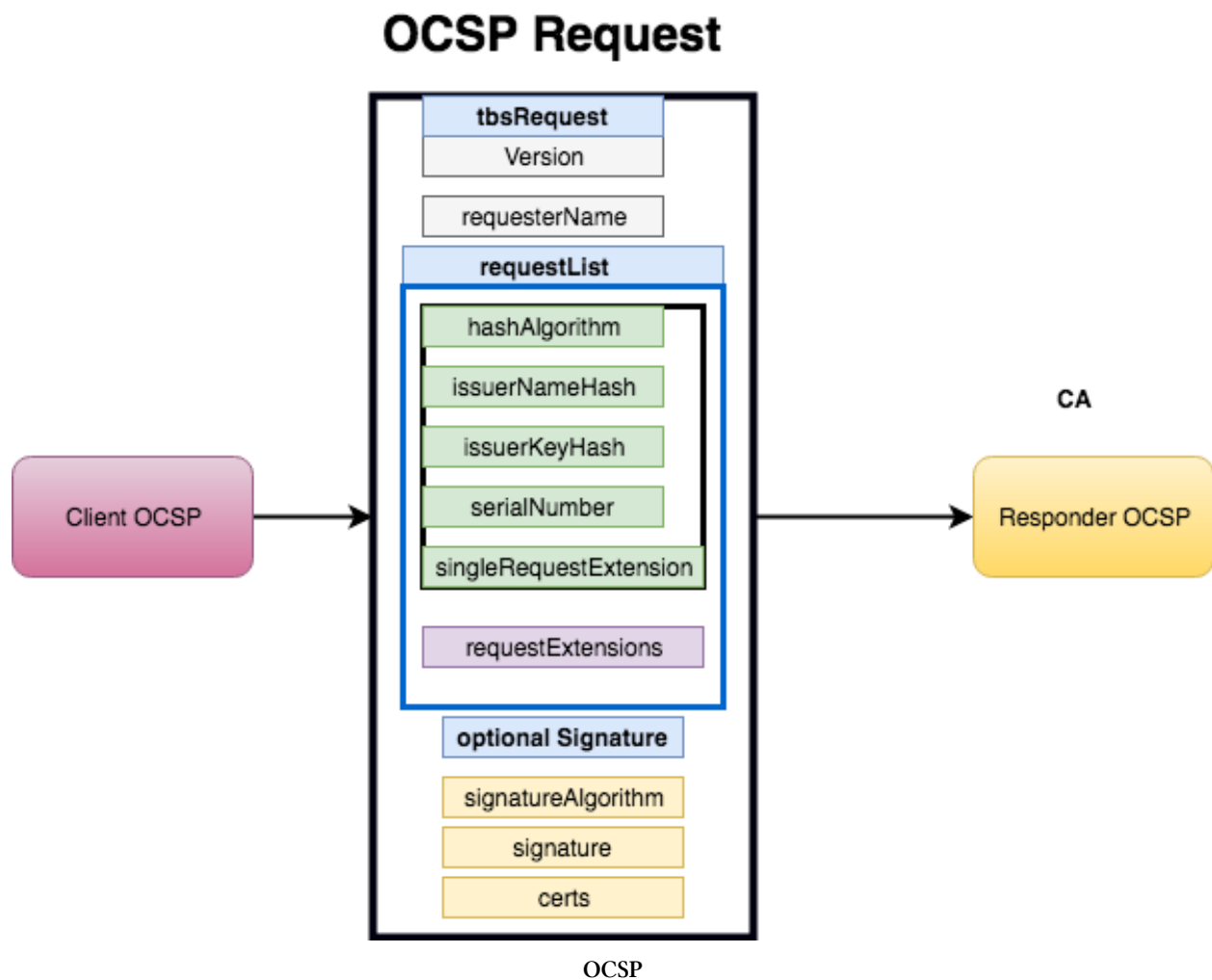
```

openssl x509 -text -in client.crt
.....
X509v3 extensions:
    X509v3 CRL Distribution Points:
        Full Name:
            URI:https://8gwifi.org/intCA.crl
        Full Name:
            URI:https://8gwifi.org/intCA.der

```

OCSP

OCSP stands for the Online Certificate Status Protocol and is one way to validate a certificate status. It is an alternative to the CRL, certificate revocation list.



The OCSP process is very simple:

1. Client receives the certificate
2. Client sends OCSP request to the OCSP server and it query by the serial number of the certificate
3. OCSP response with a certificate status Good, Revoked or Unknown

Working Demo

- Get the certificate you want to verify 8gwifi.org¹

```
openssl s_client -servername 8gwifi.org -connect 8gwifi.org:443 2>&1 < /dev/null | \
sed -n '/-----BEGIN/,/-----END/p' > 8gwifi.pem
```

- Build the certificate chain

```
openssl s_client -servername 8gwifi.org -connect 8gwifi.com:443 -showcerts 2>&1 < \
/dev/null > cacert.pem
```

edit the file cacert.pem and add necessary chain certificate

- Determine the ocsp URI

```
openssl x509 -noout -ocsp_uri -in 8gwifi.pem
http://ocsp.int-x3.letsencrypt.org
```

- Invoke the openssl ocsp client

```
openssl ocsp -no_nonce -issuer cacert.pem -cert 8gwifi.pem -VAfile cacert.pem -text \
-url http://ocsp.int-x3.letsencrypt.org/ -header Host ocsp.int-x3.letsencrypt.org \
-respout ocsp-test
```

The OCSP Response

¹<https://8gwifi.org>

OCSP Request Data:

Version: 1 (0x0)

Requestor List:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: 7EE66AE7729AB3FCF8A220646C16A12D6071085D

Issuer Key Hash: A84A6A63047DDDBAE6D139B7A64565EFF3A8ECA1

Serial Number: 03FF3497BFA5D45C36C511809F9FD5F28C20

OCSP Response Data:

OCSP Response Status: successful (0x0)

Response Type: Basic OCSP Response

Version: 1 (0x0)

Responder Id: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3

Produced At: Aug 6 08:59:00 2018 GMT

Responses:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: 7EE66AE7729AB3FCF8A220646C16A12D6071085D

Issuer Key Hash: A84A6A63047DDDBAE6D139B7A64565EFF3A8ECA1

Serial Number: 03FF3497BFA5D45C36C511809F9FD5F28C20

Cert Status: good

This Update: Aug 6 08:00:00 2018 GMT

Next Update: Aug 13 08:00:00 2018 GMT

Signature Algorithm: sha256WithRSAEncryption

39:3d:96:78:44:9f:03:29:bc:83:35:32:1a:d7:6d:05:f9:59:
63:9e:52:6f:06:8e:9f:74:d1:f9:aa:18:2b:e2:13:61:5a:d1:
ad:7a:67:9e:2b:a1:12:83:92:92:f3:c0:dc:4b:2a:ee:96:85:
f0:5b:39:30:2f:17:ed:20:a3:ae:de:c1:41:e1:26:8d:70:c5:
fe:79:9c:37:7b:b7:75:93:61:f7:5f:8b:7f:6f:99:7a:5a:19:
a4:e7:4b:41:ad:e5:92:71:44:11:75:67:68:0d:0c:b6:be:ef:
70:a0:a7:c6:fa:6e:06:08:5a:7c:2e:f0:41:7a:55:a3:21:74:
89:2c:e5:f9:ab:58:5c:97:1d:89:a8:65:a3:be:f7:0a:e5:5c:
4f:a9:61:f2:04:d5:f2:18:6b:74:e7:b5:c8:12:db:9c:70:89:
e7:c6:e3:43:70:18:41:d6:4b:a9:15:94:13:4b:00:75:d2:2a:
fe:fb:e4:a8:cf:e5:aa:56:d6:e1:91:55:06:d1:33:43:d9:4b:
82:a6:bc:10:a1:42:d0:e2:49:fe:18:08:44:d4:a7:4f:b6:3f:
00:95:72:11:d7:e3:14:eb:6b:51:7a:e7:c1:40:42:2e:da:c4:
be:1a:ce:8c:48:f2:03:ed:c0:93:19:c9:26:93:1e:f0:d2:56:
bc:70:39:db

Response verify OK

8gwifi.pem: good

This Update: Aug 6 08:00:00 2018 GMT

Next Update: Aug 13 08:00:00 2018 GMT

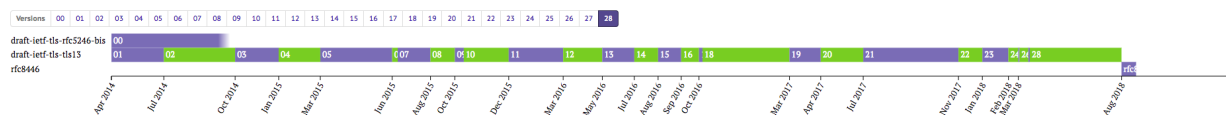
TLSv1.3

TLS stands for [Transport Layer Security](https://en.wikipedia.org/wiki/Transport_Layer_Security)² and is the successor to SSL (Secure Sockets Layer). TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery. This section mostly focus on TLSv 1.3. TLS 1.3 was defined in [RFC](https://tools.ietf.org/html/rfc8446)³ 8446⁴ in August 2018. It is based on the earlier TLS 1.2 specification.

TLS History and Development

Protocol	Published
SSL 1.0	Unpublished
SSL 2.0	1995
SSL 3.0	1996
TLS 1.0	1999
TLS 1.1	2006
TLS 1.2	2008
TLS 1.3	2018

TLS1.3 has been over eight years since the last encryption protocol update, but the final version of TLS 1.3 has now been published as of August 2018 [Image Ref](https://datatracker.ietf.org/doc/rfc8446/)⁵



²https://en.wikipedia.org/wiki/Transport_Layer_Security

³[https://en.wikipedia.org/wiki/Request_for_Comments_\(identifier\)](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier))

⁴<https://tools.ietf.org/html/rfc8446>

⁵<https://datatracker.ietf.org/doc/rfc8446/>

Major Differences from TLS 1.2

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000509	127.0.0.1	127.0.0.1	TLSv1.3	299	Client Hello
6	0.008516	127.0.0.1	127.0.0.1	TLSv1.3	2299	Server Hello, Change Cipher Spec, Application Data, Application Data, App...
8	0.009185	127.0.0.1	127.0.0.1	TLSv1.3	146	Change Cipher Spec, Application Data
9	0.009564	127.0.0.1	127.0.0.1	TLSv1.3	321	Application Data
10	0.009759	127.0.0.1	127.0.0.1	TLSv1.3	321	Application Data
12	7.048517	127.0.0.1	127.0.0.1	TLSv1.3	90	Application Data


TLSv1.3

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000508	127.0.0.1	127.0.0.1	TLSv1.2	377	Client Hello
6	0.007982	127.0.0.1	127.0.0.1	TLSv1.2	2142	Server Hello, Certificate, Server Key Exchange, Server Hello Done
8	0.008630	127.0.0.1	127.0.0.1	TLSv1.2	159	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
9	0.008886	127.0.0.1	127.0.0.1	TLSv1.2	292	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

TLSv1.2

From the Wireshark packet capture, its clearly visible the TLSv.1.3, the number of packets is being reduced this offer better speed in TLS v1.3 , and some of the major changes from TLS1.2 as follows

- The list of supported symmetric encryption algorithms has been pruned of all algorithms that are considered legacy. Those that remain are all **Authenticated Encryption with Associated Data (AEAD)** algorithms.

▼ Handshake Protocol: Client Hello 

Handshake Type: Client Hello (1)

Length: 224

Version: TLS 1.2 (0x0303)

Random: 3ab2121f850c067cc82af4c012f5ac3ddb1e7e6369c1eb91...

Session ID Length: 32

Session ID: 19785f578273db9bfa2848077cc3bb2698fc6260eec96e51...

Cipher Suites Length: 8

▼ Cipher Suites (4 suites)

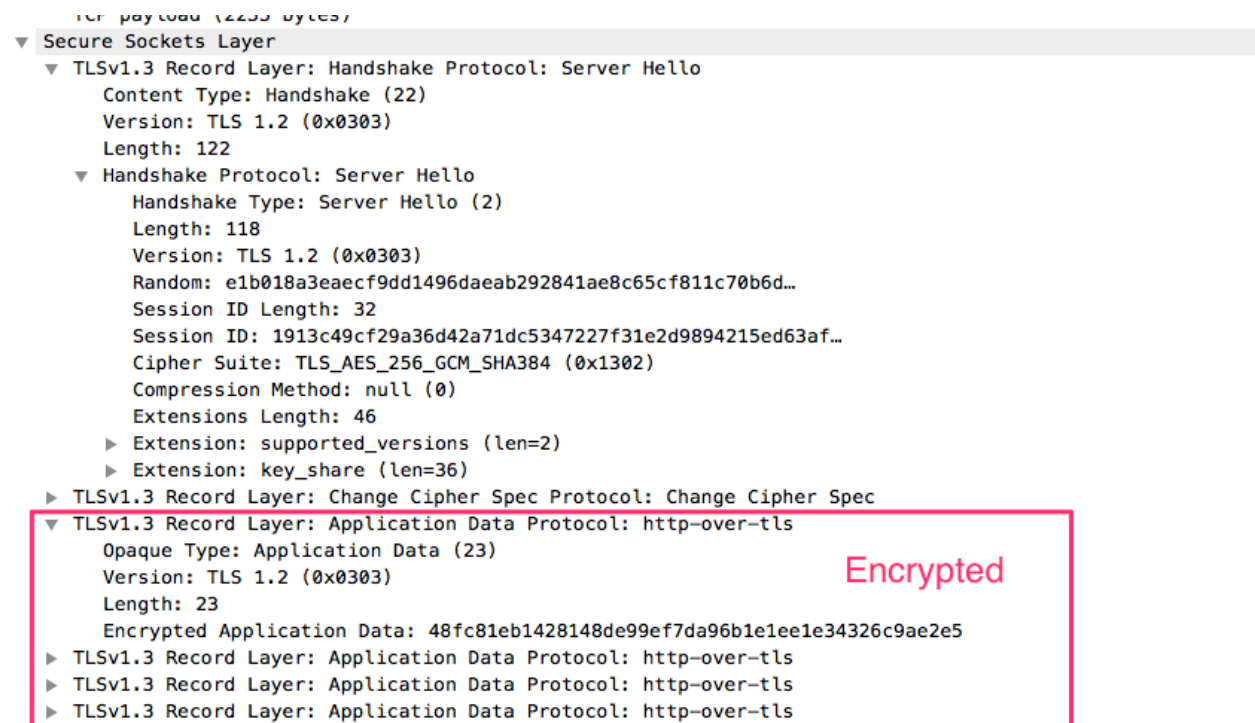
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)

Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)

Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)

Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)

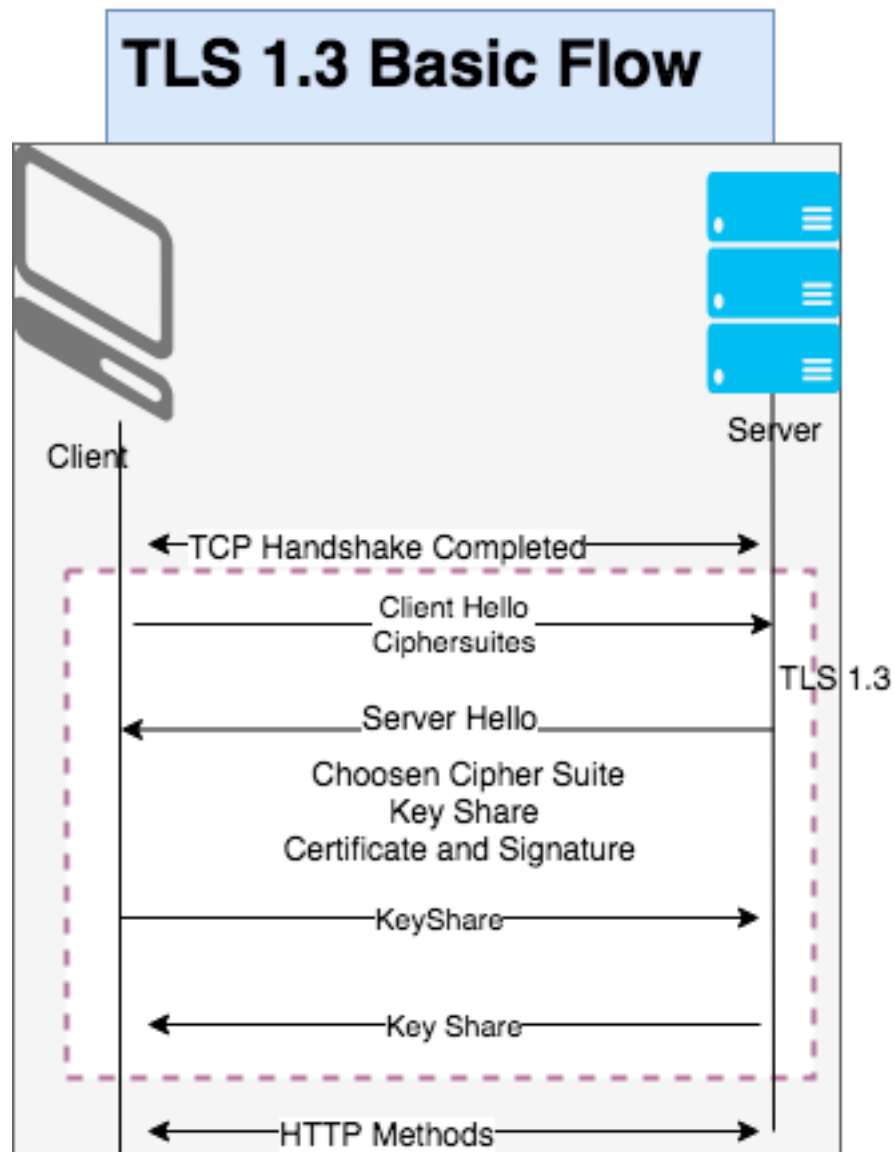
- The cipher suite concept has been changed to separate the authentication and key exchange mechanisms from the record protection algorithm (including secret key length) and a hash to be used with both the key derivation function and handshake message authentication code (MAC).
- A **zero round-trip** time (0-RTT) mode was added, saving a round trip at connection setup for some application data, at the cost of certain security properties. **IMP** 0-rtt should be avoided , there are proven replay attack has been found
- All handshake messages after the **ServerHello** are now **encrypted**. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.



- Static **RSA** and **Diffie-Hellman** cipher suites have been **removed**; all public-key based key exchange mechanisms now provide forward secrecy.
- The **key derivation** functions have been **redesigned**.
- The handshake state machine has been significantly restructured to be more consistent and to **remove** superfluous messages such as ChangeCipherSpec (except when needed for middlebox compatibility).
- Elliptic curve algorithms are now in the base spec, and new signature algorithms, such as EdDSA, are included. TLS 1.3 removed point format negotiation in favor of a single point format for each curve.
- The TLS 1.2 version **negotiation mechanism** has been **deprecated** in favor of a version list in an extension. This increases compatibility with existing servers that incorrectly implemented version negotiation.
- Session resumption with and without server-side state and the PSK-based ciphersuites of earlier versions of TLS have been replaced by a single new PSK exchange

TLS 1.3 Handshake

The handshake can be thought of as having three phases (indicated in the diagram below)



1. In the first phase, the client sends the ClientHello message, which contains
 - random nonce (ClientHello.random);
 - protocol versions;
 - symmetric cipher/HKDF hash pairs; either a set of Diffie-Hellman key shares (in the “key_ - share” extension
 - A set of pre- shared key labels (in the “pre_shared_key” extension or both;
 - And potentially additional extensions.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000493	127.0.0.1	127.0.0.1	TLSv1.3	299	Client Hello
▶ Frame 4: 299 bytes on wire (2392 bits), 299 bytes captured (2392 bits) ▶ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00) ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 ▶ Transmission Control Protocol, Src Port: 54470, Dst Port: 443, Seq: 1, Ack: 1, Len: 233 ▼ Secure Sockets Layer						
▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 228						
▼ Handshake Protocol: Client Hello Handshake Type: Client Hello (1) Length: 224 Version: TLS 1.2 (0x0303) Random: 1404b7032b72dec9c9f5d94fddb85ef0f5c7abd78c782c... Session ID Length: 32 Session ID: 30a91b02274fa33177da6ec82d43a481ee0af36386398baa... Cipher Suites Length: 8 ▼ Cipher Suites (4 suites) Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302) Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303) Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301) Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff) Compression Methods Length: 1 ▶ Compression Methods (1 method) Extensions Length: 143 ▶ Extension: server_name (len=14) ▶ Extension: ec_point_formats (len=4) ▶ Extension: supported_groups (len=12) ▶ Extension: session_ticket (len=0) ▶ Extension: encrypt_then_mac (len=0) ▶ Extension: extended_master_secret (len=0) ▶ Extension: signature_algorithms (len=30) ▶ Extension: supported_versions (len=3) ▶ Extension: psk_key_exchange_modes (len=2) ▼ Extension: key_share (len=38) Type: key_share (51) Length: 38 ▼ Key Share extension Client Key Share Length: 36 ▼ Key Share Entry: Group: x25519, Key Exchange Length: 32 Group: x25519 (29) Key Exchange Length: 32 Key Exchange: 96cd0e457fafa034bb732fd01bf206fdd79b5a899a62845f...						

8gwifi.org

The “key_share” extension contains the endpoint’s cryptographic parameters. In TLSv1.3 the client selects a “group” that it will use for key exchange.

The PSK: If clients offer “pre_shared_key” without a “psk_key_exchange_modes” extension, servers abort the handshake and used to negotiate the identity of the pre-shared key to be used with a given handshake in association with PSK key establishment

1. The server processes the ClientHello and determines the appropriate cryptographic parameters for the connection. It then responds with its own ServerHello which indicates the negotiated connection parameters. The combination of the ClientHello and the ServerHello determines the shared keys

Secure Sockets Layer

TLShv1.3 Record Layer: Handshake Protocol: Server Hello

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 122

Handshake Protocol: Server Hello

Handshake Type: Server Hello (2)

Length: 118

Version: TLS 1.2 (0x0303)

Random: 4f71e6b31534e47f5eb1f0e276873a059931ac999cb884d8...

Session ID Length: 32

Session ID: 30a91b02274fa33177da6ec82d43a481ee0af36386398baa...

Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)

Compression Method: null (0)

Extensions Length: 46

Extension: supported_versions (len=2)

Type: supported_versions (43)

Length: 2

Supported Version: TLS 1.3 (0x0304)

Extension: key_share (len=36)

Type: key_share (51)

Length: 36

Key Share extension

Key Share Entry: Group: x25519, Key Exchange length: 32

Group: x25519 (29)

Key Exchange Length: 32

Key Exchange: 570eb60a6edff40e98cbe97ea3ad32515bb6673f3cf303f7...

TLShv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.2 (0x0303)

Length: 1

Change Cipher Spec Message

TLShv1.3 Record Layer: Application Data Protocol: http-over-tls

TLShv1.3 Record Layer: Application Data Protocol: http-over-tls

TLShv1.3 Record Layer: Application Data Protocol: http-over-tls

TLShv1.3 Record Layer: Application Data Protocol: http-over-tls

Negotiated Ciphers

1. Upon receiving the server's messages, the client responds with its Authentication messages, namely Certificate and CertificateVerify (if requested), and Finished.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000493	127.0.0.1	127.0.0.1	TLShv1.3	299	Client Hello
6	0.000147	127.0.0.1	127.0.0.1	TLShv1.3	2299	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application D...
8	0.007812	127.0.0.1	127.0.0.1	TLShv1.3	146	Change Cipher Spec, Application Data
9	0.000874	127.0.0.1	127.0.0.1	TLShv1.3	305	Application Data
10	0.010379	127.0.0.1	127.0.0.1	TLShv1.3	305	Application Data
12	6.500862	127.0.0.1	127.0.0.1	TLShv1.3	90	Application Data

Secure Sockets Layer

TLShv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.2 (0x0303)

Length: 1

Change Cipher Spec Message

TLShv1.3 Record Layer: Application Data Protocol: http-over-tls

Opaque Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 69

Encrypted Application Data: 49057362f8f06950ba126a0cb831baed87514ddad1aac221...

8gwifi.org

Ciphersuites

OpenSSL has implemented support for five TLShv1.3 ciphersuites as follows:

1. TLS_AES_256_GCM_SHA384

2. TLS_CHACHA20_POLY1305_SHA256
3. TLS_AES_128_GCM_SHA256
4. TLS_AES_128_CCM_8_SHA256
5. TLS_AES_128_CCM_SHA256

```
openssl ciphers -v | grep TLSv1.3
TLS_AES_256_GCM_SHA384 TLSv1.3 Kx=any Au=any Enc=AESGCM(256) Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any Au=any Enc=CHACHA20/POLY1305(256) Mac=\
AEAD
TLS_AES_128_GCM_SHA256 **TLSv1.3** Kx=any Au=any Enc=AESGCM(128) Mac=AEAD
```

Start the TLS1.3 server in openssl

The forthcoming openssl 1.1.1-pre9 (beta) release has included support for TLSv1.3.

```
openssl version
OpenSSL 1.1.1-pre9 (beta) 21 Aug 2018
```

openssl command to start the tls1.3 server

```
openssl s_server -accept 443 -tls1_3 -ciphersuites TLS_AES_256_GCM_SHA384 -key key.\
pem -cert cert.pem
Using default temp DH parameters
ACCEPT
```

Connect to TLS1.3

The openssl command to connect to tlsv1.3.

```
openssl s_client -connect 127.0.0.1:443
-----
-----
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 4096 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
```

Verify return code: 18 (self signed certificate)

Post-Handshake New Session Ticket arrived:

SSL-Session:

Protocol : TLSv1.3

Cipher : TLS_AES_256_GCM_SHA384

Session-ID: 2BC1AB6B0BE58B527AE4CAEFEABC6D9654094BC1F4D529E5F3F0912A80C97001

Session-ID-ctx:

Resumption PSK: EA4A8E23B397F4F822B770C0922F47F7A66F6A7AA2F2DC4B94B961941AA87ACD\611AC293259EFB130887F9A2D02AC89E

PSK identity: None

PSK identity hint: None

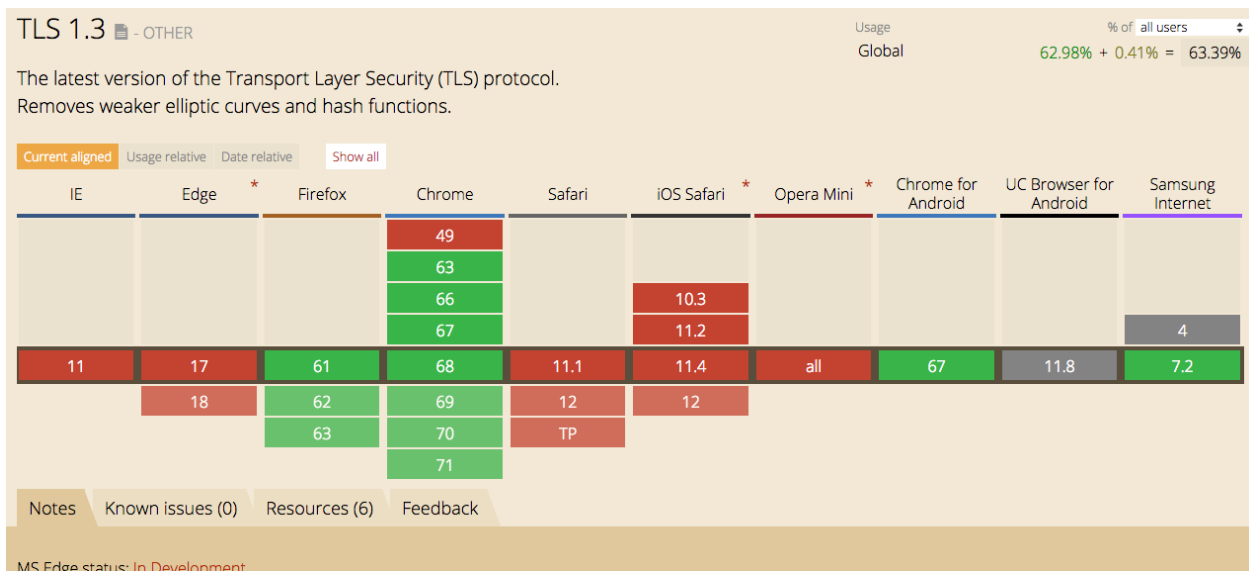
SRP username: None

TLS session ticket lifetime hint: 7200 (seconds)

TLS session ticket:

Browser Support

Checkout the browser compatibility for TLS 1.3 here : <https://caniuse.com/#feat=tls1-3>



Further Reading

RFC⁶ 8446⁷ Cloudflare TLS 1.3 and Q&A⁸

⁶[https://en.wikipedia.org/wiki/Request_for_Comments_\(identifier\)](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier))

⁷<https://tools.ietf.org/html/rfc8446>

⁸<https://blog.cloudflare.com/tls-1-3-overview-and-q-and-a/>