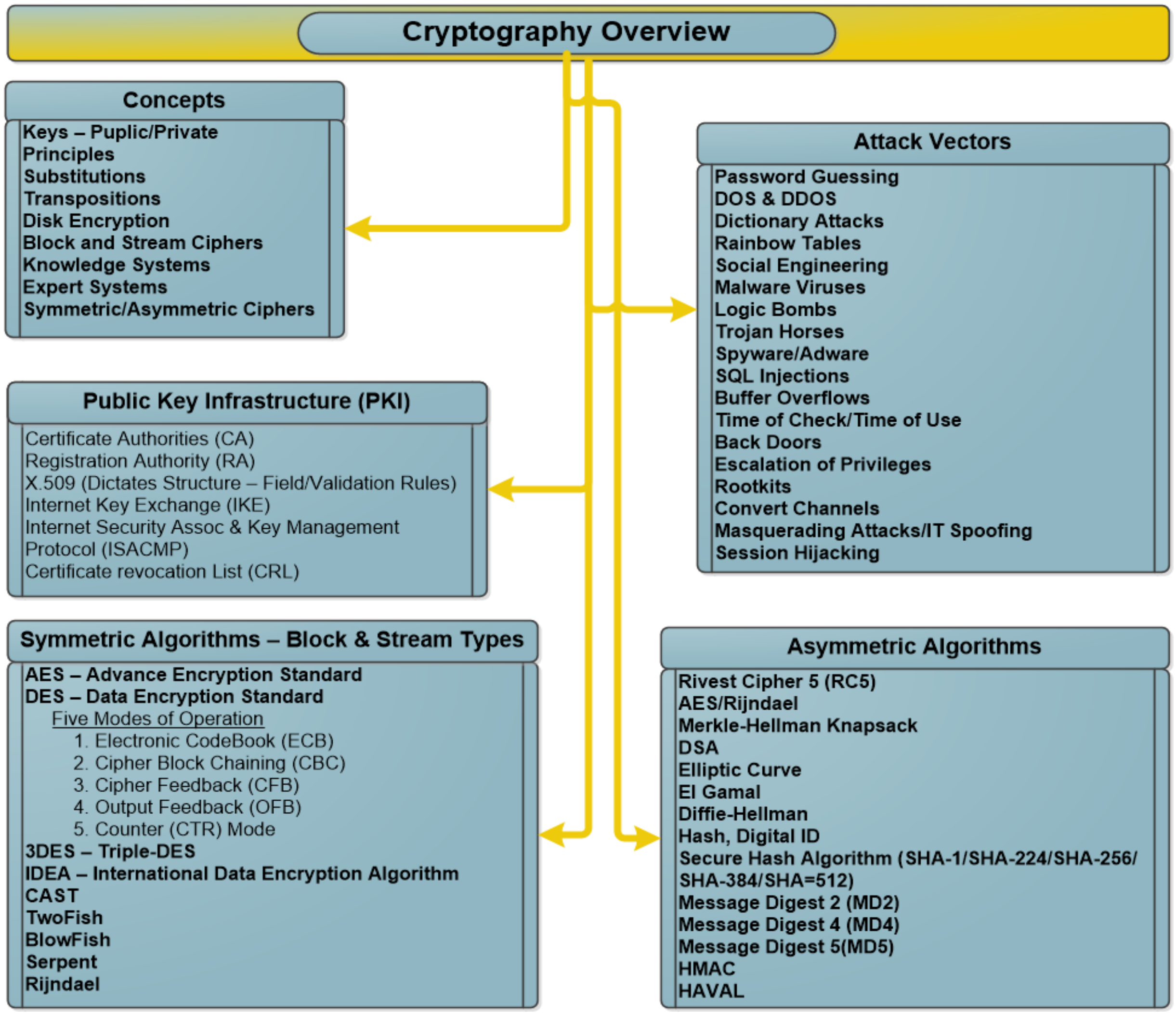
The Art & Science of Encryption Technology

Table of Content

Foreword & Overview

Encryption is just as much an art as it is a science because people have been hiding messages in one form or another for thousands of years. Moreover, it also involves a bit of math and some insight into how people think in order to decrypt some messages. If you're ever in Ft Mead Maryland stop by the "National Cryptologic Museum"; it's next to the NSA Headquarters. The museum holds a treasure of artifacts, from devices and techniques to some of the people who made history in their development.

"**https://www.nsa.gov/about/cryptologic_heritage/museum/**."



Concepts

Keys – Puplic/Private
Principles
Substitutions
Transpositions
Disk Encryption
Block and Stream Ciphers
Knowledge Systems
Expert Systems
Symmetric/Asymmetric Ciphers

Attack Vectors

Password Guessing
DOS & DDOS
Dictionary Attacks
Rainbow Tables
Social Engineering
Malware Viruses
Logic Bombs
Trojan Horses
Spyware/Adware
SQL Injections
Buffer Overflows
Time of Check/Time of Use
Back Doors
Escalation of Privileges
Rootkits
Convert Channels
Masquerading Attacks/IT Spoofing
Session Hijacking

Public Key Infrastructure (PKI)

Certificate Authorities (CA)
Registration Authority (RA)
X.509 (Dictates Structure – Field/Validation Rules)
Internet Key Exchange (IKE)
Internet Security Assoc & Key Management Protocol (ISACMP)
Certificate revocation List (CRL)

Symmetric Algorithms – Block & Stream Types

AES – Advance Encryption Standard
DES – Data Encryption Standard
Five Modes of Operation
1. Electronic CodeBook (ECB)
2. Cipher Block Chaining (CBC)
3. Cipher Feedback (CFB)
4. Output Feedback (OFB)
5. Counter (CTR) Mode
3DES – Triple-DES
IDEA – International Data Encryption Algorithm
CAST
TwoFish
BlowFish
Serpent
Rijndael

Asymmetric Algorithms

Rivest Cipher 5 (RC5)
AES/Rijndael
Merkle-Hellman Knapsack
DSA
Elliptic Curve
El Gamal
Diffie-Hellman
Hash, Digital ID
Secure Hash Algorithm (SHA-1/SHA-224/SHA-256/SHA-384/SHA=512)
Message Digest 2 (MD2)
Message Digest 4 (MD4)
Message Digest 5(MD5)
HMAC
HAVAL

The Four Pillars of Cryptography:

Encryption focuses on four fundamental premises or pillars: Authentication, Confidentiality, Integrity, and Non-repudiation.

Authentication is a major part of cryptography and its main purpose is to verify the user. It provides the assurance as to the identity of users. Authentication can be achieved using either symmetric or asymmetric systems and can be done by any number of mechanisms from passwords, smartcards to biometrics. Or as they say, something you know, something you have, or something you are!

For most of us authentication will happen when you're trying to reach your favorite web site. In this case CHAP which stands for, Challenge Handshake Authentication Protocol (CHAP) for short is working behind the scene. CHAP encrypts your username and password and establishes the communication session to the remote server housing storing the content.

There are many different types of standardized protocol that can help in the authentication process, like Password Authentication Protocol (PAP) which can transmit your user name and password in clear text - not a great idea, or Extensible Authentication Protocol (EAP) which allows for a more custom method of authenticating a user across the web. Now there is mainly two types of encryption methodologies, one is symmetric and the other is asymmetric both encrypt a message but carry out the process differently.

The Symmetric methods can only provide confidentiality and uses only a single shared key which makes it difficult when you have to disturb many keys or scaling-up the process to authenticate a larger number of users. It is also an out-of-band exchange methodology meaning if a secure electronic channel is not available an off-line key distribution method channel must be used and might not be as secure. But the symmetric method is fast which is why it is used in many protocols over the web still today.

The Asymmetric method uses a key pair sets (one public key, and one private key), and works within an In-bank exchange. It's very scalable and not only adds confidentiality, but integrity, authenticity and Non-repudiations. The drawback is it's slower than the symmetric method and works best on small blocks of data. It's also the preferred method in digital signatures, digital envelopes, and digital certificates which I will outline later in the paper.

Confidentiality Ensures secrecy and prevents unauthorized disclosure of data in transit, on disk or during transmission. Confidentiality can be achieved by encrypting data that is stored or in transit using logical or physical access controls, transmission protocols, database views, and controlled traffic flow. This also includes protecting the data from unauthorized modification or deletion. In short, confidentiality is the assurance that information is not disclosed to unauthorized people, processes or applications and only authorized people can change the data.

Integrity verifies that your data has not been altered or changed in any way except by authorized personal. Message integrity is enforced by a mean of an encrypted message digest you create and can be enforced by a public and or secret key cryptosystem. Remember the term message digest since we will also look into this, but can be called by many names: hash, hash value, CRC, checksum, and digital ID. Most message digest are 128 bits or larger, but the longer the message the better the security. I will cover Hash functions later, but the most common ones are SHA, MD2, MD4, MD5 and HMAC to name but a few.

Now there are many forms of encryption, but the major thing to remember is it can be done either through symmetric or asymmetric methods. Moreover when it's in transit you want to ensure that your data has not been altered in any way. One of the main methods of verifying the authentication of the messages is by creating a hash value or message digest guaranteeing the message sent is the same as the one received because confidentiality cannot exist without integrity.

Non-repudiation – Ensures that a sender cannot deny sending a message. Some of the techniques used can be encryption, digital signatures, and notarization.

One of the first ciphers ever used for example:

The Caesar cipher which was one of the first known ciphers to exist was a pretty simple cipher. It worked by shifting the letters three places to the right see Figure 1 below. But you have to understand back in the Roman days few people even knew how to read.

Let's same you have a message you want to encrypt, it's called a plaintext message in the cryptography world, and within the math encryption function let's give it the designation of "P." Now, once you encrypt your message it's is now a ciphertext, and let's give it a math designation of "C." The mathematical algorithm sets the rules on how the message is to be encrypted and decrypted.

The Top part is your basic alphabet and on the bottom is the Caesar Cipher shifted three places to the right.

Figure 1

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

The letter "X" would become "A" and Y becomes B…..etc..etc. The Caesar cipher would later become known as the ROT3 cipher or "Rotate 3." This type of cipher is a substitution cipher. In the English language the most repeated letters in the alphabet are E, T, A, O, N, and R. So it would not take someone long to see a common pattern in the ciphertext to break this type of cipher. The ROT3 can be expressed as a mathematical formula by just changing the letters to numbers for example: A = 0, and B = 1, up to Z equaling 25.

The final Caesar cipher encryption function to encrypt a message would look like this

"To encrypt a message," C = (P + 3) mod 26
"And to decrypt a message," P = (C - 3) mod 26

The "mod" stands for modular arithmetic. It's a system where numbers are wrapping around upon reaching a certain value. Most calculators have a function key that will do the work for you.

Let's look at an example of how to decrypt a ROT3 or Caesar cipher using the equations above.

Decrypt this message: **ZLQQHU**

Us the Caesar cipher from Figure 1, but first take the alphabet and apply numbers to a grid as below.

Figure 2

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Using this equations: P=(C-3) mod 26

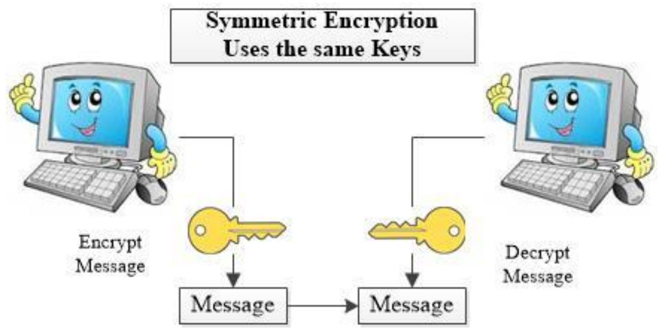
P=(25-3) mod 26 = 22 or W
P=(11-3) mod 26 = 8 or i
P=(16-3) mod 26 = 13 or n
P=(16-3) mod 26 = 13 or n
P=(7-3) mod 26 = 4 or e
P=(20-3) mod 26 = 17 or r

The Message would read: Winner.

Symmetric Cryptosystems:

A Symmetric algorithm uses a single key to encrypt and decrypt data. In other words a symmetric key cryptosystems uses a shared single secret key available to all users of the system.

Graphic 1



Strengths of Symmetric System:

- Faster and uses less computer power needed then asymmetric systems.

- Can be harder to break if using a larger key size.

Weaknesses of Symmetric System: Essentials of Cryptography for Diagrams on the different types below

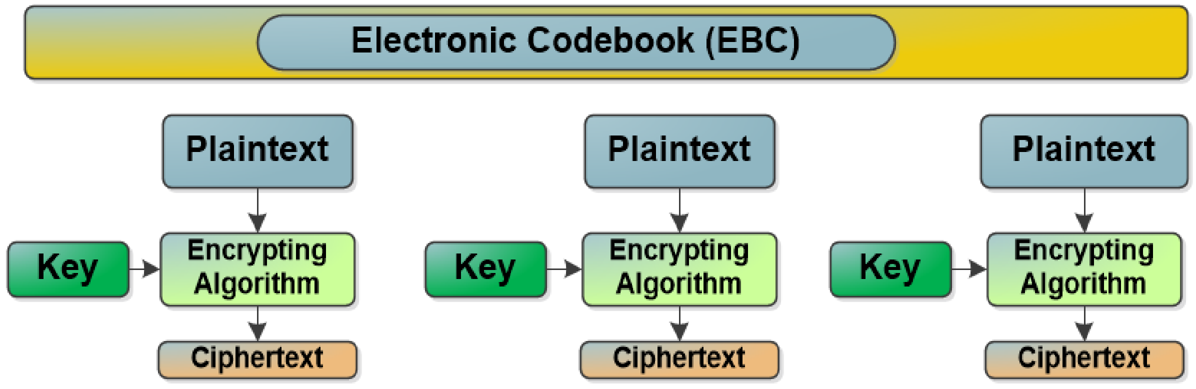
- Key delivers required a secure mechanism
- Each user needs a unique key, which makes key management overwhelming if the pool of users increases beyond a certain point.
- Provides confidentiality, but not authenticity or non-repudiation.
- Below are the main symmetric algorithms used:

Data Encryption Standard (DES) - DES is a 64-bit block cipher, 56-bits for encryption with 8-bits of parity.

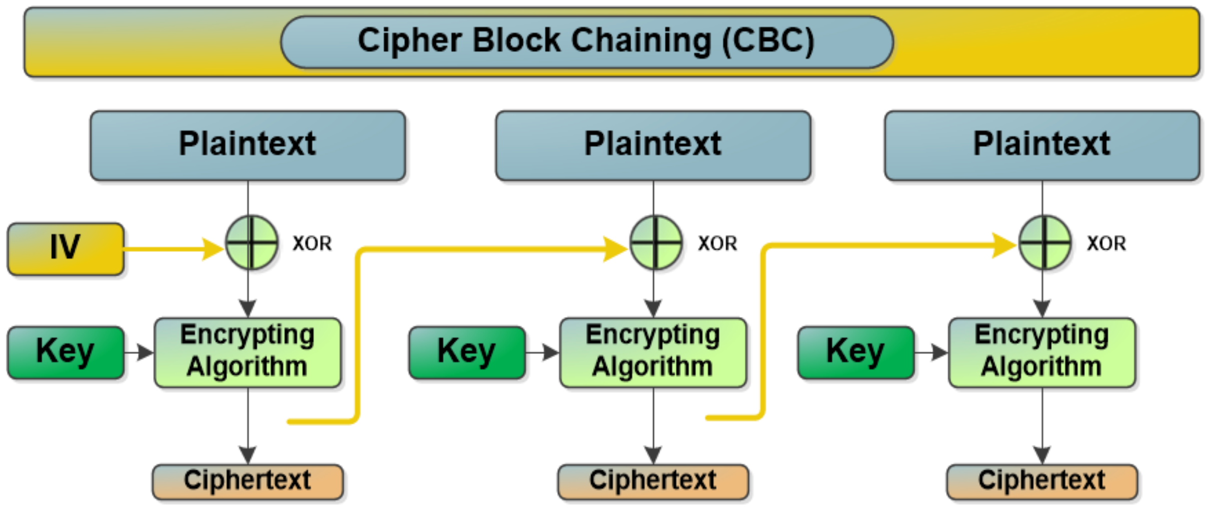
DES has five modes of operations:

1. Electronic Codebook (ECB) mode
2. Cipher Block Chaining (CBC) mode
3. Cipher Feedback (CFB) mode, Output
4. Output Feedback (OFB) mode
5. Counter (CTR) mode

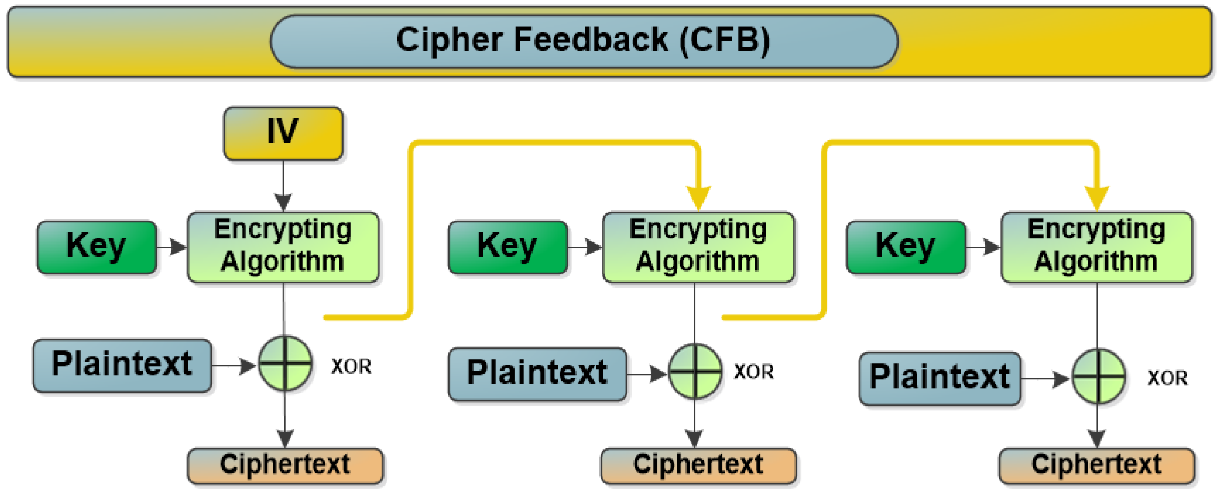
Electronic Codebook (ECB): DES uses an OR (XOR) operation to generate the ciphertext and repeats the process 16 times for encryption/decryption operation. ECB is the simplest mode because there is no feedback. Each 64 bit block is encoded independently from the other blocks, but uses the same key - meaning the same plaintext will always result in the same ciphertext



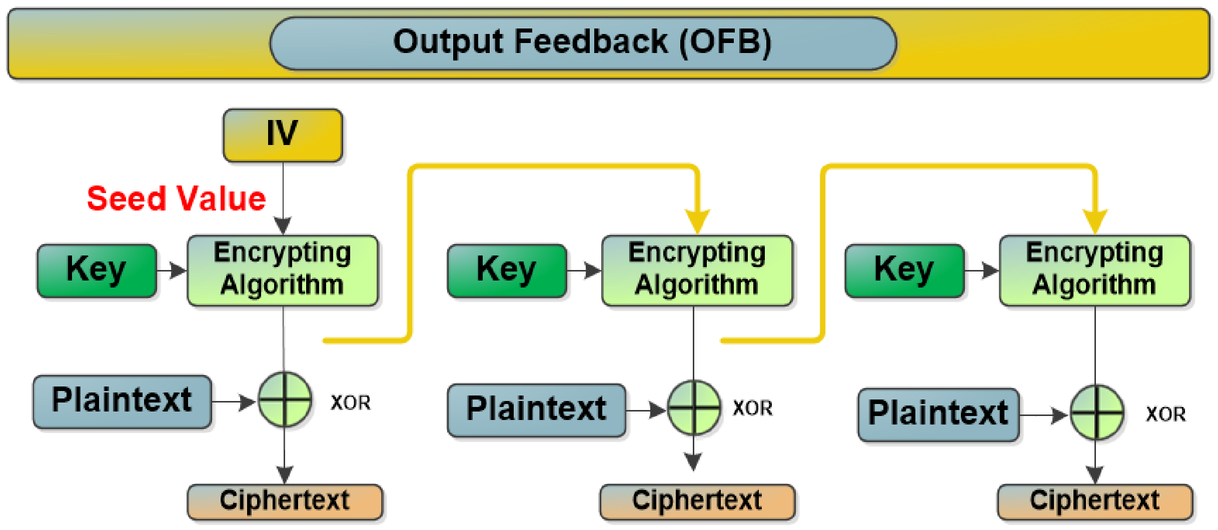
Cipher Block Chaining Mode (CBC): Each block of text is XORed with a block of ciphertext immediately preceding it before its encrypted using the DES algorithm. Meaning, it adds an Initialization Vector (IV), which is an encrypted block of data used as the first 64 bit block when it begins the chaining process. Next it processes an exclusive XOR operation with each previous block of code as it move down the chain.



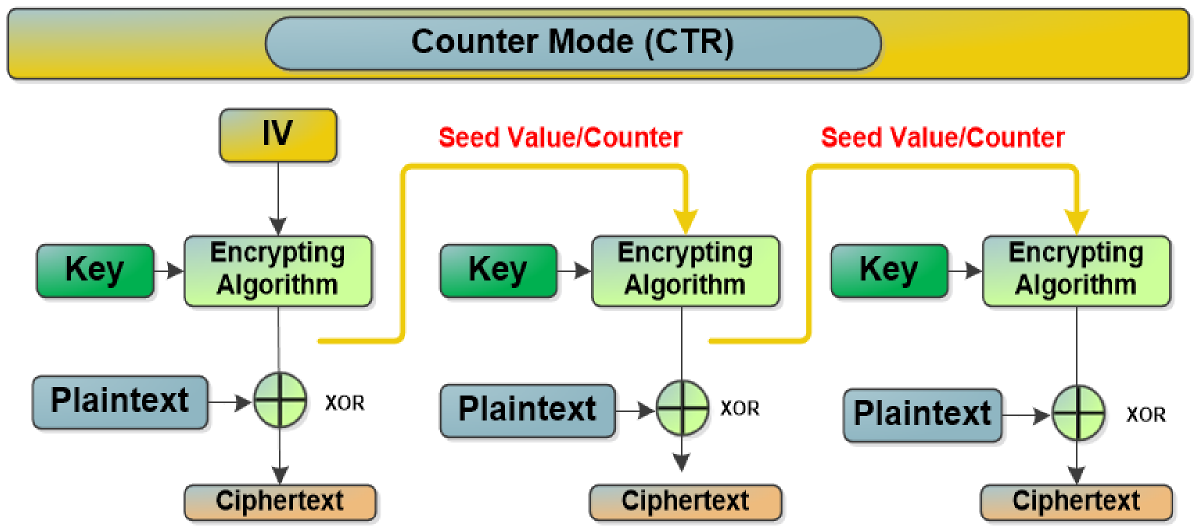
Cipher Feedback Mode (CFB): This is the streaming cipher version of CBC. It operates against data produced in real time. The cipher uses a memory buffer to hold the data as it encrypts before forwarding it on.



Output Feedback (OFB): Works like the Cipher Feedback Mode (CFB) above, but instead of using just an exclusive XOR encryption operations. It applies a seed value to the process as it moves forward. The OFB prevents early errors in the process from interfering with later encryption/decryption processes; meaning it does not carry forward errors.



Counter Mode (CTR): Uses a stream cipher like CFB and OFB. But it creates a seed value for each operation and adds a counter to increment each step.



Other Symmetric Encryption Methodologies

- **Triple DES (3DES)**
  Triple DES or 3DES, just encrypts the message three time using three different keys – example: **E(K1,E(K2,E(K3, P)))**
- **International Data Encryption Algorithm (IDEA)**
  Operates on 64-bit blocks of ciphertext, with 128-bit key where each operation is broken up into 52 16-bit sub-keys.

- **Blowfish** - Operates on 64-bit blocks, but it also allows a variable length set of keys from 32 bits to 448 bits.
- **Skipjack** - Operates on 64-bit blocks and uses 80-bit key. It also supports four mode of operation supported by DES.
- **Advanced Encryption Standard** (AES) – Is a Block Cipher and has replaced the old DES standard. It uses three key lengths, 128 bits, 192 bits, or 256 bits.
  - 128 bit key - using 10 rounds of encryption
  - 192 bit key - using 12 rounds of encryption
  - 256 bit key - using 14 rounds of encryption
- **RC5 -** Rivest Cipher 5 (RC5) uses block sizes of 32, 64 and 128 bits with key sizes ranging from 0 to 2,040 bits.

One difficulty with using the symmetric encryption methods is that it's not scalable, and you're dealing with a single key that needs to be share shared. One method of exchanging keys is to use the "Diffie-Hellman" algorithm. It was developed in the 70s, and is still used today as a means to exchange keys; because it has proven to be an extremely useful mechanism for sharing keys.

The Diffie-Hellman algorithm works as follows:

1) Sue and Jay want to communicate using an asymmetric algorithm. Each agrees before hand to use two large numbers, "P" for a prime number and "I" for the Integer. Keeping in mind that "**1 <Integer <Prime**" for numbers that each will use.

2) Jay chooses a random large integer J for his calculation raising the integer to that power: $J = I^J \bmod P$

3) Sue chooses a random large integer S for her calculation raising the integer to her power: $S = I^S \bmod P$

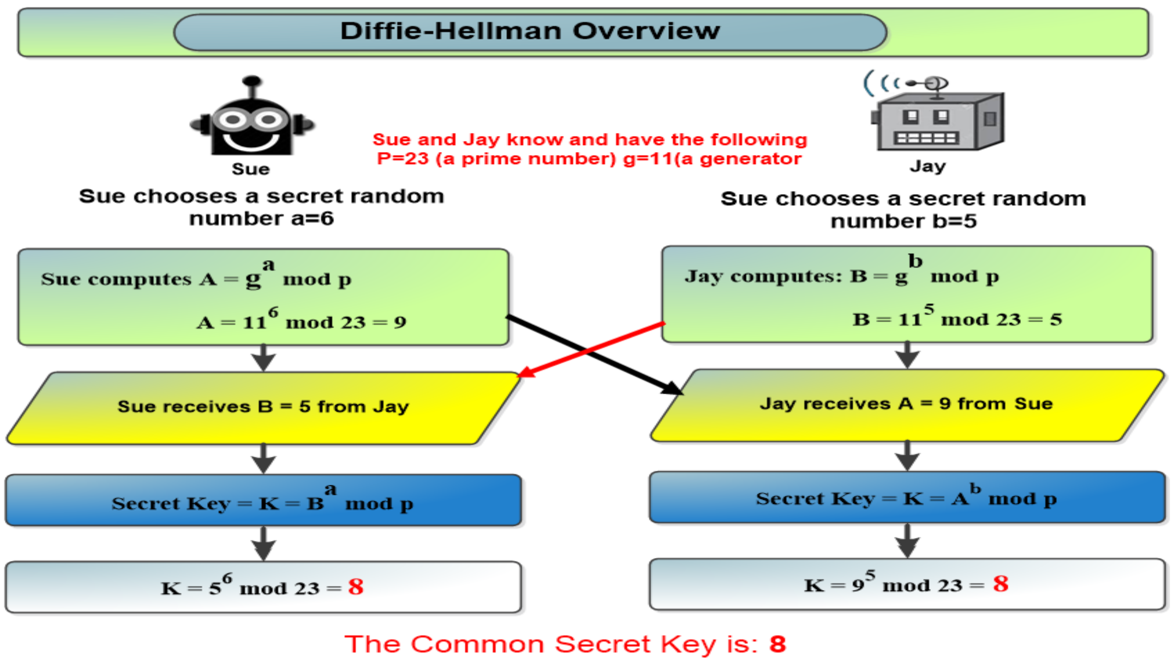
4) Jay and Sue now exchange their generated values. Jay sends "J" to Sue and Jay sends "S" to Sue.

5) Once each receives the others calculated value

6) Jay performs the final calculation: $K = S^J \bmod P$

7) Sue performs her calculation: $K = J^T \bmod P$

Now Jay and Sue will both have the same value, K, which can now be used for secret key communications.


Diffie-Hellman Overview
Sue
Sue and Jay know and have the following P=23 (a prime number) g=11(a generator
Jay
Sue chooses a secret random number a=6
Sue chooses a secret random number b=5
Sue computes $A = g^a \bmod p$
$A = 11^6 \bmod 23 = 9$
Jay computes: $B = g^b \bmod p$
$B = 11^5 \bmod 23 = 5$
Sue receives B = 5 from Jay
Jay receives A = 9 from Sue
Secret Key $= K = B^a \bmod p$
Secret Key $= K = A^b \bmod p$
$K = 5^6 \bmod 23 = 8$
$K = 9^5 \bmod 23 = 8$
The Common Secret Key is: 8

If you would like to see an example of how it works, check out this web site:

Link to online Example:

https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/diffie-hellman-key-exchange-part-2

Figure 3

Symmetric Algorithm Chart		
Name	**Block Size**	**Key Size**
Advanced Encryption Standard (AES)	128	128, 192, 256
Rijndael	Variable	128, 192, 256
Blowfish	Variable	1-448
Data Encryption Standard (DES)	64	56
IDEA (Used by PGP)	64	128
Rivest Cipher 2 (RC2)	64	128
Rivest Cipher 4 (RC4)	Streaming	128
Rivest Cipher 5 (RC5)	32, 64, 128	0-2,040
Skipjack	64	80
Triple DES (3DES)	64	112 or 168
Twofish	128	1-256

Symmetric Keys and scalability: To understand why the symmetric method is not very scalable you can use a simple calculation to see the number of keys you would need as the number of users increases. Let's input 100 users below just to see how many keys one would need?

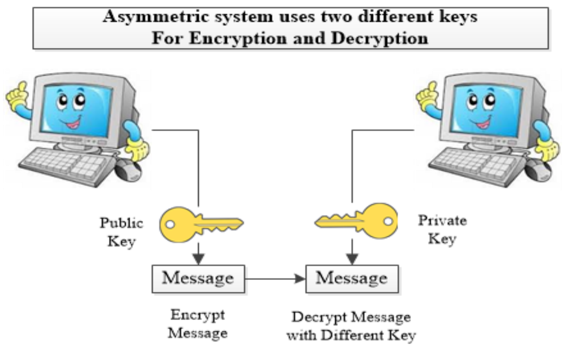
Number of key = N(N-1)/2

So let’s say you have 100 users. 100(100-1)/2 = 4,950

The number of symmetric keys required would be 4,950. Now you compare that to the same number of participants with the asymmetric method which would be only 200, and you get the idea of the scalability problem with the asymmetric method. The number of keys required would become a major security problem in managing and distribution.

Asymmetric Cryptosystems

Graphic **2**


Asymmetric system uses two different keys
For Encryption and Decryption
Public Key
Private Key
Message
Message
Encrypt Message
Decrypt Message with Different Key

Asymmetric avoids some of the problems you have with symmetric key cryptography because each user generates a pairs of public and private keys. This forgoes the difficulty of complex key distribution you have with symmetric keys as well as scalability and other issues. Three of the most common asymmetric cryptosystem today are the RSA, El Gamal and the Elliptic Curve. El Gamal is an offshoot of the Diffie-Hellman key exchange algorithm which I covered above. The El Gamal has one big disadvantage in that it doubles the length of any message it encrypts. On the other hand Elliptic Curve algorithms use elliptic curve discrete logarithms and to some it is more secure, but both keys use the same length. In addition, the Elliptic Curve is one of the few algorithms still strong enough to withstand a Quantum Crypto systems attack. This is something to think about because Quantum Computers are here now! Also note that the 1.088 RSA key is equivalent to a 160-bit Elliptic Curve cryptosystem key.

The best known of these three is the RSA system. The RSA algorithm which is named after the mathematicians who created it: Rivest, Shamir, and Adleman; is one of the best known Asymmetric key cryptography methods. The RSA method depends on the difficulty in factoring large prime numbers.

The algorithm below is how each user generates a pair of public and private keys.

1) Pick two prime numbers (approximately 20 digitals each), labeled p and q

2) Compute the product of those two numbers: n= p*q

3) Select a number, e, this must satisfy two requirements:

- **e** is less then **n**
- **e** and (n-1)(q-1) are relatively prime. Example: The two numbers have no common factors other than 1.

4) Find a number, **d**, that (ed-1) mod(p-1)(q-1) = 0

5) Distribute **e** and **n** as the public key to all cryptosystem users; Keeping **d** secret as the private key.

Figure 4

Comparison of Symmetric and Asymmetric Systems	
Symmetric	**Asymmetric**
Single Shared Key	**Key Pair Sets**
Out-of-Band Exchange	**In-Band Exchange**
Fast Processing	**Slow Processing**
Bulk Encryption	**Small Blocks of Data**
	Digital Signatures, Digital Envelopes, and Digital Certificates
Confidentiality	**Confidentiality, Integrity, Authenticity, and Non-Repudiation**

The key lengths for the best known Asymmetric keys are:

- RSA - 1,088 bits
- DSA 1,024
- Elliptic Curve 160 bits

One-Pad System aka Vernam Cipher

The One-time Pad is a very powerful substitution cipher developed by Gilbert Vernam in 1917, it is sometimes called the Vernam Cipher.

It uses a different substitution alphabet for each letter of text. This cipher does not just substitute alphabets like the Caesar or Vigenere ciphers. It uses a pad of random values compute its cipher. If used correctly is considered unbreakable even by quantum computes.

Its weakness is that computers really don’t generate a true set of random numbers, but in theory a quantum computer could generate the required perfect set of random numbers one could use. Now given that people are people we tend to repeat ourselves even if we don't realize it, and the machines we have created so far truly don’t create sets of real random numbers. But if you could in theory; the One-Time Pad system if implemented correctly is strong enough that neither a super-computer nor a quantum computer could break it?

The encryption system uses a binary mathematic function called exclusive-OR, abbreviated as XOR. XOR is a binary system and applies two bits to a message stream. If both values are the same for example (1 XOR 1 = 0); the bits are different for example (1 XOR 0 = 1). Below is how the message stream will change once the XOR is applied.

Figure 5

One-Pad System using XOR mathematics											
Message Stream	↓	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**
Keystream		**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**
Ciphertext Stream		**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**

Basic rules when using the One-Pad Systems

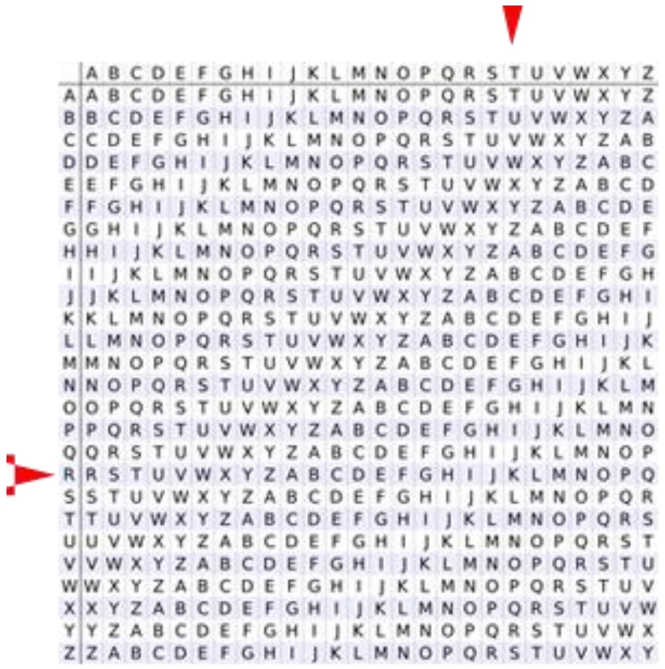
1) The pad must be used only one time.
2) The pad must be as long as the message.
3) The pad must be securely distributed and protected.
4) The pad must be a set of truly random values.

Running Key Cipher

The Running Key Cipher is also a very effective algorithm and hard to break when applied correctly by two parties. Some call it the “book cipher,” and is a type of polyalphabetic substitution cipher. This is where the two parties agree ahead of time to use text from a common book they both know and possess. Let’s say each use a favorite poem for example, and beginning on the first paragraph starting with the chapter of “Great Poems of the twenty Century.” You use every consecutive character as necessary to perform the encryption and decryption operations.

For example: You send this message to your friend that reads -- "RUN FOR IT," they are coming to get you. You can take out the spaces to make it easier to encrypt.

Figure 6



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Plaintext: RunForIt

Looking at the first paragraph the poem starts with **"Tiger Eyes in the night."**

Running Key: TIGEREYE
PlainText: RUNFORIT

Now for the Ciphertext you would have a copy of a tabula recta, a simple matrix of shifted letters. Starting from the left side going across from R, down from T, it's a K for the first letter. Now do this for each letter.

The Ciphertext: KCTJFVGX

Now to decrypt, just map the Running Key: **TIGEREYE** – this you know from the book of poems you both agreed to use.

CipherText Message you received: **KCTJFVGX**

Running Key: TIGEREYE
Ciphertext: KCTJFVGX

On the Left side starting with T across till you find K, then go up to the letter you need, R.

Next the letters I and C going across till you hit C, then scan up to find U. Do this for each letter to decrypt the message: which is "RUN FOR IT."

He or she would know what passage or poem you are using since you both decided before hand what book of poems. Below is a link to a web site you can input your own cipher and see how it works.

Some online resources:

http://crypto.interactive-maths.com/other-examples.html

http://rumkin.com/tools/cipher/

Vigenere Encryption Systems

The Vigenere cipher was developed in the 16th by Blaise de Vigenere of France. It's another polyalphabetic substitution cipher based on the Caesar cipher. Unlike the Caesar cipher that shifted letters from one shift to up to three places within a given alphabet. The Vigenere increases the complexity of the cipher by 27 shifts and the letters are shifted up only one place. The Vigenere uses a table very similar to the running key cipher above, but this type of cipher does have its weaknesses one of which is a repeating key-stream. One of the ways you could use to increase the complexity of the cipher was to add confusion and diffusion techniques.

Confusion: is a means of obscuring the relationship between the plain text message, making it so complicated the attacker cannot just alter the message and analyzing the resulting to determine a pattern.

Diffusion: this involves changing the message in multiple ways and spreading the change throughout the ciphertext. You can also add in initialization vectors to make process of breaking the message even more confusing to the attacker.

Initialization Vectors: These are random values used within the algorithm to remove patterns that an attacker might find, helping them to break the code. Here is a few other ways to that people have used to obscuring the cipher making it harder to decrypt.

Compression: Take your plaintext before you encrypt it and run compression functions on it to reduce redundancy within the text.

Expansion or Padding: Expanding the plaintext message so it maps to the key size your using or adding padding either before the message or after it before it is encrypted.

Key Mixing: Using sub-keys to limit the exposure of the main key. You can generate a sub-key based on the master keys which will enhance your overall message security. This is a little more complex than the other methods, but might be worth it if the message is critical.

You can get a taste of the Vigenere Cipher and how it works with this link:

http://www.cs.du.edu/~snarayan/crypt/vigenere.html

The Digital Signature Algorithm (DSA)

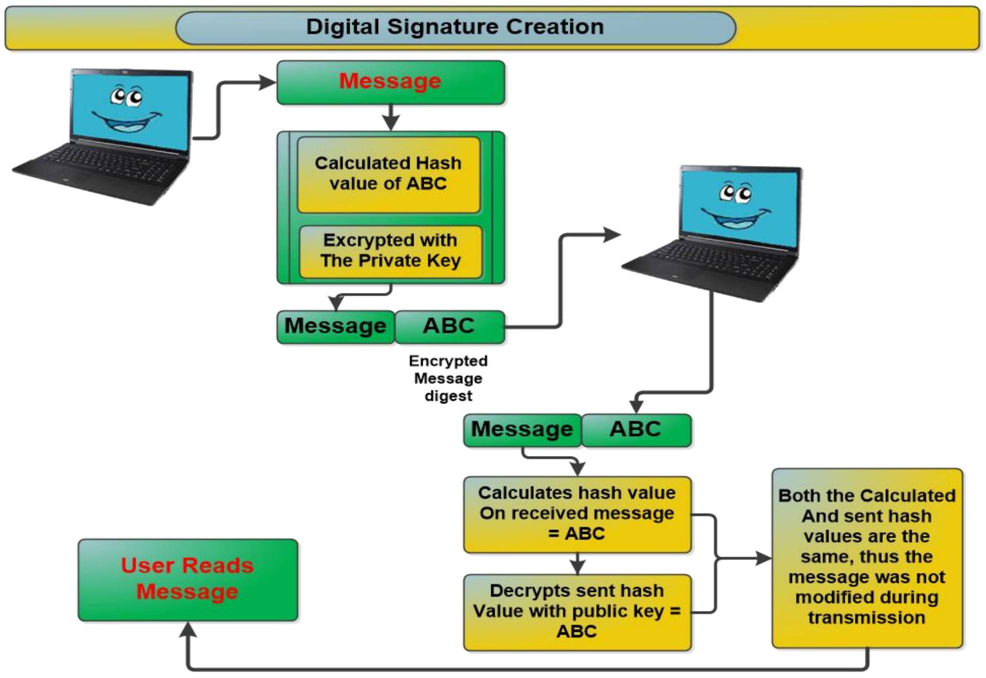
Digital Signatures are an important part of the digital world we live in today. If say a friend wants to send you a message and he or she wants to be sure it came only from you, he or she would use a digitally signed algorithm.

You can digitally sign or encrypt an entire message using one of many asymmetric algorithms, but in most cases it's more practical encrypting a hash of the message. If one creates a signature before encrypting the message, ones can authenticate the message signature itself and not the ciphertext of the message. The hashing function ensures the integrity of the message, and signing the hash value with the private key, provides authentication and non-repudiation.

In the early 1990's the government proposed a federal standard called Digital Signature (DSS) and task the National Institute of Science and Technology (NIST) to developed specifications and standards which most vendors followed in designing their products today. When the DSS was released in 1991 the document was called FIPS 186 using the Secure Hashing Algorithm (SHA). It's been updated several times since and the most recent version is FIPS 186-4. It has also expanded to include the RSA algorithm named after its inventors Rivest, Shamir, and Adleman, and elliptic curve cryptosystem (ECC) encryption methodologies.

DSA was developed by the NSA, and is only used for digital signatures. RSA which is one of the better known algorithms can be used for signatures, encryption, and secure distribution of symmetric keys.

The digital signature is a hash value that is encrypted with the sender's private key. Encrypting the messages is called the act of signing the message with a private key. Below is a graphical example of a digital signature.


Digital Signature Creation
Message
Calculated Hash value of ABC
Excrypted with The Private Key
Message
ABC
Encrypted Message digest
Message
ABC
Calculates hash value On received message = ABC
Decrypts sent hash Value with public key = ABC
Both the Calculated And sent hash values are the same, thus the message was not modified during transmission
User Reads Message

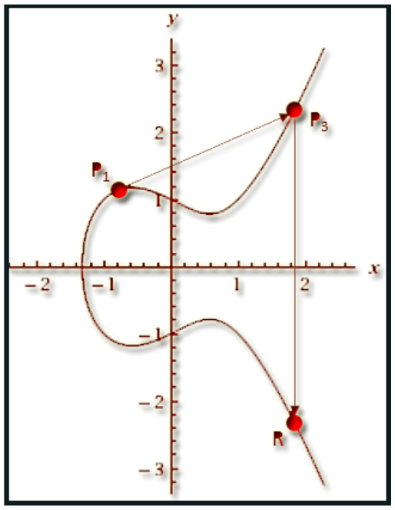
- The RSA Algorithm is an asymmetric algorithm used for encryption of digital signatures, and key exchange. It's based on the difficulty of factoring large numbers into their original prime numbers.

- El Gamal algorithm is another asymmetric algorithm based upon the Diffie-Hellman algorithm used for digital signature, encryption, and key exchange
- Elliptic curve cryptosystem algorithm is an asymmetric algorithm based upon the algebraic structure of elliptic curves over finite fields. Used in the creation of digital signature, encryption, and key exchange.
- Message authentication code (MAC) keyed cryptographic hash function used for data integrity and data origin authentication.

The Elliptic Curve (ECC)

The Elliptic Curve uses a shorter key length then the RSA algorithm but has many of the same functionality. It also requires less computing power which is one of the reasons it's used in cell phones and other wireless devices. ECC is based on the idea of using points on a curve which is less mathematically intensive then other encryption methodologies. The RSA algorithm provides digital signatures, secure key distribution, and encryption. ECC differs in that it's more efficient then other asymmetric algorithms.

In mathematical theory points on a curve compose a structure called a group. These points are values used in ECC formula to encryption and decryption processes. The Diffie-Hellman and El Gamal algorithms use logarithms in a finite field. ECC compute discrete logarithms of elliptic curves – (see graphic). ECC fills a need since it requires less resource's to compute so devices like cell phones and tablets with limited processing capacity, storage, power, and bandwidth use it.



ECC does this by using a smaller key length, but offers the same level of protection when compared to the longer key length used by RSA. So it's not always the case that a longer key size adds more protection. The link below is a Youtube video about the Elliptical Curve.

https://www.youtube.com/watch?v=dCvB-mhkT0w

ECC vs RSA

ECC Advantages:

- Smaller keys, ciphertexts and signatures.
- Very fast key generation.
- Fast signatures.
- Moderately fast encryption and decryption.

- Signatures can be computed in two stages, allowing latency much lower than inverse throughput.
- Good protocols for authenticated key exchange (FH-ECMQV et al)
- Better US government support.
- Special curves with bilinear pairings allow new-fangled crypto.
- Binary curves are really fast in hardware.
- One of the Few Algorithms strong enough to withstand a Quantum Computer cipher attack

ECC Disadvantages:

- Complicated and tricky to implement securely, particularly the standard curves.
- Standards aren't state-of-the-art, particularly ECDSA which is kind of a hack compared to Schnorr signatures.
- Signing with a broken random number generator compromises the key.
- Still some patent problems, especially for binary curves.
- Newer algorithms could theoretically have unknown weaknesses. Binary curves are slightly scary.
- Don't use DUAL_EC_DRBG, since it has a back door.

RSA advantages:

- Very fast, very simple encryption and verification.
- Easier to implement than ECC.
- Easier to understand.
- Signing and decryption are similar; encryption and verification are similar.
- Widely deployed, better industry support.

RSA Disadvantages:

- Very slow key generation.
- Slow signing and decryption, which are slightly tricky to implement securely.
- Two-part key is vulnerable to GCD attack if poorly implemented.

The Enigma Machine

The most famous rotor encryption machine is the Enigma used by the Germans in World War II.

Arthur Scherbius was the German engineer who developed the Enigma machine. The first version of the machine used three to five notched wheels or rotors. Once the wheels are set to a set of letters the operator can type in a message and it will scramble the letters based on those settings. You need to know the setting of the wheels in order to describe the message. In later versions of the machine a plug board was added to make it even more complicate.

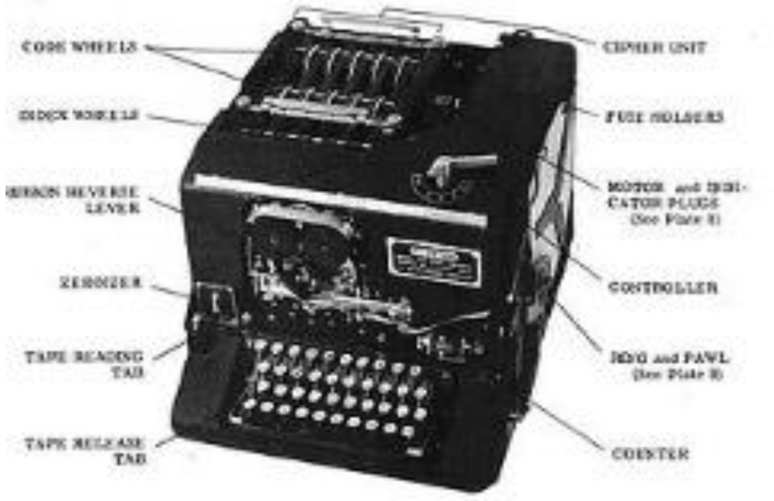
Before the start of the II world war the Polish Cipher Bureau between the years of “1933 to 1938,” where one of the first to decipher Enigma messages. They also managed to reconstruct an Enigma machine and it internal wiring. This was possible because of their extensive links to the German engineering industry before the war. Later they shared that information with the British who finally broke the upgraded versions of Enigma – calling their program “Ultra,” which was based out of Bletchley Park in Buckinghamshire.

http://www.bbc.co.uk/history/places/bletchley_park

Alan Turning was one of the brilliant mathematicians who played a key role in breaking the Enigma code along with Gordon Welchman they build one of the first computers known as the bombe because when it was working it ticked like a bomb. Alan Turning is credited helping shorten the war by about two years, and saving around 14 million people.

http://www.iwm.org.uk/history/how-alan-turing-cracked-the-enigma-code

A lot of historians forget we also had our own version of the Enigma machine called the ECM Mark II, also known as the SIGABA. It worked similar to the Enigma machine using multiple stepping rotors to encipher text. But the Enigma machines had a weakness in that it advanced its rotors regularly; rotor 1 would advance rotor 2 by one every 26 letters, rotor 2 would advance rotor 3 by one every 26 letters…etc.



The ECM Mark II used two banks of five rotors that controlled the advancement of the rotors and varied it advancing of the rotor wheels based on any number of letters. This machine generated several orders of magnitude when compared to the German version. During the war the German Codebreaking organization called B-DIENST never broke any code generated by the ECM Mark II.

In addition, Alan Turning also helped break the “Lorenz Cipher Machine” which too many say was even more complex than the Enigma Machine. Turning developed a process he called ‘Turingery’.

Reading a Turning message required three key things to be understood.

1. Firstly that the logical structure of the system has to be known.
2. Secondly that the periodically changed pattern of active cams on the wheels was derived.

3. Thirdly that the starting positions of the scrambler wheels for this message—the message key—was established.

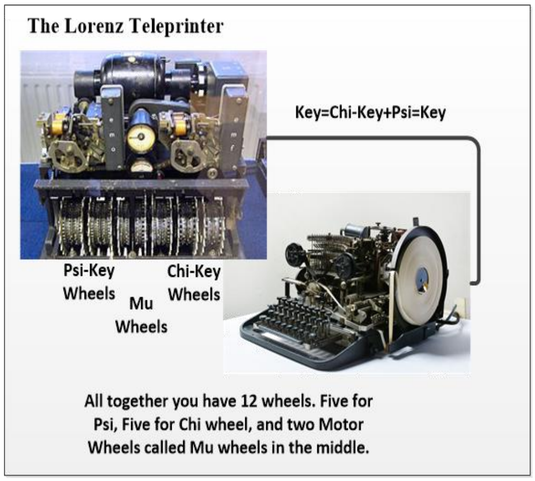
Below is a link to see how the Enigma Machine Worked.

http://enigmaco.de/enigma/enigma.html

Lorenz Teleprinter

The Lorenz cipher uses a 32-symbol baudot code unlike the Enigma machine which uses a 26 letter alphabet code. On each wheel there is a set of pins which act as ON or OFF switches telling the machine it can send or not send based on how the pins are set. There are a total of 501 pins which means there is a **$2^{501} = 6.5 \text{ X } 10^{150}$** ways to set the wheels, which is something like a "One hundred Trillion, Trillion Trillion….googles"….!

Each wheel also has different starting positions and if you add that together would be another large number of settings something like Sixteen million, million, million, or **$1.6 \text{ X } 10^{19}$,** so the total number of settings for the Lorenz machine is something like, **$1.0 \text{ x } 10^{170}$**.


The Lorenz Teleprinter
Key=Chi-Key+Psi=Key
Psi-Key Wheels
Mu Wheels
Chi-Key Wheels
All together you have 12 wheels. Five for Psi, Five for Chi wheel, and two Motor Wheels called Mu wheels in the middle.

How they broke the code was one day an operator sent a message and the receiver asks him to resend for some reason. He resent the message using the same setting or key as before and also the abbreviated some of the words somewhat to speed-up redoing the message. This gave the code breakers two sets of messages they could use in deciphering the machines settings. In addition, machines do not generate completely random sequence of characters. They generally create what is called pseudo-random sequences. Unfortunately for the Germans it was more pseudo than random which gave the code breakers the edge in decrypting the systems generated key.

Below is some links as to how the Lorenz Teleprinter code worked:

https://www.codesandciphers.org.uk/lorenz/fish.htm

http://cs.stanford.edu/people/eroberts/courses/soco/projects/2008-09/colossus/baudot.html

https://www.youtube.com/watch?v=GBsfWSQVtYA

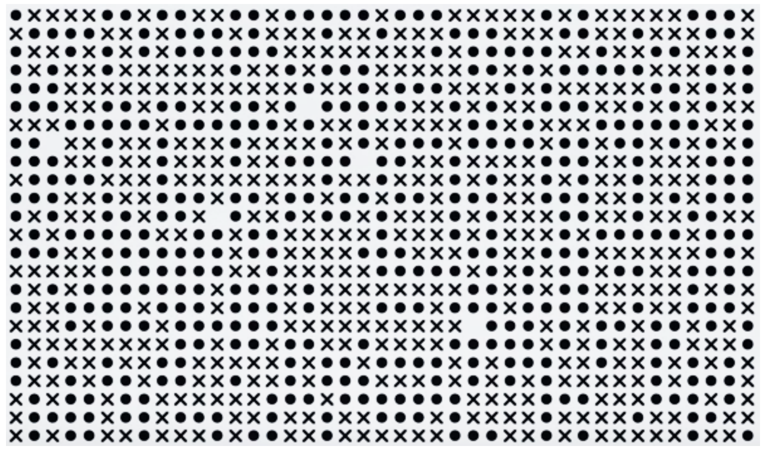
Using the baudot code which when on it's a cross (X), if it's off it adds a dot(.) For example you want to send the letter C which is a set of dots and X's. Next the machine would add on a random letter which for our example we added the letter M.

Message C:	.	X	X	X	.
Key M:	.	.	X	X	X
Code Generated L:	.	X	.	.	x

The machine finally generates the random letter L. What is interesting if you add back the key M letter again, you get back to where you started.

Now once you have two sets of the same message using the same key. You begin to look for patterns and one way you do that is to write out the code in rows to see what types of patterns might show up.

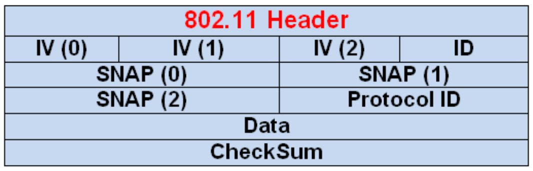
Example of one of the patterns:



This helped them work out the length of the keys because when they wrote it out in rows of 41 a pattern started to show up. This pattern lead to others that finally help them decipher how the other wheels on the machine worked together to create the machine. Once you understand how the machine works then you need to understand all the setting. Alan Turing had already worked out a method to figure out how the pins got set on the outside of the wheels. So once you have a method to work out the wheel and pins setting you can decrypt about ninety percent of the messages sent by the machine.

Wireless Encryption Algorithms – WEP, WPA, and WPA2

One of the first wireless encryption algorithms created was called the Wired Equivalent Privacy (WEP). WEP comes in two favorers or sizes 40-bit and 104-bits with a 24-bit initialization vector (IV). When it was first introduced was quickly hacked because it had a serious design flaw that let hackers derive the encryption key and see all traffic going across the wireless network. At the time WEP used the RC4 algorithm and was considered sufficient for its time.

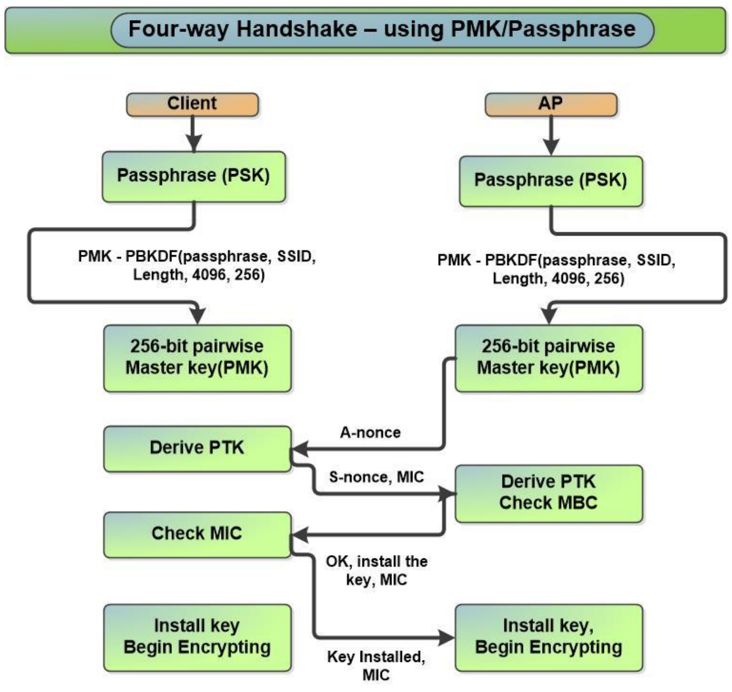


802.11 Header			
IV (0)	IV (1)	IV (2)	ID
SNAP (0)		SNAP (1)	
SNAP (2)		Protocol ID	
Data			
CheckSum			

But it’s how the WEP algorithm uses the secret key that is shared between the Access Point (AP) and the client node on the wireless network that makes it acceptable to hacking. The 802.11 network WEP does not encrypt the header, nor does it encrypt the initialization Vector (IV), or the ID portions of the packet.

WEP uses the initialization vector (IV) and appends the IV to the secret key before feeding the key into RC4 algorithm. This means that if an attacker can collect enough data can easily figure out the key. So as this shared key is going across the encrypted network all data between your computer and the AP node can be seen. Moreover, WEP does not provide network authentication via the use of this shared secret and is acceptable to a number of attack vectors, from replay attacks to dictionary attacks.

WPA/WPA2 is a subset of the 802.11i standard and increased the level of security given the types of encryption used moving from RC4 to AES, and TKIP vs. Cipher Block Chaining Message Authentication Code Protocol (CCMP). The Temporal Key Integrity Protocol (TKIP) which you may know took over after the initial shared secret is entered in your wireless devices and handles the encryption and automatic rekeying, but now is replaced with the CCMP protocol.


Four-way Handshake – using PMK/Passphrase
Client
AP
Passphrase (PSK)
Passphrase (PSK)
PMK - PBKDF(passphrase, SSID, Length, 4096, 256)
PMK - PBKDF(passphrase, SSID, Length, 4096, 256)
256-bit pairwise Master key(PMK)
256-bit pairwise Master key(PMK)
A-nonce
Derive PTK
S-nonce, MIC
Derive PTK Check MBC
Check MIC
OK, install the key, MIC
Install key Begin Encrypting
Key Installed, MIC
Install key, Begin Encrypting

WPA/802.11i has two modes of operation, home or enterprise mode if your running in home mode the AP and all clients use a pre-shared key (PSK). Enterprise mode means you organization is using a RADIUS server to authenticate. The diagram shows how WPA pre-shared key works.

WPA2 Vulnerability

A major weakness in the WPA2 protocol has come out in the last few months. It allows attackers to intercept passwords, emails, and other types of data. The proof-of-concept that exploited this flaw is called KRACK or (Key Reinstallation Attacks). The hack is most effective on Android, Linux and OpenBSD systems, and Windows systems to a lesser extent; but virtually all systems that use WPA2 for encryption on their networks.

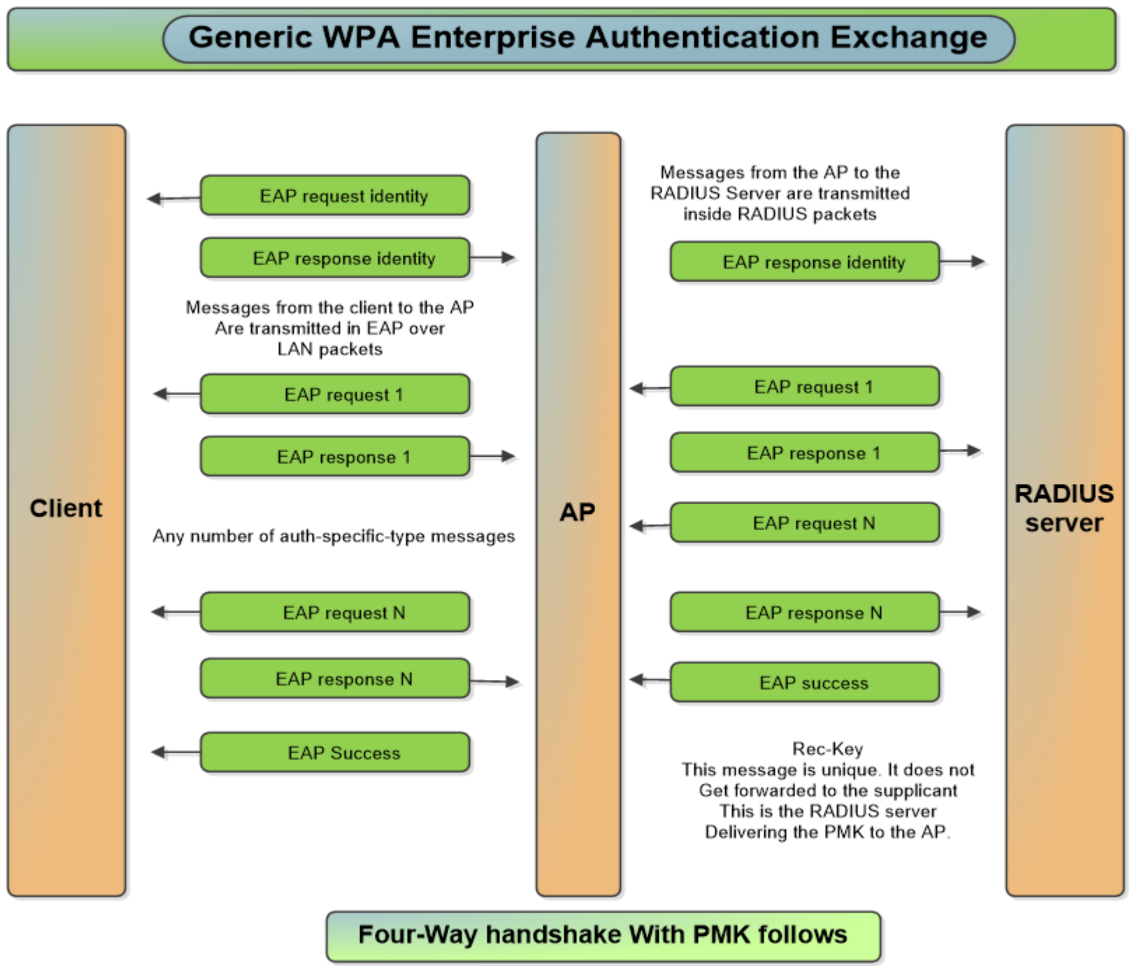
KRACK works by targeting the four-way handshake when a client wants to joining a WPA2 network. KRACK tricks the client into reinstalling an already in use key. This forces the client to reset packet numbers resetting them to their initial values. In a word it works by forcing a reinstall of keys with all-zero encryption keys, rather than the real keys.

The attack works against all modern Wi-Fi networks depending on their configuration. When implemented it can not only steal credit card and other personal information it can even give the attacker the option to inject and manipulate data going across the wireless network.

Even if the user is using an HTTPS-protected web page it does not mean you're protected given so many web servers are configured incorrectly. The attacker can use a script like SSLstrip to force the web site to downgrade its connection from an HTTPS session to a HTTP one.

Remember this WPA2 vulnerability affects not only computers, but smart-phones, tablets or anything that uses a Wi-Fi connection. On a positive note neither Windows nor iOS are vulnerable to the most severe attacks. This is not to say it cannot be done because each platform has its own level of difficulty, and it depends on the configuration, OS version, and browser being used during the connection.

The graphic below shows you a WPA Enterprise Level Authentication Exchange


Generic WPA Enterprise Authentication Exchange
Client
AP
RADIUS server
EAP request identity
EAP response identity
Messages from the client to the AP Are transmitted in EAP over LAN packets
EAP request 1
EAP response 1
Any number of auth-specific-type messages
EAP request N
EAP response N
EAP Success
Messages from the AP to the RADIUS Server are transmitted inside RADIUS packets
EAP response identity
EAP request 1
EAP response 1
EAP request N
EAP response N
EAP success
Rec-Key This message is unique. It does not Get forwarded to the supplicant This is the RADIUS server Delivering the PMK to the AP.
Four-Way handshake With PMK follows

SSL (Secure Socket Layer), TLS (Transport Layer Security)

SSL was originally developed by Netscape and first came onto the scene way back in 1995 with SSL 2.0. Version 2.0 was quickly replaced by SSL 3.0 in 1996 after a number of vulnerabilities were found. Later another version was developed called TLS but the differences between SSL 3.0, and TLS 1.0 was not that significant, and the two versions do not interoperate. In addition, the name was changed to avoid any legal issues with Netscape so that the protocol could be "open source, in a word free software."

SSL utilizes two keys:

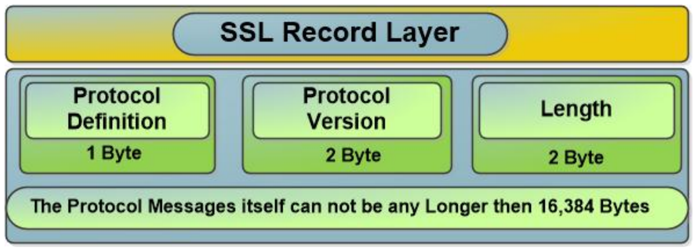
- Public keys which are knowable by everyone.
- Private keys known only by the person receiving the message.

The two keys work together to form an encrypted connection via the Web, it also uses a specific port for its secure connection, port 443.

The Sockets Layer (SSL) protocol uses one of the most commonly implemented stream ciphers called RC4. But, as good things go; it was also poorly implemented in the 802.11 WEP protocol, and was quickly hacked. The RC4 cipher was developed by Ron Rivest in the 80's and is a very simple and fast algorithm which is one of the reasons it became so popular and is still used today.

SSL is made up of four protocol layers which encapsulate the communication between the client and the server

1) Record Layer
2) Change CipherSpecProtocol
3) Alert Protocol
4) HandShake



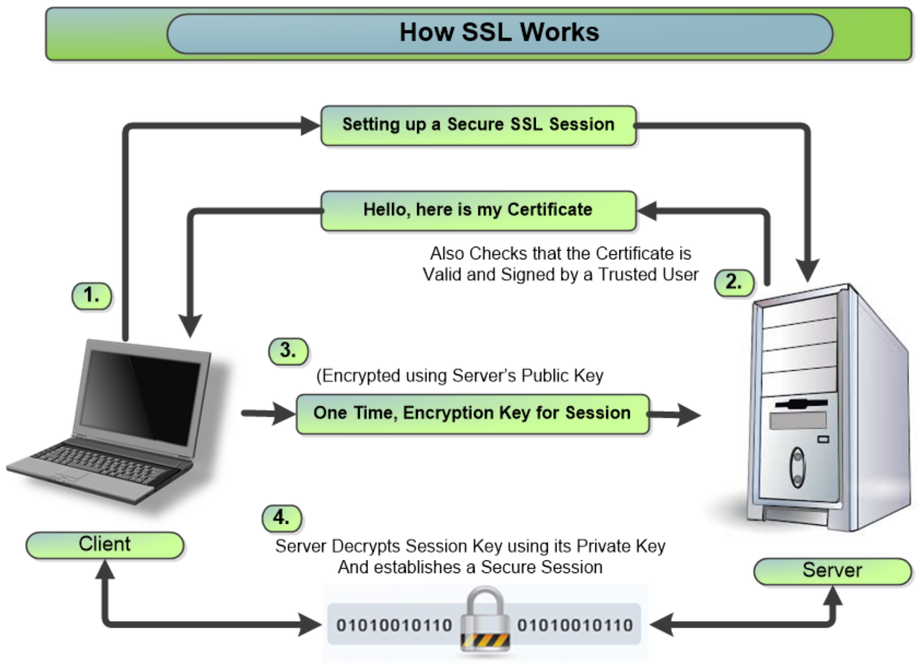
The record layer formats Alerts, ChangeChipherSpec, Handshake and application protocol messages. It’s comprised of five bytes.

Now for the most part the SSL protocol lies beneath the application layer and above the network layer. But others might put it in other areas of the OSI model, which can be confusing. This is because SSL is made up of two protocols: it works at the lower end of the session layer, and at the top end of the transport layer. In any case there are a number of reasons to use an HTTPS connection:

1. It helps to establish a secure communication between server and browser
2. It secures websites against tampering activities or eavesdropping
3. It protects users from man-in-the-middle attacks
4. It is used worldwide by business of all sizes to process secure payment transactions
5. It is used by banking, healthcare, e-commerce, social media and government industries

SSL is now considered insecure and its final released version was SSL version 3.0 must be replaced by all users as of June 2016.

The replacement for SSL is called Transport Layer Security (TLS), which I believe is up to version 2 and version 3 is close to release, as of this writing. Many thought there was very little difference between SSL 3.0 and TLS Version 1.2. But after the 2014 Oracle on Downgraded Legacy Encryption (POOBLE) attack happened in which it showed everyone the weaknesses within the older SSL protocol; it sealed SSL's downfall. This figure is an overview of just how the SSL worked in the past.


How SSL Works
Setting up a Secure SSL Session
Hello, here is my Certificate
Also Checks that the Certificate is Valid and Signed by a Trusted User
1.
2.
3.
(Encrypted using Server's Public Key
One Time, Encryption Key for Session
4.
Server Decrypts Session Key using its Private Key And establishes a Secure Session
Client
Server
01010010110
01010010110

In the past when SSL version 2 and later SSL version 3 came out there was still some effort in backwards compatibility, but that changed when TLS version 1.0 came out! So the differences between protocol SSL 3.0 and TLS 1.2 began to happen because of the flaws and dangers of being hack just because of its backward compatible. In 2015 because of the POODLE attacks network administrator began to update their web servers/browsers, and removing support for older versions of SSL within the code.

<u>**Here are the Steps that the SSL uses for a secure connection:**</u>

ChangeCipherSpec: layer signals the beginning of a secure connection.

Alert Protocol: Uses to fields, the Severity Level and the Alert Description which indicates the specific error that caused the problem.

The SSL handshake uses specific steps:

1) **ClientHello**: The client machine is requesting a secure communication session. It includes a set of options that the client is willing to use in order to communicate with the server.

2) **ServerHello:** Based on the ClientHello message the server makes a few choices. It returns five fields like the ClientHello message, but fills in SessionID, and picks the version of SSL to use. The Compression Method and CipherSuite protocols.

3) **ServerKeyExchange**: Now the method for transmitting the data is set by the server, information is passed to the client to determine how it will be encrypted. The server encrypts a separate session key for secure communications. Both Client and Server will use the key to transmit secure data. To ensure all parties are who they claim to be, a digital certificate is use to provide electronic identification.

4) **ServerHelloDone**: Once the Server has completed the ServerKeyExchange message. A ServerHelloDone is sent to the client to indicate that the server is through with its messages.

5) **ClientKeyExchange**: Since SSL does not require a client to have public and private keys in order to establish a SSL session, the ClientKeyExchange message contains information about the key that the client and server will use to communicate.

6) **ChangeCipherSpec**: The two ChangeCipherSpec messages signal the change of data transmission from an insecure state to a secure state. The final message within the SSL handshake looks at three things. Key Information, Contents of all previous SSL handshake messages. A special value that indicates if the sender is a client or server.

7) **Lock Icon**: appears in the browser indicating a secure protocol, which is used by the browser and the Web e-mail server.

Transport Layer Security Protocol (TLS)

The goal of TLS is cryptographic security, interoperability, extensibility, and relative efficiency.

TLS Record Protocol: Negotiates a private, reliable connection between the client and server. It uses a symmetric cryptography keys for its connection, and is secured through the use of a hash function generated by the Message Authentication code.

TLS Handshake Protocol: It uses the same protocol methods as SSL, and provides authentication for the server, and client, but several change have were made to enhance the handshake.

There are several main differences between SSL and TLS.

1) **Alert Protocol Message Types:** Additionally, several more descriptions have been added to bring the number of Alert Descriptions to 23 from 12

- CloseNotify; UnexpectedMessage; BadRecordMAC; DecryptionFailure; RecordOverflow; DecompressionFailure; HandshakeFailure; BadCertificate; UnsupportedCertificate; CertificateRevoked; CertificateExpired; CertificateUnknown; IllegalParameter; UnknownCA; AccessDenied; DecodeError; DecryptError; ExportRestriction; ProtocolVersion; InsufficientSecurity; InternalError; UserCancelled; NoRenegotiation.

2) **Message Authentication:** TLS implements a standardized MAC (H-MAC) The main benefit to this change is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.

3) **Key Material Generation:** TLS uses the HMAC standard and its pseudorandom function (PRF) output to generate key material. Each system starts out with a premaster secret; next it creates the master secret. Then it generates the required key material.

4) **SSL** uses RSA, Diffie-Hellman or Fortezza/DMS output to create key material. This output generates secret information based on the cipherSuite and Parameters selected during session negotiations.

5) **In SSL, the CertificateVerify** message requires a complex procedure of messages. In TLS the verified information is completely contained in the handshake messages previously exchanged during the session.

6) **Finished** in TLS, the PRF output of the H-MAC algorithm is used with the master secret and either a “client finished” or a “server finished” designation to create the Finished message.

7) **In SSL**, the finished message is created in the same ad-hoc manner that key material is generated: using a combination of hash output, selected ciphersuite and parameter information.SSL specifically supports RSA, Diffie-Hellman and Fortezza/DMS ciphersuites.

8) TLS has stopped allowing Fortezza/DLS support, but allows for ciphersuites to be added to the protocol in future revisions

Here is but a few of the other difference between SSL and TLS: HTTPS is the code-text that is written using a standard HTTPS format and secured with SSL/TLS to encrypt the HTTP text and ensure the communication is protected at all times.

- In the Client-Hello message (first message sent by the client, to initiate the handshake), the version is {3.0} for SSLv3, {3,1} for TLSv1.0 and {3,2} for TLSv1.1. The Client-Key-Exchange differs.
- The MAC/HMAC differs (TLS uses HMAC whereas SSL uses an earlier version of HMAC). The key derivation differs.
- The client application data can be sent straight after sending the SSL/TLS Finished message in SSLv3. In TLSv1, it must wait for the server's finishes the message.
- The list of cipher suites differ (and some of them have been renamed from SSL_* to TLS_*, keeping the same id number).
- There are also differences regarding the new re-negotiation extension.

In addition, TLS has two distinct layers:

- TLS Record Protocol establishes a secure connection with encryption methods like data encryption standard.
- TLS Handshake Protocol allows authentication for the servers and clients together. Before data can be exchanged, it has to convert cryptographic keys and algorithms.

But with any new technology there is a few problems one must be mindful of when it comes to TLS.

There was a problem with re-negotiation of extension if a certificate based client authentication is used, the server will see a stream of bytes where the initial bytes are protected but unauthenticated by TLS and subsequent bytes are authenticated by TLS and bound to the client's certificate. In some protocols (notably HTTPS), thus no distinction is made between pre-and post-authentication stages and the bytes are handled uniformly, resulting in the server believing that the initial traffic corresponds to the authenticated client identity.

This opened up a variety of attacks in which the attacker can convince the server to accept data from it as data from the client. These attacks can be prevented by cryptographically binding renegotiation handshakes to the enclosing TLS cryptographic parameters, thus allowing the server to differentiate renegotiation from initial negotiation, as well as preventing renegotiations from being spliced in between connections.

Now some protocols like IMAP and SMTP have explicit transitions between authenticated and unauthenticated phases thus requiring that the protocol state machine be partly or fully reset at such transitions. But there is no requirement's for state machine resets at TLS renegotiation, and thus there is still a window of vulnerability.

There is a number of ways TLS implementers can demand documentation that clearly lays out how renegotiation interacts with a developers API. If the certificate changes or let's say the server name changes for any reason once authentication has happened. The program has the option to abort the renegotiation. In addition, you will have to know how your other applications are using renegotiation because these apps could break if renegotiation within the app selects a different cipher that requires a different certificate. These are just a few of the problems you might encounter with implementing TLS. Below is some great videos on the subject of SSL and TLS.

Youtube on SSL

https://www.youtube.com/watch?v=Z7Wl2FW2TcA

Hash Functions

Hash function have a purpose in that they take long messages and generate a unique output value derived from the content of the message creating what is referred to as a message digest.

The creator of the message sends or transmits it to the recipient along with the full message.

1. The recipient then uses the hash function to re-compute the message digest from the full message. They can now compare the computed message digest to the transmitted one. This ensures that the message sent came from the sender, originator of the message.
2. The message digest can be used to create a digital signature algorithm.

There are 5 requirements for the creation of a Hash Function

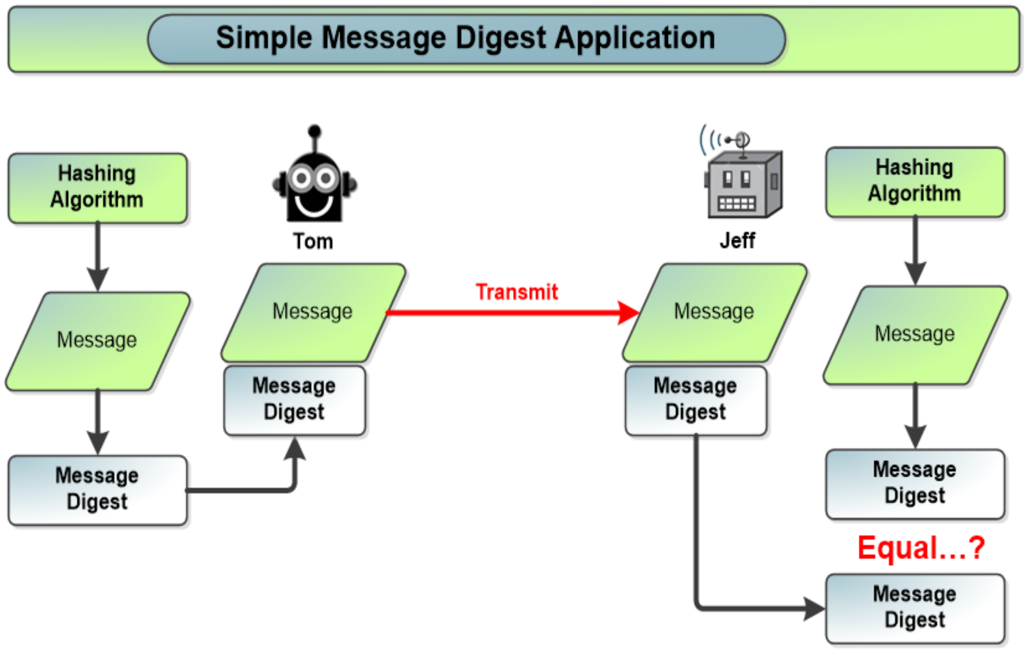
1) Input can be of any length

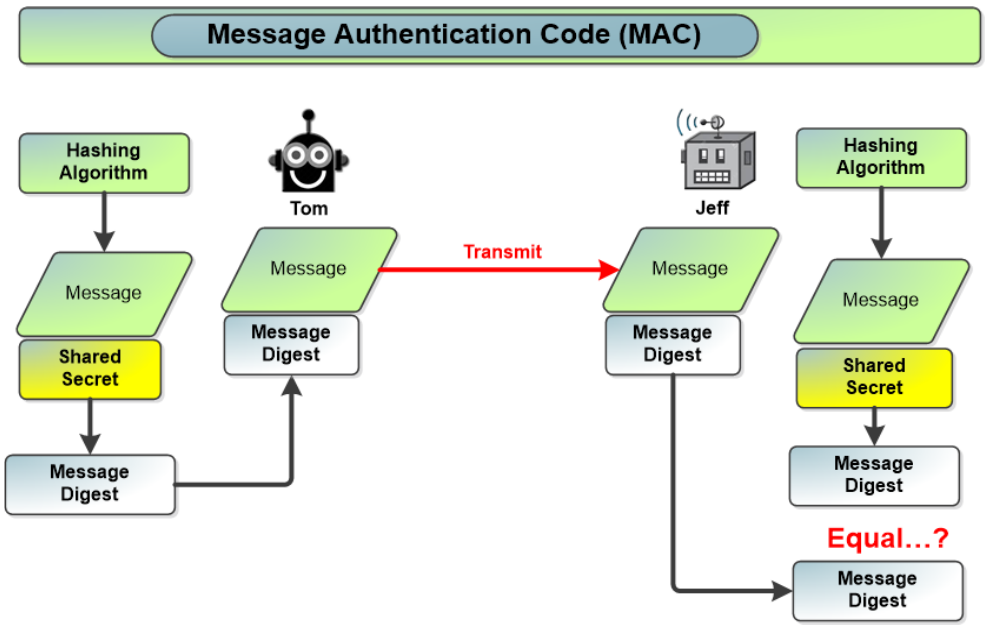
2) Output has a fixed length

3) The Hash function is easy to compute input

4) A Hash function is one-way; meaning hard to determine the input value.

5) Collision Free; Hard to find two messages that produce the same hash value.





There are many types of hashing algorithms for example: SHA, MD2, MD4, MD5, HMAC, and HAVAL to name a few of the top ones.

The SHA-1 and SHA-2 are government standard hash functions.

Since SHA-1 had some major weakness discovered within its algorithm. This lead to the creation of SHA-2 which has a number of variations and block sizes added for security. Below are some of the block sizes used within the SHA family of algorithms.

1. SHA-224, 224-bit message digest using 512-bit block size
2. SHA-256, 256-bit message digest using 512-bit block size
3. SHA-512, 512-bit message digest using a 1,024-bit block size
4. SHA-384, 384-bit digest, with 1,024-bit block size

There is a SHA-3 version coming out given that SHA-2 had some of the same weakness as in SHA-1, but it's still under development.

SHA - SHA was developed by the U.S. Government. It produces a 160-bit hash value or message digest. SHA is similar to MD4 with some added mathematical functions, but was found to be vulnerable to collisions. So SHA-2 and SHA-3 versions were later developed to fix the problems with applications that required collision resistance.

Here is a list of some other Hashing methods, HMACs, CBC-MACs, CMACs

There differences, and Steps to create.

Function	Steps	Security Method
HASH	1) Sender creates hashing algorithm and generates a Message Digest 2) Sender sends message and Message Digest 3) Receiver runs message through the hashing algorithm to create his own Message Digest 4) Receiver compares his Message Digest to see if it matches senders Message Digest	Hash only offers Integrity, does not offer confidentiality or authentication to the user
HMAC	1) Sender concatenates a message with a secret key. Creates the hashing algorithm that creates a MAC value. 2) Sender Appends MAC value to message and sends to receiver 3) Receiver takes message and concatenates it with his/her own symmetric key creating an independent MAC value 4) Receiver compares the two MAC values. If the two MAC values are the same. The message has not been modified.	Provides Integrity and data origin authentication, but no confidentiality is provided with HMAC.
CBC-MAC	1) Sender encrypts a message with a symmetric block algorithm in CBC mode. 2) Last block is used as the MAC 3) The Plaintext message and the appended MAC are sent to receiver. 4) Receiver encrypts the message, creates a new MAC, compares the two values and if the same. The receiver knows the message was not modified.	No confidentiality is provided. But Integrity and data authentication is provided.
CMAC	CMAC works like CBC-MAC, but the mathematical logic and far more complex	

MD4

Ron Rivest designed the MD4 and it produces a 128-bit message digest, block size of 512-bit with three rounds of encryption. It is no longer considered secure because it's particularly vulnerable to collisions. It can also be cracked by a brute-force attack technique.

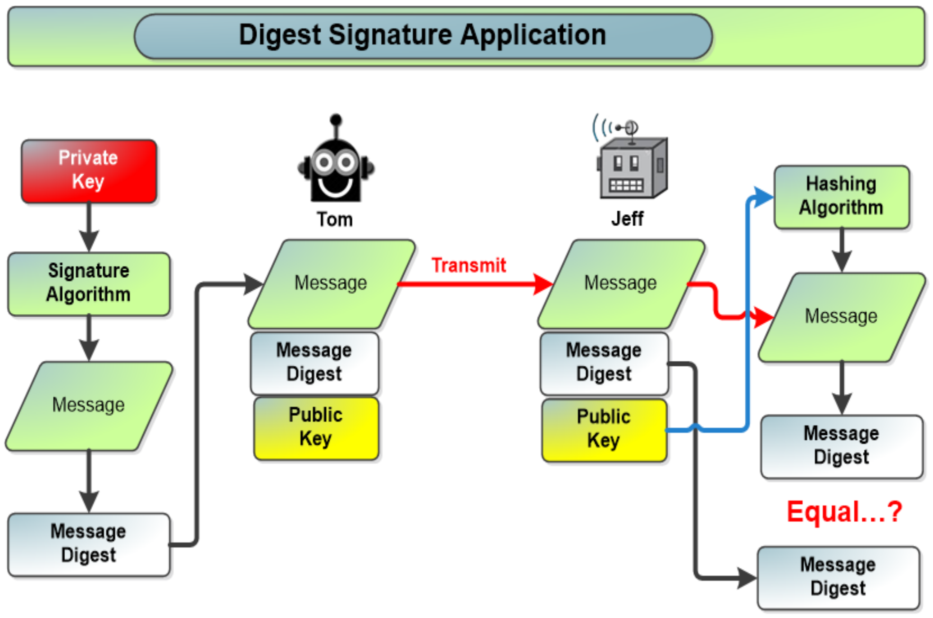
MD5

This is the newest addition of the algorithm created by Ron Rivest. It still uses a 128-bit hash, with a block size of 512-bits but is far more complex. Some of the differences between the two versions of the algorithm are that in MD5 has an added fourth round of encryption, it also includes a unique additive constant on each step. It promotes a faster avalanche effect by adding the results from the previous step. In addition, each round the input word pattern in rounds 2 and 3 are scrambled making it harder to see patterns.

Hashing Algorithms Overview	
Algorithm	**Description**
Message Digest 4 (MD4)	128-bit hash value, 512-bits, 3 rounds
Message Digest 5 (MD5)	128-bit hash value, 512-bits, 4 rounds
Secure Hash Algorithm (SHA)	160-bit hash value, used with Digital signature Algorithm (DSA)
SHA-1, SHA-256, SHA-384, SHA-512	160-bit hash value, SHA-256, 384, 512..etc

Digital Signatures

A digital signature is a hash value that has been encrypted with the sender's private key. Signing a message means encrypting the message's hash value with a private key, as shown below.



"The One-Way Hash," if Tom wants to ensure that the message he sends to Jeff is not modified and he wants to be sure it came only from him he can digitally sign the message. This means that a one-way hashing function would be run on the message, and then Tom would encrypt that hash value with his private key

Documents relating to HASH function and algorithms put out by NIST

National Institute of Standards and Technology | NIST

- NIST FIPS 202
- NIST FIPS 180-4
- NIST SP 800-57
- NIST SP 800-107

Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) enables a large number of people to communicate securely. It combines a number of key platforms, from procedures, security policies, encryption mechanisms, working together creating a level of trust within the structure based on X.509 standards. It provides people on different networks and over the internet authentication, confidentiality, non-repudiation, and integrity. Moreover, it's a system hybrid using symmetric and asymmetric key algorithms within its protocols.

PKI Components
A certification authority (CA) is like a notary republic. They confirm the identities of all communicating parties. In a word they make sure you are, and who you say you are for identification purposes.

Registration Authority (RA) is trusted by the CA to register and create vouchers for the identity of users to a CA.

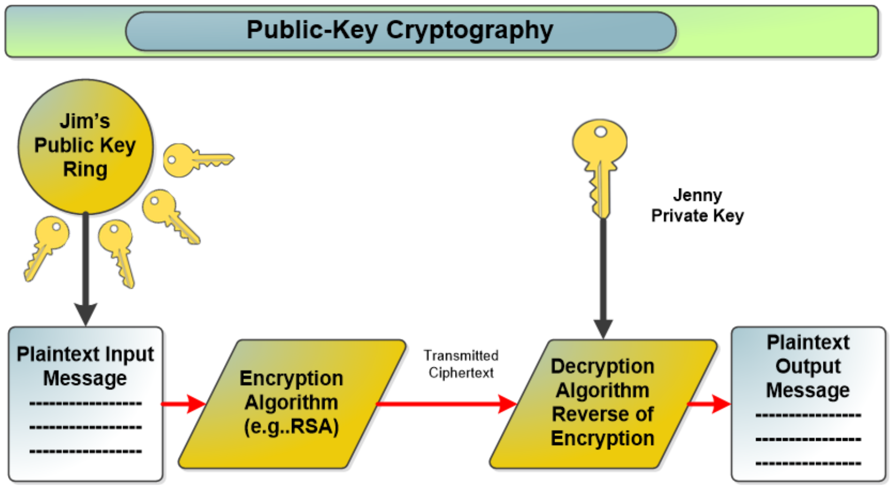
Repository is a database of active digital certificates for a CA system. It provides data that allows users to confirm the status of digital certificates. The CA post and updates certificates to the certificate revocation lists (CRL) in the repositories.

Archive keeps information about certificates for any future disputes, and other information that might be needed for history reasons.

The CA issues the public key certificate, confirming the identity has the correct credentials. The certificate includes public key, identity information of holder to the corresponding private key, dates/times of effective periods, CA digital signature. In addition, it may include other information outlined in the table below. A subscriber is an individual/business entity that has contracted with a CA to receive a digital certificate.

Moreover, a digital certificate allows the user to communication across many different types of networks including the Internet. The PKI supports authentication, confidentiality, non-repudiation, and integrity within the message exchange. In enables symmetric and asymmetric key algorithms making it a hybrid framework of communication techniques.

In other words, PKI is an infrastructure and not an algorithm.



Each person within the PKI is required to obtain a digital certificate from the certificate authorities (CA). This certificate contains the public key along with identifying information. The certificate has to be created and signed by a trusted third party.

The CA validates all certificates issued, and acts as the trusted third partner. The CA establishes a trusted middleman relationship between all users as if each individual certificate was issued by them.

Certificates

The digital certificate is the most important pieces of a PKI since the certificate is the mechanism that associates a public key with a claimed owner. The X.509 is the standard used for the creation of the certificate and it outlines the fields and values that are required in order to be valid.

The Registration Authority

The registration duties for a CA are carried out by the registration authority (RA). The RA confirms the identity of the individual, and initiates the certificate process with the CA on behalf of the user. In a word the RA is the broker between the CA and the user. In addition, if the user requires a new certificates for some reason. The request goes to the RA, and the RA verifies all identification information before sending the request onto the CA.

Certificate Structure

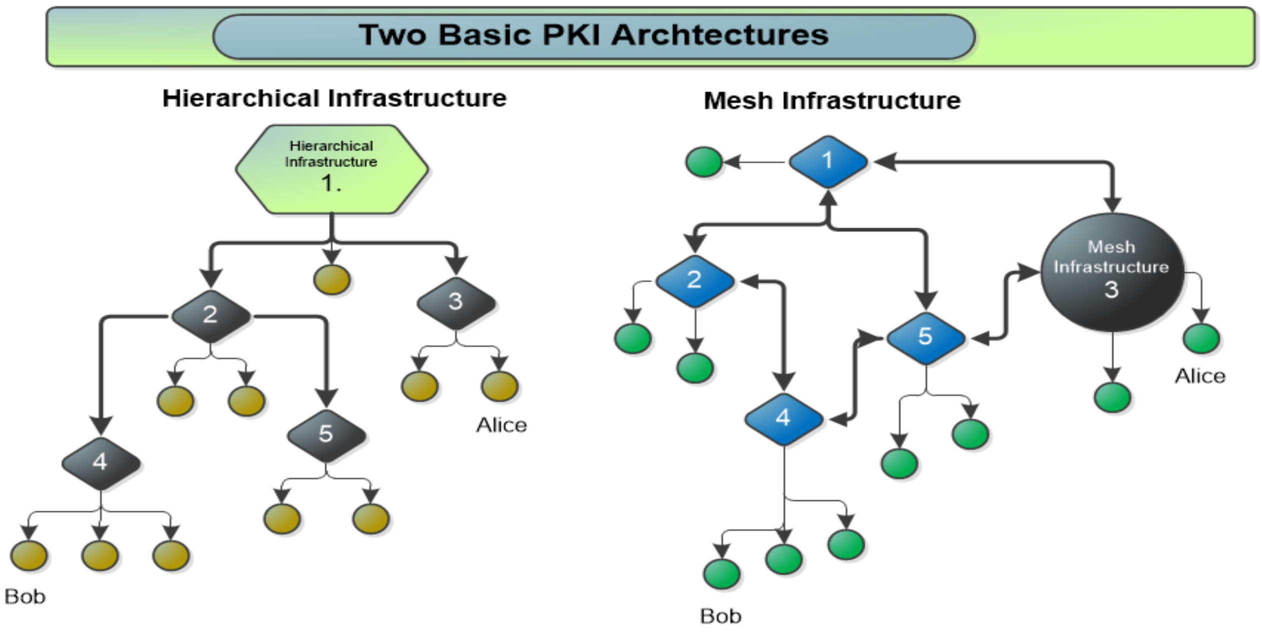
Version	Serial Number	Signature	Issuer	Validity	Subject	Subject Public Key Info	Issuer Unique ID	Subject Unique ID	Extensions
Identifies version of Certificate	Unique ID	Algorithm ID used to Sign Certificate	Name of Certificate Issuer	Validity Dates	Name of Owner	Public Key of Owner	ID of Issuing CA	ID of Subject	Optional Extensions

Two Basic PKI Architectures

Hierarchical: Authorities are arranged under a "root" CA that issues certificates to subordinate CAs. These CAs may issue certificates to CAs below them or to users. In a hierarchical PKI, every relying party knows the public key of the root CA. Any certificate may be verified by verifying the certification path of certificates from the root CA. Example: Alice verifies Bob's certificate, issued by CA 4, then CA 4's certificate, issued by CA 2, and then CA 2's certificate issued by CA 1, the root, whose public key she knows.

Mesh: In this type of configuration each CA's issue certificates cross certifying each other. This creates a mesh of trust relationships between peer CAs. A relying party knows the public key of a CA near himself, normally one that issued his certificate so each party within the path is in a since a forming a cross-Certificate-Pair. Example; Alice knows the public key of CA 3, while Bob knows the public key of CA 4. There are several certification paths that lead from Bob to Alice. The shortest requires Alice to verify Bob's certificate, issued by CA 4, then CA 4's certificate issued by CA 5 and finally CA 5's certificate, issued by CA 3. CA 3 is Alice's CA and she trusts CA 3 and knows its public key cross certify each other.

Graphical Overview of the two basic PKI Structures:


Two Basic PKI Archtectures
Hierarchical Infrastructure
Hierarchical Infrastructure 1.
2
3
5
4
Alice
Bob
Mesh Infrastructure
1
2
Mesh Infrastructure 3
5
4
Alice
Bob

Trusted Platform Module
The Trusted Computing Group (TCG) is the organization that devised the standards to help strengthen computing platforms. The TPM is a microchip dedicated to carrying out security functions and is installed on the motherboard. It carries out storage and processing of keys, hashes, digital certificates using symmetric and asymmetric algorithm methods.

The TPM organization promotes open standards that help strength computing platforms. Its microcontroller security chip stores keys, passwords, and digital certificates data. This hardware based security platform significantly improves the Root-of-Trust within computing systems. TPM binds to a hard drive in a way where the content is encrypted to a particular computer systems hard-drive. The key to the TPM chip is stored in another location making access to the information basically inaccessible. The content will be rendered useless if the key is not backed up and escrowed in a safe place. There are some great resources on PKI at the NIST web site, or do a search for these articles below.

NIST (National Institute of Standards and Technology)

http://csrc.nist.gov/groups/ST/crypto_apps_infra/pki/

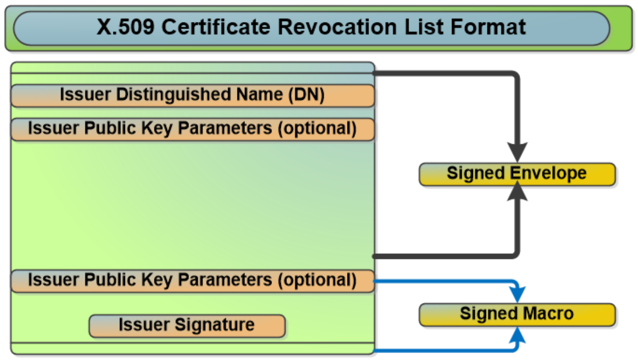
NIST Special Publication 800-57
NIST Special Publication 800-32

X.509 Standard

X.509 is a standard published by the **International Telecommunication Union** **(ITU)** that specifies the standard format for digital certificates use in combination with creating a Public Key Infrastructure. It in since it outlines a series of conceptual, legal and directory objectives providing users with a means of resolving certification issues. The main focus of a PKI is to provide confidentiality, integrity, access control, authentication, and most importantly, non-repudiation.

Now the X.509 does have many good features that aid in securing communication between users, but it does have its problems with implementation flaws and other issues which I will outline. The Internet as you know is an open connectionless system where identities who want to communicate with each other face the risk of active interference from any number of methods one can use to eavesdropping on your interaction.

It is true that public-key cryptography has solved many problems, but not all when it comes to the question of public-key acquisition, recognition, revocation lists, distribution, and re-distribution of validation. Moreover, key-binding to an identifier, and or key-attribution to a real-world entity bags the question, “is that key from the sender, and is that key still valid?”


X.509 Certificate Revocation List Format
Issuer Distinguished Name (DN)
Issuer Public Key Parameters (optional)
Signed Envelope
Issuer Public Key Parameters (optional)
Signed Macro
Issuer Signature

Take the case of binding a key with a common name or identifier because clearly you could just be having a private communication with a thief given if that thief was creative enough to steal, or create a false certificate. The certification process does introduce a tamper-proof attribute for key-binding to an identifier and key-attributions. But the big but you never really know given past cases of people creating or stealing certificates and using them to access systems.

PKI does provide **key exchange** functionality that facilitates a secure exchange of public keys such that the authenticity of the parties can be verified. There are just a few things one has to keep in mind during and after creating the keys one plans on using with their PKI.

Here are a few key essentials to follow.

Key management concerning public and private keys:

1. Secure generation of keys—Ensures that private and public keys are generated in a secure manner.
2. Secure storage of keys—Ensures that keys are stored securely.
3. Secure distribution of keys—Ensures that keys are not lost or modified during distribution.
4. Secure destruction of keys—Ensures that keys are destroyed completely once the useful life of the key is over.

In regards to security issues within the X.509 framework:

There is question regarding the contents of certificates and their issuance conditions for example. The naming scheme, authentication relies on each user possessing a unique distinguished name. But how are these distinguished name (DN) assigned? Are they really unique? The DN is created by the Naming Authority (NA) and accepted by the Certificate Authority (CA). But users can have different DN's in different CA's, or in some cases the same DN in different CAs?

Moreover, regarding the validation procedures for the certified data and a user's identity within the X.509 rules state, **"A certification authority shall be satisfied of the identity of a user before creating a certificate for it."** Thus meaning the identity validation procedures can be satisfied by the CA's own self-defined rules, which can be different from one CA to another. In addition, Certification Practice Statements (CPS) accepts indirect reference when issuing certificates, using an ID as identifying proof, which can be subject to fraud. The X.509 needs to focus on creating a better way of securing third-party certificates which is something it has not addressed in its management structure yet.

Moreover, the purpose of a CA is to bind the public key to the name in the certificate, but how are these "name/key" validated? Do these credentials correspond to a person, but it turns out this is outside the scope of the X.509 and depends on each CA to self-defined CPA and on each NA.

This is just a few of the things I found in my research of the PKI structure and how the NA's and CA's work together to create a secure platform to communicate.

Types of keys within Key Management

NIST Special Publication 800-57 titled Recommendation for Key Management

1. Private signature key: It is a private key of public key pairs and is used to generate digital signatures. It is also used to provide authentication, integrity, and non-repudiation.
2. Public signature verification key: It's the public key of the asymmetric (public) key pair. It is used to verify the digital signature.
3. Symmetric authentication key: It is used with symmetric key algorithms to provide assurance of the integrity and source of the messages.
4. Private authentication key: It is the private key of the asymmetric (public) key pair. It is used to provide assurance of the integrity of information.
5. Public authentication key: Public key of an asymmetric (public) pair that is used to determine the integrity of information and to authenticate the identity of entities.
6. Symmetric data encryption key: It is used to apply confidentiality protection to information.
7. Symmetric key wrapping key: It is a key-encrypting key that is used to encrypt the other symmetric keys.
8. Symmetric and asymmetric random number generation keys: They are used to generate random numbers.
9. Symmetric master key: It is a master key that is used to derive other symmetric keys.
10. Private transport key: They are the private keys of asymmetric (public) key pairs, which are used to decrypt keys that have been encrypted with the associated public key.
11. Public key transport key: They are the public keys of asymmetric (public) key pairs that are used to decrypt keys that have been encrypted with the associated public key.
12. Symmetric agreement key: It is used to establish keys such as key wrapping keys and data encryption keys using a symmetric key agreement algorithm.
13. Private static key agreement key: It is a private key of asymmetric (public) key pairs that is used to establish keys such as key wrapping keys and data encryption keys.
14. Public static key agreement key: It is a public key of asymmetric (public) key pairs that is used to establish keys such as key wrapping keys and data encryption keys.
15. Private ephemeral key agreement key: It is a private key of asymmetric (public) key pairs used only once to establish one or more keys such as key wrapping keys and data encryption keys.
16. Public ephemeral key agreement key: It is a public key of asymmetric (public) key pairs that is used in a single key establishment transaction to establish one or more keys.
17. Symmetric authorization key: This key is used to provide privileges to an entity using symmetric cryptographic method.

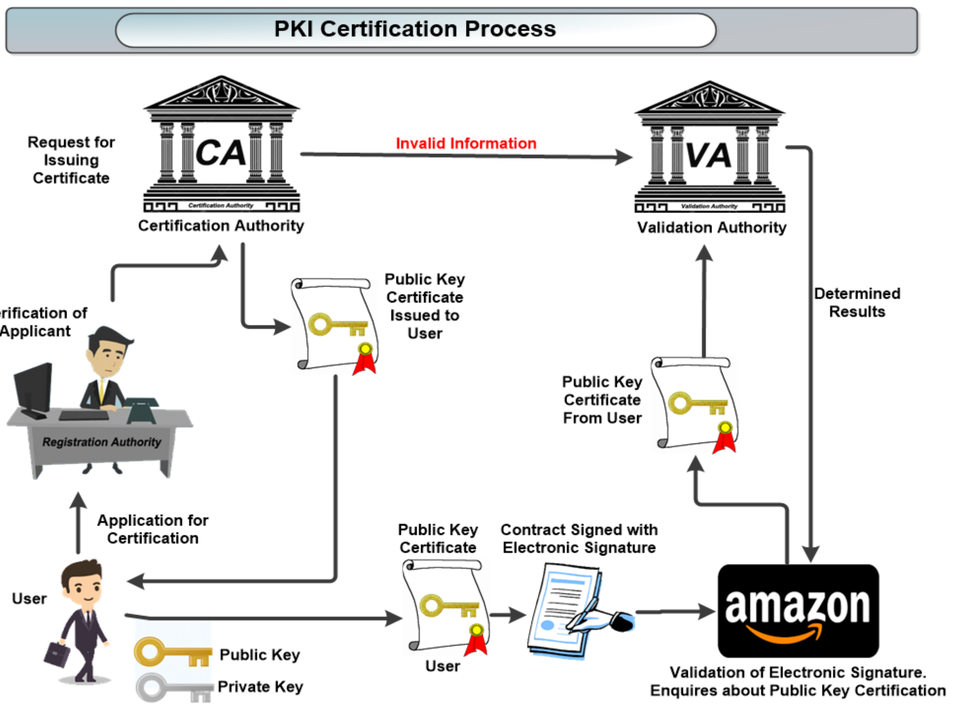
18. Private authorization key: It is a private key of an asymmetric (public) key pair that is used to provide privileges to an entity.
19. Public authorization key: It is a public key of an asymmetric (public) key pair that is used to verify privileges for an entity that knows the associated private authorization key.

Assurance requirements within the key management process:

- **Integrity protection**—Assuring the source and format of the keying material by verification
- **Domain parameter validity**—Assuring parameters used by some public key algorithms during the generation of key pairs and digital signatures, and the generation of shared secrets that are subsequently used to derive keying material
- **Public key validity**—Assuring that the public key is arithmetically correct
- **Private key possession**—Assuring that the possession of the private key is obtained before using the public key

A key goes through six key states outlined by the NIST SP800-57 document.

- Pre-activation state—The key has been generated, but not yet authorized for use
- Active state—The key may used to cryptographically protect information
- Deactivated state—The crypto-period of the key is expired, but the key is still needed to perform cryptographic operations
- Destroyed state—The key is destroyed
- Compromised state—The key is released or determined by an unauthorized entity
- Destroyed compromised state—The key is destroyed after a compromise or the comprise is found after the key is destroyed


PKI Certification Process
Request for Issuing Certificate
CA
Certification Authority
Invalid Information
VA
Validation Authority
Public Key Certificate Issued to User
Determined Results
Verification of Applicant
Registration Authority
Public Key Certificate From User
Application for Certification
Public Key Certificate
Contract Signed with Electronic Signature
User
User
Public Key
Private Key
amazon
Validation of Electronic Signature. Enquires about Public Key Certification

<u>Types of other Cyber-Attacks on PKI, and X.509</u>

1. **Ciphertext only attack**: This type of attack refers to the availability of the ciphertext (encrypted text) to the cryptanalyst. With large ciphertext data, it may be possible to decipher the ciphertext by analyzing the pattern.
2. **Known-plaintext attack**: This type of attack happens when a cryptanalyst obtains a ciphertext as well as the corresponding plaintext. In this scenario, even if the data is small, it is possible to understand the algorithm.
3. **Chosen-plaintext attack**: This type of attack refers to the availability of a corresponding ciphertext to the block of plaintext chosen by the analyst.
4. **Adaptive-chosen-plaintext attack**: This type of cryptanalytic attack is known as an adaptive-chosen-plaintext attack if the cryptanalyst can choose the samples of the plaintext based on the results of previous encryptions in a dynamic passion.
5. **Chosen-ciphertext attack**: This type of attack is used to obtain the plaintext by choosing a sample of ciphertext by the cryptanalyst.
6. **Adaptive-chosen-ciphertext attack**: This type of attack is similar to the chosen-ciphertext attack, but the samples of ciphertext are dynamically selected by the cryptanalyst and the selection can be based on the previous results as well.
7. **Spoofing Chaining**: In a Spoof Chain operation it tries to obfuscate false data by giving it a shroud of credibility based on secondary steps that may not be perceived as insecure by a third person. Example: obtain a true birth certificate of a deceased person, create a faking mailing address. In addition, to creating secondary ID's like library cards, SSN, and school ID's.

For more information take a look at the Federal Information Processing Standard (**FIPS-140),** which is part of the NIST series of 140 Publication requirements and standards for Crytographic Modules

<u>The core structure of FIPS-140 recommends four security levels for cryptographic modules for Federal Systems</u>.

1. **FIPS140 Security Level 1**—It is the basic or lowest level of security that prescribes basic security requirements for a cryptographic module.
2. **FIPS140 Security Level 2**—Tamper evidence mechanisms is a requirement in this level. This enhances the physical security of the device. Tamper-evident seals or coatings should be used to physically protect the device or storage that contains the cryptographic module.
3. **FIPS140 Security Level 3**—The primary requirement is preventing an intruder from gaining access to the cryptographic modules and the Critical Security Parameters (CSP) contained within.
4. **FIPS140 Security Level 4**—This is the highest level and the physical security mechanisms. A complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access is provided. This level requires a two-factor authentication. This level also requires the control of environmental conditions such as preventing damage to cryptographic modules due to temperature, heat, and voltage.

This part covers some of the important concepts relating to PKI and X.509 standard's which also cover's FIPS140 cryptographic modules.

Questions one can ask in relationship to the X.509 for further study.

Almost all issues involving the CA's Certification Practice Statement (CPA) is about trust. The CPA is the governing law that the CA presents to clients, but each CA can have their own rules designed for different needs. It can lead to a "laissez faire" attitude leaving ample room for interpretations of X.509 implementations.

For example take the Secure Socket Layer (SSL) and version 3.0 with an IETP equivalent specification being developed as TSL. Netscape's SSL could be accessible to this platform, but not to Microsoft's or for that case RSA or some another vendor? Then you take the case of cross-certification in a global internet, when different CA's need to scale within different systems, and businesses. This could lead to some major security problems and trust relationships if they are not address and as of today no one has forced the issue.

Here are a few thoughts about implementation flaws and processes when it comes to PKI and X.509 structures for CA's and their Certification Practice Statement (CPS).

- The server defines which certificates can be accepted, not the user….good/bad depends?
- Certificates can have short life spans from as short as 4 weeks. However, they are usually issued with a one-year lifetime
- CA root certificates tend to be issued for lifetimes, or up to twenty-years(Canada)
- Certificates can be compromised by chain events outside the control of the user or subscriber.
- Verification of data on the subscriber is limited if a user had to check. This could be because of privacy rights, which is your basic security vs. privacy paradox.
- Certificate Revocation list is no assurance that all certificate copies will be revoked for a given key.
- Client and S/MIME certificates are issued using insecure on-line protocols in a browser, and you know how unsecure Netscape to Explorer can be given the version you're on?
- Certificate Authorities is a misnomer. For the most part a CA is a self-appointed issuer and a "certificate" does not convey any authorization.
- Moreover, the certificate which have no relationship to the data supplied by the subscriber but which are necessary for the proper use of the certificate, as a secure transport for information in the X.509 model: Example, Serial Number, Data of issuance, validity, the CA's signature...etc. In addition, the Certification Practice Statement (CPS) has a disclaimer in case of fraud, computer viruses and warranty limits, which can further jeopardize the validity of the certificate.

Now I have only touched on some of the issues relating to Certificates. But let's talk about auditing? The ways that CA's are put together in practice are for the most part self-regulated and bags to the question of trust management. As outlined by many expert in the field of digital certification have a tested too is that policies rules within a Certification Practice Statement (CPS) are self-made by each CA, and are so designed not to be audited. So there clearly needs to be a defined standard for CA practices, but as of yet none has been implemented.

Trust is an interesting thing when it comes to Third Party or even a trusted CA; just remember?

- Trusted in relationship to whom?
- Trusted by whom?
- Trusted for what?
- Trusted for how long?

The question of the PKI and its structure is a very big topic and I have only touch on some of the key issues when it comes to security and who, what and how are PKI's set up.

Here is a few options for further reading on the subject of PKI's

National Institute of Science & Technology – Document: **NIST.SP.800-57**

Public Key Infrastructure (PKI) Wikipedia
https://www.saylor.org/site/wp-content/uploads/2012/07/Public-key-infrastructure1.pdf

RSA Data Security Understanding Public Key Infrastructure
ftp://ftp.rsa.com/pub/pdfs/understanding_pki.pdf

How to avoid the Breakdown of Public Key Infrastructures
https://huelsing.files.wordpress.com/2013/05/paper.pdf

X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP
https://maikel.pro/published/crls-ocsp-stapling.pdf

10. Cracking and Hacking code: Frequency Analysis, Statistics, Finding Patterns and Identifying Weaknesses.

Here is a list of some of the basic techniques used by hackers.

- **Dictionary attack***:* This attack uses a file that contains a list of words that are found in the dictionary or put together based on a profile of the person or business your going after.

- **Brute force attack***:* Apart from the dictionary words, brute force attack makes use of non-dictionary words too. Some of the newest methods include using none standard word groups and unique symbols. Example: Pa66w00$ for passwords.
- **Rainbow table attack***:* This attack comes along with pre-computed hashes or you can create your own based on the target your after

Types of Attacks on Cryptography

- **Eavesdropping**
  Eavesdropping and sniffing data as it passes over a network are considered *passive attacks* because the attacker is not affecting the protocol, algorithm, key, message, or any parts of the encryption system. Passive attacks are hard to detect, so in most cases methods are put in place to try to prevent them rather than to detect and stop them.
- **Ciphertext-Only Attacks**
  In this type of attack, the attacker has the ciphertext of several messages. Each of the messages has been encrypted using the same encryption algorithm. The attacker's goal is to discover the key used in the encryption process. Once the attacker figures out the key, she can decrypt all other messages encrypted with the same key. A *ciphertext-only attack* is the most common type of active attack because it is very easy to get ciphertext by sniffing someone's traffic, but it is the hardest attack to actually be successful at because the attacker has so little information about the encryption process.
- **Differential Cryptanalysis**
  This type of attack also has the goal of uncovering the key that was used for encryption purposes. This attack looks at ciphertext pairs generated by encryption of plaintext pairs with specific differences and analyzes the effect and result of those differences. One such attack was invented in 1990 as an attack against DES, and it turned out to be an effective and successful attack against DES and other block algorithms
- The attacker takes two messages of plaintext and follows the changes that take place to the blocks as they go through the different S-boxes. (Each message is being encrypted with the same key.) The differences identified in the resulting ciphertext values are used to map probability values to different possible key values.
- **Linear Cryptanalysis**
  Linear cryptanalysis is another type of attack that carries out functions to identify the highest probability of a specific key employed during the encryption process using a block algorithm. The attacker carries out a known-plaintext attack on several different messages encrypted with the same key. The more messages the attacker can use and put through this type of attack, the higher the confidence level in the probability of a specific key value.
- **Side-Channel Attacks**
  All of the attacks we have covered thus far have been based mainly on the mathematics of cryptography. Using plaintext and ciphertext involves high-powered mathematical

tools that are needed to uncover the key used in the encryption process. In all in a side-channel attack is any type of attack based on information gained from the physical implementation of a cryptosystem, rather than brute-force or theoretical attack.

- **Replay Attacks**
  One of the biggest concerns in distributed environments is the replay attack, in which an attacker captures some type of data and resubmits it with the hopes of fooling the receiving device into thinking it is legitimate information. Many times, the data captured and resubmitted is authentication information, and the attacker is trying to authenticate themselves as someone else to gain unauthorized access.
- **Timestamps**
  Timestamps and sequence numbers are two countermeasures to replay attacks. Packets can contain sequence numbers, so each machine will expect a specific number on each receiving packet.
- **Algebraic Attacks**
  Algebraic attacks analyze the vulnerabilities in the mathematics used within the algorithm and exploit the intrinsic algebraic structure. For instance, attacks on the "textbook" version of the RSA cryptosystem exploit properties of the algorithm, such as the fact that the encryption of a raw "0" message is "0."
- **Analytic Attacks**
  Analytic attacks identify algorithm structural weaknesses or flaws, as opposed to brute-force attacks, which simply exhaust all possibilities without respect to the specific properties of the algorithm. Examples include the Double DES attack and RSA factoring attack.
- **Statistical Attacks**
  Statistical attacks identify statistical weaknesses in algorithm design for exploitation—for example, if statistical patterns are identified, as in the number of zeros compared to the number of ones.
- **Social Engineering Attacks**
  Attackers can trick people into providing their cryptographic key material through various social engineering attack types. Social engineering attacks are carried out on people with the goal of tricking them into divulging some type of sensitive information that can be used by the attacker.
- **Meet-in-the-Middle Attacks**
  This term refers to a mathematical analysis used to try and break a math problem from both ends. It is a technique that works on the forward mapping of a function and the inverse of the second function at the same time.

Malware Viruses and Other ways to hack.

“There is a saying if you can’t break it, infect it…!” What better way to cripple a system then with a Malware virus. They can be written in many different platforms from Visual Basic, Dot Net, Java, JScript to Python. Some viruses go after the boot sector, others the Operating System.

- **Stealth** virus hides its modifications to the systems in files or boot records. Stealth virus will hide its tracks after infecting a system, and even show the original file size of a file it’s infected.
- **Polymorphic** virus makes copies of itself so if one copy is found others can still be activated. It can use different encryption schemes to fool the antivirus software by using bogus instructions, and added noise features to confuse the antivirus software. Polymorphic virus can even change their own code to a variant the antivirus software will not detect it.
- **Multipart** viruses uses several components to its structure that can be distributed to different parts of the system; for example put code in the boot sector of the hard drive or added as part of an executable file.
- **Meme** viruses are a sort of social engineering viruses because they are spread not by computers, but by users through e-mail hoaxes, religious messages, or pyramid selling schemes. People will forward them for any number of reasons like fear, fun or profit. They can do harm like take up bandwidth and clog up e-mail servers.
- **Script** viruses are files that are executed by the system itself, either by Microsoft Windows Script Host or some other event that triggers it to activate. One of the most used methods is to embed the script within a web browser and the minute you click on something it activates.

Malware Components have six main elements

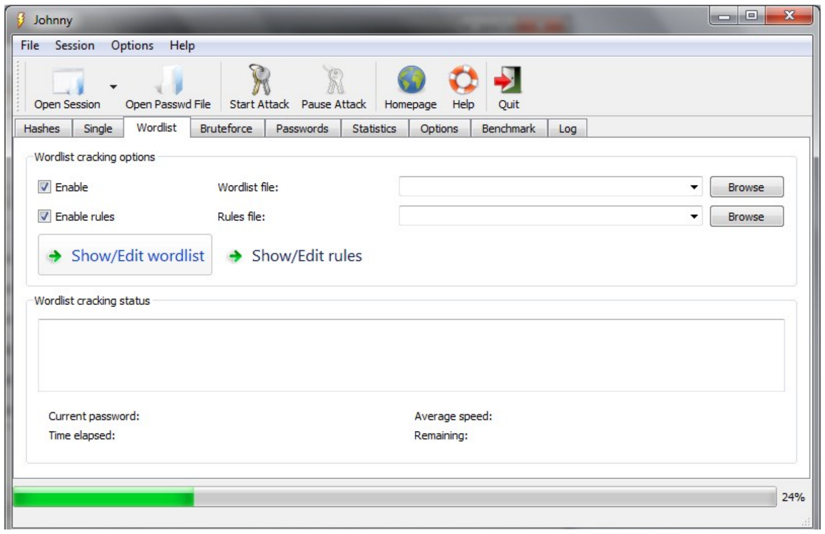
• **Insertion** Installs itself on the victim’s system
• **Avoidance** Uses methods to avoid being detected
• **Eradication** Removes itself after the payload has been executed
• **Replication** Makes copies of itself and spreads to other victims
• **Trigger** Uses an event to initiate its payload execution
• **Payload** Carries out its function (Example: deletes files, installing back doors, or some other vulnerability to a system)

Worms are different type of digital animal. They can reproduce on their own without a host application. In a sense it’s a self-sufficient program that does not require a host environment in order to carry out its activities. One of the most famous computer worms is the Stuxnet which targeted Siemens supervisory control and data acquisition (SCADA) software.

RootKits - A rootkit is a bundle of tools that once installed give the hacker complete control over a system. In a word he/she is now the administrator of the system. Most rootkit tools replace the default system tools called "Trojaned Programs." These programs maliciously carry out activities in the background and are not listed in a standard service request. These kits also include network sniffers to capture data and put the Network Interface Card (NIC) into promiscuous mode so it can listen to all the traffic on the network. It also has an ipconfig program which helps it hide the fact that the NIC is in promiscuous mode.

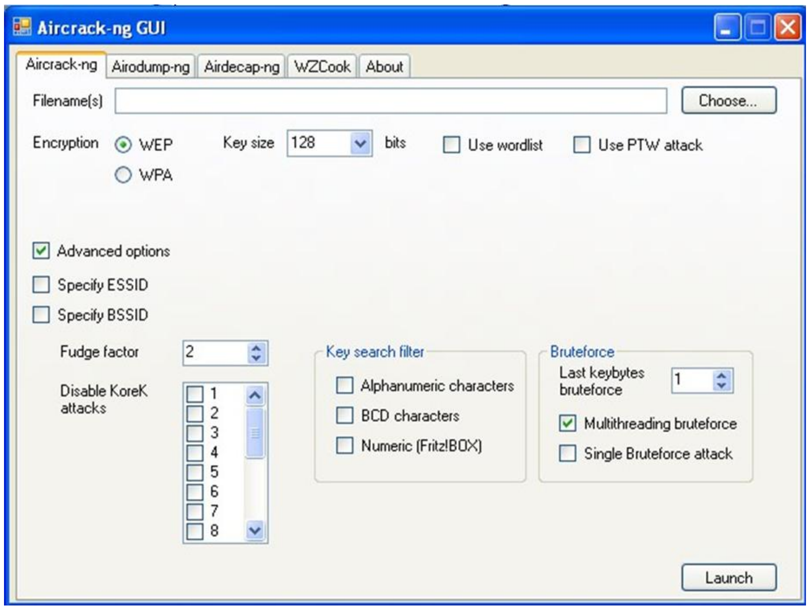
10 of the best Cracking/Hacking Tools Of 2016

John the Ripper - written in C programming language, and encompassing a customizable password cracker; it has the ability to autodetect password hashtypes and is a favorite of hackers and ethical hackers to ensure security. John the Ripper belongs to a family of tools by Raspid7.



The Tool works on all major platforms including Linux, Windows, DOS, and OS X

Aircrack-ng – One of the best cracking tools for hackers and has a number of other tools one can use with its suite of software. It’s also included within Kali..aka (Wireshark) and is great tool for analyzing password packets, and cracking their algorithm.

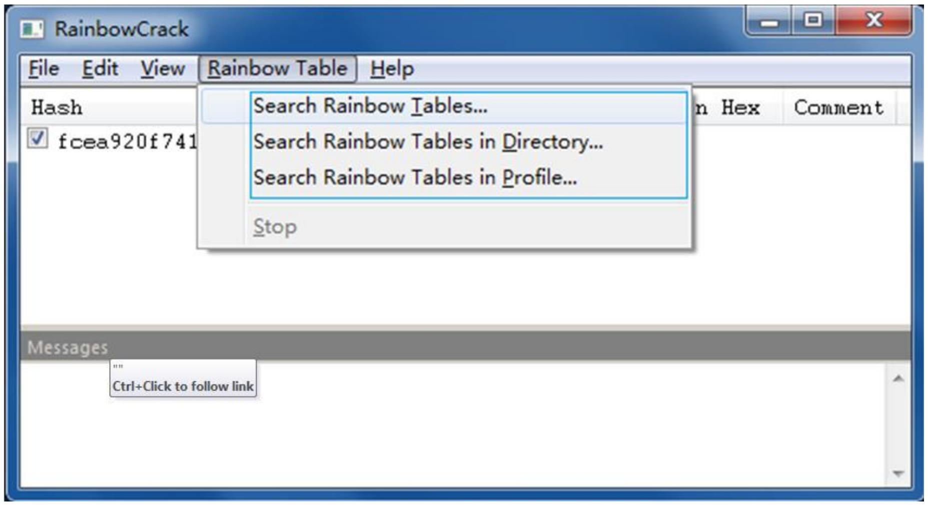


Like John the Ripper, Aircrack-ng includes a complete software suite that many use to troubleshoot Wi-Fi networks. Like aircrack hackers use it to crack WPA or WEP passwords

You can also use the well know attack techniques called FMS. FMS attacks rely on capturing an enormous amount of encrypted traffic, then using a probabilistic algorithm to crack the key. FMS crack scales linearly, which means that cracking a 128-bit key takes only slightly longer to crack then a 64-bit key, but you need to capture enough weak keys. Just recently a new attack technique has been added to the suite call "PTW." An important limitation is that the PTW attacks currently can only crack 40 and 104 bit WEP keys. The main advantage of the PTW approach is that very few data packets are required to crack the WEP key.

Supported platforms: Linux, OpenBSD, FreeBSD, OX X, Windows, Android

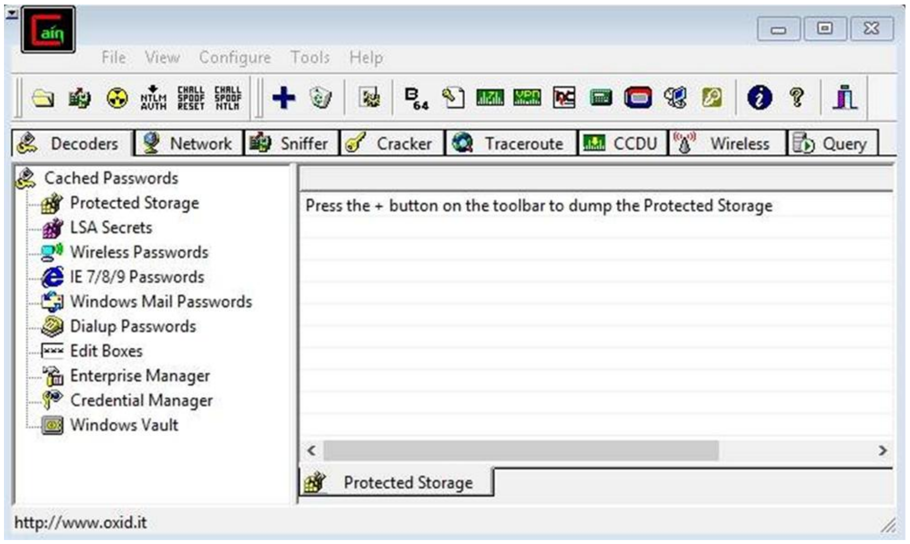
Rainbow Crack - This application makes use of rainbow tables to crack password hashes. It uses a large-scale time-memory trade-off and performs an advance cracking time computation.


RainbowCrack
File Edit View Rainbow Table Help
Hash
fcea920f741
n Hex
Comment
Search Rainbow Tables...
Search Rainbow Tables in Directory...
Search Rainbow Tables in Profile...
Stop
Messages
Ctrl+Click to follow link

This tool is about hundreds of times faster than a brute force attack. You can use rainbow tables off the Internet or create your own rainbow table. There are many different types of tables from LM, NTML, MD5 and SHA1

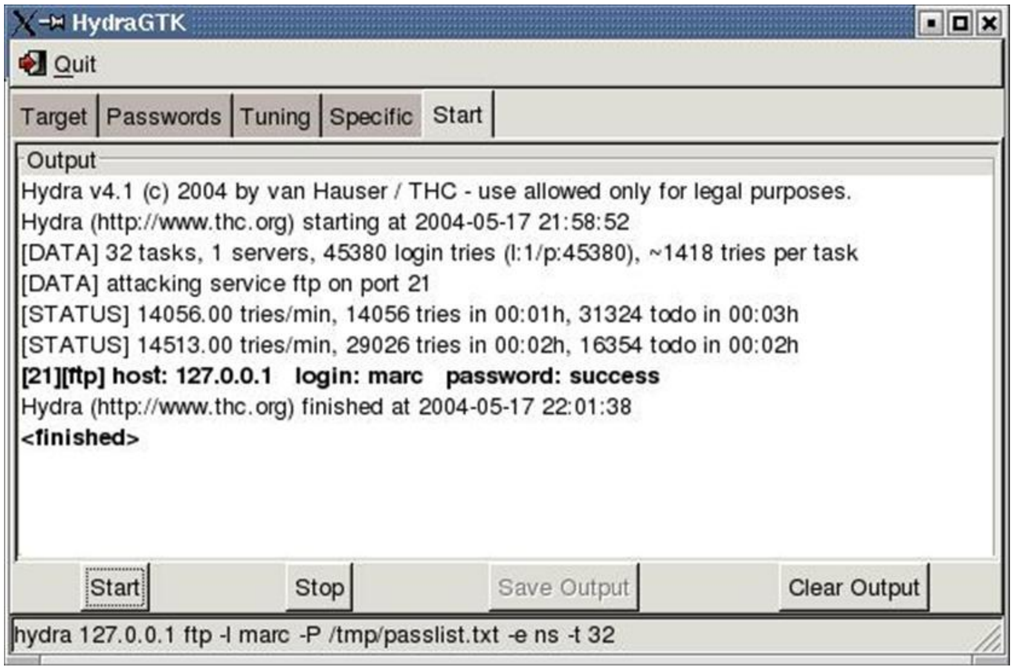
Supported platforms: Linux and Windows

Cain and Abel – It can be used to recover various types of passwords using multiple techniques. Like Dictionary, Brute-Force, and Cryptoanalysis attacks. You can also sniff networks, and record VoIP communications. In addition, to recover network keys, and decode scrambled passwords and routing protocols.



Supported Platforms: Windows

THC Hydra - This tool uses numerous network protocols, including Asterisk, FTP, HTTP-Proxy, MySQL, XMPP, Telnet and more to crack passwords. It is a fast brute-force and dictionary attack tool used to crack login pages. Network admins, pentesters and security researchers use it to test the strength of their systems.

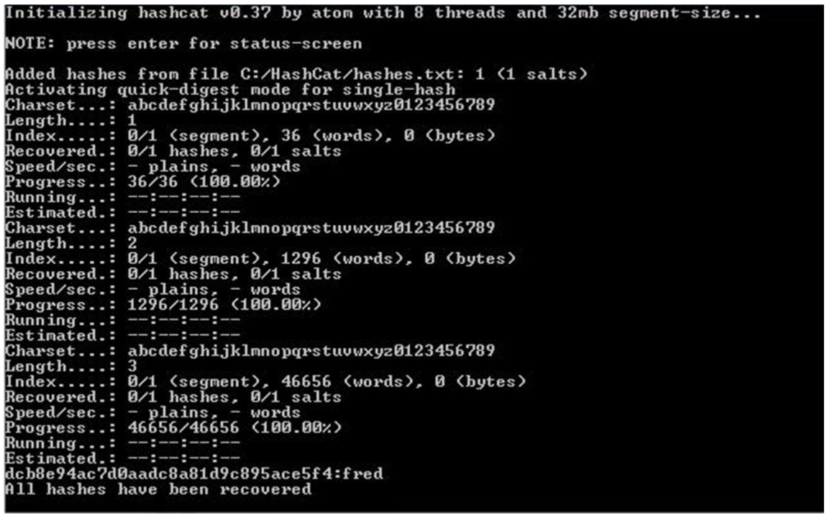


```
Hydra v4.1 (c) 2004 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2004-05-17 21:58:52
[DATA] 32 tasks, 1 servers, 45380 login tries (l:1/p:45380), ~1418 tries per task
[DATA] attacking service ftp on port 21
[STATUS] 14056.00 tries/min, 14056 tries in 00:01h, 31324 todo in 00:03h
[STATUS] 14513.00 tries/min, 29026 tries in 00:02h, 16354 todo in 00:02h
[21][ftp] host: 127.0.0.1   login: marc   password: success
Hydra (http://www.thc.org) finished at 2004-05-17 22:01:38
<finished>
```

```
hydra 127.0.0.1 ftp -l marc -P /tmp/passlist.txt -e ns -t 32
```

Supported Platforms: Windows, Linux, Solaris, FreeBSD, OS X

HashCat – HashCat supports algorithms MD4, MD5, Microsoft LM hashes, SHA-family, MySQL, Cisco PIX and Unix Crypt formats. It claims to be one of the best, and uses a well-documented GPU acceleration method. Just some of the attack vectors included are brute-force, combinator attack, fingerprint attack, dictionary attack, bybrid attack, mask attack, table-lookup attack, PRINCE attack, permutation attacks to name a few.



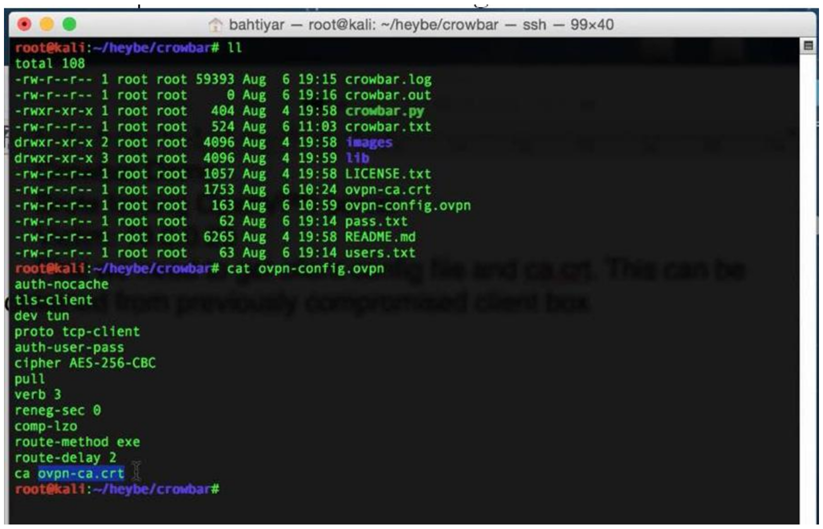
```
Initializing hashcat v0.37 by atom with 8 threads and 32mb segment-size...

NOTE: press enter for status-screen

Added hashes from file C:/HashCat/hashes.txt: 1 (1 salts)
Activating quick-digest mode for single-hash
Charset...: abcdefghijklmnopqrstuvwxyz0123456789
Length....: 1
Index.....: 0/1 (segment), 36 (words), 0 (bytes)
Recovered.: 0/1 hashes, 0/1 salts
Speed/sec.: - plains, - words
Progress..: 36/36 (100.00%)
Running...: --:--:--:--
Estimated.: --:--:--:--
Charset...: abcdefghijklmnopqrstuvwxyz0123456789
Length....: 2
Index.....: 0/1 (segment), 1296 (words), 0 (bytes)
Recovered.: 0/1 hashes, 0/1 salts
Speed/sec.: - plains, - words
Progress..: 1296/1296 (100.00%)
Running...: --:--:--:--
Estimated.: --:--:--:--
Charset...: abcdefghijklmnopqrstuvwxyz0123456789
Length....: 3
Index.....: 0/1 (segment), 46656 (words), 0 (bytes)
Recovered.: 0/1 hashes, 0/1 salts
Speed/sec.: - plains, - words
Progress..: 46656/46656 (100.00%)
Running...: --:--:--:--
Estimated.: --:--:--:--
dcb8e94ac7d0aadc8a81d9c895ace5f4:fred
All hashes have been recovered
```

Supported Platforms: Windows, Linux, OS X

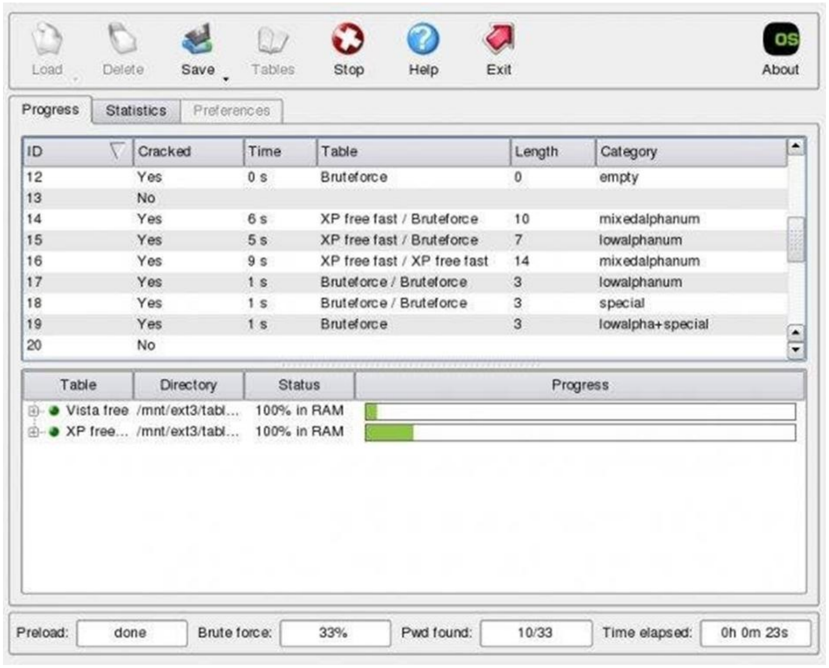
CrowBar – This is a brute force tool used by a lot of pen testers. It lets you decide how and what you want to send to the web server. Not only does Crowbar use attributes like username and passwords within it brute force algorithm, but it make use of SSH keys acquired during penetration testing. The tool supports VNC key authentication, OpenVPN, SSP private key authentication, and Remote Desktop Protocol with NLA support.



```
bahtiyar — root@kali: ~/heybe/crowbar — ssh — 99×40
root@kali:~/heybe/crowbar# ll
total 108
-rw-r--r-- 1 root root 59393 Aug  6 19:15 crowbar.log
-rw-r--r-- 1 root root     0 Aug  6 19:16 crowbar.out
-rwxr-xr-x 1 root root   404 Aug  4 19:58 crowbar.py
-rw-r--r-- 1 root root   524 Aug  6 11:03 crowbar.txt
drwxr-xr-x 2 root root  4096 Aug  4 19:58 images
drwxr-xr-x 3 root root  4096 Aug  4 19:59 lib
-rw-r--r-- 1 root root  1057 Aug  4 19:58 LICENSE.txt
-rw-r--r-- 1 root root  1753 Aug  6 10:24 ovpn-ca.crt
-rw-r--r-- 1 root root   163 Aug  6 10:59 ovpn-config.ovpn
-rw-r--r-- 1 root root    62 Aug  6 19:14 pass.txt
-rw-r--r-- 1 root root  6265 Aug  4 19:58 README.md
-rw-r--r-- 1 root root    63 Aug  6 19:14 users.txt
root@kali:~/heybe/crowbar# cat ovpn-config.ovpn
auth-nocache
tls-client
dev tun
proto tcp-client
auth-user-pass
cipher AES-256-CBC
pull
verb 3
reneg-sec 0
comp-lzo
route-method exe
route-delay 2
ca ovpn-ca.crt
root@kali:~/heybe/crowbar#
```

Supported Platforms: Windows, Linux, OS X

OphCrack – This tool is used to crack passwords and hashes using rainbow tables. It's a favorite to crack windows log-in passwords. You can import a large number of hashes from multiple formats to use with the tool. This tool works on Windows XP, Vista, and Windows 7, and leaves no trace behind...!



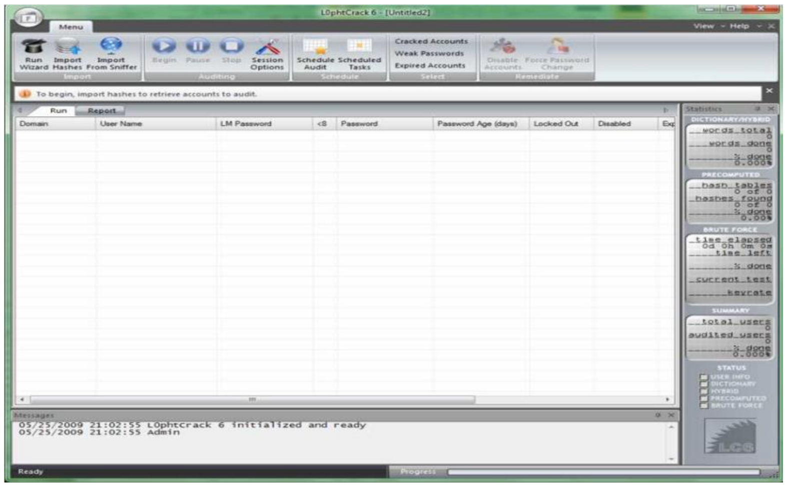
ID	Cracked	Time	Table	Length	Category
12	Yes	0 s	Bruteforce	0	empty
13	No				
14	Yes	6 s	XP free fast / Bruteforce	10	mixedalphanum
15	Yes	5 s	XP free fast / Bruteforce	7	lowalphanum
16	Yes	9 s	XP free fast / XP free fast	14	mixedalphanum
17	Yes	1 s	Bruteforce / Bruteforce	3	lowalphanum
18	Yes	1 s	Bruteforce / Bruteforce	3	special
19	Yes	1 s	Bruteforce	3	lowalpha+special
20	No				

Table	Directory	Status	Progress
Vista free	/mnt/ext3/tabl...	100% in RAM	
XP free...	/mnt/ext3/tabl...	100% in RAM	

Preload: done Brute force: 33% Pwd found: 10/33 Time elapsed: 0h 0m 23s

Supported Platforms: Windows

L0phtCrack – L0phtCrack is widely used to crack Windows passwords. It uses a large set of attack vectors, like dictionary, hybrid, brute force, and rainbows tables, and is useful in sniffing hashes. It has a routine audit function that lets you scans at convenient times. It's also a good tool to target network servers, Active Directories, and domain controllers.

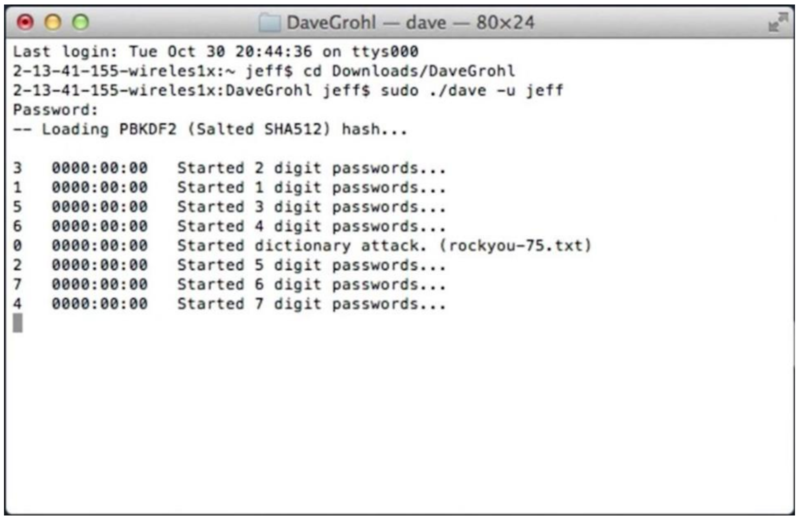


To begin, import hashes to retrieve accounts to audit.

```
05/25/2009 21:02:55 L0phtcrack 6 initialized and ready
05/25/2009 21:02:55 Admin
```

Supported Platforms: Windows

DaveGrohl - Apple security professionals preferred this tool since it works well on Max OS X. It's an object-oriented codebase platform making it a very useful tool for developers to crack passwords, and hash algorithms using dictionary attacks.



```
Last login: Tue Oct 30 20:44:36 on ttys000
2-13-41-155-wireles1x:~ jeff$ cd Downloads/DaveGrohl
2-13-41-155-wireles1x:DaveGrohl jeff$ sudo ./dave -u jeff
Password:
-- Loading PBKDF2 (Salted SHA512) hash...

3   0000:00:00   Started 2 digit passwords...
1   0000:00:00   Started 1 digit passwords...
5   0000:00:00   Started 3 digit passwords...
6   0000:00:00   Started 4 digit passwords...
0   0000:00:00   Started dictionary attack. (rockyou-75.txt)
2   0000:00:00   Started 5 digit passwords...
7   0000:00:00   Started 6 digit passwords...
4   0000:00:00   Started 7 digit passwords...
```

Supported Platforms: OS X

References

Stack Overflow
http://stackoverflow.com/questions/3690734/difference-between-ssl-tls

TLS Renegotiation Extension

https://tools.ietf.org/html/rfc5746#section-4.5

The WebLogic Server Blog – TroubleShooting SSL/TSL

https://blogs.oracle.com/WebLogicServer/entry/ssl_troubleshooting_and_debugg

Eric Rescorla's book - *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, 2001 ISBN 0-201-61598-3

Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", March 1997.
https://tools.ietf.org/html/bcp14
https://tools.ietf.org/html/rfc2119

Dierks, T. and E. Rescorla, "The Transport Layer Security
(TLS) Protocol Version 1.2", August 2008.
https://tools.ietf.org/html/rfc5246

Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", April 2006.
https://tools.ietf.org/html/rfc4347

Williams, N., "On the Use of Channel Bindings to Secure Channels", November 2007.
https://tools.ietf.org/html/rfc5056

Altman, J., Williams, N., and L. Zhu, "Channel Bindings
for TLS", Work in Progress, October 2009.

Rescorla, E., "HTTP Over TLS", May 2000.
https://tools.ietf.org/html/rfc2818

Kristol, D. and L. Montulli, "HTTP State Management
Mechanism", October 2000.
https://tools.ietf.org/html/rfc2965

Ray, M., "Authentication Gap in TLS Renegotiation",
November 2009, <http://extendedsubset.com/?p=8>.

Freier, A., Karlton, P., and P. Kocher, "The SSL Protocol
Version 3.0", Work in Progress, November 1996.

SANS Institute InfoSec Reading Room
https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029

http://www.tcs.hut.fi/Studies/T-79.159/2004/slides/L4.pdf

FIPS PUB 186, May 19, 1994, page 7, Section 6.

Responses to NIST DSS Proposal, Ron Rivest, Communication of the ACM, July 1992, Page 43.

A Course in Number Theory and Cryptography, Neal Koblitz, Springer-Verlag, Second Edition.

Recommendation X.509 and ISO 9594-8, Information Processing System - Open Systems Interconnection - The Directory - Authentication Framework, 1988.

10 Best Password Cracking Tools Of 2016 | Windows, Linux, OS X***https://fossbytes.com/best-password-cracking-tools-2016-windows-linux-download/***