**Symmetric Encryption**

Symmetric encryption utilises one secret key to encrypt and decrypt a message. This means that both the sender and recipient must either securely exchange the secret key, or inform each other beforehand which secret key will be used. The message is encrypted using the current encryption standard, the Advanced Encryption Standard (AES) which takes the original plaintext message in 128-bit blocks, and converts them into ciphertext using 128, 192 or 256 bit keys. For a 128 bit key, there are 9 rounds of jumbling in the AES encryption algorithm, 11 rounds for 192 bit key and 13 rounds of 256 bit key. As the recipient, all they need to do is decrypt using the same secret key which will then turn the ciphertext back into the original plaintext message. Symmetric Algorithms are further divided into Block & Stream algorithms. A block algorithm like AES breaks the input into fixed-size blocks and then progresses the crypto operations. Stream algorithms on the other hand perform “bit-by-bit” crypto operations which are commonly used for live broadcasts or web services where incoming data has to be transmitted immediately.

**Strengths and Weaknesses**

**Strengths**

Depending on the situation, symmetric encryption has clear advantages over asymmetric encryption. Due to its small size key, symmetric encryption requires less computing resources and is much faster at encrypting message so typically used for encrypting large amounts of data such as databases. Symmetric encryption keys based on today’s computing power is considered very secure offering 2^128 different possible key combinations for a 128 bit key, provided you’re able to safety store the actual key safe. Symmetric encryption speeds are faster than asymmetric encryption owing to the smaller key size, 128 bit key compared to 2048 bit key.

**Weaknesses**

The downsides to using symmetric encryption is that while it offers confidentiality by encrypting the message, it offers no mechanism to guarantee the integrity or authenticity of the message. The main weakness is how do you securely get the key to the other person, and how do you safely store this key? Since the key is used to decrypt and encrypt, if the other person is irresponsible in securely storing the shared secret key, both of your communications become compromised. This problem grows as the number of connected users increases because you’ll need a unique secret key for each unique user.

**Asymmetric Encryption**

Asymmetric encryption utilises two sets of keys called public and private keys. The public key is used to encrypt the message which is publicly known, while the private key is used to decrypt the file and is kept private. If you wanted to send a secure message to your friend, you would encrypt the message with the public key and your friend would decrypt the message with his private key which is known only to him. An example of asymmetric encryption is RSA which creates and publishes a public key based on two large prime numbers. The first step of RSA is to turn the message into a number, then encrypt that number by raising the message to the power of the public key. When choosing the prime numbers, the multiple of the number (modulus) must be big enough to encode the intended message so if your message is a 6-digit number, your modulus must be greater than 6 digits. RSA is one way function with a secret trapdoor (private key) which is used to re-assemble the encrypted message. To decrypt, you raise the encrypted text to the power of the private key to get the original plaintext message.

Asymmetric encryption can also used as a form of authentication through RSA. The owner of the private key raises a message (x) to the power of their private key (d), x^d. The resulting ciphertext (y) is the unique form of authentication because only the holder of that private key could create that ciphertext. To decrypt, you would use the public key (e) which is publicly available and raise the encrypted text to the power of the public key to get the original message, y^e.

**Strengths and Weaknesses**

**Strengths**

The big advantage is that secure communication is no longer necessary. An attacker could intercept your encrypted message but it can only be decrypted using the private key, which is held by the recipient. This solves one of the key weaknesses of symmetric encryption which is how the sender would establish secure communication with the intended recipient. Additionally, if the sender were to lose their key, doesn’t compromise communication data between sender and recipient as the case with symmetric encryption.

**Weaknesses**

Key sizes for asymmetric encryption are many times longer than symmetric encryption so it requires more time and computing power to encrypt and decrypt. If you’re using RSA, your encoded message must be smaller than the modulus, which limits the type of message you can send. When you use their public key, the sender can authenticate the identity of the recipient but the recipient has no way of authenticating the identity of the sender. This makes it vulnerable to attackers who may intercept a message with their public key imbedded on its way to one of the recipients and instead pass along their key instead. There’s no way of detecting this ever happened on both sides.

**Hashing**

Hashing is one-way cryptographic function that takes a plaintext message or file and summaries the contents into a series of numbers and letters of fixed size. The output of a hash function is referred to the hash value or digest. The hash value is a summary of the file content so by comparing two different files, you can verify the integrity of the file by comparing the two hash values. Hash values are of fixed length no matter the size of the content so this makes it a great tool in verifying file integrity. A key property of hashing is the avalanche effect whereby small changes in the input lead to large changes in the hash value. This means the hash value seems almost random and getting two similar messages won’t produce similar hash values. This is because the hashing function breaks the message into certain block sizes and applies a hash function to the first block. This hash value is attached to the second block and a hash value is computed for the second block and so on so forth.

Other key properties of hashing are that the same plaintext inputs should always produce the same hash output. For this reason, it makes it a great cryptographic function for password storage and file integrity verification. By saving the hashed password, exposed databases don’t automatically reveal the passwords of users. The attackers must try every input and match the hash value to the hash values of the databases which require significant computing power.

Three important properties include:

**Preimage** **Resistance**: given a hash value, it should be difficult to reverse a hash function to find the initial input, making it a one-way function

**Second** **Preimage** **Resistance**: given a plaintext input, it should be difficult to find a different plaintext input with the same hash value. This protects against attackers who has an input value and its hash, and wants to substitute a different value as legitimate in place of the original hash value

**Collision** **Resistance**: it should be difficult to find two different inputs of any length that results in the same hash.

Key difference between collision resistance and second preimage resistance is that for second preimage resistance, the attacker is handed a fixed plaintext input and must find another plaintext input with the same hash value. In collision resistance, the attacker can choose both plaintext messages with the only requirement being they find two plaintext messages with the same hash value. This is what a Birthday attack is used for

**Strengths and Weaknesses**

**Strengths**

The advantage of hashing is that it’s a one-way function making it suitable for storage of sensitive information like passwords. Hashing also can also be used as proxies for much larger sized files since the hash value is just a summary of the file’s contents compressed into a fixed length. This makes it a convenient cryptographic function to quickly verify file integrity because any small change will be expressed through a different hash value output.

**Weaknesses**

A key weakness in hashing is the way the hash values are created for each block. A hash function breaks a file into blocks of a certain size and applies a hash function to the first block. The hash value output is attached to the second block and computes a hash function for the second block and so on so forth. This means its vulnerable to a length extension attack whereby an attacker could attach his own message at the end of the hash value. The attacker would use the previous hash value and combine it with the message of his choice to continue the hash function. As the number of hashed values increases, it runs the risk of collision. Ideally, each plaintext input should result in a unique hash value but given a large enough database, you could find a situation where two different values have the same hash values, meaning that the attacker could use a different input to crack your password.

**Combined Cryptographic Primitives**

**Session Key (Symmetric Encryption + Asymmetric Encryption)**

One way of taking advantage of the strengths of symmetric and asymmetric encryption while suffering none of its weakness is by combining the cryptographic protocols as exemplified with SSL session keys. SSL Session keys are used to establish secure communication from web browser to web server.

1. Upon request, the web server sends a copy of its public key to the web browser.
2. The web browser creates a secret key which will be used as a session key and encrypts using the browser’s public key.
3. Server decrypts the session key with its private key
4. The server and browser can now communicate using the session key, allowing for quicker communication

You might be wondering why the web browser and web server would switch from asymmetric encryption to symmetric encryption. The reason being that asymmetric keys comprise of 2048 bit keys compared to 128 to 256 bit keys for symmetric encryption. This means transmitted data take considerably longer to encrypt and size of the message would have to be smaller than the size of the modulus.

Few benefits of using an SSL session key or any session key used in communication are:

1. Crypto-analyses of the communication data becomes more difficult because each session data is encrypted with a different key
2. Ensures that the symmetric key is delivered securely to the intended recipient as only they can decrypt the contents with their private key. Solves one of the main problems of symmetric encryption
3. After establishing secure communication, you can switch to symmetric encryption allowing for faster transmission of data between sender and recipient

**HMAC (Hash-based Message Authentication Code) (Hashing + Symmetric Encryption)**

HMAC combines elements of symmetric encryption and hashing in order to verify the message. In symmetrical encryption, the sender sends an encrypted message to the recipient but the recipient has no way of verifying the integrity of the message. Hashing by itself is vulnerable to length extension attacks since the current hash is generated by combining the new block (which you decide) and the hash function of the previous block. HMAC works by choosing a secret key known by both the sender and recipient like symmetric encryption but the key difference is that two keys are created derived from the original key, k1 and k2. The first key is created by XOR with the innerpad (usually 0x36) and the message is appended to k1. It is then hashed to create the first hash value. In order to stop extension attacks, a second key is created by XOR with the outerpad (usually 0x5c) and the previous hash is appended to k2. It’s quite impossible to use an extension attack because you won’t know the previous hash value required to create the current block’s hash because that’s been hashed with the second key. This hash value is the HMAC and is sent alongside the original plaintext message to the recipient. The recipient can re-compute the hash value using the keys and verifies message integrity by comparing the HMAC. It also provides authentication because only those that know the original key can re-create the HMAC.