LEARNING FROM SSL

BURTON ROSENBERG
UNIVERSITY OF MIAMI
6 APRIL 2025

Contents

1. Overview of SSL

Secure Socket Layer is a software library that sits between the communication sock (e.g. the Berkeley socket) and the application, and ensures security communication between the communication endpoints. It can be considered a layer 4 or layer 5 technology, in the OSI terminology. It's full implementation requires the existence of the PKI (Public Key Infrastructure). It is built out of standard encryption algorithms such as AES, CBC and RSA.

1.1. **The Design Principles.** SSL was proposed by the browser maker Netscape to enable web commerce. It was analyzed, flaws corrected, standardized, and then passed to the IETF for standardization in the IETF suite as TLS.

The design principles were,

- Aim for high adoption.
- Avoid messing with the operating system.
- Favor the most important cases.

- Use standard cryptography.
- Be end to end, without relying on the security properties of the channel or the operating system.

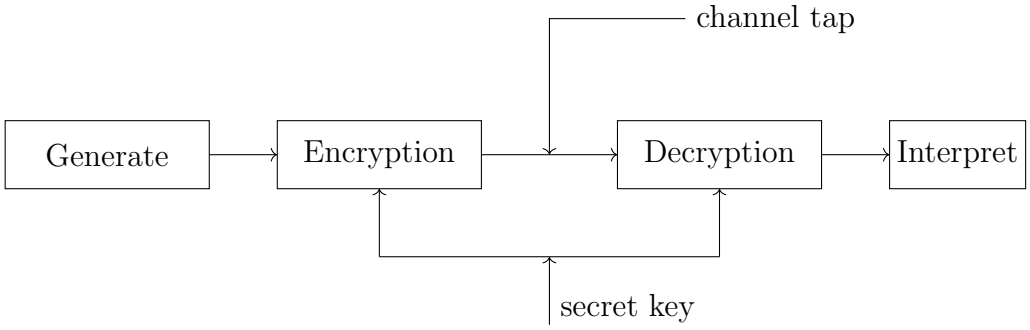
1.2. **The Four Elements.** The SSL system of encrypted communication between applications ensures four security qualities,

(1) Confidentiality: given a public channel, observers of the public channel learn little from observing the messages. They might learn *traffic analysis*, but nothing more.
(2) Integrity: the data received is not the data sent, the receiver knows this is the case.
(3) Authenticity: the messages can be verified to be from the said sender.
(4) Key Agreement: required encryption keys are shared.

2. Encryption

The first element of SSL security is privacy. This is accomplished with encryption. Encryption is what many people think of when they think "cryptography".

We will discuss both *public key* and *secret key* encryption (also called *asymmetric* and *symmetric* key encryption, respectively). Secret key encryption is the traditional encryption type. The model is as follows,



(1) The channel connecting the encryption output to the decryption input is public. There can be a eavesdropper as shown. This would be a *passive* adversary that simply observes. Adversaries can also be more active: they can impede or modify messages.
(2) An encryption and decryption needs a key. For secret key, this key is shared, as shown. How this key is shared is a matter outside of this current diagram. It is interesting to think of this scheme as "privacy in a bottle", where the privacy was created where and when the key was shared. The privacy of the public channel is not the privacy of that location and moment.

(3) The generation abstract is needed in order to talk mathematically about the system. It is generally a finite probability space of possible *messages*. The probably on the messages is the a priori information the eavesdropper has about the message. The game is that the eavesdropper begins with this probability distribution and on observing the encrypted data on the channel perhaps updates this probability distribution to sharpen it around certain messages.
(4) The interpret box allows for the fact that the receiver is also engaged in the same process as the eavesdropper. It too has an a priori distribution and will update this with the reception of data. Typically, the data is decoded to exactly the message selected by the encryptor, so that after reception the distribution is probability one on the message.

2.1. **Perfect Secrecy.** It is possible to have a perfect encryption scheme. This was established by Claude Shannon in *Communication Theory of Secrecy Systems* (1949). The mathematical description of *perfect secrecy* is,

$$P(M) = P(M \,|\, C)$$

This is a very compact way of saying that the a priori probability distribution $P(M)$ over the space of messages M when conditioned on "learning"' a ciphertext in C, does not change the probably distribution.

A *Vernon cipher* gives perfect security. It is also called the *one time pad*. Let the message be a bit string $p \in \{\, 0, 1 \,\}^*$. To encrypt, create a random string of bits $r \in \{\, 0, 1 \,\}^*$ and set

$$\mathcal{E}_r(p) = r \oplus p$$

The bit string r is the same length of p, and is a truly random sequence of coin flips of an unbiased coin. To decrypt, assuming the decryption machine has r, exclusive or again by r,

$$\mathcal{D}_r(p) = r \oplus \mathcal{E}_r(p) = r \oplus (r \oplus p) = p.$$

The problem of this method is that shared key is long and it cannot be reused. So it is inconvenient because the two parties when they meet share a random key that is very long and will run out.

This can be seen to be perfectly secret by this argument. Consider a one bit message. Whatever it is, even if the message is certain, a coin flip with probability $1/2$ flips it and with probably $1/2$ leaves it alone. Therefore the result is a bit that is $1/2$ of the time 0 and the other $1/2$ 1, regardless of what the message is. Hence the observer of this bit learns nothing.

To test your understanding, if the coin used to encrypt is biased, then watching the ciphertext is helpful. If the coin is more likely 0, then it is more likely that the message was left alone than that it was inverted. Hence the ciphertext hints at the

message value. Whether it is worth it to exploit this is another matter. For instance, if the observer knew that only the message 0 will be sent, that is the $P(M)$ in the above equation is concentrated on 0; then it is correct to guess 0 no matter what is seen.

2.2. **Model encryptions.** Because the one time pad is inconvenient, other methods are used to encrypt which are not perfect. A block cipher tries to emulate a perfect encryption. The current block cipher to use is called AES and it comes in two flavors depending on key size, AES–128, AES–192 and AES–256.

If the block cipher were perfect, each key chooses a random (but deterministically reproducible) permutation of the 2^{128} bit patterns of the block size. If this were true, the only information leaked in the pair $c = \mathcal{E}_k(p)$ is that for any $p' \neq p$ then $\mathcal{E}_k(p') \neq c$.

Whether this is true depends on the cipher being good; and in our discussions let us assume it is good. We can learn a great deal once we fix this ideal case in our minds. Reality is more difficult but we still learn a lot.

2.3. **Randomized encryption.** Among the things we can learn, even in this idealized model are,

- Even an ideal cipher does not disguise the length of a message.
- Even an ideal cipher does not disguise the timing of a message.
- Even an ideal cipher does not disguise the pattern of senders and receipients.
- And even an ideal cipher does not disguise when two cipher texts are of the same or different plain texts.

The first four problems are real, and or often called *traffic analysis*. I do not pursue here a discussion of these problems.

The last problem however should be handled by using randomization to insure that the cipher text is always diversified. A *nonce* is a string that is used only once. One possible defense against this last problem is to encrypt as,

$$\mathcal{E}'_k(p) = \langle\, n, \mathcal{E}_k(p \oplus n) \,\rangle$$

and to ensure that given two nonce choices n and n' that the distribution of $n \oplus n'$ is uniformly random.

2.4. **Attacks on encryption schemes.** The most definitive success for an attack would be *key recovery*. That means, the key becomes known by analysis of the message stream. But it is not the only sort of success. Any improvement in the attackers guess about the message contents is a problem. There are several classes of attack, depending on the attackers access,

(1) Known plain text: passive eavesdropping gives pairs (p_i, c_i) where $c_i = \mathcal{E}_k(p_i)$.

(2) Chosen plain text: the attacker provides a batch of plain texts $\{ p_i \}$ and receives $\{ c_i \}$ where $c_i = \mathcal{E}_k(p_i)$.
(3) Adaptive chosen cipher text: the attacker gets to pick c_i and receives $p_i = \mathcal{D}_k(c_i)$. The difference with the above is the choice c_i can depend on the previous decryptions.

These attacks are more threatening in the order given. To illustrate the power of the Adaptive chosen cipher text attack (CCA) consider our nonce scheme. It is broken by a CCA attack.

Suppose we happen to have

$$\langle\, n, \mathcal{E}_k(p \oplus n) \,\rangle$$

that we are curious about. In CCA attack the attacker asks for a decryption of

$$\langle\, n', \mathcal{E}_k(p \oplus n) \,\rangle$$

with $n \neq n'$. This is an allowed query because the victim does not notice this is a variant of an already sent message. The attacker then receives

$$p \oplus n \oplus n'$$

and can subtract off the two nonces to get p.

2.5. **Modes of operation.** The encryption of blocks is extended to the encryption of messages. A message is a sequence of blocks $p_1, p_2, \ldots$. Since a message might not be a multiple of the block size, a *padding scheme* is used to fill out the last block to block size.

2.5.1. *Cipher Block Chaining.* A popular method to encrypt the block sequence is called *cipher block chaining* (CBC) and is given my the equation,

$$c_i = \mathcal{E}_k(c_{i-1} \oplus p_i).$$

The first encryption uses c_0 set to a nonce called the *initial vector*. This is information that is agreed upon between the sender and receiver, and in some systems this is done by sending it clear as the zero-th block of data. This allows for randomization of all the encryptions.

2.5.2. *Output Feedback.* Another method is *output feedback mode* (OFB) which is described as,

$$r_i = \mathcal{E}_k(r_{i-1}), \quad c_i = r_i \oplus p_i$$

with r_0 an Initial Vector.

A way to look at OFB is that it is a one-time pad, where the masking string is not generated by coin flips but by continuous sampling of the random permutation $\mathcal{E}_k$.

Note that to decrypt we reapply the encryption. An inverse to $\mathcal{E}_k$ is not needed. Rather than AES, a one-way function, such as a cryptographic hash function, is usable in this encryption scheme.

This encryption is *malleable*, meaning the cipher text can be manipulated in a way that changes the decryption in a known manner. If c_i decrypts to p_i, then $c_i \oplus r$ decrypts to $p_i \oplus r$. This might seem to be a terrible weakness, but the issue of data corruption has not yet been addressed.

2.5.3. *Counter Mode.* The last mode to look at is *counter mode*, which simplifies OFB by sampling the encryption of inputs $r_0, r+1, \ldots$ for a random starting point r_0.

$$r_i = r_{i-1} + 1, \quad c_i = \mathcal{E}_k(p_i \oplus r_i).$$

This method has the advantage of working in parallel over the encryption stream. For instance, full disk encryption can use the disk block number for r_i.

3. Message Integrity

Message integrity is also called message authentication. This is because in the symmetric key setting, integrity is assured using a roll of of the message into a key-dependent tag. This tag is created by the sender, and recalculated by the receiver. If they match the message is considered good. Else the message is bad. The tag is called a *message authentication code* (MAC).

There are two inferences made of the tag being good,

(1) The message is not corrupted.
(2) The message is authentic to the key-holders.

The combination of a encryption and integrity is called *authenticated encryption*. Some systems are built specifically for both aims. Some are composites of an encryption scheme and a separate authentication scheme.

Consider the the *CBC MAC* authentication tag. Given a text use CBC encryption with a distinct key and IV and the last block is the MAC. All other blocks must be kept secret and discarded, and the MAC is sent as the tag. We will symbolically show this system as $MAC_k(x)$. Let $\mathcal{E}_k(x)$ be the encryption scheme for messages (multiple blocks). These can be combined in these ways,

(1) MAC and Encrypt: Send

$$\langle\, \mathcal{E}_k(m), MAC_{k'}(m) \,\rangle$$

for message m, note that $k \neq k'$. This is not good, as we made no assurance about information leakage about the message m by the MAC algorithm.

(2) MAC then Encrypt: Send

$$\mathcal{E}_k\big(m \,||\, MAC_{k'}(m)\big).$$

for message m, note that $k \neq k'$. SSL does it this way.

(3) Encrypt then MAC: Send

$$\langle\, c = \mathcal{E}_k(m), MAC_{k'}(c) \,\rangle.$$

for message m, note that $k \neq k'$. The cipher text is MAC'ed. The advantage of this method is that a request to decrypt will not even be entertained unless the MAC on the cipher text is incorrect. Therefore, before any attack can be made on the encryption, an attack has to be success against the MAC.

3.1. **Authenticated Encryption.** There are mechanisms that both encrypt and authenticate as part of a single standard. The Galois/Counter Mode (GCM) applies an Encrypt then MAC where the MAC is similar to a CRC, where the cipher texts are coefficients in a polynomial. The polynomial is evaluated at a key dependent location and finally encrypted with the first block of randomness from the encryption stream.