CCA Security

Lecturer: David Heath
Scribe: Mayank Hirani

September 26, 2023

Introduction

This lecture is focused on the concept of CCA security, which stands for "Chosen-Ciphertext Attack". CCA security ensures confidentiality *and* authenticity. With CPA security, the weaker security discussed in previous lectures, Alice and Bob can send messages to each other with a shared key and eavesdropper Eve has no way to understand them.

We will combine the ideas of encryption and MAC tags to define the Encrypt-Then-MAC scheme. We will take a look at the proof of why Encrypt-Then-MAC is CCA secure. This lecture uses all the ideas and concepts from previous lectures to funnel into the best scheme for achieving confidentiality and authenticity that we have analyzed so far.

This lecture will also briefly talk about malleability and revisit the CBC (cipher block chaining) mode to see why it is not CCA secure.

Background

To understand CCA security, we must review the concepts of CPA security and MAC tagging.

Definition 1. CPA Security: We say an encryption scheme Σ (Enc, Dec) is CPA secure if the following two pieces of code are indistinguishable

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\textsc{eavesdrop}(m_L, m_R \in \Sigma.\mathcal{M})$:
$\quad c := \Sigma.\text{Enc}(k, m_L)$
$\quad \text{return } c$

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\textsc{eavesdrop}(m_L, m_R \in \Sigma.\mathcal{M})$:
$\quad c := \Sigma.\text{Enc}(k, m_R)$
$\quad \text{return } c$

[Ros21]

This means a scheme is CPA security if an adversary cannot tell, with high probability, if they exist in the universe where the left message is being encoded, or if the right message is. The key is set one time, and not changed every time the EAVESDROP function is called.

It is impossible to achieve CPA security with a deterministic encryption scheme that always produces the same ciphertext ct for the same encryption of a message m.

Here is an example of an attack that demonstrates why deterministic encryption schemes can easily fail to meet CPA security.

$\mathcal{A}$
arbitrarily choose *distinct* plaintexts $x, y \in \mathcal{M}$
$c_1 := \textsc{eavesdrop}(x, x)$
$c_2 := \textsc{eavesdrop}(x, y)$
return $c_1 \stackrel{?}{=} c_2$

[Ros21]

In the above example, the code represents the adversary's attack. Because, and this is important, the adversary can call EAVESDROP as many times as they want, with whatever string they choose, they can simply call it with the same string twice, and again with string x and string y.

Case 1, we are in world 1 (encrypting the left message):

c_1 will be set to $Enc(k, x)$.

c_2 will be set to $Enc(k, x)$.

Since Enc is deterministic, $c_1 = c_2$. Since they are equal, the adversary knows they must be in the first world, encrypting the left message.

Case 2, we are in world 2 (encrypting the right message):

c_1 will be set to $Enc(k, x)$.

c_2 will be set to $Enc(k, y)$.

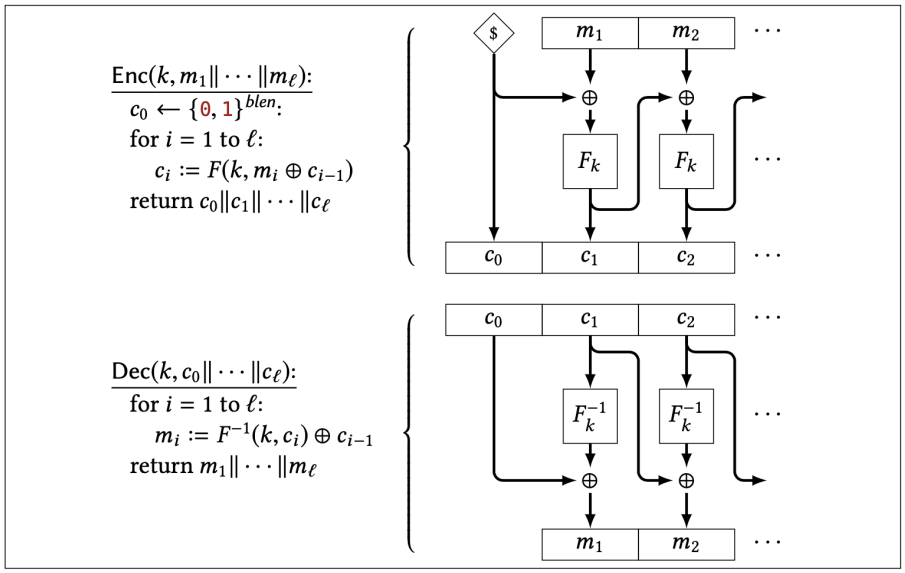
Hence, $c_1 \neq c_2$ because Enc is invertible and thus bijective.

Since $c_1 \neq c_2$, the adversary knows they must be in the second world, encrypting the right message.

The main idea here is that since the adversary has access unlimited use of the EAVESDROP function, with any messages, and with constant key, a deterministic scheme will not be secure enough. Something needs to change each time to prevent the adversary from breaking the scheme.

CPA security is commonly achieved using the CBC (Cipher Block Chaining) and CTR (Counter) modes.

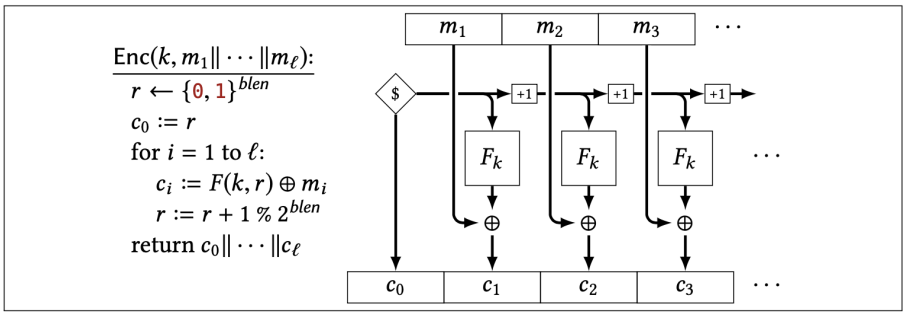
Definition 2. Cipher Block Chaining (CBC) Mode: We define CBC mode to be the following



[Ros21]

The message m is broken into chunks, and a uniformly randomized initialization vector is passed into the starting block. Each block of the message is bitwise XOR'd with the previous output, or the initialization vector for the first one, and then passed through a pseudo-random function to create a block of the cipher text. This mode is a CPA secure mode.

Definition 3. Counter (CTR) Mode: We define CTR mode to be the following



[Ros21]

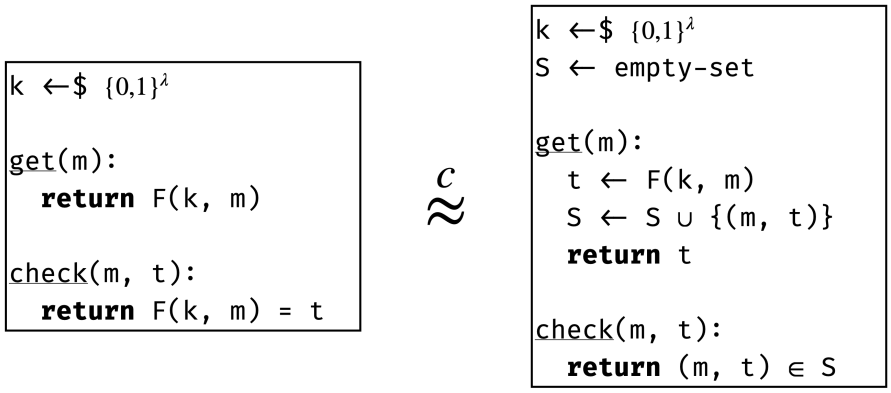
CTR mode works similarly to CBC mode, but does not encrypt each message block using the PRF, and instead uses an incremented initialization vector each time. CTR mode also achieves CPA security.

Definition 4. Message Authentication Codes (MAC): The last bit of background used for today's lecture is the concept of message authentication modes. Message authentication code is an extra tag sent along with a message to ensure **authenticity**.

A MAC scheme consists of *KeyGen* and *tag* functions. *KeyGen* used for generating keys to be used by *tag*, which generates a tag specific to the input and key.

The general idea to tagging is that every message has one single unique tag, and that tag is mostly unique to the message. By sending the message and the tag, if the message is modified, it *should not* match with the tag when the receiver tries to create a tag to match it. If the adversary modifies the tag, then the tag *should* not match the message. This method ensures that when the receiver, Bob, gets the message, he can tell if Eve changed any part of the message, **even if she could not understand the contents**.

The formal definition of a MAC scheme is as follows:



[DH]

get generates a tag for a message, *check* compares a message to a tag to see if the tag is the correct corresponding tag for the message.

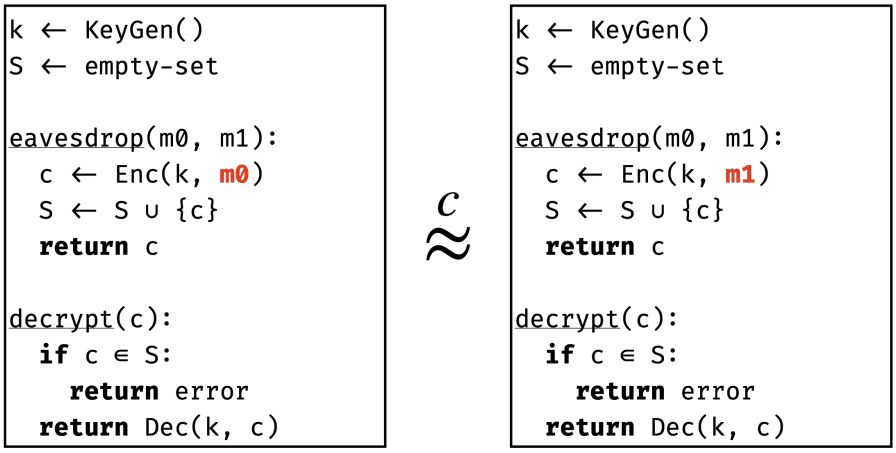
Here, the formal definition is saying that our tag creation and checking system should be indistinguishable from a system of storing every pair of message and tag in a set, and only having a tag match a message if it was generated before. What this means is that the adversary Eve can generate a valid tag for any message, but she cannot check the tag for any message she has not generated a tag for before.

It is effectively impossible for the adversary to guess the tag of a new message that they have not tagged themselves, but Alice and Bob "share" the set of tagged messages in a way, and can therefore check the message tags. This is the ideal behaviour that the MAC scheme behaves as.

Since PRFs only work with certain length strings, we can use a CBC-MAC scheme to split a message into smaller chunks and XOR each output of the previous chunk through a PRF.

Lecture Notes

Definition 5. CCA Security is defined as follows, for an encryption scheme (Enc, Dec):

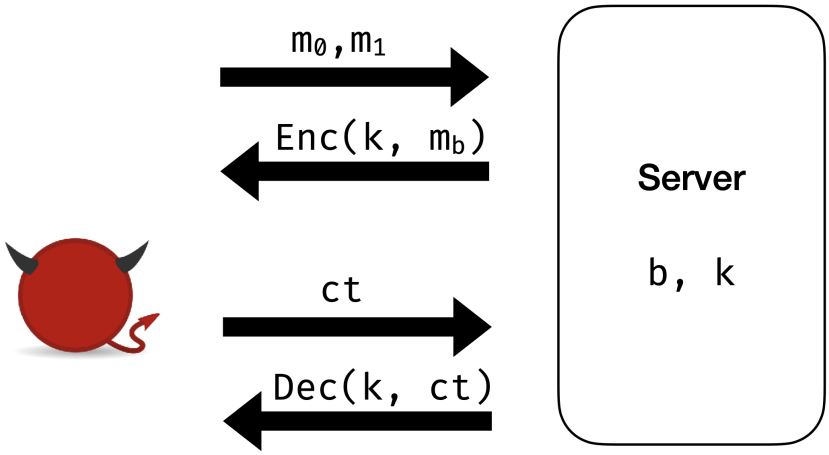


[DH1]

The big difference between CCA security and CPA security is that in CCA security, the **adversary has access to decryption**. The adversary can try decrypting messages of their choice, but only if that message has previously been encrypted by them. As you may immediatley be able to tell, this is a lot stronger than CPA security.

To beat CCA security, an adversary must be able to distinguish, again, which world they are in. They must be able to tell with high certainty whether they are encrypting the first message or the second one. If they could decrypt any message, this would be a simple task, as they could decrypt the output of eavesdrop.

One way to think of this is that there is some server that everyone has access to. The server has some set constant key. By interacting with the server and giving it ciphertexts, the server emits certain behaviour. The adversary can interact with this server to try and learn an attack.

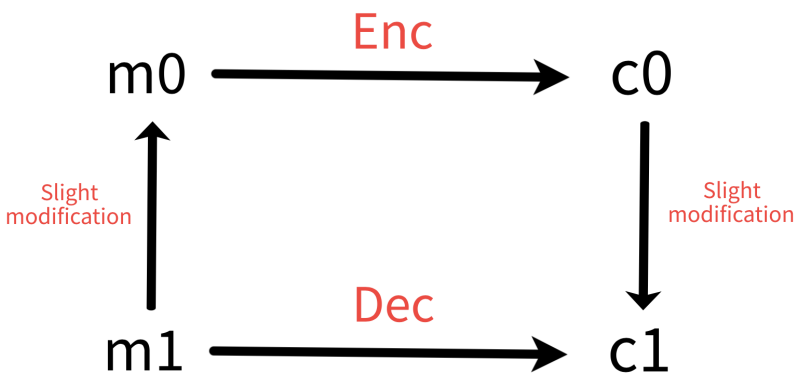


[DH1]

It becomes clear why **every scheme that is CCA secure is also CPA secure**. If the adversary cannot attack a CCA secure scheme, and we take some of their functionality or tools away, they are definitely not going to be able to attack the scheme with even less tools.

Definition 6. In this lecture we also bring up the concept of **malleability** again. If an encryption scheme is malleable, the adversary can change the cipher-text a little and decrypt to compare with the original text. This becomes a bigger deal with CCA security because the adversary has access to the decryption tools.

Here is a general diagram how this is possible.



The adversary can make a slight modification to the cipher-text they have intercepted, and then use the decryption functionality to find the original message of the slightly modified cipher-text. If the encryption scheme is malleable, then the message $m1$ will be a slight modification of the original message $m0$, and the adversary can use this information to gain some knowledge about $m0$.

CBC mode, mentioned above, is one example of an encryption scheme we discussed that *is* malleable.

Using the decryption functionality, the adversary can build a distinguisher as follows:

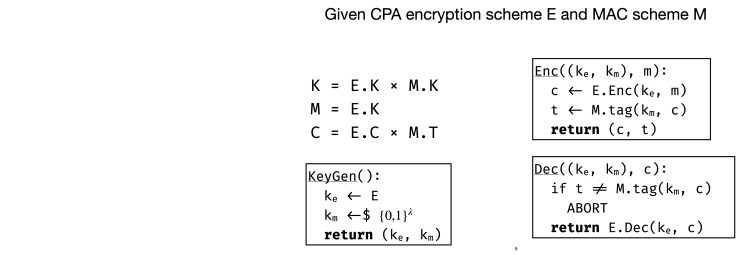
$c_0|c_1|c_2 \longleftarrow eavesdrop(0^{2\lambda}, 1^{2\lambda})$
$m \longleftarrow decrypt(c_0|c_1)$
return if m all 0s or all 1s

Here, $c_0|c_1|c_2$ is the cipher-text consisting of each block c_0, c_1, c_2 from each block of the CBC mode. This cipher-text is either an encryption of all 0s or all 1s, and the adversary wants to figure out which one.

Here is where the magic comes in. The adversary cannot simply decrypt the cipher-text by itself. That is not possible by the definition of CCA security. What the adversary *can* do however, is decrypt just a chunk of that. That's what's going on in line 2. The adversary can decrypt the first two thirds of the cipher-text, which since we used CBC chaining will just correspond to the first two thirds of the message, and then see part of the message. They can then tell if quickly if the piece of the original message they have is all 0s or all 1s, and thus distinguish the world they are in. So in conclusion, CBC mode is not CCA secure.

Definition 7. Encrypt-Then-MAC: The Encrypt-Then-MAC scheme is the scheme that brings together everything we have talked about to create a CCA secure encryption scheme. In its construction, the Encrypt-Then-MAC scheme is very simple. As the title suggests, we first encrypt a message, ensuring confidentiality, then we tag that encrypted cipher-text, ensuring authenticity, and then send it.

Here is what the scheme would look like.



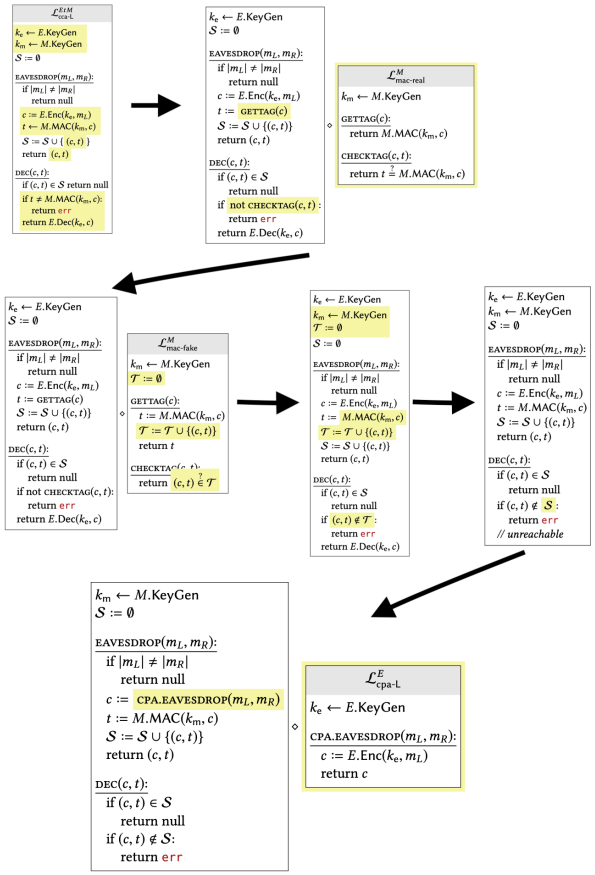
[DH1]

We can see here that there are two different keys, one for encryption, and one for creating the tag. The encryption encrypts the message using the key for encryption, then tags that cipher text with the key used for MAC tagging. It then returns that cipher-text along with the tag. The decrypt works as we would expect, checking the tag first, and then decrypting the cipher-text if the tag was correct.

Proof of CCA Security:

We can describe the server as having two values, k and b, where k is the constant key and b is the bit for which world we are in, the world in which we are in. The world we are in is either the one in which the server always gives an error, or one where it properly decrypts the given cipher-text.

We can show by replacing libraries that our scheme is essentially the same as the server returning an error every time the adversary tries to decrypt a new message, so it is CCA secure.



[Ros21]

By replacing different sections of the scheme with indistinguishable pieces of code, we can see that our scheme is indistinguishable from a scheme that checks if the tag and cipher-text pair has already been created. In this version it is impossible to get decrypt a cipher-text that has not been added to the set, so it is CCA secure.

References

[DH] David Heath, CS 407 Cryptography Fall 2023 Lecture 9.

[DH1] David Heath, CS 407 Cryptography Fall 2023 Lecture 11.

[Ros21] Mike Rosulek. *The Joy of Cryptography*. 2021.