University of Illinois, Urbana Champaign
CS/ECE 498AC3/4 Applied Cryptography

Instructor: Dakshita Khurana
Scribe: Jong Chan Lee
Date: September 15, 2020

LECTURE
7

Authenticated Symmetric Encryption

Thus far in our exploration of the realm of private key cryptography, we have encountered two key concepts of cryptography that are two of the basic principles of "secure communication": confidentiality and integrity. Confidentiality, or secrecy, is a security property that prevents potential adversaries from obtaining information about any messages that we wish to communicate securely. Previous lectures dealt with the general properties and various examples of (symmetric key) encryption schemes proposed and employed mainly for the purpose of guaranteeing varying degrees of data confidentiality. One rather reasonable definition of confidentiality that we expect of, or require from real-world encryption schemes is semantic security against a chosen plaintext attack.

Confidentiality does not, however, by itself ensure secure communication between two parties. In fact, many of the encryption schemes covered earlier in this course only provide security against passive adversaries that are merely "eavesdropping" with the goal of obtaining what should be secret information. Against a more active adversary whose intentions are not limited to simply the extraction of secret information but may also include, for example, intercepting and tampering with a (possibly encrypted) message before sending it on its way to its originally intended receiver, confidentiality is not enough. Integrity, however, is a security property that prevents precisely this second kind of adversary from achieving their malicious dreams. In past lectures, we explored message authentication codes(MACs) as a way to ensure data integrity; a definition of integrity that we were the most concerned with was existential unforgeability under a chosen message attack. MACs, unfortunately, do not then necessarily guarantee confidentiality.

Naturally, the contents of this lecture are thus centered around the concept of authenticated (symmetric) encryption, which is literally defined as encryption that gives us both confidentiality and integrity. We first discuss the definition of secure authenticated encryption and the implications of an encryption scheme satisfying that definition, before moving on to the problem of constructing secure authenticated encryption schemes. It turns out that we are in luck and there exist generic ways to construct secure authenticated encryption schemes from the very cryptographic devices we have dedicated the past few lectures to familiarizing ourselves with: (secure) encryption schemes and (secure) MACs. Of the several ways of combining encryption with MACs that immediately come to mind, we first look at the ones that work and why they do. As for the ones that do not always work, we

introduce a particularly interesting attack to which some encryption schemes formed by one of these methods are vulnerable–the padding oracle attack–before concluding the lecture.

7.1 Preliminaries and Motivation

As the objective of authenticated encryption(AE) is to achieve both confidentiality and integrity, it is essential to first clearly understand the definitions of semantic security against a chosen plaintext attack and existential unforgeability under a chosen message attack. These will be the target properties that we desire from an AE scheme. Each of these definitions are closely tied to that of their respective adversary-challenger games.

7.1.1 Semantic Security Against a Chosen Plaintext Attack

Semantic security against chosen plaintext attacks is defined as the following:

DEFINITION 7.1. For a computational cipher $\Phi = (E, D)$ such that $E : \mathcal{K} \times \mathcal{M} \times \mathcal{N} \to \mathcal{C}$ and $D : \mathcal{K} \times \mathcal{C} \times \mathcal{N} \to \mathcal{M}$, the *semantic security against a chosen plaintext attack games* Game_b for $b \in \{0, 1\}$ between an adversary A and challenger C are defined as the following:

1. C selects $k \leftarrow_\$ \mathcal{K}$.
2. A selects a pair of messages $m_0, m_1 \in \mathcal{M}$ of the same length and sends it to C. C responds to the query by sampling a random nonce $n \leftarrow_\$ \mathcal{N}$, computing $c \leftarrow E(k, m_b, n)$ and sending it to A.
3. A outputs its guess $b' \leftarrow \{0, 1\}$ for b.

$\Phi = (E, D)$ is *semantically secure against a chosen plaintext attack* if for any efficient adversary A, the semantic security advantage $|\Pr[b' = 1|\mathsf{Game}_0] - \Pr[b' = 1|\mathsf{Game}_1]|$ of A is negligible. For convenience, let us refer to this as Φ being *CPA-secure*.

7.1.2 Existential Unforgeability Under a Chosen Message Attack

Existential unforgeability against a chosen message attack is defined as the following:

DEFINITION 7.2. For a MAC system $I = (S, V)$ such that $S : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ and $V : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0, 1\}$, the *existential unforgeability against a chosen message attack game* Game between an adversary A and a challenger C is defined as follows:

1. C samples a random key $k \leftarrow_\$ \mathcal{K}$.
2. A submits a sequence of queries to C. For $i = 1, 2, \cdots$, the ith signing query is a message $m_i \in \mathcal{M}$ of A's selection. For each ith query, C computes a tag $t_i \leftarrow_\$ S(k, m_i)$ and sends it to A.
3. A outputs a candidate forgery pair $(m, t) \in (\mathcal{M} \times \mathcal{T}) \setminus \{(m_1, t_1), (m_2, t_2), \cdots\}$.

$I = (S, V)$ is *existentially unforgeable under a chosen message attack* if for any efficient adversary A, the advantage $\Pr[V(k, m, t) = 1|\mathsf{Game}]$ of A is negligible. For convenience, let us refer to this as I being CMA-secure.

7.1.3 Motivating Example

Consider the following example. Let $\Phi = (E, D)$ be a CPA-secure block cipher over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ in CBC mode with random initialization vectors. Suppose Alice and Bob have a shared secret key $k \in \mathcal{K}$ and Alice is trying to send Bob a single-block message $m = m_0 \in \mathcal{M}$ securely using Φ. The encryption of m with k and a randomly sampled initialization vector IV results in a ciphertext of the form $c = IV||c_0$, where $c_0 = E(k, m_0 \oplus IV)$, which Alice sends to Bob for Bob to decrypt. Due to the CPA-security of Φ, any eavesdropping adversary Eve who intercepts the ciphertext c should not be able to (efficiently) obtain information about the message m. However, a more active adversary Mallory can implement the following attack in order to disrupt communication of the message from Alice to Bob:

1. Possibly by posing to be Bob, Mallory intercepts the ciphertext $c = IV||c_0$ that Alice attempts to send to Bob.

2. Now possibly posing as Alice, Mallory sends a corrupted ciphertext $c' = (IV \oplus 1)||c_0$ to Bob.

Recall that Bob would originally recover Alice's message by computing $m_0 = D(k, c_0) \oplus IV$. Upon receiving c' in place of c, however, the supposed plaintext recovered by Bob is then $m_0' = D(k, c_0) \oplus (IV \oplus 1) = m_0 \oplus 1$ instead of m_0, and Mallory succeeds in tampering with the message. This illustrates that a block cipher Φ, even if it is CPA-secure, is not sufficient by itself to provide message integrity.

In fact, we desire something even stronger from Φ than just being able to resist the attack described above. We would like to require that Bob reject any supposed ciphertext forged by an (efficient) adversary that does not have access to the key k. On the other hand, for a CMA-secure MAC $I = (S, V)$ over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, say that Alice sends $(m, t = S(k, m))$ over to Bob. Clearly, this does not satisfy any form of confidentiality as any eavesdropping adversary can learn all sorts of information about the "secret" message m. In order for Alice to be able to securely send a message to Bob without leaking any information and at the same time letting Bob know that it is indeed she who is sending the message, we require an encryption mechanism that is beyond simply either a CPA-secure cipher or a CMA-secure MAC. This role is filled by authenticated encryption schemes.

7.2 Ciphertext Integrity and Authenticated Encryption

A secure authentication encryption scheme is one that has both CPA-security and ciphertext integrity–no efficient adversary can generate a ciphertext that the receiver will perceive as valid and decrypt.

Definition 7.3. Let $\Phi = (E, D)$ be a computational cipher, with encryption algorithm $E : \mathcal{K} \times \mathcal{M} \times \mathcal{N} \to \mathcal{C}$ and decryption algorithm $D : \mathcal{K} \times \mathcal{C} \times \mathcal{N} \to \mathcal{M} \cup \{\perp\}$. Consider the following game Game between an adversary A and challenger C:

1. C selects a random key $k \leftarrow_\$ \mathcal{K}$.

2. A submits a sequence of queries to C. For $i = 1, 2, \cdots$, the ith query consists of a message $m_i \in \mathcal{M}$. For each ith query, C computes $c_i \leftarrow_\$ E(k, m_i)$ and sends c_i to A.

3. A outputs a candidate ciphertext $c \in \mathcal{C} \setminus \{c_1, c_2, \cdots\}$ and sends it to C.

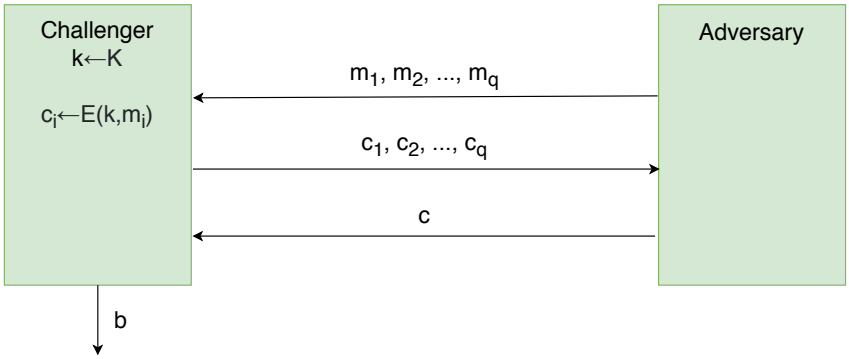

FIGURE 7.1: Confidential Integrity

4. C outputs $b \in \{0,1\}$, such that $b = 1$ if $D(k,c) \neq \perp$ and $b = 0$ otherwise.

$\Phi = (E,D)$ provides *ciphertext integrity*(CI), if for any efficient adversary A, the advantage $|\Pr[b = 1|\mathsf{Game}]|$ is negligible.

DEFINITION 7.4. A cipher $\Phi = (E,D)$ provides *authenticated encryption*(AE), or is simply *AE-secure*, if

1. Φ is semantically secure under a chosen plaintext attack, and
2. Φ provides ciphertext integrity.

By this definition, we have from our motivating example in Subsection 7.1.3 that CBC mode block ciphers with random initialization vectors are not AE-secure. There are generic ways through which we can achieve AE-security by employing some combination of CPA-secure encryption schemes and CMA-secure MACs, which we will discuss in Section 7.4.

7.3 Implications

Let us then consider the implications of a cipher $\Phi = (E,D)$ over $(\mathcal{K},\mathcal{M},\mathcal{C})$ being AE-secure. It turns out that for confidentiality, we end up having something that is stronger than our original target of CPA-security.

7.3.1 Authentication

By defining AE-security to include ciphertext integrity, we are able to defend against tampering attacks such as the one described in the previous section. Suppose that Alice is trying to send messages to Bob using Φ, and consider an adversary, say Mallory, that has "CPA-adversary" power. In other words, we are assuming that Mallory has access to some kind of encryption oracle–for example, let us say that Mallory can send a sequence of messages $m_1, m_2, \cdots$ to Alice and get her to respond with the corresponding ciphertexts $c_i = E(k, m_i)$. As a result of Φ being AE-secure, Mallory, without knowing the secret key

k herself, should not be able to generate a new ciphertext $c \notin \{c_1, c_2, \cdots\}$, such that Bob would decrypt to anything other than $\perp$.

7.3.2 Semantic Security Against a Chosen Ciphertext Attack

In terms of confidentiality, any secure authenticated encryption scheme satisfies a notion of security that is stronger than CPA-security, called semantic security under a chosen ciphertext attack, that we will refer to as CCA-security for short. It is based on a more conservative assumption about the power that an adversary may have, one that is closer to that of certain real-world adversaries.

Consider the following example: Suppose an adversary intercepts and wants to decrypt an encrypted message of the form $\mathsf{dest} = 80||\mathsf{data}$, where $\mathsf{dest} = 80$ specifies the originally intended target of the message. The adversary could then maybe fool a server into decrypting and sending them the contents of data if they were to replace the value of dest with something of their own choosing, say 25, and send $\mathsf{dest} = 25||\mathsf{data}$ to the server. If the messages are encrypted and decrypted via a CPA-secure CBC mode block cipher with random IVs such as the example in Section 7.1, the attack described in Subsection 7.1.3 would be precisely a way to do that; the adversary could replace the IV attached to the original encryption of $80||\mathsf{data}$ with $IV \oplus 80 \oplus 25$. The forged ciphertext would then be decrypted by the server as $80 \oplus (80 \oplus 25)||\mathsf{data} = 25||\mathsf{data}$, as was intended by the adversary.

The example above demonstrates how an adversary may be able to decrypt certain ciphertexts, and necessitates a new notion of secrecy that prevents the leaking of messages from adversaries that can not only 1) obtain the encryptions of arbitrary messages(usual CPA adversaries), but also 2) decrypt any ciphertext other than the one that their goal is to decrypt. This is a more conservative threat model than that of CPA-security, and we refer to any encryption scheme that can withhold information from such adversaries as being *semantically secure under a chosen ciphertext attack*.

Definition 7.5. Let $\Phi = (E, D)$ be a computational cipher over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. For $b \in \{0, 1\}$, define the following games Game_b between an adversary A and challenger C:

1. C selects a random key $k \leftarrow_\$ \mathcal{K}$.
2. A makes a series of decryption queries to C. For each $i = 1, 2, \cdots$, the ith query consists of a ciphertext c^i. C responds to the ith query by computing $m^i \leftarrow D(k, c^i)$ and sending it to A.
3. A selects a pair of messages $m_0, m_1 \in \mathcal{M}$ of the same length and sends them to C. C responds to the query by computing $c \leftarrow E(k, m_b)$ and sending it to A.
4. A makes another series of decryption queries to C. This time, the ciphertexts that A is querying must be from $\mathcal{C} \setminus \{c\}$.
5. A outputs its guess $b' \leftarrow \{0, 1\}$ for b.

Φ is *semantically secure against a chosen ciphertext attack* if for any efficient adversary A, A's advantage $|\Pr[b' = 1|\mathsf{Game}_0] - \Pr[b' = 1|\mathsf{Game}_1]|$ is negligible. For simplicity, let us refer to that is Φ being CCA-secure.

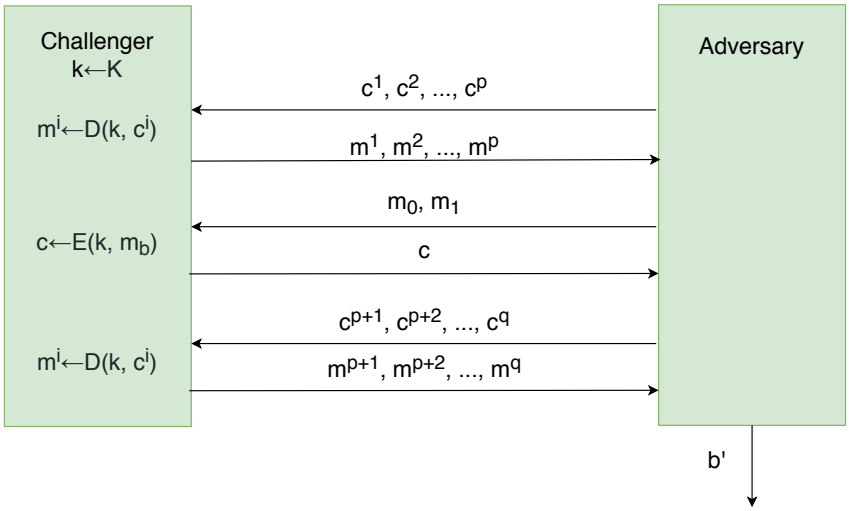

FIGURE 7.2: CCA-Security Game_b

REMARK 7.6. The adversary's ability to make decryption oracle queries after the encryption oracle query in step 3 makes this an *adaptive* version of CCA-security.

While an AE-secure encryption scheme is necessarily CCA-secure, the converse is not true. A scheme can be CCA-secure and yet not AE-secure when an adversary may be able to construct new ciphertexts c that decrypt to something other than $\perp$, but never anything that is related to its decryption oracle queries c_i and their corresponding responses m_i. The example of Section 7.1 is not CCA-secure, since by the attack previously described, upon receiving ciphertext $c \leftarrow E(k, m_b)$, an adversary should be able to forge a ciphertext c' that decrypts to $D(k, c') = m_b \oplus 1$. Then by querying the decryption oracle for c', the adversary can always deduce b by comparing m_0, m_1 with $m_b = D(k, c') \oplus 1$. The non-CCA-security of this scheme then implies that it is not AE-secure.

The following theorem states and proves our earlier claim that all AE-secure schemes are CCA-secure:

THEOREM 7.7. *Any secure authentication encryption scheme $\Phi = (E, D)$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ is semantically secure under a chosen ciphertext attack.*

Proof. Recall our previous definitions for the CCA-security games $\mathsf{Game}_b (b \in \{0, 1\})$. Now define the following games $\mathsf{Game'}_b$ between adversary A and challenger C that differ from Game_b only in steps 2 and 4:

1. C samples a random key $k \leftarrow_\$ \mathcal{K}$.
2. A makes a series of decryption queries to C. For each $i = 1, 2, \cdots$, the ith query is for a ciphertext c^i. C **responds to all queries by returning $\perp$ to A.**
3. A selects a pair of messages $m_0, m_1 \in \mathcal{M}$ of the same length and sends them to C. C responds to the query by computing $c \leftarrow E(k, m_b)$ and sending it to A.

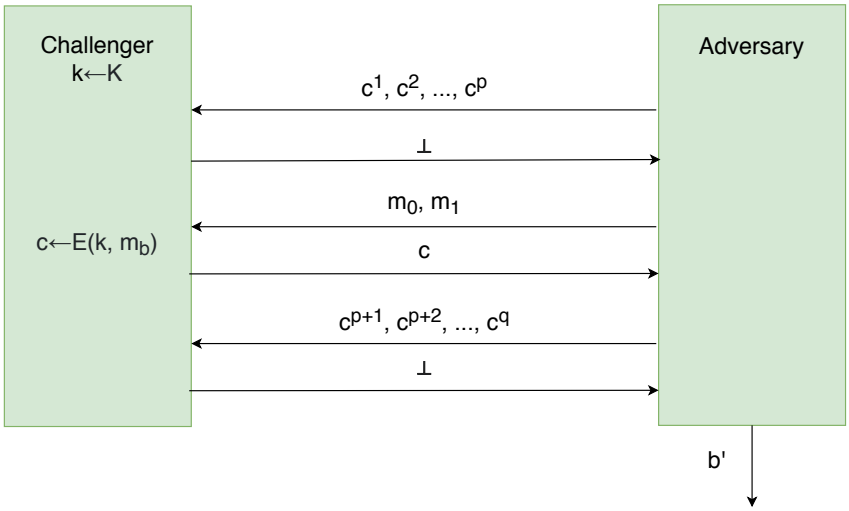

FIGURE 7.3: Game'_b

4. A makes another series of decryption queries to C. This time, the ciphertexts that A is querying all come from $\mathcal{C} \setminus \{c\}$. **As in step 2,** C **responds to all queries with** $\perp$**.**

5. A outputs its guess $b' \leftarrow \{0, 1\}$ for b.

Due to the ciphertext integrity of Φ, for each decryption oracle query in Game_b, there is at most some negligible probability ϵ_{CI} such that the response to the ith query $m^i = D(k, c^i) \neq \perp$. Being otherwise would imply that A already has some way to generate a ciphertext c^i that decrypts to something other than $\perp$ with non-negligible probability. If A makes q_D decryption oracle queries in total, then by the union bound, the games Game_b and Game'_b can only go differently with probability at most $q_D \cdot \epsilon_{CI}$. Thus $|\Pr[b' = 1|\mathsf{Game}_b] - \Pr[b' = 1|\mathsf{Game}'_b]| \leq q_D \cdot \epsilon_{CI}$.

Now observe that in the games Game'_b, the decryption oracles do not help the adversary in any way. The A still has no way of "accessing the key" whatsoever, and is playing a game that is essentially no different from the corresponding CPA-security game. Then from the CPA-security of Φ, we have that $|\Pr[b' = 1|\mathsf{Game}'_0] - \Pr[b' = 1|\mathsf{Game}'_1]| \leq \epsilon_{CPA}$ for some negligible ϵ_{CPA}.

Combining these inequalities, we can conclude that
$|\Pr[b' = 1|\mathsf{Game}_0] - \Pr[b' = 1|\mathsf{Game}_1]| \leq |\Pr[b' = 1|\mathsf{Game}_0] - \Pr[b' = 1|\mathsf{Game}'_0]| + |\Pr[b' = 1|\mathsf{Game}'_0] - \Pr[b' = 1|\mathsf{Game}'_1]| + |\Pr[b' = 1|\mathsf{Game}_1] - \Pr[b' = 1|\mathsf{Game}'_1]| \leq 2q_D \cdot \epsilon_{CI} + \epsilon_{CPA}$, and thus that Φ is CCA-secure. □

7.3.3 Limitations

While authenticated encryption provides us with confidentiality against active adversaries that can decrypt ciphertexts, that does not mean that AE-secure schemes are invulnerable to any kind of adversarial attack. Authenticated encryption schemes have their limitations

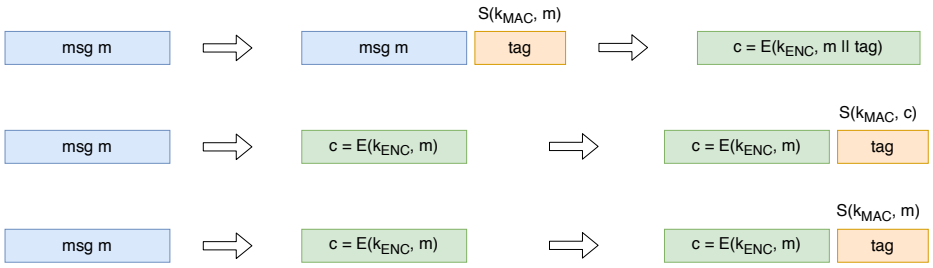

FIGURE 7.4: MAC-Then-Encrypt, Encrypt-Then-MAC, Encrypt and MAC (top to bottom)

in that AE-security is not enough to prevent real-world attacks including replay attacks and side channel(timing) attacks. These attacks are topics that we may cover in future lectures.

7.4 Authenticated Encryption by Generic Composition

Authenticated encryption, while not without its own limitations, is definitely a stronger version of security than that which we have been requiring from encryption schemes or MACs. Thankfully, however, we do not have to build such more secure encryption schemes from scratch. As AE-security requires basically a combination of confidentiality(CPA-security) and integrity(CI), we can construct AE-secure schemes by combining CPA-secure encryption schemes with CMA-secure MACs. There are three methods for doing this that easily come to mind:

1. using a MAC to generate a tag for a given message and then encrypting the entire message-tag pair;
2. encrypting the message first and using a MAC system to generate a tag for the ciphertext; and
3. encrypting the message while also generating a MAC tag for it and sending both the ciphertext and the tag to the receiver.

7.4.1 MAC-Then-Encrypt

The first approach is referred to as *MAC-then-encrypt*, or MtE for short. Suppose we have a CMA-secure MAC system $I = (S, V)$ over $(\mathcal{K}_{MAC}, \mathcal{M}, \mathcal{T})$ and a CPA-secure encryption scheme $\Phi = (E, D)$ over $(\mathcal{K}_{ENC}, \mathcal{M} \times \mathcal{T}, \mathcal{C})$. If Alice wants to send messages to Bob, they must first both share a MAC key $k_{MAC} \leftarrow \mathcal{K}_{MAC}$ and an encryption key $k_{ENC} \leftarrow \mathcal{K}_{ENC}$, sampled randomly and independently from their respective domains. Say Alice wants to send the message m to Bob. Alice first computes the tag $t \leftarrow S(k_{MAC}, m)$, then encrypts the message-tag pair (m, t) with the encryption key to obtain $c \leftarrow E(k_{ENC}, (m, t))$, which is what she sends to Bob. Upon receiving such a ciphertext, Bob first decrypts it using his encryption key to get $(m', t') \leftarrow D(k_{ENC}, c)$, before using his MAC key to compute $\mathsf{accept/reject} \leftarrow V(k_{MAC}, m', t')$ to verify the message.

A cipher constructed using the MAC-then-encrypt approach is not generally secure, as we will be able to show via the padding oracle attack of Section 7.5. MtE ciphers are not

necessarily even CCA-secure in general. However, when the encryption scheme Φ that we are working with has special properties, such as being a rand-CTR mode or rand-CBC mode block cipher, the resulting MtE scheme may provide authenticated encryption. In the case that Φ is a rand-CTR mode cipher, even I being a one-time MAC is sufficient for the MtE scheme to be AE-secure. Well-known examples of AE-secure MtE schemes can be found in the SSL/TLS protocols.

7.4.2 Encrypt-Then-MAC

The second method of composition, called *encrypt-then-MAC* or EtM, on the other hand, necessarily results in a scheme that is AE-secure. Suppose Φ and I are a CPA-secure encryption scheme and a CMA-secure MAC, respectively, as they were in the subsection. This time, however, Φ is over $(\mathcal{K}_{ENC}, \mathcal{M}, \mathcal{C})$, while I is over $(\mathcal{K}_{MAC}, \mathcal{C}, \mathcal{T})$. Whenever Alice wants to send a message m to Bob, she first encrypts m using her encryption key to get $c \leftarrow E(k_{ENC}, m)$. Then she signs it with her MAC key by computing the tag $t \leftarrow S(k_{MAC}, c)$, and sends over (c, t) to Bob. Once Bob receives this (c, t), he first verifies it using his MAC key via $\mathsf{accept/reject} \leftarrow V(k_{MAC}, c, t)$. If V rejects, then Bob receives $\perp$ as his message. Otherwise, Bob decrypts the ciphertext using his decryption key to obtain the message $m \leftarrow D(k_{ENC}, c)$.

THEOREM 7.8. *Let $\Phi = (E, D)$ be a cipher over $(\mathcal{K}_{ENC}, \mathcal{M}, \mathcal{C})$ and $I = (S, V)$ be a MAC system over $(\mathcal{K}_{MAC}, \mathcal{C}, \mathcal{T})$. Then the encryption scheme Φ_{AE} described above is AE-secure assuming that Φ is CPA-secure and I is CMA-secure.*

The following is a proof for the above theorem found in the Boneh-Shroup textbook.

Proof. Let us first prove that Φ_{AE} is CI. Suppose instead that there is an adversary A_{CI} that breaks the ciphertext integrity of Φ_{AE} with some non-negligible advantage ϵ_{CI}. Consider the following adversary B playing the CMA-security game for I against challenger C:

1. B runs an instance of A_{CI}, playing the role of challenger for A_{CI}.
2. B selects a random encryption key $k'_{ENC} \leftarrow_{\$} \mathcal{K}$.
3. Whenever B receives a query $m_i \in \mathcal{M}$ from A_{CI}, it first computes $c_i \leftarrow E(k'_{ENC}, m_i)$. Then B queries C on c_i and obtains $t_i \leftarrow S(k_{MAC}, c_i)$ in response. B responds to A_{CI}'s query with (c_i, t_i).
4. When A_{CI} eventually outputs a ciphertext $(c, t) \in \mathcal{C} \times \mathcal{T}$, B outputs (c, t) as its own candidate message-tag pair.

Clearly, from A_{CI}'s point of view, it is playing the ciphertext integrity game for Φ_{AE} where B, with the help of its oracle queries to C, is playing the role of challenger. Then with probability ϵ_{CI}, A_{CI} outputs a ciphertext (c, t) that is does not decrypt to $\perp$, which means that $\mathsf{accept} \leftarrow V(k_{MAC}, c, t)$. Thus B is able to generate a message-tag pair (c, t) that breaks the CMA-security of I with non-negligible probability of at least ϵ_{CI}. This contradicts our assumption of the CMA-security of I, and therefore no such adversary A_{CI} can exist that breaks the CI of Φ_{AE}. In other words, Φ_{AE} provides ciphertext integrity.

As for the CPA-security of Φ_{AE}, notice that a ciphertext of Φ_{AE} is simply a ciphertext of Φ paired with a tag generated for that ciphertext using the MAC key k_{MAC}. Recall that k_{ENC} and k_{MAC} were sampled independently from their respective domains. Then

the added bonus of having a tag for the ciphertext does not in any way give the adversary any hints about decryption or access to the key k_{ENC}. The tag is simply a predicate of the ciphertext that anyone with knowledge of $I = (S, V)$ and k_{MAC} could have computed. Thus it is straightforward that Φ_{AE} retains the CPA-security of its component Φ.

Consequentially, Φ_{AE} is a secure authenticated encryption scheme. □

The IPSec protocol suite employs authenticated encryption schemes formed in this manner.

7.4.3 Encrypt and MAC

The third and last approach is named *encrypt and MAC*, or EaM. For Φ and I as defined in the last two subsections, this time over $(\mathcal{K}_{ENC}, \mathcal{M}, \mathcal{C})$ and $(\mathcal{K}_{MAC}, \mathcal{M}, \mathcal{T})$, respectively, the following depicts how Alice would send a message $m \in \mathcal{M}$ to Bob. Alice uses each of her keys k_{ENC} and k_{MAC} to compute $c \leftarrow E(k_{ENC}, m)$ and $t \leftarrow S(k_{MAC}, m)$, respectively. She sends Bob (c, t), which Bob then first decrypts with k_{ENC} to obtain $m \leftarrow D(k_{ENC}, c)$ and then verifies with $\mathsf{accept/reject} \leftarrow V(k_{MAC}, m, t)$. If V rejects, Bob outputs $\perp$ as his decrypted message; otherwise he should retrieve the message m output by D.

It is easy to see that unless the MAC system I has mechanisms of its own to provide confidentiality, there is no guarantee that the tag $t = S(k_{MAC}, m)$ will not leak information about the secret message m. Therefore an EaM construction does not provide AE-security, given the CPA- and CMA-security of its component systems. Notably, the SSH protocol uses a secure AE scheme constructed via this EaM approach. It is AE-secure, however, because the MAC system it is built upon does not leak information about the plaintext, not because of any security guarantees that come from EaM-style composition itself.

7.4.4 Standards and AE with Associated Data

There are several widely accepted standards for authenticated encryption. GCM, CCM, and EAX are among those, the first two being NIST standards, and all three even support a somewhat stronger version of AE called *authenticated encryption with associated data*, or AEAD. An AEAD encrypted message consists of both some ciphertext of encrypted data and plaintext associated data. While only part of the communicated cipher is actually encrypted, both the plaintext and ciphertext portions of the message must be authenticated. This means that the plaintext associated data must be bound to the ciphertext part in a way such that even if an adversary were to replace the ciphertext with another valid ciphertext that was generated in a different context with different associated data, the receiver should be able to reject the corrupted cipher forged by joining together the original associated data with this new ciphertext.

A GCM, or Galois/Counter Mode AE scheme is an EtM composition of a random CTR mode encryption scheme with a CW-MAC. A CW-MAC, or a MAC in the style of Wegman and Carter, is one that essentially creates a tag for a message by first hashing the message, then encrypting it with a PRF. A CCM, or Counter with CBC-MAC mode AE scheme is a MtE composition of a CBC-MAC system with CTR mode encryption. An EAX mode AE scheme is an EtM composition of CTR mode encryption with a CMAC, or Cipher-based MAC.

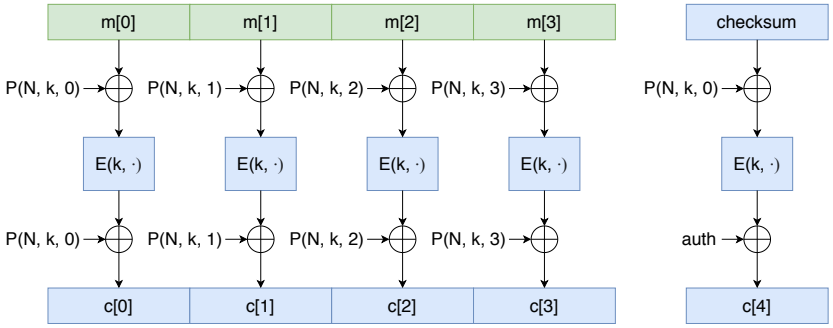

FIGURE 7.5: OCB

7.4.5 Case Study: OCB

As much as these general composition methods present us with a convenient way to build AE-secure ciphers, it is also important to note that not all authenticated encryption schemes must be constructed this way. An example that illustrates this point is the OCB, or Offset Codebook mode of authenticated encryption. An OCB mode AE scheme is built directly from a pseudorandom permutation(PRP) as can be seen in Figure 7.5. Observe that it is "built from scratch" without resorting to the generic EtM method. While OCB mode encryption is more efficient than other modes such as CCM mode, it is not used as often in practice.

7.5 Padding Oracle Attack

Recall our previous mention of secure MtE-constructed AE schemes being applied to the SSL/TLS protocols. In this section, we look into the TLS record protocol in more detail and introduce the *padding oracle attack*, an attack that exploits a vulnerability in now deprecated versions of this protocol that arise as a consequence of using a CBC mode block cipher in a MtE construction.

7.5.1 TLS Record Protocol

The TLS record protocol uses CBC mode encryption and HMAC-SHA1. As it is a MtE composition of the two, the plaintext that is input to the CBC mode block cipher is of the form $\mathsf{data}||\mathsf{tag}||\mathsf{pad}$. For the purposes of analyzing just the security of the AE scheme, let us ignore any details regarding the internal structure of data. tag is the tag generated by the MAC, and pad is padding to ensure that the padded plaintext is of such length that it can be evenly divided up into blocks for encryption. Since the intended receiver of the record should be able to identify where the data and tag end and where the padding begins, pad adheres to a specific, public format and depends only on the length of $\mathsf{data}||\mathsf{tag}$ modulo the block size. Upon receiving an encrypted record of the form $\mathsf{record} \leftarrow E(k_{ENC}, \mathsf{data}||\mathsf{tag}||\mathsf{pad})$, decryption can be broken down into 3 steps.

1. CBC decrypt the record using k_{ENC} to obtain $\mathsf{plaintext} \leftarrow D(k_{ENC}, \mathsf{record})$.
2. Check if $\mathsf{plaintext}$ is correctly padded. If it is not, abort and report "padding error".

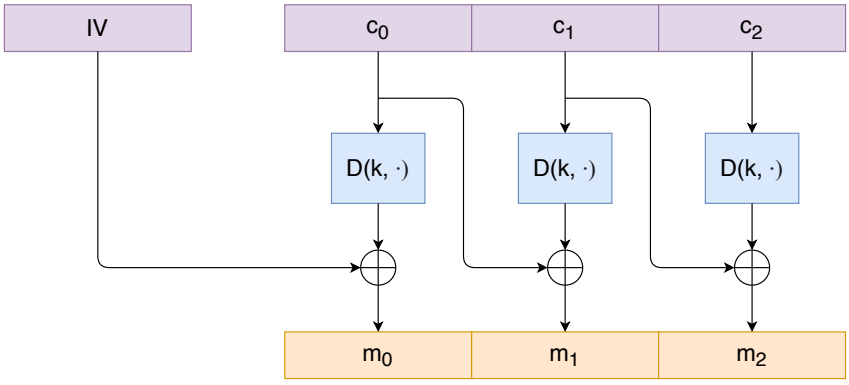

FIGURE 7.6: TLS Decryption

3. Remove the padding, separate the data and tag parts and verify the tag using k_{MAC}. If verification rejects, abort and report "MAC error". Otherwise return the data part of the plaintext.

In this decryption process lies the vulnerability exploited by the padding oracle attack. Suppose an adversary A has access to a decryption oracle, which is likely the case in the real world. While the CBC encryption and HMAC-SHA1 try to make it hard for A to generate a valid ciphertext that passes decryption, the decryption oracle gives A valuable information about even invalid ciphertexts. This is because there are two different types of error that it reports: padding and MAC. The attacker can submit a ciphertext and learn whether or not the last bytes of plaintext are a valid pad. If they do form a valid padding, the decryption oracle proceeds all the way to step 3 of the decryption process and returns "MAC error", otherwise it aborts at step 2 and returns "padding error". This is referred to as a *padding oracle*.

7.5.2 The Attack

Let us demonstrate the attack by providing an example. Suppose an attacker A has intercepted a ciphertext of 3 blocks $c = c_0||c_1||c_2$ and the corresponding initialization vector IV. We know that such a ciphertext corresponds to some 3-block plaintext $m = m_0||m_1||m_2||\mathsf{pad}$, where $m_2||\mathsf{pad}$ fills out one block. Let us say that A wants to find out the contents of m_1, and has access to a padding oracle that decrypts the CBC encryption as in Figure 7.6. Let n be the block size used for CBC encryption. For our padding scheme, assume that for a plaintext that ends with just $n-k$ bytes of data content for its last block, we have $|\mathsf{pad}| = k$ bytes and $\mathsf{pad} = k||0||0||\cdots||0$.

A begins by guessing the value of the last byte of m_1, or $m_1[n-1]$. Let $g \in \{0, \cdots, 255\}$ be A's guess for $m_1[n-1]$. A can query the decryption/padding oracle on IV and the fake ciphertext $c^1 = c_0^1||c_1$, where $c_0^1[0 \cdots (n-2)] = c_0[0 \cdots (n-2)]$ and $c_0^1[n-1] = c_0[n-1] \oplus g \oplus 1$. Observe that since c should correctly decrypt to m, we know that $m_1[n-1] = c_0[n-1] \oplus (D(k_{ENC}, c_1))[n-1]$, or equivalently, $(D(k_{ENC}, c_1))[n-1] = m_1[n-1] \oplus c_0[n-1]$. The decryption oracle should then CBC decrypt c^1 to $m^1 = m_0||m_1^1$, where $m_1^1[0 \cdots (n-2)] = m_1[0 \cdots (n-2)]$ and $m_1^1[n-1] = c_0^1[n-1] \oplus (D(k_{ENC}, c_1))[n-1] = (c_0[n-1] \oplus g \oplus 1) \oplus$

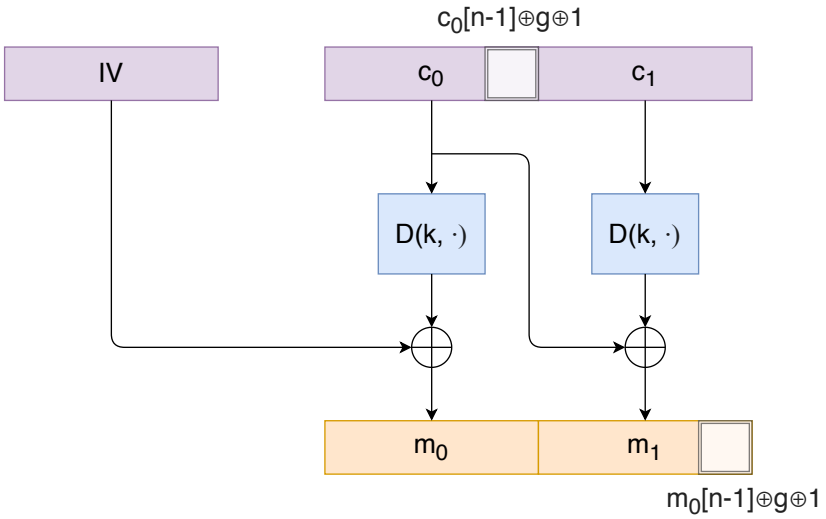

FIGURE 7.7: Padding Oracle Attack

$(m_1[n-1] \oplus c_0[n-1]) = (m_1[n-1] \oplus g) \oplus 1$. Thus if A is correct and indeed $g = m_1[n-1]$, we have $m_1^1 = m_1[0 \cdots (n-2)]||1$, which to the decryptor will appear to be a "properly padded" plaintext. Then the decryptor will proceed to verify the MAC and either return "MAC error" or the decrypted plaintext m^1 to A based on the result. If A is not as lucky and $g \neq m_1[n-1]$, then m^1 will not be a properly padded plaintext in the eyes of the decryptor, and the oracle will return the "padding error" message to A.

So depending on what the padding oracle returns, A can tell whether or not their guess g was correct. Since there are only 256 possible values of g, if A keeps checking different values of g each time, they will eventually discover the correct value of $m_1[n-1]$ after at most 256 tries.

Once A knows the value of $m_1[n-1]$, they can then conduct a similar exhaustive search for $m_1[n-2]$. A can check the correctness of a guess g by querying the oracle on $c^2 = c_0^2||c_1$, where $c_0^2[0 \cdots (n-3)] = c_0[0 \cdots (n-3)]$, $c_0^2[n-2] = c_0[n-2] \oplus g \oplus 2$, and $c_0^2[n-1] = c_0[n-1] \oplus m_1[n-1]$. This will decrypt to $m^2 = m_0||m_1^2$ with $m_1^2[0 \cdots (n-3)] = m_1[0 \cdots (n-3)]$, $m_1^2[n-2] = (m_1[n-2] \oplus g) \oplus 2$, and $m_1^2[n-1] = 0$, and the oracle will return something other than "padding error" if and only if $g = m_1[n-2]$.

Repeating this exhaustive guess-search for all n bytes of m_1 lets A recover the entirety of m_1 in at most $256n$ guesses. This is therefore a very efficient attack that allows an adversary to learn the contents of all but the last block of the plaintext, and can be applied to any MtE construction that uses CBC mode encryption in general that allows for a padding oracle. More recent versions of TLS are no longer vulnerable to this attack as they no longer return different values for "padding error"s and "MAC errors".

7.5.3 IMAP Over TLS

Finally, let us take a look at an example of a defense that does not work against the padding oracle attack. IMAP, or the Internet Message Access Protocol, is an Internet standard

protocol used by email clients to retrieve email messages from a mail server over a TCP/IP connection. For better security, the TLS protocol has the server and client renegotiate keys when an invalid record is received. When using IMAP over TLS, the client sends a login message "LOGIN {username} {password}" to the mail server every five minutes. This means that for an adversary that is attempting to break the security of IMAP over TLS, the secret key used by the encrypting and decrypting parties is effectively refreshed every 5 minutes. While one may be tempted to think that this should prevent padding oracle attacks, it does not, and an adversary using the padding oracle attack can recover a secret password in only a few hours without any knowledge of any of the keys.

This is because the progress of A's guesswork does not ever get reset or undone by the refreshing of encryption keys. In the padding oracle attack described in the previous subsection, for each byte m_ℓ of the message, A exhaustively searches through $\{0, \cdots, 255\}$ to find the correct value of that byte. For each value $g \in \{0, \cdots, 255\}$, A is able to determine whether or not $g = m_\ell$ with a single query to the padding oracle. A's behavior is never once affected by the value of k_{ENC}, other than the fact that whenever k_{ENC} changes, A will need to intercept a new ciphertext c to work with, as the old one will no longer be valid. The padding mechanism is unchanged; the values of g that A has ruled out as not m_ℓ are still not m_ℓ; and bytes of the message m that A has already recovered are still the same bytes of m. So A can simply continue on with their exhaustive search for the bytes of m.

7.5.4 Conclusion

The padding oracle attack is an efficient attack on MAC-then-CBC type constructions for encryption. AE schemes constructed via the EtM method are not at all vulnerable to this type of attack, since during decryption, the MAC is verified first and the entire ciphertext is simply discarded if the tag is invalid. As the security of the MAC system makes it infeasible for an adversary to forge fake CBC ciphertext-tag pairs, they cannot make queries to the decryption oracle to obtain any information about the CBC plaintext. Any tampered ciphertext with which they wish to query the oracle with will simply get rejected at the MAC verification stage and thrown out (with all but negligible probability). Also, as the plaintext is in this case added before CBC encryption, and the adversary cannot get the decrypting oracle to even begin CBC decryption of any corrupted ciphertext, there is no way for an adversary to use their knowledge about the padding mechanism to their advantage either.

It is also not true that the padding oracle attack disqualifies all MAC-then-CBC constructions from secure authenticated encryption schemes. As we mentioned earlier, more recent versions of TLS which still run MAC-then-CBC systems or not vulnerable to the padding oracle attack. It was the "padding oracle" that was formed as a result of the flawed decryption algorithm for TLS 1.3 that broke the AE-security of that particular MAC-then-CBC construction. Even without an explicit oracle, however, MAC-then-CBC constructions, or even other AE-secure encryption systems, can still be vulnerable to other attacks such as timing attacks, which may be discussed in future lectures.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography.* 2020.

[2] D. Khurana. CS498 AC3/AC4 Lecture Slides, 2020.