Cryptography Part VII: CCA, HMAC and Unforgeability

ECE 4156/6156 Hardware-Oriented Security and Trust

Spring 2024

Assoc. Prof. Vincent John Mooney III

Georgia Institute of Technology

Reading

- Introduction to Modern Cryptography, Chapter 3.7 (CCA-Security), Chapter 4, Chapter 5.1 and Chapter 5.3

Notation

- π_E = (Gen, Enc, Dec) is an encryption scheme
- π_M = (Gen, Mac, Vrfy) is a message authentication code or MAC
- Probabilistic Polynomial Time or PPT refers to algorithms which take at most polynomial time while having free use of a true random number generator
- $\mathrm{PrivK}^{\mathrm{cca}}_{A,\pi}(n)$ is an experiment involving a private key encryption scheme π with a key of size *n* and a PPT adversary A with access to ciphertext, an encryption oracle (without limits other than time) and a decryption oracle (but the challenge ciphertext may not be submitted)
- $H^s(x) \stackrel{\text{def}}{=} H(s, x)$ where the keyed hash function take inputs *s* and x in order to produce output *h*
  - A superscript is used for *s*, i.e., H^s, instead of a subscript, i.e., H_s in order to emphasize the fact that the typical attack surface includes scenarios where the adversary may have possession of the key

CCA-Security

- For Chosen Ciphertext Attack (CCA) security, the attacker has access to a decryption oracle
  - Experiment $\mathrm{PrivK}_{A,\pi}^{\mathrm{cca}}(n)$ is run with two messages m_0 and m_1 encrypted to c_0 and c_1 where the adversary A has to guess which message was encrypted given only the corresponding encrypted ciphertext
  - For obvious reasons, the adversary may not submit c_0 or c_1 to the decryption oracle!
- Some practical situations where partial access to a decryption oracle exists occur when error messages are provided
  - Based on which error message occurs, a CCA may commence where, for example, incorrect padding allows one to correctly guess the value of a byte
  - Padding oracle attack! (not covered this year in ECE 4156 / 6156)

Message Authentication Code (MAC) Design

- In Lecture 3, Intro to SHA-2, hash functions were introduced
  - Collision resistance
  - Target-collision resistance (a.k.a. second preimage resistance)
  - Preimage resistance
- SHA-2 is keyless (or you can say that the initial conditions are fixed)
- However, is this lecture we will introduce the concept of a MAC which is a keyed hash
- In Lecture 4, Authentication I, it was observed that typically what is meant by "Message Authentication" in a MAC is in fact message integrity, i.e., verification that a message has not been altered after being sent

MAC Definition

- A Message Authentication Code (MAC) π_{M} is composed of three PPT functions Gen, Mac and Vrfy
- As with an encryption scheme π_{E}, Gen generates a key
  - We will denote the key for π_{M} as k_{M}
  - As with symmetric key encryption, we assume that key k_{M} is provided to both parties (e.g., Alice and Bob) without being revealed to the adversary
- $\mathrm{Mac}_{k_{\mathrm{M}}}(m)$ takes as input a message m and uses k_{M} to output a tag t
- $\mathrm{Vrfy}_{k_{\mathrm{M}}}(m,t)$ takes as inputs message m and tag t
  - $\mathrm{Vrfy}_{k_{\mathrm{M}}}$ uses k_{M} to output a '1' if tag t corresponds to message m
  - Otherwise $\mathrm{Vrfy}_{k_{\mathrm{M}}}$ outputs a '0'

Verification that a Message is Unaltered

- The concept of a verifier Vrfy can also, in principle, be applied to keyless hashes, e.g., SHA2 or SHA3
- For a keyless hash such as SHA2 it is assumed that the tag t and message m are not easily replaced in transit (since the adversary clearly can calculate a new tag!)
  - One possibility is to send tag t encrypted
- In this case there is no key k_M used to compute tag t given message m
- In this case (which is not included in Katz and Lindell!) $\mathrm{Vrfy}(m,t)$ verifies if the appropriate keyless hash when provided message m as input gives as output tag t
- Canonical verification occurs with deterministic MACs and keyless hashes when the verifier simply recomputes tag t and checks for equality

(c_0, c_1, c_2, c_3)

c_1, t

Toward the formal definition, consider the following experiment for a message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$, an adversary $\mathcal{A}$, and value n for the security parameter:

The message authentication experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$**:**

1. *A key k is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to* $\mathsf{Mac}_k(\cdot)$. *The adversary eventually outputs (m, t). Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its oracle.*
3. $\mathcal{A}$ succeeds *if and only if (1)* $\mathsf{Vrfy}_k(m, t) = 1$ *and (2)* $m \notin \mathcal{Q}$. *In that case the output of the experiment is defined to be 1.*

Existentially Unforgeable under an Adaptive Chosen-Message Attack

- Given π_{M} and adversary A, $\mathrm{Mac\text{-}forge}_{A,\pi_{\mathrm{M}}}(n)$ checks to see if A can come up with a valid MAC tag *t* given message *m* and oracle access to $\mathrm{Mac}_{k_{\mathrm{M}}}$ except that *m* may not be submitted to the oracle
  - The requirement that $m \notin Q$, where Q is the set of all oracle queries, enforces that *m* may not be submitted to the oracle
- A tag is *existentially unforgeable*[1] for an arbitrary message *m* if an adversary has only a negligible change of generating a valid tag *t* given only message *m* (and, of course, no access to key k_{M}, i.e., a keyless hash does not fit this experiment)
  - The *adaptive chosen-message attack*[1] refers to the adversary's ability to arbitrarily choose message *m* during the attack itself, e.g., by adding spaces or commas to a legal statement contained in a message-e
  - The oracle access of the attacker models the case where the attacker can induce some messages (other than *m*) and obtain their corresponding tags

[1] Page 112 of Katz and Lindell.

DEFINITION 4.2 *A message authentication code* π_M = (Gen, Mac, Vrfy) *is* **existentially unforgeable under an adaptive chosen-message attack***, or just* **secure***, if for all* PPT *adversaries* A *there is a negligible function* negl *such that, for all n,*

$$\Pr\left[\text{Mac–forge}_{A,\pi_M}(n) = 1\right] \le \text{negl}(n).$$

Replay Attacks

- replay attacks are not prevented
- need to protect against replay attacks in the overall protocol

Replay Attacks

- Note that as presented the verifier has no access to any kind of history or record of previous messages
- Without any notion of state, the protocols presented will not be able to prevent replay attacks
- In practice, the two most popular approaches to prevent replay attacks are (i) use of a counter and (ii) use of a timestamp
- Use of a counter has the problem of synchronization
- Use of a timestamp has the problem of slack or clock skew
  - Attacks that are “fast enough” (i.e., within acceptable skew) may succeed
- Katz and Lindell pages 113-114

if Gen omitted, assume key from uniform dist: TRNG

CONSTRUCTION 4.18

Let $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme and let $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code, where in each case key generation is done by simply choosing a uniform n-bit key. Define a private-key encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Gen': on input 1^n, choose independent, uniform $k_E, k_M \in \{0,1\}^n$ and output the key (k_E, k_M).
- Enc': on input a key (k_E, k_M) and a plaintext message m, compute $c \leftarrow \mathsf{Enc}_{k_E}(m)$ and $t \leftarrow \mathsf{Mac}_{k_M}(c)$. Output the ciphertext $\langle c, t \rangle$.
- Dec': on input a key (k_E, k_M) and a ciphertext $\langle c, t \rangle$, first check whether $\mathsf{Vrfy}_{k_M}(c, t) \stackrel{?}{=} 1$. If yes, then output $\mathsf{Dec}_{k_E}(c)$; if no, then output $\perp$.

Error

A generic construction of an authenticated encryption scheme.

Throughout, let $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure encryption scheme and let $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ denote a message authentication code, where key generation in both schemes simply involves choosing a uniform n-bit key. There are three natural approaches to combining encryption and message authentication using independent keys[4] k_E and k_M for Π_E and Π_M, respectively:

1. *Encrypt-and-authenticate:* In this method, encryption and message authentication are computed independently in parallel. That is, given a plaintext message m, the sender transmits the ciphertext $\langle c, t \rangle$ where:

$$c \leftarrow \mathsf{Enc}_{k_E}(m) \quad \text{and} \quad t \leftarrow \mathsf{Mac}_{k_M}(m).$$

   The receiver decrypts c to recover m; assuming no error occurred, it then verifies the tag t. If $\mathsf{Vrfy}_{k_M}(m, t) = 1$, the receiver outputs m; otherwise, it outputs an error.

[4] *Independent* cryptographic keys should always be used when different schemes are combined. We return to this point at the end of this section.

2. *Authenticate-then-encrypt:* Here a MAC tag t is first computed, and then the message and tag are encrypted together. That is, given a message m, the sender transmits the ciphertext c computed as:

$$t \leftarrow \mathsf{Mac}_{k_M}(m) \quad \text{and} \quad c \leftarrow \mathsf{Enc}_{k_E}(m\|t).$$

The receiver decrypts c to obtain $m\|t$; assuming no error occurs, it then verifies the tag t. As before, if $\mathsf{Vrfy}_{k_M}(m,t) = 1$ the receiver outputs m; otherwise, it outputs an error.

3. *Encrypt-then-authenticate:* In this case, the message m is first encrypted and then a MAC tag is computed over the result. That is, the ciphertext is the pair $\langle c, t \rangle$ where:

$$c \leftarrow \mathsf{Enc}_{k_E}(m) \quad \text{and} \quad t \leftarrow \mathsf{Mac}_{k_M}(c).$$

(See also Construction 4.18.) If $\mathsf{Vrfy}_{k_M}(c, t) = 1$, then the receiver decrypts c and outputs the result; otherwise, it outputs an error.

Keyed MAC	Adv.	Dis adv.
1. Enc and ~~then~~ Auth	run encr. + MAC in parallel (=> faster computation)	cryptanalysis on tag can potentially reveal info about m
2 Authenticate then Encrypt		~~(" ")~~ m‖t provides a relationship which may be exploited ~~by~~
3. Encrypt then Authenticate	information leakage through the tag reveals nothing about the message	

Unkeyed MAC

tag generated on ciphertext can be recomputed by an adversary for change in any ciphertext block

Let's leave unkeyed MACs for later discussion ...

Later Discussion!

- For a keyless hash intended to attest to the integrity of a message, which of the three approaches to combine encryption and message integrity are preferred and why?
- 1) Encrypt-and-authenticate
  - $c := Enc_k(m)$, $t := H^s(m)$, send <c,t>
- 2) Authenticate-then-encrypt
  - $t := H^s(m)$, $c := Enc_k(m \parallel t)$, send <$c$>
- 3) Encrypt-then-authenticate
  - $c := Enc_k(m)$, $t := H^s(c)$, send <c,t>

DEFINITION 4.16 *A private-key encryption scheme* Π *is* unforgeable *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *there is a negligible function* negl *such that:*

$$\Pr[\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

Paralleling our discussion about verification queries following Definition 4.2, here one could also consider a stronger definition in which $\mathcal{A}$ is additionally given access to a decryption oracle. One can verify that the secure construction we present below also satisfies that stronger definition.

We now define a (secure) *authenticated encryption* scheme.

DEFINITION 4.17 *A private-key encryption scheme is an* authenticated encryption scheme *if it is CCA-secure and unforgeable.*

Forgeability Experiment

- Fix A, $\Pi_M(n)$
- Experiment $\text{Forge}_{A,\Pi_M}(n)$
  1. $k_M \leftarrow \text{Gen}(1^n)$
  2. Adversary A interacts w/ MAC_{k_M} (let M be a set of messages submitted to the oracle)
  3. A outputs m, t
  4. A succeeds if $\text{Vrfy}(m,t) = 1$ and $m \notin M$

The unforgeable encryption experiment $\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n)$**:**

1. *Run* $\mathsf{Gen}(1^n)$ *to obtain a key* k.
2. *The adversary* $\mathcal{A}$ *is given input* 1^n *and access to an encryption oracle* $\mathsf{Enc}_k(\cdot)$. *The adversary outputs a ciphertext* c.
3. *Let* $m := \mathsf{Dec}_k(c)$, *and let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *asked its encryption oracle. The output of the experiment is* 1 *if and only if (1)* $m \neq \perp$ *and (2)* $m \notin \mathcal{Q}$.

4.4.1 The Basic Construction

CBC-MAC is a standardized message authentication code used widely in practice. A basic version of CBC-MAC, secure when authenticating messages of any *fixed* length, is given as Construction 4.11. (See also Figure 4.1.) We caution that this basic scheme is *not* secure in the general case when messages of different lengths may be authenticated; see further discussion below.

CONSTRUCTION 4.11

Let F be a pseudorandom function, and fix a length function $\ell > 0$. The basic CBC-MAC construction is as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$, do the following (we set $\ell = \ell(n)$ in what follows):

  1. Parse m as $m = m_1, \ldots, m_\ell$ where each m_i is of length n.
  2. Set $t_0 := 0^n$. Then, for $i = 1$ to ℓ:
     Set $t_i := F_k(t_{i-1} \oplus m_i)$.

  Output t_ℓ as the tag.

- Vrfy: on input a key $k \in \{0,1\}^n$, a message m, and a tag t, do: If m is not of length $\ell(n) \cdot n$ then output 0. Otherwise, output 1 if and only if $t \stackrel{?}{=} \mathsf{Mac}_k(m)$.

Basic CBC-MAC (for fixed-length messages).

Secure CBC-MAC for arbitrary-length messages. We briefly describ two ways Construction 4.11 can be modified, in a provably secure fashior to handle arbitrary-length messages. (Here for simplicity we assume that a messages being authenticated have length a multiple of n, and that Vrfy reject

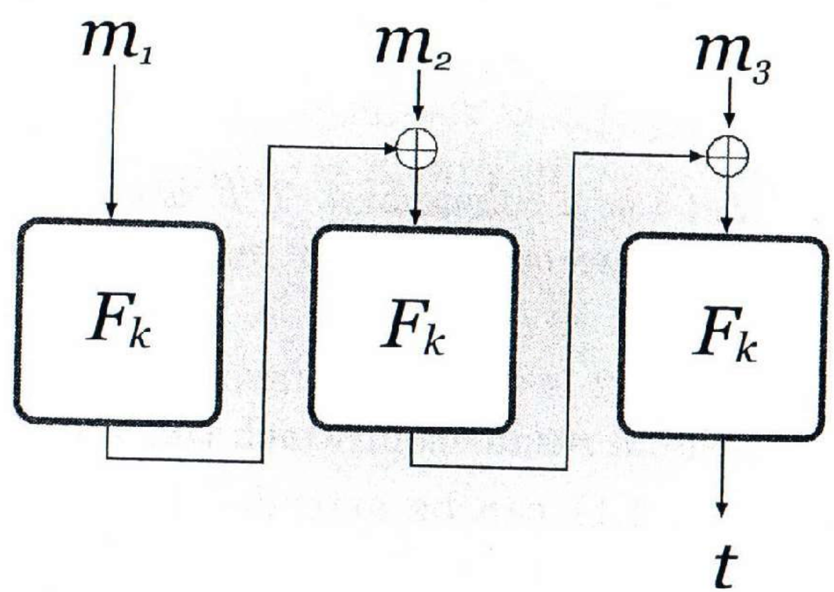

FIGURE 4.1: Basic CBC-MAC (for fixed-length messages).

CONSTRUCTION 4.5

Let F be a pseudorandom function. Define a fixed-length MAC for messages of length n as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, output the tag $t := F_k(m)$. (If $|m| \neq |k|$ then output nothing.)
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^n$, and a tag $t \in \{0,1\}^n$, output 1 if and only if $t \stackrel{?}{=} F_k(m)$. (If $|m| \neq |k|$, then output 0.)

A fixed-length MAC from any pseudorandom function.

THEOREM 4.6 *If F is a pseudorandom function, then Construction 4.5 is a secure fixed-length MAC for messages of length n.*

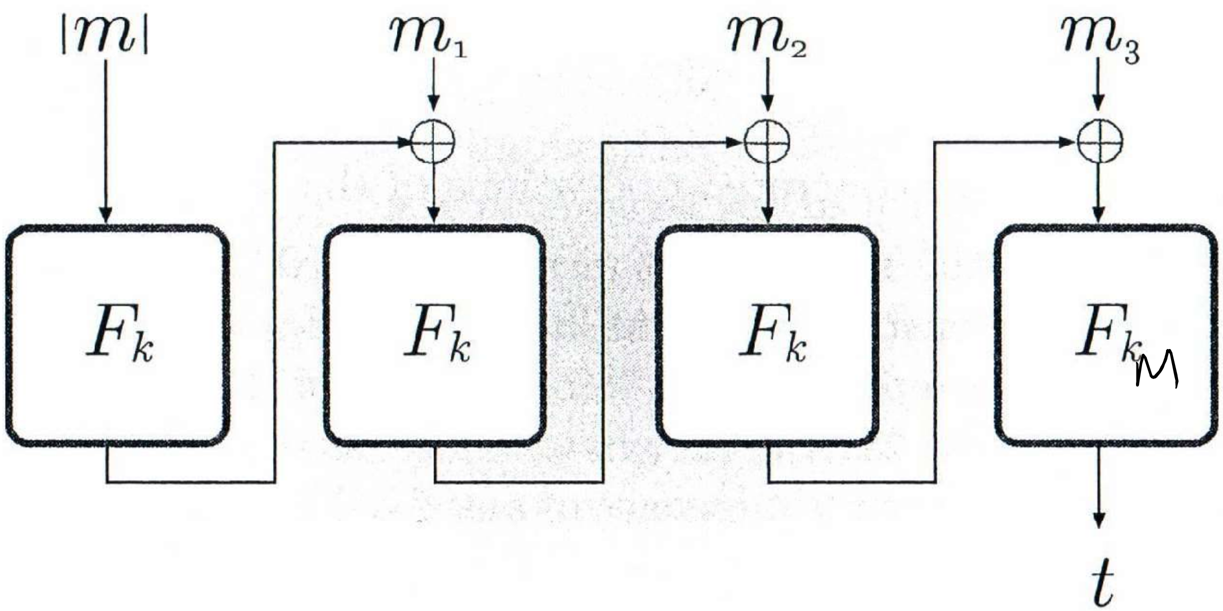

FIGURE 4.2: A variant of CBC-MAC secure for authenticating arbitrary-length messages.

CONSTRUCTION 4.7

Let $\Pi' = (\mathsf{Mac}', \mathsf{Vrfy}')$ be a fixed-length MAC for messages of length n. Define a MAC as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$, parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Choose a uniform identifier $r \in \{0,1\}^{n/4}$.

  For $i = 1, \ldots, d$, compute $t_i \leftarrow \mathsf{Mac}'_k(r\|\ell\|i\|m_i)$, where i, ℓ are encoded as strings of length $n/4$.† Output the tag $t := \langle r, t_1, \ldots, t_d \rangle$.
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = \langle r, t_1, \ldots, t_{d'} \rangle$, parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Output 1 if and only if $d' = d$ and $\mathsf{Vrfy}'_k(r\|\ell\|i\|m_i,\, t_i) = 1$ for $1 \le i \le d$.

† Note that i and ℓ can be encoded using $n/4$ bits because $i, \ell < 2^{n/4}$.

A MAC for arbitrary-length messages from any fixed-length MAC.

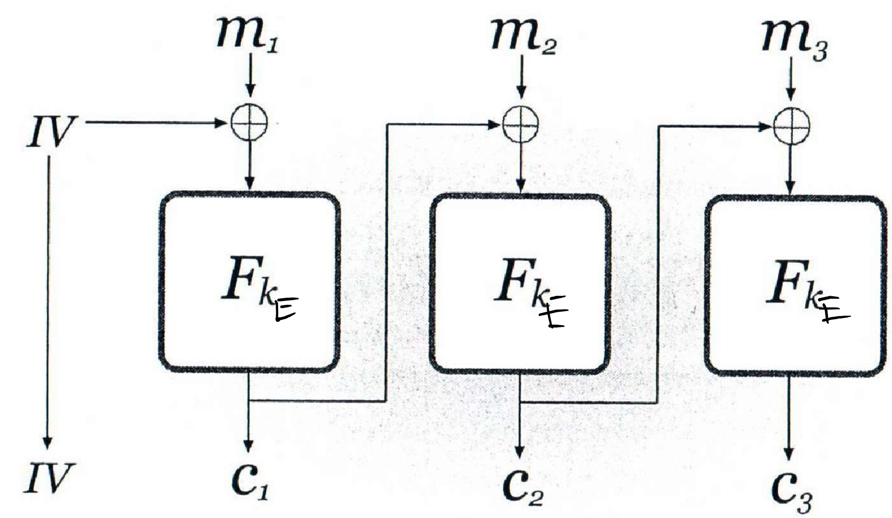

FIGURE 3.7: Cipher Block Chaining (CBC) mode.

Cipher Block Chaining (CBC) mode. To encrypt using this mode, a uniform initialization vector (IV) of length n is first chosen. Then, ciphertext blocks are generated by applying the block cipher to the XOR of the current plaintext block and the previous ciphertext block. That is, set $c_0 := IV$ and then, for $i = 1$ to ℓ, set $c_i := F_k(c_{i-1} \oplus m_i)$. The final ciphertext is $\langle c_0, c_1, \ldots, c_\ell \rangle$. (See Figure 3.7.) Decryption of a ciphertext $c_0, \ldots, c_\ell$ is done

$$\langle c_0, c_1, c_2, c_3, t \rangle$$

$128 \times 5 = 640$
bits