Lecture 5 – Message Authentication Codes, Hash Functions, and Authenticated Encryption

François Dupressoir

2025/26

So far, we have focused on protecting the *confidentiality* of messages. But our motivation is to protect the *security* of messages. In particular, we have done nothing at all to prevent our adversaries from modifying messages: in fact, in most of the schemes and constructions we studied in depth, the adversary can cause a predictable change in the plaintext by modifying a ciphertext.

We'll focus on integrity and authenticity—two related notions with subtle differences we won't really explore here, and consider security definitions and constructions for Message Authentication Codes (MACs)—keyed functions allowing a sender and recipient with a shared secret to protect messages against modification; and we will consider security notions for hash functions—*public* functions meant to ensure a similar property in the presence of a separate high integrity channel.

We will then see how to combine confidentiality and integrity by defining a security notion for *authenticated encryption*, and considering some generic ways of composing nonce-based encryption schemes and MACs to obtain secure authenticated encryption. These are as close as we will get to a true secure channel, and will work as long as we know how to securely establish short secrets from public information. (For example, using Diffie-Hellman.)

5.1 Message Authentication Codes

Message authentication codes operate by producing—from the key and message—an *authentication tag* (or simply a tag) that can be used—jointly with the key and message—to verify that the message was not modified since the tag was computed.

5.1.1 Syntax and Security

We define the syntax and correctness of those schemes formally as follows.

Definition 5.1 (Message Authentication Code (MAC)). A *message authentication code* $\textsc{Mac} = (\mathsf{Kg}, \mathsf{Tag}, \mathsf{Vfy})$, where Kg randomly generates a key $k \in \mathcal{K}$, Tag takes a key k and a message

```
$\mathsf{Exp}^{\text{euf-cma}}_{\textsc{Mac}}(\mathbb{A})$
$k \leftarrow_\$ \mathsf{Kg}$
$(\widehat{m}, \widehat{t}) \leftarrow_\$ \mathbb{A}^{\mathcal{T}_{\text{cma}}}$

    $\mathcal{T}_{\text{cma}}(m)$
    $t \leftarrow \mathsf{Tag}_k(m)$
    return $t$
```

```
$\mathsf{Exp}^{\text{uuf-cma}}_{\textsc{Mac}}(\mathbb{A})$
$k \leftarrow_\$ \mathsf{Kg}$
$m^* \leftarrow_\$ \mathcal{M}$
$\widehat{t} \leftarrow_\$ \mathbb{A}^{\mathcal{T}_{\text{cma}}}(m^*)$

    $\mathcal{T}_{\text{cma}}(m)$
    require $m \neq m^*$
    $t \leftarrow \mathsf{Tag}_k(m)$ return $t$
```

Figure 5.1: Two different unforgeability experiments for $\textsc{Mac}$

$m \in \mathcal{M}$ to output tag $t \leftarrow \mathsf{Tag}_k(m) \in \mathcal{T}$, and Vfy takes a key k and a message–tag pair (m, t) to output either $\top$ (valid) or $\bot$ (invalid).

The MAC scheme is *correct* iff, for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$, $\mathsf{Vfy}_k(m, \mathsf{Tag}_k(m)) = \top$.

Definition 5.1 leaves open the possibility that the Tag algorithm is probabilistic. For most practical schemes, Tag is in fact deterministic, and verification simply recomputes the tag: $\mathsf{Vfy}_k(m, t)$ outputs $\top$ exactly when $\textsc{Mac}_k(m) = t$. In this unit, we consider only schemes that use this type of verification, and leave the Vfy algorithm unspecified from now on. Still, it is sometimes useful to refer to the act of "verifying a tag with respect to a message and a key"', so we will keep the terminology and notation.

Definition 5.2 (Existential Unforgeability under Chosen Message Attack)**.** The *advantage of an adversary* $\mathbb{A}$ *in existentially forging a* $\textsc{Mac}$ *tag under chosen message attack* is defined as follows—where $\mathsf{Exp}^{\text{euf-cma}}_{\textsc{Mac}}(\mathbb{A})$ is defined in Figure 5.1, and a message being *fresh* in a given run means that it was never used as an input to the $\mathcal{T}_{\text{cma}}$ oracle.

$$\mathsf{Adv}^{\text{euf-cma}}_{\textsc{Mac}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{euf-cma}}_{\textsc{Mac}}(\mathbb{A}) : \mathsf{Vfy}_k(\widehat{m}, \widehat{t}) = \top \wedge \ \widehat{m} \text{ is fresh}\right]$$

We say that $\textsc{Mac}$ is (t, q, ϵ)*-existentially unforgeable under chosen message attack* if, for every $\mathbb{A}$ that runs in time at t and makes at most q queries to their $\mathcal{T}_{\text{cma}}$ oracle, we have $\mathsf{Adv}^{\text{euf-cma}}_{\textsc{Mac}}(\mathbb{A}) \leq \epsilon$.

Definition 5.3 (Universal Unforgeability under Chosen Message Attack)**.** The *advantage of an adversary* $\mathbb{A}$ *in existentially forging a* $\textsc{Mac}$ *tag under chosen message attack* is defined as follows—where $\mathsf{Exp}^{\text{uuf-cma}}_{\textsc{Mac}}(\mathbb{A})$ is defined in Figure 5.1.

$$\mathsf{Adv}^{\text{uuf-cma}}_{\textsc{Mac}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{euf-cma}}_{\textsc{Mac}}(\mathbb{A}) : \mathsf{Vfy}_k(\widehat{m}, \widehat{t}) = \top\right]$$

We say that $\textsc{Mac}$ is (t, q, ϵ)*-universally unforgeable under chosen message attack* if, for every $\mathbb{A}$ that runs in time at t and makes at most q queries to their $\mathcal{T}_{\text{cma}}$ oracle, we have $\mathsf{Adv}^{\text{uuf-cma}}_{\textsc{Mac}}(\mathbb{A}) \leq \epsilon$.

```
$\mathrm{CBC\text{-}MAC}_k(m)$
$(m[1], \ldots, m[n]) \leftarrow \mathsf{parse}(m)$
$X[0] \leftarrow 0^\ell$
for $i \in [1, \ldots, n]$
    $Y[i] \leftarrow X[i-1] \oplus m[i]$
    $X[i] \leftarrow \mathsf{E}_k(Y[i])$
return $X[n]$
```

```
$\mathrm{C^*\text{-}MAC}_{k_1,k_2}(m)$
$(m[1], \ldots, m[n]) \leftarrow \mathsf{pad}(m)$
$X[0] \leftarrow 0^\ell$
for $i \in [1, \ldots, n]$
    $Y[i] \leftarrow X[i-1] \oplus m[i]$
    $X[i] \leftarrow \mathsf{E}_{k_1}(Y[i])$
$t \leftarrow \mathsf{F}_{k_2}(X[n])$
return $t$
```

Figure 5.2: CBC-MAC: the vanilla version for $\mathcal{M} = \{0,1\}^{\ell \cdot n}$ in the left panel; the usual template for dealing with $\mathcal{M} = \{0,1\}^*$ in the right panel.

Guessing attack and lower bound on insecurity. As with previous definitions, let us first consider the best we could hope to do. Let us consider the weakest notion we can think of: UUF-PAS (universal unforgeability under passive attack). In that case, the very best even a clever adversary can do is guess a tag, which succeeds with probability at least $1/|\mathcal{T}|$. Here again, this implies that tags should be long enough *at least* for you to be happy with the level of insecurity implied by this best case attack.

5.1.2 CBC-MAC

A popular way of building MACs is to use a blockcipher in CBC mode, retaining only the last block of ciphertext. The resulting construction, CBC-MAC, is shown in Figure 5.2. We can prove it is EUF-CMA secure as long as the underlying blockcipher is IND secure and as long as the length of messages is fixed *a priori* to some $n \cdot \ell$ (where ℓ is the block size).

To make it secure in practice, some form of post-processing is needed. One simple form of post-processing is shown in the right pane of Figure 5.2, and consists in running the tag through an independent blockcipher (or the same blockcipher with an independent key) before output. CMAC, a standard based on this, is slightly more involved because it attempts to minimise padding—which we discuss now.

5.1.3 Padding: Dealing with Arbitrary-Length Messages

So far, we've defined our constructions only on well-behaved messages that could easily be parsed into blocks. We need to explain how this parsing can be done in practice, in a way that doesn't weaken security.[1]

The most pervasive way of allowing arbitrary-length inputs is *padding*, which consists in defining an *injective* function $\mathsf{pad} \in \{0,1\}^* \to (\{0,1\}^\ell)^*$—that is, a function that turns a string of bits into a string of blocks in—at least theoretically—invertible way.

For MACs, the inverse does not need to be efficiently computable for correctness (but it might need to be efficiently computable for security proofs to make sense). For encryption,

[1] We had scope to discuss padding in relation to encryption as well, but there, getting it wrong in the normal ways only threatens correctness, which we don't care overmuch about in this unit.

the inverse does need to be efficiently computable for correctness, as well as for security proofs.

One widely-used padding scheme is the 10^* padding scheme (or some byte-level variant), which involves padding the message with at least one 1 bit, followed by as many zeroes as needed to align with the block length. It can easily inverted by looking back from the end of the padded string for the last 1 bit, and dropping it and all following bits. (Note that this is partial! It is important that "unpadding" can fail.)

5.2 Cryptographic Hash Functions

MACs are powerful, but require that the sender and recipient share a key and trust each other to not misuse it. This is not useful if a single sender wants to send to multiple recipients: anyone who can verify the tag can also compute it! Cryptography offers a public alternative—*hash functions*—which provide some form of integrity protection, and often also serve as a building block in many other constructions. For technical reasons, we must define hash functions as keyed functions instead.

5.2.1 Syntax and Security

Definition 5.4 (Hash Function)**.** A hash function is a $\mathcal{K}$-indexed family of algorithms $\mathsf{H}_k : \mathcal{M} \to \mathcal{D}$ that take as input a message $m \in \mathcal{M}$ and outputs a *digest* $d \in \mathcal{D}$.

In order to speak of a hash function we require that the function *compresses*, that is $|\mathcal{M}| > |\mathcal{D}|$.

Typically, the cardinality $|\mathcal{M}|$ of the message space is a *whole lot* larger than that of the digest space $|\mathcal{D}|$. For instance, $\mathcal{D} = \{0,1\}^d$ for say $d = 256$, yet $\mathcal{M}$ consists of all bitstrings up to length 2^{64}.

Cryptographic hash functions are typically expected to have three security properties: collision resistance, preimage resistance, and second preimage resistance. We define the corresponding experiments and advantages in Figure 5.3, without formally defining the detailed notions.

As always, generic attacks help us pick parameters such as the digest length. Collision resistance is vulnerable to birthday attacks and are the main constraint on digest length: for the same level of security, collision resistance requires digests to be twice as long as preimage resistance or second preimage resistance.

5.3 Authenticated Encryption

5.3.1 Syntax and Security

Definition 5.5 ((Nonce-Based) Authenticated Encryption)**.** A nonce-based authenticated encryption scheme $\mathsf{E} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of algorithms where Kg randomly generates a key $k \in \mathcal{K}$, Enc takes a key k, a nonce $n \in \mathcal{N}$ and a message $m \in \mathcal{M}$ to output ciphertext

```
$\mathsf{Exp}^{\mathsf{cr}}_{\mathsf{H}}(\mathbb{A})$
$k \leftarrow_\$ \mathcal{K}$
$(\widehat{m_1}, \widehat{m_2}) \leftarrow_\$ \mathbb{A}(k)$
```

$$\mathsf{Adv}^{\mathsf{cr}}_{\mathsf{H}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{cr}}_{\mathsf{H}}(\mathbb{A}) : \begin{array}{l} \widehat{m_1} \neq \widehat{m_2} \\ \wedge\ \mathsf{H}_k(m_1) = \mathsf{H}_k(m_2) \end{array}\right]$$

```
$\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{H}}(\mathbb{A})$
$k \leftarrow_\$ \mathcal{K}$
$m^* \leftarrow_\$ \mathcal{M}$
$d^* \leftarrow \mathsf{H}_k(m^*)$
$\widehat{m} \leftarrow_\$ \mathbb{A}(k, d^*)$
```

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{H}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{H}}(\mathbb{A}) : \mathsf{H}_k(\widehat{m}) = d^*\right]$$

```
$\mathsf{Exp}^{\mathsf{pr2}}_{\mathsf{H}}(\mathbb{A})$
$k \leftarrow_\$ \mathcal{K}$
$m^* \leftarrow_\$ \mathcal{M}$
$\widehat{m} \leftarrow_\$ \mathbb{A}(k, m^*)$
```

$$\mathsf{Adv}^{\mathsf{pr2}}_{\mathsf{H}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{pr2}}_{\mathsf{H}}(\mathbb{A}) : \begin{array}{l} \widehat{m} \neq m^* \\ \wedge\ \mathsf{H}_k(\widehat{m}) = \mathsf{H}_k(m^*) \end{array}\right]$$

Figure 5.3: Hash function security notions: collision resistance (top), preimage resistance (bottom left), and second preimage resistance (bottom right)

$c \leftarrow \mathsf{Enc}^n_k(m) \in \mathcal{C}$, and Dec takes a ciphertext c, a nonce n and a key k to output a message m or $\perp$ (denoting a decryption failure).

The authenticated encryption scheme is correct iff, for all $k \in \mathcal{K}$, $n \in \mathcal{N}$ and $m \in \mathcal{M}$, it holds that $\mathsf{Dec}^n_k(\mathsf{Enc}^n_k(m)) = m$.

All notations here assume that $\perp$ is not a valid message (that is, $\perp \notin \mathcal{M}$) so we can safely denote with m the representation of m in the extended set $\mathcal{M} \cup \{\perp\}$. (In practice, how to encode errors is one of those pitfalls that even experienced cryptography engineers get caught in.)

Definition 5.6 (Authenticated Encryption Security)**.** The *advantage of an adversary* $\mathbb{A}$ *in distinguishing an authenticated encryption scheme* Enc *from an ideal encryption scheme* is defined as follows, where experiments $\mathsf{Exp}^{\mathsf{ae\text{-}real}}_{\mathsf{Enc}}(\mathbb{A})$ and $\mathsf{Exp}^{\mathsf{ae\text{-}ideal}}_{\mathsf{Enc}}(\mathbb{A})$ are defined in Figure 5.4.

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{Enc}}(\mathbb{A}) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{ae\text{-}real}}_{\mathsf{Enc}}(\mathbb{A}) : \widehat{b}\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ae\text{-}real}}_{\mathsf{ENC}}(\mathbb{A}) : \widehat{b}\right]\right|$$

An authenticated encryption scheme Enc is said to be $(t, q_{\mathcal{E}}, q_{\mathcal{D}}, \epsilon)$*-AE-secure* if, for every $\mathbb{A}$ that runs in time at most t, and makes at most $q_{\mathcal{E}}$ queries to its encryption oracle, and at most $q_{\mathcal{D}}$ queries to its decryption oracle, we have $\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{Enc}}(\mathbb{A}) \leq \epsilon$.

It might be an interesting exercise to show that the EUF-CMA notion we defined on MACs is equivalent to an indistinguishability notion inspired by the decryption oracle in the above. The security notion we use is in fact equivalent to being (N)IND-secure and being EUF-CMA secure (seeing the encryption algorithm as Tag, and the decryption algorithm as Vfy).

```
$\mathsf{Exp}^{\text{ae-real}}_{\text{Enc}}(\mathbb{A})$
    $k \leftarrow_\$ \mathsf{Kg}$
    $\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$

    $\mathcal{E}(n, m)$
        no repeated nonces
        $c \leftarrow \mathsf{Enc}^n_k(m)$
        return $c$

    $\mathcal{D}(n, c)$
        $c$ not output by $\mathcal{E}(n, \cdot)$
        $m \leftarrow \mathsf{Dec}^n_k(c)$
        return $m$
```

```
$\mathsf{Exp}^{\text{ae-ideal}}_{\text{Enc}}(\mathbb{A})$
    $k \leftarrow_\$ \mathsf{Kg}$
    $\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$

    $\mathcal{E}(n, m)$
        no repeated nonces
        $c \leftarrow_\$ \mathcal{C}(|m|)$
        return $c$

    $\mathcal{D}(n, c)$
        $c$ not output by $\mathcal{E}(n, \cdot)$
        return $\perp$
```

Figure 5.4: All-in-one AE security experiment

5.3.2 Chosen Ciphertext Attacks

We now have a security definition that allows the adversary to not only ask for encryptions of chosen plaintexts, but also for decryptions of chosen ciphertexts. This kind of threat model is in fact also useful for non-authenticated encryption, where the decryption oracle might in fact leak more than success. We describe the security experiments for nonce-based indistinguishability under chosen ciphertext attacks (IND-CCA) in Figure 5.5, without further formally defining the security notion. (Which goes as usual.)

It should be intuitively clear that any AE-secure scheme is (N)IND-CCA secure.

5.3.3 Constructing AE: Generic Composition

We can construct an AE-secure scheme from an (N)IND-secure encryption scheme and an EUF-CMA-secure MAC scheme. Figure 5.6 shows the three natural ways of doing this.

All three are AE-secure under reasonable assumptions on the encryption and MAC schemes, but Encrypt-then-MAC (EtM) is the most widely used because it is harder to implement insecurely: for MtE and E+M, it is very easy to leak more information than success or failure upon decryption failures, which will reveal more information than safe about the plaintext.

```
$\mathsf{Exp}^{\text{(n)ind-cca-real}}_{\textsc{Enc}}(\mathbb{A})$
    $k \leftarrow_\$ \mathsf{Kg}$
    $\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$

    $\mathcal{E}(n, m)$
        no repeated nonces, and
        $m$ not output by $\mathcal{D}(n, \cdot)$
        - - - - - - - - - - - - - - - -
        $c \leftarrow \mathsf{Enc}^n_k(m)$
        **return** $c$

    $\mathcal{D}(n, c)$
        $c$ not output by $\mathcal{E}(n, \cdot)$
        - - - - - - - - - - - - - - - -
        $m \leftarrow \mathsf{Dec}^n_k(c)$
        **return** $m$
```

```
$\mathsf{Exp}^{\text{(n)ind-cca-ideal}}_{\textsc{Enc}}(\mathbb{A})$
    $k \leftarrow_\$ \mathsf{Kg}$
    $\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$

    $\mathcal{E}(n, m)$
        no repeated nonces, and
        $m$ not output by $\mathcal{D}(n, \cdot)$
        - - - - - - - - - - - - - - - -
        $c \leftarrow_\$ \mathcal{C}(|m|)$
        **return** $c$

    $\mathcal{D}(n, c)$
        $c$ not output by $\mathcal{E}(n, \cdot)$
        - - - - - - - - - - - - - - - -
        $m \leftarrow \mathsf{Dec}^n_k(c)$
        **return** $m$
```

$$\mathsf{Adv}^{\text{(n)ind-cca}}_{\textsc{Enc}}(\mathbb{A}) = \left|\Pr\left[\mathsf{Exp}^{\text{(n)ind-cca-real}}_{\textsc{Enc}}(\mathbb{A}) : \widehat{b} = 1\right] - \Pr\left[\mathsf{Exp}^{\text{(n)ind-cca-ideal}}_{\textsc{Enc}}(\mathbb{A}) : \widehat{b} = 1\right]\right|$$

Figure 5.5: The (N)IND-CCA experiment and security notion

```
$\textsc{MtE}^n_{k_a,k_e}(m)$
    $t \leftarrow \mathsf{Tag}_{k_a}(n, m)$
    $c \leftarrow \mathsf{Enc}_{k_e}(n, m \| t)$
    **return** $c$
```

```
$\textsc{EtM}^n_{k_a,k_e}(m)$
    $c \leftarrow \mathsf{Enc}_{k_e}(n, m)$
    $t \leftarrow \mathsf{Tag}_{k_a}(n, c)$
    **return** $c \| t$
```

```
$\text{E+M}^n_{k_a,k_e}(m)$
    $c \leftarrow \mathsf{Enc}_{k_e}(n, m)$
    $t \leftarrow \mathsf{Tag}_{k_a}(n, m)$
    **return** $c \| t$
```

Figure 5.6: Generic composition for authenticated encryption: mac-then-encrypt (left), encrypt-then-mac (middle), and encrypt-and-mac (right)

Bibliography