COMS30023/ Cryptology

Lecture Notes

Dr François Dupressoir

2022

Contents

Lecture 0 – Introduction

Cryptology is old: ever since we started mistrusting each other, we've sought to hide secrets from each other—and that's the first thing cryptography does.

But *modern* cryptology does more: it is not only a set of tools to protect data and the information it contains (whether it is at rest, in transit, or even in use), but also a set of techniques that allow us to precisely analyse the actual security guarantees of cryptographic tools—both by establishing lower bounds on security (by proving that breaking security would imply solving some hard problem), and by establishing upper bounds on security (by studying generic attacks against constructions and against the hard problems they rely on).

The core principle of this analysis is known as *Kerckhoffs' Principle* [Ker83], which can be roughly summed up as:

> Design and analyse your system assuming that your opponents know it in detail.

In other words: whatever algorithm you come up with, its security must not rely on the fact that it is unknown. The only secret you should assume is the *cryptographic key*.

In this unit, we will see how this principle shapes the way in which the modern cryptographer:

1. defines security;
2. designs cryptographic schemes;
3. chooses cryptographic assumptions; and
4. analyses cryptographic security;

and in which the modern cryptanalyst:

1. breaks security in practice;
2. attempts to undermine cryptographic assumptions.

We will do so considering the simple case of two mutually trusting participants attempting to exchange a message over an insecure network.

Lecture 1 – One-Time Cryptosystems

We first consider a simple setting: Aniket and Barbara want to securely exchange a single message whose length they know in advance. This setting gives rise to simple constructions—*enciphering schemes* that we use to introduce a number of basic and standard methods in modern cryptography.

First, we'll specify which algorithms can be considered enciphering schemes by defining their *syntax*. Then, we'll specify the most basic properties such enciphering schemes should possess: *correctness*. Finally, and most importantly, we will discuss what exactly it might mean for an enciphering scheme to be *secure*.

This will lead us to the modern practice of game-based (or property-based) definitions of security.

1.1 Enciphering Schemes: Syntax and Correctness

Informally, we are interested in taking a message in plain text—often referred to as the *plaintext*, taken from some *message space* $\mathcal{M}$—and some key—taken from some *key space* $\mathcal{K}$; and outputs an enciphered message—often referred to as *the ciphertext*—taken from some *cipher space* $\mathcal{C}$; in such a way that the original message can be recovered given knowledge of ciphertext and key, but not without the key.

An enciphering scheme (for us) is such that $\mathcal{M} = \mathcal{C}$, and is specified by three algorithms:

- A probabilistic algorithm that generates keys in $\mathcal{K}$;
- An algorithm that *enciphers* a plaintext under a key, and into a ciphertext; and
- An algorithm that *deciphers* a ciphertext under a key, and into a plaintext.

This exactly specifies the syntax of enciphering scheme. The formal definition (Definition 1.1) does a bit more legwork in introducing some notation, and giving names to those algorithms. This will later give us nice ways of abstracting over specific enciphering schemes.

Definition 1.1 (Symmetric Enciphering Scheme)**.** A *symmetric enciphering scheme* E is a triple of algorithms Kg, E, and D, where:

- Kg randomly generates a $k \in \mathcal{K}$;
- E takes a key k and a message $m \in \mathcal{M}$ to output ciphertext $c \leftarrow \mathsf{E}_k(m) \in \mathcal{C}$, with $\mathcal{C} = \mathcal{M}$; and
- D takes a key k and a ciphertext $c \in \mathcal{C}$ and to output a purported message $m' \leftarrow \mathsf{D}_k(c)$.

With this definition in place, we can generically define what it means for an enciphering scheme to be *correct*.

Definition 1.2 (Correctness of Enciphering Schemes)**.** An enciphering scheme $E = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ is correct iff, for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$, we have $\mathsf{D}_k(\mathsf{E}_k(m)) = m$.

1.2 The One-Time Pad

The one-time pad is a very simple enciphering scheme where keys, messages and ciphertexts are all bitstrings of some fixed length ℓ. Figure 1.1 specifies its algorithms Kg, E and D for some $\ell > 0$. Note also the notation—which will become pervasive—for sampling a value x uniformly at random in a (finite) set S: $x \leftarrow_{\$} S$. (We will also use it to denote storing in a variable the result of running a probabilistic algorithm.) $\oplus$ is bitwise exclusive or (XOR).

$\mathsf{Kg}()$
$k \leftarrow_{\$} \{0,1\}^{\ell}$
return k

$\mathsf{E}_k(m)$
$c \leftarrow m \oplus k$
return c

$\mathsf{D}_k(c)$
$m \leftarrow c \oplus k$
return m

Figure 1.1: The one-time pad; ℓ is the intended message length

1.3 Security of Enciphering Schemes

A natural question, then is: how much *security* does such a simple construction as the one-time pad provide? The answer, as we see next, is simultaneously "it provides perfect security," and "it provides no security whatsoever". The main crux of what looks like a paradox right now, is that we do not even know what it means for an enciphering scheme to be secure. Let's remedy that.

1.3.1 Key Recovery

By Kerckhoffs' principle, we must certainly ensure that the key cannot be recovered from the system—if an adversary can recover the key from a ciphertext—and is assumed to know all details of the algorithm used, then they can surely decipher that ciphertext as well.

Adversary Goal So we first attempt to define what it means for an enciphering scheme (*any* enciphering scheme) to be secure against key recovery. We do so using a *security experiment* (or *security game*), and defining an *adversarial advantage*—which essentially measures the *insecurity* of a scheme against an adversary.

Definition 1.3 (Passive Key Recovery Security for Enciphering Schemes)**.** Let E be an enciphering scheme. The *advantage of* $\mathbb{A}$ *in passively recovering the key* is defined as follows, for the experiment $\mathrm{Exp}^{\text{kr-pas}}_{E}()$ defined in Figure 1.2.

Figure 1.2: The passive key-recovery game $\mathsf{Exp}^{\text{kr-pas}}_{E}()$, and the "guessing" adversary $\mathbb{A}_{\text{guess}}$ (right).

$$\mathsf{Adv}^{\text{kr-pas}}_{E}(\mathbb{A}) \stackrel{def}{=} \Pr\left[\mathsf{Exp}^{\text{kr-pas}}_{E}(\mathbb{A}) : \widehat{k} = k^*\right]$$

An enciphering scheme E is said to be (t, ϵ)-*secure against passive key recovery* if, for every algorithm $\mathbb{A}$ running in time at most t, we have $\mathsf{Adv}^{\text{kr-pas}}_{E}(\mathbb{A}) \leq \epsilon$.

With this definition of security, it is clear that the adversary cannot do much: they get to see nothing that depends on the key, so the best they can do is guess. Such an adversary is given on the right hand side of Figure 1.2: they simply run the key generation algorithm, and hope it outputs the same key. If, say, Kg samples its output uniformly at random in the key space $\mathcal{K}$, then $\mathsf{Adv}^{\text{kr-pas}}_{E}(\mathbb{A}_{\text{guess}}) = 1/|\mathcal{K}|$.

Adversarial powers Clearly, beyond allowing us to introduce concepts and notations slowly, passive key recovery isn't very interesting as a security notion. Can the adversary recover the key when observing a ciphertext? We certainly hope not! But what other powers could we give the adversary?

Figure 1.3 shows three different experiments for key recovery under increasing adversary powers: (one-time) *known ciphertext attacks* (kr-1kca), (one-time) *known plaintext attack* (kr-1kpa), and (one-time) *chosen plaintext attack*. The shape of their advantage expression is determined entirely by the goal of key recovery: the only thing that changes is which game the adversary plays, but their winning condition is the same.

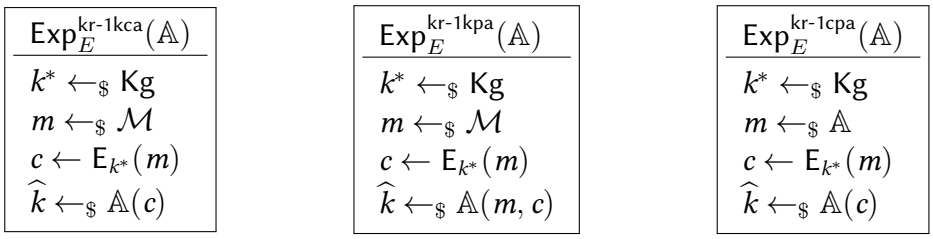


Figure 1.3: Adding one-time powers to the key-recovery game in three different ways.

1.3.2 One-Wayness

The goal of enciphering is not to protect the key, but to protect the plaintext. How can we express that no adversary can reasonably do so? Let us first consider the notion of one-

wayness, which captures the idea that recovering the plaintext *in full* from the ciphertext should be hard.

$\mathsf{Exp}_E^{\text{ow-pas}}(\mathbb{A})$

$k \leftarrow_\$ \mathsf{Kg}$
$m^* \leftarrow_\$ \mathcal{M}$
$c^* \leftarrow \mathsf{E}_k(m^*)$
$\widehat{m} \leftarrow_\$ \mathbb{A}(c^*)$

Figure 1.4: Passive one-time one-way security for symmetric enciphering scheme E

Definition 1.4 (Passive One-Time One-Way Security for Enciphering Schemes)**.** Let E be an enciphering scheme. We define the *advantage of* $\mathbb{A}$ *in passively breaking one-wayness* as follows, for the experiment $\mathsf{Exp}_E^{\text{ow-pas}}()$ defined in Figure 1.4.

$$\mathsf{Adv}_E^{\text{ow-pas}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}_E^{\text{ow-pas}}(\mathbb{A}) : \widehat{m} = m^*\right]$$

An enciphering scheme E is said to be (t, ϵ)*-secure against passive one-wayness attacks* if, for every algorithm $\mathbb{A}$ running in time at most t, we have $\mathsf{Adv}_E^{\text{ow-pas}}(\mathbb{A}) \leq \epsilon$.

1.3.3 Perfect Secrecy

Ensuring that no adversary is able to recover the message in full is nice. Ensuring that the adversary learns *no information* about the message at all is nicer.

Definition 1.5 captures this by expressing the fact that, whatever distribution the message is sampled from, the distribution over ciphertexts induced by enciphering a random plaintext under a freshly generated key is independent from the plaintext being enciphered.

Definition 1.5 (Perfect Secrecy)**.** A symmetric enciphering scheme E satisfies perfect secrecy if and only if for all message distributions over $\mathcal{M}$, the following holds.

$$\forall c \in \mathcal{C}, m \in \mathcal{M}. \Pr\left[m^* = m \mid c^* = c\right] = \Pr\left[m^* = m\right]$$

The probabilities are taken over $k \leftarrow_\$ \mathsf{Kg}$ and $m^* \leftarrow_\$ \mathcal{M}$ (according to the aforementioned message distribution), which fixes $c^* = \mathsf{E}_k(m^*)$.

Security of the One-Time Pad The One-Time Pad turns out to be perfectly secure.

Theorem 1.1 (Security of the One-Time Pad)**.** *The One-Time Pad satisfies perfect security.*

Shannon's Theorem Unfortunately for us, the One-Time Pad turns out to be (up to isomorphism) the only enciphering scheme that is perfectly secure.

Theorem 1.2 (Shannon's Theorem)**.** *Let* $E = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ *be an enciphering scheme with* $\mathcal{K} = \mathcal{M}$*. Then* E *is perfectly secure iff* Kg *draws from* $\mathcal{K}$ *uniformly at random and* E *satisfies that for all* (m, c) *pairs, there exists a unique key* k *such that* $\mathsf{E}_k(m) = c$*.*

1.3.4 Indistinguishability

We want to weaken perfect secrecy a little bit so that more schemes satisfy the notion, but without weakening it so much as to make it meaningless. We've already seen a few notions that allowed a bit of sloppiness. Can we express perfect secrecy as a game, then loosen it a little bit?

There are a few equivalent ways of expressing perfect secrecy. That given in Definition 1.5 is the original one given by Shannon [Sha49]. However, it is not directly useful to express security as a game: it quantifies over the plaintext distribution, and it directly talks about the independence of some distribution.

Definition 1.6

Definition 1.6 (Perfect Indistinguishability)**.** A symmetric enciphering scheme E satisfies perfect indistinguishability if and only if the following holds.

$$\forall c \in \mathcal{C}, m \in \mathcal{M}. \Pr\left[c^* = c \mid m^* = m\right] = |\mathcal{C}|^{-1}$$

The probability is taken over $k \leftarrow_{\$} \mathsf{Kg}$.

Theorem 1.3. *An enciphering scheme has perfect secrecy if and only if it has perfect indistinguishability.*

We can express this property as a game—however, the adversary's goal here is no longer to *recover* or compute a value, but to distinguish two different experiments.

$\mathsf{Exp}^{\text{1ind-real}}_{E}(\mathbb{A})$
$k \leftarrow_{\$} \mathsf{Kg}$
$m \leftarrow_{\$} \mathbb{A}$
$c^* \leftarrow \mathsf{E}_k(m)$
$\widehat{b} \leftarrow_{\$} \mathbb{A}(c^*)$

$\mathsf{Exp}^{\text{1ind-ideal}}_{E}(\mathbb{A})$
$k \leftarrow_{\$} \mathsf{Kg}$
$m \leftarrow_{\$} \mathbb{A}$
$c^* \leftarrow_{\$} \mathcal{C}$
$\widehat{b} \leftarrow_{\$} \mathbb{A}(c^*)$

Figure 1.5: One-time indistinguishability

Definition 1.7 (Game-Based Perfect Indistinguishability)**.** Let E be an enciphering scheme. We define the *advantage of* $\mathbb{A}$ *in one-time distinguishing* E *from random* as follows, for the experiments $\mathsf{Exp}^{\text{1ind-real}}_{E}()$ and $\mathsf{Exp}^{\text{1ind-ideal}}_{E}()$ defined in Figure 1.5.

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

An enciphering scheme E is said to be *perfectly indistinguishable* if, for every algorithm $\mathbb{A}$, we have $\mathsf{Adv}^{\text{1ind}}_{E}(\mathbb{A}) = 0$.

Now *this* definition can effectively be weakened in two ways: first, we can—instead of considering all possible algorithms—only consider adversaries with bounded resources, as we did for key recovery and one-wayness; and second, we can—instead of requiring that the adversary's advantage be 0—consider a scheme secure if any (bounded) adversary's advantage is small.

Definition 1.8 (Game-Based Indistinguishability)**.** An enciphering scheme E is said to be (t, ϵ)-*indistinguishable* if, for every algorithm $\mathbb{A}$ that runs in time at most t, we have $\mathsf{Adv}_E^{1\mathsf{ind}}(\mathbb{A}) \leq \epsilon$.

This will be our baseline security notion for confidentiality in the rest of this unit.

Lecture 4 – Symmetric Encryption

We've now seen how to build a couple more (asymmetric) enciphering schemes—which do not require pre-sharing secret information and even allow sending multiple messages using the same key. (Do they really? Give this a think!) More importantly, we've seen that—even with a whole lecture worth of effort, public key cryptography is slow as a goat compared to a bitwise XOR.

The way we really want to use public key cryptography is as first outlined: use it to establish a short, shared secret key, then use *that* to encrypt large messages fast. Except… we still don't know how to do that! Let's remedy this.

Along the way, as has become our thing, we'll define syntax and security, and we'll consider generic attacks that give us a hint about the best we can do. But this time, we'll also start proving that our constructions are as secure as their building blocks.

4.1 Blockciphers

To do anything worth while, we need to allow ourselves to define security when the same key can be used multiple times—recall that we've so far only defined notions under one-time attacks! We take the smallest possible step in this direction by defining blockciphers.

Definition 4.1 (Blockcipher)**.** A *blockcipher* E with *block length* ℓ is a symmetric enciphering scheme with $\mathcal{M} = \mathcal{C} = \{0,1\}^\ell$.

Lemma 4.1 (Blockciphers as Keyed Permutations)**.** *Let* $E = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ *be a* correct *blockcipher. Then, for any fixed key* k *output by* Kg*, the enciphering function* E_k *is a permutation.*

Proof. Left as an exercise to the reader, recalling the definition of correctness for enciphering schemes. □

We do not discuss the construction (or *realisation*) of blockciphers in this unit. The second (optional) half of the worksheet explores this question in depth—mostly negatively, to show that designing a good blockcipher is hard and that you are strongly encouraged to show humility if you try. But let us consider instead what kind of security we might want, how we can define it, and the boringest ways in which we can break it—this will inform some design constraints.

4.1.1 Security: Key Recovery

Let us first revisit key recovery under chosen plaintext attacks. Unlike previously—where we were considering one-time security notions, we now want to allow the adversary to interact with the key *multiple times*—but we still can't give the adversary they key!

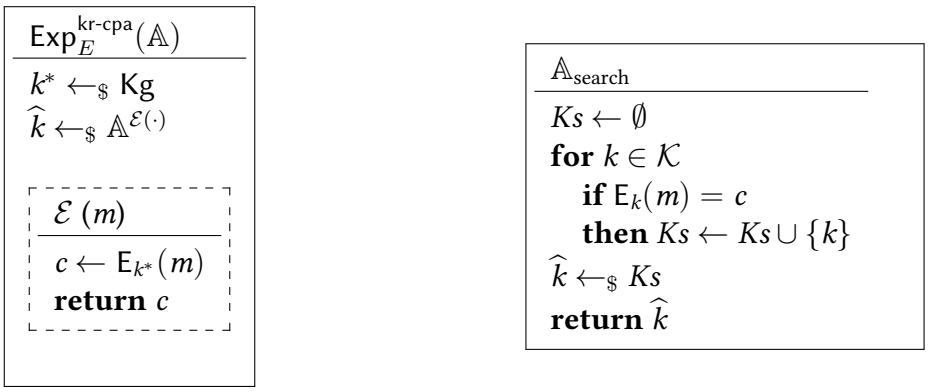


Figure 4.1: The "key-recovery under chosen plaintext attack game" (left) and the "exhaustive search" adversary that uses a single known-plaintext-ciphertext pair (right).

The solution here is to consider adversaries that have *oracle access* to the encryption algorithm with a fixed key: this controls the way in which the adversary is allowed to make use of the key in a minimally intrusive way. In particular, unless a specific note is made otherwise, the adversary can choose their queries to their oracles *adaptively*: make a query, see the result, *then* choose the next query.

Definition 4.2 (Key Recovery Security for Blockciphers)**.** Let E be a blockcipher. We define the *advantage of* $\mathbb{A}$ *in recovering the key from* E *under chosen plaintext attack* as follows, where experiment $\mathsf{Exp}^{\text{kr-cpa}}_E(\mathbb{A})$ is defined in Figure 4.1.

$$\mathsf{Adv}^{\text{kr-cpa}}_E(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{kr-cpa}}_E(\mathbb{A}) : \widehat{k} = k^*\right]$$

E is said to be (t, q, ϵ)*-secure against chosen plaintext key recovery* if, for every algorithm $\mathbb{A}$ running in time at most t and making at most q queries to its chosen plaintext oracle $\mathcal{E}(\cdot)$, we have $\mathsf{Adv}^{\text{kr-cpa}}_E(\mathbb{A}) \leq \epsilon$.

Exhaustive search as baseline security level. A simple (but costly) attack, given one or several plaintext-ciphertext pairs, is to simply iterate through all the keys and eliminate those that fail to map the plaintexts to the corresponding ciphertexts.

The number of encipherings it takes to run an exhaustive search (or rather, its base 2) is often used as a baseline for the security of a blockcipher. When a blockcipher's actual key recovery security strays a bit too far from this then the blockcipher is considered broken. This gives somewhat uniform measures for all notions of security: "we want 256-bit security" translates to "we want breaking whatever security we just asked you to obtain to be as costly as running 2^{256} encipherings of something". However, it's a lot less uniform in practice than we'd like—as a science—to pretend.

Current recommendations: if you really can't do anything else, 112 bit security is OK (lightweight cryptography); if you don't really care about the data long-term but need to show you did something short-term, 128 bit security is what you want; if you care about the data long-term, you must aim for 256 bit security.

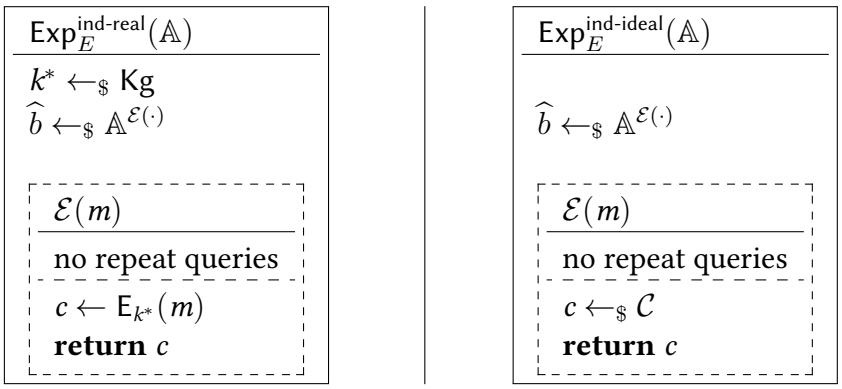


Figure 4.2: The real and ideal indistinguishability experiments.

4.1.2 Pseudorandomness

As before, key recovery is a nice baseline, but what we want is *indistinguishability*. For blockciphers, for some reason, it's called pseudorandomness.

Definition 4.3 (Pseudorandom Permutation)**.** Let E be a blockcipher. We define the *advantage of $\mathbb{A}$ in distinguishing E from a random permutation* as follows, where experiments $\mathsf{Exp}^{\text{ind-real}}_E(\mathbb{A})$ and $\mathsf{Exp}^{\text{ind-ideal}}_E(\mathbb{A})$ are defined in Figure 4.2.

$$\mathsf{Adv}^{\text{ind}}_E(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{ind-real}}_E(\mathbb{A}) : \widehat{b} = 1\right] - \Pr\left[\mathsf{Exp}^{\text{ind-ideal}}_E(\mathbb{A}) : \widehat{b} = 1\right]$$

E is said to be a (t, q, ϵ)-*secure pseudorandom permutation* if, for every algorithm $\mathbb{A}$ running in time at most t and making at most q queries to its $\mathcal{E}(\cdot)$ oracle, we have $\mathsf{Adv}^{\text{ind}}_E(\mathbb{A}) \leq \epsilon$.

Note here that we *must* restrict the adversary from querying the same input twice to the $\mathcal{E}$ oracle: in the real world, they would get the same response to both identical queries; in the ideal world, they would only get the same response with low probability—a clear and trivial distinguishing attack!

The birthday bound. What we cannot rule out immediately is repeat responses: those never happen in the real world, but could happen in the ideal world. This is known as a *collision*—two messages $m \neq m'$ such that $\mathcal{E}(m) = \mathcal{E}(m')$. An adversary that makes q queries to a random permutation will find such a collision with probability roughly $\frac{q \cdot (q-1)}{2 \cdot |\mathcal{C}|}$ (the birthday bound).

If exhaustive search placed a constraint on key size, the birthday bound places a constraint on the block length ℓ of a blockcipher if we are hoping for it to be pseudorandom.

4.2 Nonce-Based Encryption

We now have a building block which allows us to use a single, relatively short key, to encrypt multiple messages—as long as they fit in a block and are never repeated. We now take

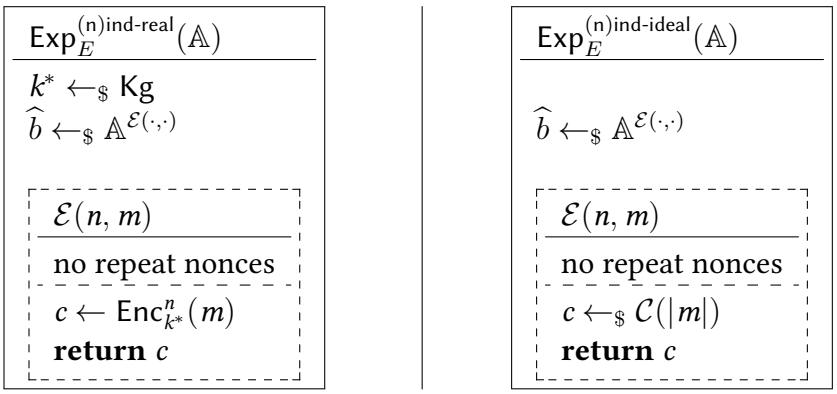


Figure 4.3: The real and ideal nonce-based indistinguishability experiments.

our final step towards the construction of an encryption primitive: we *use* blockciphers in structured ways to *encrypt* long messages while allowing repetitions. We do so by instead requiring that some public value called a *nonce* (number used only once) never be repeated instead—this is safer because the nonce can be entirely controlled by the cryptographic layer above, whereas plaintexts come from strange and unknown distributions—you might, for example, be hard-pressed to send three distinct messages from the set $\{Yes, No\}$.

Definition 4.4 (Nonce-Based Encryption Scheme)**.** A *nonce-based encryption scheme* E is a triple of algorithms $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$, 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 $c \leftarrow \mathsf{Enc}_k^n(m) \in \mathcal{C}$, and Dec takes a nonce n, a ciphertext $c \in \mathcal{C}$ and key k to output a purported message $m' \leftarrow \mathsf{Dec}_k^n(c)$.

E is said to be *correct* iff, for all $k \leftarrow_\$ \mathsf{Kg}$, $n \in \mathcal{N}$, and $m \in \mathcal{M}$, $\mathsf{Dec}_k^n(\mathsf{Enc}_k^n(m)) = m$.

Definition 4.5 (Nonce-Based Indistinguishability)**.** Let E be a nonce-based encryption scheme. We define the *advantage of* $\mathbb{A}$ *in distinguishing* E *from random ciphertexts* as follows, where experiments $\mathsf{Exp}_E^{(\mathrm{n})\mathrm{ind\text{-}real}}(\mathbb{A})$ and $\mathsf{Exp}_E^{(\mathrm{n})\mathrm{ind\text{-}ideal}}(\mathbb{A})$ are defined in Figure 4.3.

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

E is said to be a (t, q, ϵ)*-indistinguishable nonce-based encryption scheme* if, for every algorithm $\mathbb{A}$ running in time at most t and making at most q queries to its CPA oracle $\mathcal{E}(\cdot, \cdot)$, we have $\mathsf{Adv}_E^{(\mathrm{n})\mathrm{ind}}(\mathbb{A}) \leq \epsilon$.

4.2.1 Modes of Operation

All that is left for us to do (before we can prove something useful) is to *generically* build nonce-based encryption from any blockcipher. This is done using a *mode of operation*.

Counter mode (or CTR), shown in Figure **??**, is the most basic mode of operation given what we've already seen: use the blockcipher to expand the nonce into as many blocks of pseudorandom bits as needed, then use those as a one-time pad on the message.

It is nonce-based indistinguishable from random as long as the blockcipher it is constructed upon is pseudorandom. After a quick aside, we'll consider how to prove this.

$\mathsf{Enc}^n_k(m)$

Assume block length ℓ and $m \in \{0,1\}^{\ell \cdot n}$, $\ell < 2^{\ell/2}$, $n \in \{0,1\}^{\ell/2}$

$(m[1], \ldots, m[n]) \leftarrow \mathsf{parse}(m)$
for $i \in \{1, \ldots, n\}$
  $X[i] \leftarrow n \| \langle i \rangle_{\ell/2}$
  $Y[i] \leftarrow \mathsf{E}_k(X[i])$
  $c[i] \leftarrow m[i] \oplus Y[i]$
$c \leftarrow c[1] \| \ldots \| c[n]$
return c

$\mathsf{Dec}^n_k(c)$

Assume block length ℓ and $c \in \{0,1\}^{\ell \cdot n}$, $\ell < 2^{\ell/2}$, $n \in \{0,1\}^{\ell/2}$

$(c[1], \ldots, c[n]) \leftarrow \mathsf{parse}(c)$
for $i \in \{1, \ldots, n\}$
  $X[i] \leftarrow n \| \langle i \rangle_{\ell/2}$
  $Y[i] \leftarrow \mathsf{E}_k(X[i])$
  $m[i] \leftarrow c[i] \oplus Y[i]$
$m \leftarrow m[1] \| \ldots \| m[n]$
return m

Figure 4.4: Nonce-Based Counter Mode (CTR) over a blockcipher $E = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$; key generation is that of the blockcipher

$\mathsf{Enc}^n_k(m)$

Require $m \in \{0,1\}^{\ell \cdot n}$ and $n \in \{0,1\}^{\ell}$

$(m[1], \ldots, m[n]) \leftarrow \mathsf{parse}(m)$
$c[0] \leftarrow n$
for $i \in [1, \ldots, n]$
  $X[i] \leftarrow m[i] \oplus c[i-1]$
  $c[i] \leftarrow \mathsf{E}_k(X[i])$
$c \leftarrow c[1] \| \ldots \| c[n]$
return c

$\mathsf{Dec}^n_k(c)$

Require $c \in \{0,1\}^{\ell \cdot n}$ and $n \in \{0,1\}^{\ell}$

$(c[1], \ldots, c[n]) \leftarrow \mathsf{parse}(c)$
$c[0] \leftarrow n$
for $i \in [1, \ldots, n]$
  $X[i] \leftarrow \mathsf{D}_k(c[i])$
  $m[i] \leftarrow c[i-1] \oplus X[i]$
$m \leftarrow m[1] \| \ldots \| m[n]$
return m

Figure 4.5: Cipher Block Chaining Mode (CBC) over a blockcipher $E = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$; key generation is that of the blockcipher

Other Modes of Operation Other modes of operation exist. Some should not be used (Electronic Codebook, or ECB), some are so secure we can't even talk about their security until later in the unit (GCM, OCB), others yet are not nonce-based secure, but are secure under some additional conditions on the nonce.

The most notorious of these—for having been used in TLS, and for being used in SSH—is Cipher Block Chaining mode (CBC), which is shown in Figure **??**. We discuss it a bit in the problem sheet—mostly, again, destructively.

All those modes of operation—and more!—have their performance and use case advantages and drawbacks. Exploring all of them is not particularly useful for generalist cryptographers, although knowing that the variety exists is.

4.2.2 Reductions

Let's get back to CTR: how do we prove the earlier statement we made about its security—that if a blockcipher E is pseudorandom, then CTR over E is nonce-based secure?

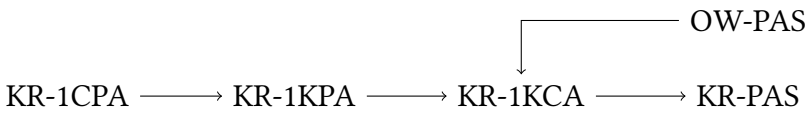


Figure 4.6: Relations between one-time security notions; arrows correspond to security implications (omitting those obtained by transitivity).

Let's consider again the statement "if 'A' is secure against this, then 'B' is secure against that." Logically, this is equivalent to "if 'B' is not secure against that, then 'A' is not secure against this." By definition, not being secure corresponds to the existence of a successful adversary, so we can restate to "if there is a successful that-adversary against 'B', then there is a successful this-adversary against 'A'." Finally, we have arrived at a statement we can deal with constructively. We will assume the existence of some $\mathbb{A}_{\text{b,that}}$ and use it to construct an adversary $\mathbb{B}_{\text{a,this}}$ where we can relate the respective adversarial advantages.

We won't prove CTR secure just yet, but we'll illustrate the concept of a reduction by proving some relations between the one-time security notions we came across in Lecture 1. The relevant notions and their relations are summarized in Figure 4.6.

From strong to weak powers. To start, we keep the security goal the same and look at what happens when the adversary gets more or less power. This corresponds to the horizontal implications in Figure 4.6. Intuitively, more power should help an adversary, so security against "stronger" adversaries should imply security against "weaker" adversaries.

For this example, we will construct a reduction showing that (t, ϵ)-KR-1CPA security implies (t, ϵ)-KR-1KPA-security.[1] Recall the logic—so we can recall what we assume, and what we construct.

i. If E is (t, ϵ)-KR-1CPA secure then it is (t, ϵ)-KR-1KPA secure.

ii. If E is (t, ϵ)-KR-1KPA *insecure* then it is (t, ϵ)-KR-1CPA *insecure*

iii. If there exists a KR-1KPA adversary against E that runs in time at most t and wins with an advantage greater than ϵ, then there exists a KR-1CPAadversary against it that runs in time at most t and wins with an advantage greater than ϵ.

Thus, given a KR-1KPA adversary $\mathbb{A}_{\text{kr-1kpa}}$, we need to construct a KR-1CPA adversary $\mathbb{B}_{\text{kr-1cpa}}$ in such a way that:

1. $\mathbb{B}_{\text{kr-1cpa}}$ is (roughly) as efficient as $\mathbb{A}_{\text{kr-1kpa}}$; and
2. $\mathsf{Adv}^{\text{kr-1kpa}}_{E}(\mathbb{A}_{\text{kr-1kpa}}) \leq \mathsf{Adv}^{\text{kr-1cpa}}_{E}(\mathbb{B}_{\text{kr-1cpa}})$.

Here $\mathbb{B}_{\text{kr-1cpa}}$ is called the *reduction*, and the two claims relating to efficiency and advantage are the *analysis* of the reduction. Figure 4.7 shows the reduction (in the dashed box headed $\mathbb{B}_{\text{kr-1cpa}}$). We are now left to analyse its efficiency and advantage.

[1] Note the same t and ϵ; this is happy land.

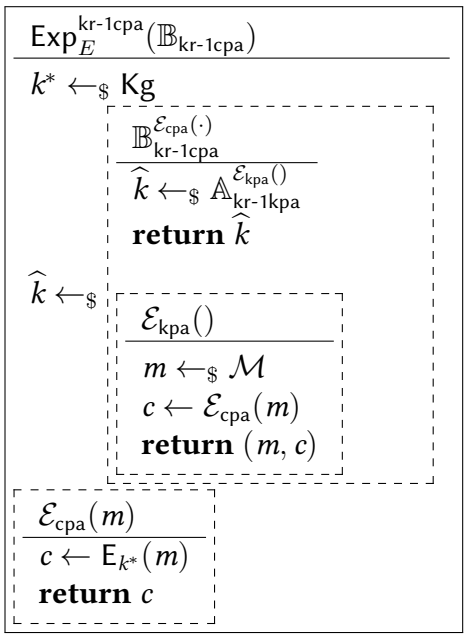


Figure 4.7: A reduction for KR-1CPA $\Rightarrow$ KR-1KPA.

- $\mathbb{B}_{\text{kr-1cpa}}$ runs $\mathbb{A}_{\text{kr-1kpa}}$ once, with the only overhead being that of sampling a message when $\mathbb{A}_{\text{kr-1kpa}}$ makes an oracle query—so $\mathbb{B}_{\text{kr-1cpa}}$ runs (roughly) in time t if $\mathbb{A}_{\text{kr-1kpa}}$ runs in time t;
- $\mathbb{B}_{\text{kr-1cpa}}$ and $\mathbb{A}_{\text{kr-1kpa}}$ are facing experiments with the same challenge key k^*, and share also their key guess, so whenever one wins, the other does as well, and we have

$$\mathsf{Adv}^{\text{kr-1kpa}}_{E}(\mathbb{A}_{\text{kr-1kpa}}) = \mathsf{Adv}^{\text{kr-1cpa}}_{E}(\mathbb{B}_{\text{kr-1cpa}})$$

From hard to easy goals. The same way adversary capabilities can be ranked, there is a hierarchy of goals as well. Typically,[2] indistinguishability notions are strongest, key recovery is weakest, and one-wayness sits in the middle. We will prove that (t, ϵ)-OW-PAS security implies (t, ϵ)-KR-1KCA security using a reduction.

First, we figure out the logic of the reduction. We are given an adversary $\mathbb{A}_{\text{kr-1kca}}$ and need to create a reduction $\mathbb{B}_{\text{ow-pas}}$. The 'skin' of the reduction is shown in the left pane of Figure 4.8: $\mathbb{B}_{\text{ow-pas}}$ must simulate $\mathbb{A}_{\text{kr-1kca}}$'s one-time KCA oracle (which gives it a valid ciphertext under the challenge key), and must somehow turn $\mathbb{A}_{\text{kr-1kca}}$'s output—a key guess—into a message guess.

The overall reduction is shown in Figure 4.8, and we can analyse it. Again, the running time of $\mathbb{B}_{\text{ow-pas}}$ is essentially that of $\mathbb{A}_{\text{kr-1kca}}$ as the overhead is minimal. Whenever $\mathbb{A}_{\text{kr-1kca}}$ wins by outputting the correct key, then $\mathbb{B}_{\text{ow-pas}}$ is guaranteed to win as well. Additionally, $\mathbb{B}_{\text{ow-pas}}$ might end up lucky even if $\mathbb{A}_{\text{kr-1kca}}$ doesn't return the correct key, so we have

$$\mathsf{Adv}^{\text{kr-1kca}}_{E}(\mathbb{A}_{\text{kr-1kca}}) \leq \mathsf{Adv}^{\text{ow-pas}}_{E}(\mathbb{B}_{\text{ow-pas}})$$

[2] of those we discuss in this unit

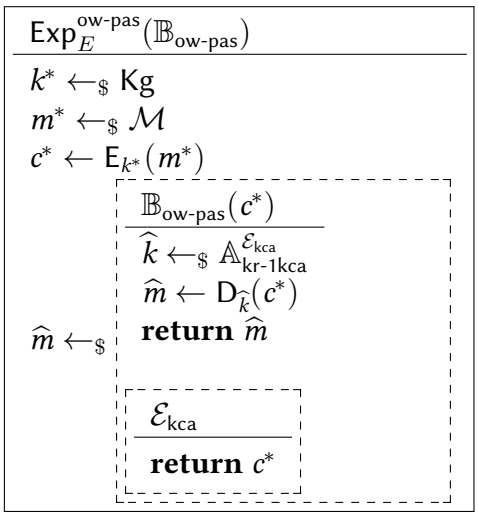


Figure 4.8: Reduction for OW-PAS ⇒ KR-1KCA.

This is exactly what we needed to prove.

Lecture 5 – Authentication

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 $m \in \mathcal{M}$ 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

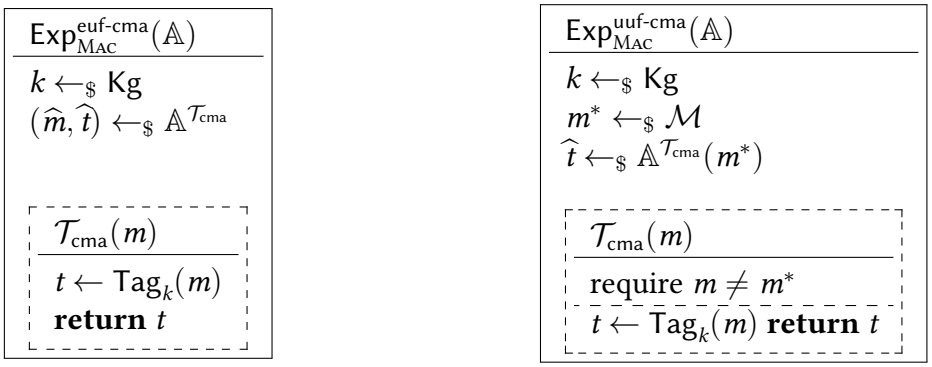


Figure 5.1: Two different unforgeability experiments for MAC

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* MAC *tag under chosen message attack* is defined as follows—where $\mathsf{Exp}^{\text{euf-cma}}_{\text{MAC}}(\mathbb{A})$ 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.

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

We say that 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 $\mathsf{Adv}^{\text{euf-cma}}_{\text{MAC}}(\mathbb{A}) \leq \epsilon$.

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

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

We say that 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 $\mathsf{Adv}^{\text{uuf-cma}}_{\text{MAC}}(\mathbb{A}) \leq \epsilon$.

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.

CBC-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 $X[n]$

C*-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.

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, 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.)

[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.

$$\boxed{\begin{array}{l}\mathsf{Exp}^{\mathsf{cr}}_{\mathsf{H}}(\mathbb{A})\\ \hline k \leftarrow_{\$} \mathcal{K}\\ (\widehat{m_1}, \widehat{m_2}) \leftarrow_{\$} \mathbb{A}(k)\end{array}}$$

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

$$\boxed{\begin{array}{l}\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{H}}(\mathbb{A})\\ \hline k \leftarrow_{\$} \mathcal{K}\\ m^* \leftarrow_{\$} \mathcal{M}\\ d^* \leftarrow \mathsf{H}_k(m^*)\\ \widehat{m} \leftarrow_{\$} \mathbb{A}(k, d^*)\end{array}}$$

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

$$\boxed{\begin{array}{l}\mathsf{Exp}^{\mathsf{pr2}}_{\mathsf{H}}(\mathbb{A})\\ \hline k \leftarrow_{\$} \mathcal{K}\\ m^* \leftarrow_{\$} \mathcal{M}\\ \widehat{m} \leftarrow_{\$} \mathbb{A}(k, m^*)\end{array}}$$

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

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

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.

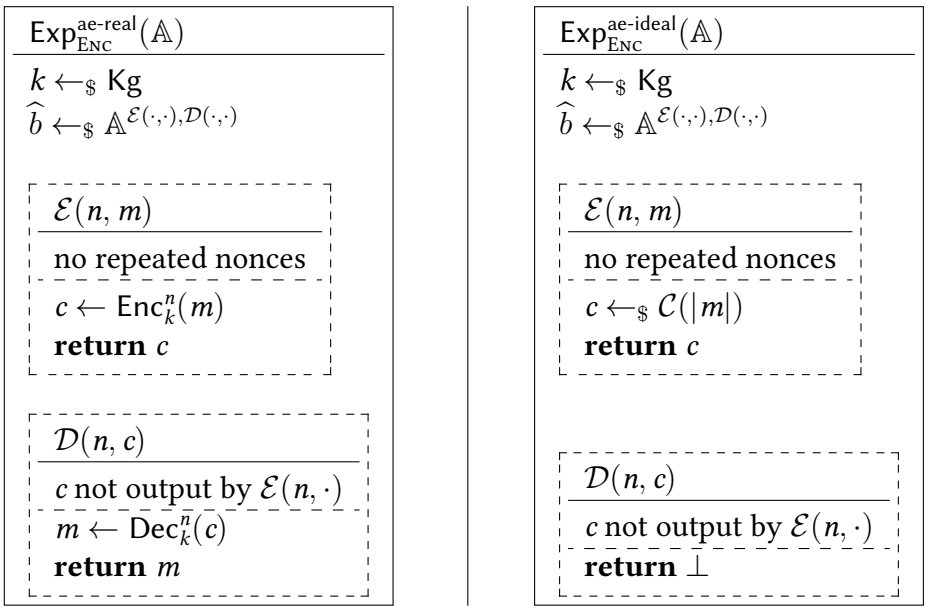


Figure 5.4: All-in-one AE security experiment

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

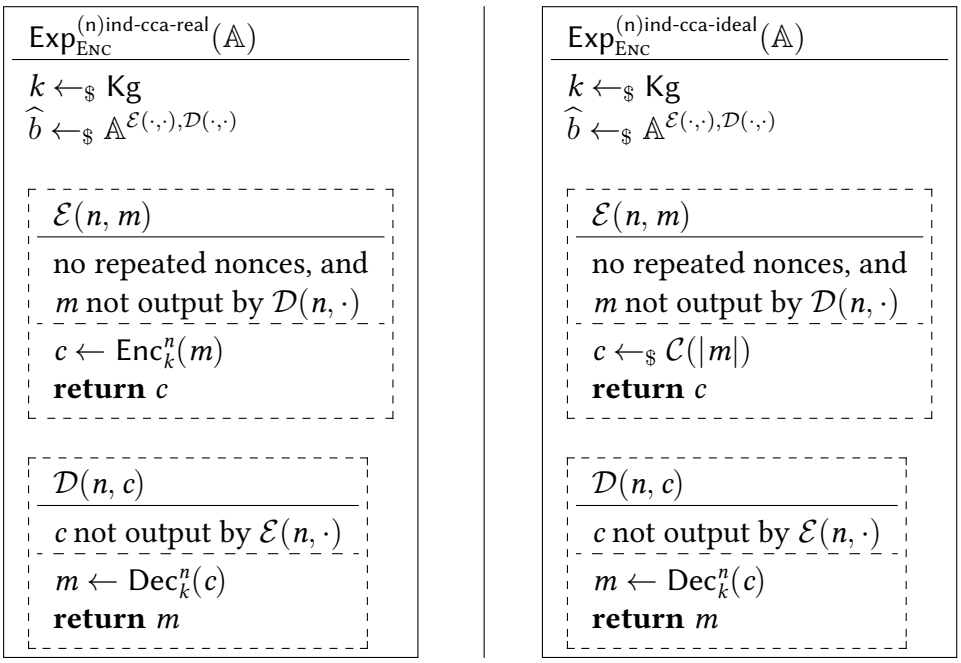
5.3.1 Syntax and Security

Definition 5.5 ((Nonce-Based) Authenticated Encryption)**.** A nonce-based authenticated encryption scheme $\mathrm{E} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ 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 $c \leftarrow \mathsf{Enc}^n_k(m) \in \mathcal{C}$, 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 $\mathsf{Exp}^{\text{ae-real}}_{\mathsf{Enc}}(\mathbb{A})$ and $\mathsf{Exp}^{\text{ae-ideal}}_{\mathsf{Enc}}(\mathbb{A})$ are defined in Figure 5.4.



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

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

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

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 $\mathsf{Adv}^{\text{ae}}_{\text{Enc}}(\mathbb{A}) \leq \epsilon$.

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).

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.

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

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

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

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

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.

Bibliography

[Ker83] Auguste Kerckhoffs. 'La Cryptographie Militaire'. In: *Journal des sciences militaires* IX (Feb. 1883), pp. 161–191.

[Sha49] Claude Elwood Shannon. 'Communication theory of secrecy systems'. In: *The Bell System Technical Journal* 28.4 (1949), pp. 656–715.