This lecture covers *message-authentication codes* and their security.

1 Games, packages and reductions

Recall that last week, we defined pseudorandom functions. Let us recall their definition.

Definition 1.1 (Security of PRFs)**.** A (λ, λ)-PRF f is *secure* if for all probabilistic poly-time (PPT) adversaries $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{\mathtt{Gprf}_f^0, \mathtt{Gprf}^1}_{f,\mathcal{A}} := \left| \Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}_f^0\right] - \Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}_f^1\right] \right|$$

is negligible in λ, where $\mathtt{Gprf}_f^0$ and $\mathtt{Gprf}^1$ are defined in Figure 2.

In last week's lecture notes, we already discussed that we can consider *games* as formal objects with a *parameters*, *state* and *oracles* which they provide. We write $[\to \mathtt{G}]$ for the oracles which a game $\mathtt{G}$ provides, or, more precisely, the *names* of the oracles which a game provides to the adversary, e.g.,

$$[\to \mathtt{Gprf}_f^0] = \{\mathsf{EVAL}\} = [\to \mathtt{Gprf}^1].$$

Why are packages useful? Well, in this lecture, we will introduce the notion of a *reduction* formally, and a reduction is, perhaps surprisingly, a *package*. Why is a reduction a package? Well, a reduction is something that turns an adversary against one *problem* into an adversary against another *problem*.

Reductions Let's say that we want to prove the security of some system S, under the assumption that primitive P is secure. In a reduction proof, we do this via a *proof by contradiction* (which can take some time to get used to): we assume that S is *insecure* and prove that P is then also insecure. If we can carry out such a proof, we can then argue that S is secure. After all, if S were insecure, then our proof would say that P is also insecure, but this in contradiction to our assumption that P is *secure*.

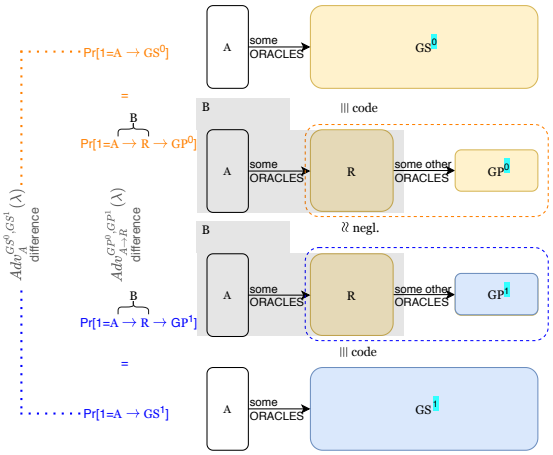


Figure 1: New adversary $\mathtt{B} = \mathcal{A} \to \mathtt{R}$

So, how do we prove that S being insecure implies P being insecure? Here it is important to realize what it means for S to be insecure. It means that there is some (efficient) adversary $\mathcal{A}$ that can break the security of S. Similarly, to prove that P is insecure, we have to provide a new efficient adversary $\mathtt{B}$ that can break it. However, due to our assumption, we are allowed to use $\mathcal{A}$ in the code

of $\mathtt{B}$. Note that we don't know the *code* of $\mathcal{A}$, so $\mathtt{B}$ can only use $\mathcal{A}$ in a black-box way, and we can't make any assumptions about the precise functioning of $\mathcal{A}$. Thus, we construct adversary $\mathtt{B}$ as a *composition* of $\mathcal{A}$ and a package $\mathtt{R}$, the latter of which we call reduction. To practice our notation, we will have that $\mathtt{B}$ is code-equivalent to the composition $\mathcal{A} \to \mathtt{R}$, written $\mathtt{B} \stackrel{\text{code}}{\equiv} \mathcal{A} \to \mathtt{R}$, and we'll have matching *oracles* and *dependencies*, which we write as $[\mathcal{A} \to] = [\to \mathtt{R}]$.
Reductions can be described explicitly, by defining reduction $\mathtt{R}$ in pseudocode. Let us make this more explicit. For simplicity, let us consider that the security of P is defined as the indistinguishability between a real game $\mathtt{GP}^0$ and an ideal game $\mathtt{GP}^1$. Let us further consider that the security of S is defined as the indistinguishability between a real game $\mathtt{GS}^0$ and an ideal game $\mathtt{GS}^1$. Then a reduction $\mathtt{R}$ from the security of S to the security of P is an (efficient!) package $\mathtt{R}$ such that

$$\begin{aligned} \mathtt{GS}^0 &\stackrel{\text{code}}{\equiv} \mathtt{R} \to \mathtt{GP}^0 \text{ and} \\ \mathtt{GS}^1 &\stackrel{\text{code}}{\equiv} \mathtt{R} \to \mathtt{GP}^1. \end{aligned} \tag{1}$$

With Equation 1 at hand, we can prove[1] that for all adversaries $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}(\lambda), \tag{2}$$

and thus, if there is an efficient adversary $\mathcal{A}$ which has non-negligible advantage against the pair $(\mathtt{GS}^0, \mathtt{GS}^1)$, then there is also an efficient adversary $\mathcal{A} \to \mathtt{R}$ against the pair $(\mathtt{GP}^0, \mathtt{GP}^1)$ which has the same advantage. However, we assume that there is no efficient $\mathcal{B}$ with non-negligible advantage $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{B}}$, thus taking $\mathcal{B} = \mathcal{A} \to \mathtt{R}$ leads to a contradiction. In conclusion, there cannot be an efficient $\mathcal{A}$ such that $\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}$ is non-negligible.
The *core argument*, here, is that (1) $\mathcal{A} \to \mathtt{R}$ is efficient (i.e., PPT) whenever $\mathcal{A}$ is efficient, and (2) Equation 2. Before we can state the notion of a reduction formally, let us actually reflect on whether we needed Equation 2, as an equality between advantage statements. Actually, we only need an argument which justifies that if $\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}$ is non-negligible, then $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}$ is non-negligible. Any of the following inequalities would establish the desired result:

$$\begin{aligned} &\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}(\lambda) \\ &\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}(\lambda) \\ &\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) \leq \mathsf{poly}(\lambda) \cdot \mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}(\lambda) \\ &\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) \leq \mathsf{poly}(\lambda) \cdot \sqrt{\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}(\lambda)} \\ &\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) \leq \mathsf{poly}(\lambda) \cdot \sqrt[5]{\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}(\lambda)} \end{aligned}$$

The reason that all of these inequalities work is that multiplying (or dividing) by constants, polynomials and squaring (or taking square roots), a negligible

[1] See Lecture Notes 3 and Lecture Video 4, minutes 16:30 and 27:56, for a concrete such argument: The main reason that code equivalence implies that two games behave exactly the same and thus, and adversary has exactly the same probability against the two games. We use this argument a lot in this course.

function cannot become non-negligible (or vice versa). We are almost ready to state the notion of a reduction, but actually, we still need to make this statement a bit more general. The reason is that often, we have more than one algorithm $\mathtt{R}$, and, e.g., might have inequalities such as

$$\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}_1}(\lambda) + \mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}_2}(\lambda), \tag{3}$$

for two different, efficient $\mathtt{R}_1$ to $\mathtt{R}_1$. This equation still suffices, since the sum of two negligible functions is negligible again, and if $\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}$ was non-negligible, then $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}_1}$ or $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}_2}$ would need to be non-negligible. As you can imagine, we can also combine this flexibility: Have *several* efficient $\mathtt{R}_i$ and some complex equation. To jump ahead, let me already mention the equation which we will see shortly:

$$\begin{aligned}&\mathsf{Adv}^{\mathtt{Gunf\text{-}cma}^0_{m_f},\mathtt{Gunf\text{-}cma}^1_{m_f}}_{m_f,\mathcal{A}}(\lambda)\\ &\quad\leq \mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_1}(\lambda) + \mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_2}(\lambda) + \frac{\mathsf{poly}_{\mathcal{A}}(\lambda)}{2^\lambda}\end{aligned} \tag{4}$$

Here, we add two advantages and some negligible term $\frac{\mathsf{poly}_{\mathcal{A}}(\lambda)}{2^\lambda}$, where $\mathsf{poly}_{\mathcal{A}}$ is some polynomial which depends on the adversary (in this case an upper bound on the number of queries that $\mathcal{A}$ makes). Thus, also Inequality 4 suffices to establish that if there is no efficient adversary such that

$$\mathsf{Adv}^{\mathtt{Gprf}^0_f,\mathtt{Gprf}^1_f}_{f,\mathcal{B}}$$

is non-negligible, then there is also no efficient adversary such that

$$\mathsf{Adv}^{\mathtt{Gunf\text{-}cma}^0_{m_f},\mathtt{Gunf\text{-}cma}^1_{m_f}}_{m_f,\mathcal{A}}$$

is non-negligible.

> How do we know which combination is the right one in which situation?

We don't really know this. Moreover, there are often more than one right way to do this. In essence, any approach that works is good, and we will see many such examples of proofs in this course (and the advanced course) so we can build our *cryptographer's toolkit*, i.e., a list of many examples for how to do proofs. We will not go in too many proof techniques in this course, since we balance seeing proof techniques with seeing concepts, and we focus more on concepts than on proof techniques in this course. Nevertheless, we'll see many examples of proofs so they will hopefully become familiar over time.
Let us now state the notion of a reduction:

Definition 1.2 (Reduction)**.** We call a package $\mathtt{R}$ a reduction from the indistinguishability of $\mathtt{GS}^0$ and $\mathtt{GS}^1$ to the indistinguishability of $\mathtt{GP}^0$ and $\mathtt{GP}^1$, if

- $\mathtt{R}$ is efficient, i.e., probabilistic polynomial-time (PPT), and
- for any efficient adversary $\mathcal{A}$, it holds that if $\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}$ is non-negligible, then $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}}$ is non-negligible.

Additionally, we call two packages $\mathtt{R}_1$ and $\mathtt{R}_2$ reductions from the indistinguishability of $\mathtt{GS}^0$ and $\mathtt{GS}^1$ to the indistinguishability of $\mathtt{GP}^0$ and $\mathtt{GP}^1$, if

- $\mathtt{R}$ is efficient, i.e., probabilistic polynomial-time (PPT), and
- for any efficient adversary $\mathcal{A}$, it holds that if $\mathsf{Adv}^{\mathtt{GS}^0,\mathtt{GS}^1}_{\mathcal{A}}$ is non-negligible, then $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}_1}$ or $\mathsf{Adv}^{\mathtt{GP}^0,\mathtt{GP}^1}_{\mathcal{A}\to\mathtt{R}_2}$ is non-negligible.

In this case, we refer to $\mathtt{R}_1$ and $\mathtt{R}_2$ as a reductions, even though one of them alone does not match the first part of the definition.

Key Packages One useful package which we will use a lot is the `Key` package. We will externalize the key of a package into a package in order to make the main package(s) of a game stateless. Let us see this on the example of the games $\mathtt{Gprf}^0_f$ and $\mathtt{Gprf}^1_f$. We will split game $\mathtt{Gprf}^0_f$ into two packages, `Key` (given in the figure on the right of this page) and $\mathtt{Prf}^0$ (given in Figure 2). When we *inline* the code of `Key` into $\mathtt{Prf}^0$, then we obtain $\mathtt{Gprf}^0_f$ again. We call merging two packages in this way *inlining* and we call the converse process, i.e. splitting a package into two, *outlining*, since it is the reverse of inlining.
One important observation of splitting a game into two is that the security definition is equivalent to the original one, i.e., both security definitions are satisfied by the same functions. Let us prove this on the example of the PRF security game. Let us start by formulating a modular version of the definition, using the `Key` package we just defined.

`Key`

GET()

if $k = \bot$:
  $k \leftarrow\!\!\$\ \{0,1\}^\lambda$
return k

Package Param.

λ:	security par.

Package State

k:	key

Definition 1.3 (Modular Security of PRFs)**.** A (λ,λ)-PRF f is *secure according to the modular security definition for PRFs* if for all PPT adversaries $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{\mathtt{Prf}^0_f\to\mathtt{Key},\mathtt{Gprf}^1}_{f,\mathcal{A}} := \left|\Pr\left[1=\mathcal{A}\overset{\mathsf{EVAL}}{\to}\mathtt{Prf}^0_f\overset{\mathsf{GET}}{\to}\mathtt{Key}\right]-\Pr\left[1=\mathcal{A}\overset{\mathsf{EVAL}}{\to}\mathtt{Gprf}^1\right]\right|$$

is negligible in λ.

With this definition at hand, we can now formulate the code equivalence between the composed game and the monolithic (i.e., single code-piece) game.

Claim 1 (Code Equivalence)**.**

$$\mathtt{Prf}^0_f\overset{\mathsf{GET}}{\to}\mathtt{Key}\overset{\text{code}}{\equiv}\mathtt{Gprf}^0_f$$

From here, we obtain the equivalence of the two definitions, as formulated in the following statement.

Claim 2. A (λ,λ)-PRF f is *secure* if and only if it is *secure according to the modular security definition for PRFs.*

$\underline{\underline{\texttt{Gprf}^0_f}}$

Package Parameters

λ:	security parameter
in:	input length λ
f:	function, (in, λ)-PRF

Package State

k:	key

$\mathsf{EVAL}(x)$

assert $x \in \{0,1\}^{\text{in}}$
if $k = \bot$:
  $k \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$y \leftarrow f(k, x)$
return y

$\underline{\underline{\texttt{Gprf}^1}}$

Package Parameters

λ:	security parameter
in:	input length λ

Package State

T:	**table** [bitstring, integer $\rightarrow$ bitstring]

$\mathsf{EVAL}(x)$

assert $x \in \{0,1\}^{\text{in}}$
if $T[x] = \bot$
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$y \leftarrow T[x]$
return y

$\underline{\underline{\texttt{Prf}^0_f}}$

Package Parameters

λ:	security parameter
in:	input length λ
out:	output length λ
f:	function, (in, out)-PRF

Package State

no state

$\mathsf{EVAL}(x)$

assert $x \in \{0,1\}^{\text{in}}$
$k \leftarrow \mathsf{GET}$

$y \leftarrow f(k, x)$
return y

$\underline{\underline{\texttt{Prf}^1}}$

Package Parameters

λ:	security parameter
in:	input length λ
out:	output length λ

Package State

T:	**table** [bitstring, integer $\rightarrow$ bitstring]

$\mathsf{EVAL}(x)$

assert $x \in \{0,1\}^{\text{in}}$
if $T[x] = \bot$
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$y \leftarrow T[x]$
return y

Figure 2: The games $\texttt{Gprf}^0_f$ and $\texttt{Gprf}^1_f$ and the packages $\texttt{Prf}^0_f$ and $\texttt{Prf}^1$.

Due to Claim 2, for a PRF, from now on, we only use the term *secure*, regardless of whether we aim to use Definition 1.1 or Definition 1.3 in a statement/proof. Claim 1 is shown by *inlining* the Key package. In the following, we show how Claim 2 follows from Claim 1.

Proof. Let $\mathcal{A}$ be an efficient adversary. Due to the code equivalence (Claim 1), we have that

$$\Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{EVAL}}{\rightarrow}\ \mathtt{Gprf}_f^0}\Big] = \Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{EVAL}}{\rightarrow}\ \mathtt{Prf}_f^0\ \overset{\mathsf{GET}}{\rightarrow}\ \mathtt{Key}}\Big]. \tag{5}$$

Therefore,

$$\begin{aligned}\mathsf{Adv}_{f,\mathcal{A}}^{\mathtt{Gprf}_f^0,\mathtt{Gprf}^1} &\overset{\text{def}}{=} \Big|\Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{EVAL}}{\rightarrow}\ \mathtt{Gprf}_f^0}\Big] - \Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{EVAL}}{\rightarrow}\ \mathtt{Gprf}^1}\Big]\Big| \\ &\overset{\text{Eq. 5}}{=} \Big|\Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{EVAL}}{\rightarrow}\ \mathtt{Prf}_f^0\ \overset{\mathsf{GET}}{\rightarrow}\ \mathtt{Key}}\Big] - \Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{EVAL}}{\rightarrow}\ \mathtt{Gprf}^1}\Big]\Big| \\ &\overset{\text{def}}{=} \mathsf{Adv}_{f,\mathcal{A}}^{\mathtt{Prf}_f^0\to\mathtt{Key},\mathtt{Gprf}^1}\end{aligned}$$

Thus, $\mathsf{Adv}_{f,\mathcal{A}}^{\mathtt{Gprf}_f^0,\mathtt{Gprf}^1}$ is negligible if and only if $\mathsf{Adv}_{f,\mathcal{A}}^{\mathtt{Prf}_f^0\to\mathtt{Key},\mathtt{Gprf}^1}$ is negligible. □

2 PRFs imply PRGs

We defined security of a PRG g via an experiment where, if the secret bit $b = 0$, then the adversary receives $y := g(x)$ for a uniformly random x, and if $b = 1$, then the adversary receives a uniformly random string y of the same length as $g(x)$. We can also security of PRGs via indistinguishability between two games, each of which provide one oracle SAMPLE which can only be queried once (this is ensured via an **assert** in the beginning of the oracle—the oracle just checks whether it has been called before.), and in the real game ($b = 0$), SAMPLE samples a uniformly random x and returns $y := g(x)$, whereas in the ideal game ($b = 1$), SAMPLE samples a uniformly random string y of the same length as $g(x)$. We now define the games and PRG security formally.

Definition 2.1 (PRG Security)**.** A PRG g with stretch $s : \mathbb{N} \to \mathbb{N}$ with $s(\lambda) > 0$ for all λ is secure if for all PPT adversaries $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A},g}^{\mathtt{Gprg}_g^0,\mathtt{Gprg}^1}(\lambda) := \Big|\Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}\ \mathtt{Gprg}_g^0}\Big] - \Pr\Big[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}\ \mathtt{Gprg}^1}\Big]\Big|$$

is negligible.

$\underline{\underline{\mathsf{Gprg}_g^0}}$

Package Parameters

λ:	security parameter
$s(\lambda)$:	length-expansion
g:	function, $\lvert g(x)\rvert = \lvert x\rvert + s(\lvert x\rvert)$

Package State

y: image value

SAMPLE()

assert $y = \bot$
$x \leftarrow\!\$\ \{0,1\}^\lambda$
$y \leftarrow g(x)$
return y

$\underline{\mathsf{Gprg}^1}$

Package Parameters

λ:	security parameter
$s(\lambda)$:	length-expansion

Package State

y: random value

SAMPLE()

assert $y = \bot$
$y \leftarrow\!\$\ \{0,1\}^{\lambda+s(\lambda)}$
return y

$\underline{\underline{\mathtt{R}_g}}$

Package Parameters

λ:	security parameter
$s(\lambda)$:	length-expansion
g:	function, $\lvert g(x)\rvert = \lvert x\rvert + s(\lvert x\rvert)$

Package State

y: image value

SAMPLE()

assert $y = \bot$
$y_\ell \leftarrow \mathsf{EVAL}(0^\lambda)$
$y_r \leftarrow \mathsf{EVAL}(1^\lambda)$
$y \leftarrow y_\ell || y_r$
return y

Theorem 1 (PRFs imply PRGs)**.** Let f be a (λ, λ)-PRF, then

$$g(k) := f(k, 0^n) || f(k, 1^n)$$

is a PRG with stretch $s(\lambda) = \lambda$.

Proof. Assume towards contradiction that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathtt{Gprg}_g^0, \mathtt{Gprg}^1}_{\mathcal{A},g}(\lambda)$ is non-negligible, i.e.,

$$\left| \Pr\left[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}\ \mathtt{Gprg}_g^0}\right] - \Pr\left[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}\ \mathtt{Gprg}^1}\right] \right| \tag{6}$$

is non-negligible. We need to construct an efficient adversary against the PRF. Now, consider reduction $\mathtt{R}_g$ as defined above. We show that the following two equations hold:

$$\mathtt{Gprg}_g^0 \overset{\text{code}}{\equiv} \mathtt{R}_g \rightarrow \mathtt{Gprf}_f^0 \tag{7}$$

$$\mathtt{Gprg}_g^1 \overset{\text{code}}{\equiv} \mathtt{R}_g \rightarrow \mathtt{Gprf}_f^1 \tag{8}$$

Before proving (7) and (8), we show that they imply that $\mathcal{A} \rightarrow \mathtt{R}_g$ has non-negligible advantage against the PRF game of f, leading to a contradiction. Namely,

$$
\begin{aligned}
& \mathsf{Adv}^{\mathtt{Gprg}_g^0,\mathtt{Gprg}^1}_{\mathcal{A},g}(\lambda) \\
\overset{\text{def}}{=} & \left|\Pr\left[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}} \mathtt{Gprg}_g^0\right] - \Pr\left[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}} \mathtt{Gprg}^1\right]\right| \\
\overset{(7)\ \&\ (8)}{=} & \left|\Pr\left[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}} (\mathtt{R}_g \rightarrow \mathtt{Gprf}_f^0)\right] - \Pr\left[1 = \mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}} (\mathtt{R}_g \rightarrow \mathtt{Gprf}_f^1)\right]\right| \\
= & \left|\Pr\left[1 = (\mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}} \mathtt{R}_g) \rightarrow \mathtt{Gprf}_f^0\right] - \Pr\left[1 = (\mathcal{A}^{\overset{\mathsf{SAMPLE}}{\rightarrow}} \mathtt{R}_g) \rightarrow \mathtt{Gprf}_f^1)\right]\right| \\
\overset{\text{def}}{=} & \mathsf{Adv}^{\mathtt{Gprf}_f^0,\mathtt{Gprf}^f}_{\mathcal{A}\rightarrow\mathtt{R}_g,f}(\lambda)
\end{aligned}
$$

In other words, (7) and (8) imply that

$$\mathsf{Adv}^{\mathtt{Gprg}_g^0,\mathtt{Gprg}^1}_{\mathcal{A},g}(\lambda) = \mathsf{Adv}^{\mathtt{Gprf}_f^0,\mathtt{Gprf}^f}_{\mathcal{A}\rightarrow\mathtt{R}_g,f}(\lambda)$$

and since $\mathsf{Adv}^{\mathtt{Gprg}_g^0,\mathtt{Gprg}^1}_{\mathcal{A},g}(\lambda)$ is non-negligible, the advantage $\mathsf{Adv}^{\mathtt{Gprf}_f^0,\mathtt{Gprf}^f}_{\mathcal{A}\rightarrow\mathtt{R}_g,f}(\lambda)$ is non-negligible, too. Thus, it suffices to prove (7) and (8).

Proof of (7) The proof proceeds by inlining. Below, in the left-most column, we display $\mathtt{Gprg}_g^0$. In the second column, we inline the code of g into $\mathtt{Gprg}_g^0$. In the right-most column, we display $\mathtt{R}_g$. In the second column from the right, we inlined the code of the EVAL oracle of $\mathtt{Gprf}_f^0$ into $\mathtt{R}_g$. From the second column from the right to the third column from the right, we observe that the second **if** $k = \bot$ **then** $k \leftarrow\$ \{0,1\}^\lambda$ is superfluous since k has already been sampled. Moreover, we observe that when $y = \bot$, then also $k = \bot$, we thus do not need to check whether $k = \bot$. Comparing the second column from the left and the third column from the right, we notice that they are equal by renaming variable k to x and omitting the intermediate assignment to y_ℓ and y_r. Thus, (7) follows.

$\underline{\underline{\mathsf{Gprg}_g^0}}$	$\underline{\underline{\mathsf{Gprg}_g^0}}$	$\underline{\underline{\mathtt{R}_g \rightarrow \mathtt{Gprf}_f^0}}$	$\underline{\underline{\mathtt{R}_g \rightarrow \mathtt{Gprf}_f^0}}$	$\underline{\underline{\mathtt{R}_g}}$
$\mathsf{SAMPLE}()$	$\mathsf{SAMPLE}()$	$\mathsf{SAMPLE}()$	$\mathsf{SAMPLE}()$	$\mathsf{SAMPLE}()$
assert $y = \bot$	**assert** $y = \bot$	**assert** $y = \bot$	**assert** $y = \bot$	**assert** $y = \bot$
$x \leftarrow\$ \{0,1\}^\lambda$	$x \leftarrow\$ \{0,1\}^\lambda$		**if** $k = \bot$ **then**	$y_\ell \leftarrow \mathsf{EVAL}(0^\lambda)$
$y \leftarrow g(x)$	$y \leftarrow f(x, 0^n)\|\|f(x, 1^n)$	$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$	$y_r \leftarrow \mathsf{EVAL}(1^\lambda)$
return y	**return** y	$y_\ell \leftarrow f(k, 0^\lambda)$	$y_\ell \leftarrow f(k, 0^\lambda)$	$y \leftarrow y_\ell\|\|y_r$
			if $k = \bot$ **then**	**return** y
			$k \leftarrow\$ \{0,1\}^\lambda$	
		$y_r \leftarrow f(k, 1^\lambda)$	$y_r \leftarrow f(k, 1^\lambda)$	
		$y \leftarrow y_\ell\|\|y_r$	$y \leftarrow y_\ell\|\|y_r$	
		return y	**return** y	

Proof of (8) The proof proceeds by inlining. Below, in the left-most column, we display $\mathtt{Gprg}^1_{s(\lambda)=\lambda}$. In the right-most column, we display $\mathtt{R}_g$. In the second column from the right, we inlined the code of the EVAL oracle of $\mathtt{Gprf}_f^1$ into $\mathtt{R}_g$.

In the second column from the right, we inlined the code of the EVAL oracle of $\mathtt{Gprf}_f^1$ into $\mathtt{R}_g$. From the second column from the right to the third column from the right, we observe that the assignments to $T[0^n]$ and $T[1^n]$ can be skipped since these variables are not used. Comparing the left-most column and the 2nd column from the right, we notice that they are equal by observing that sampling y_ℓ and y_r of length λ each is the same as sampling y of length 2λ uniformly at random at once. Thus, (8) follows.

$\underline{\underline{\mathsf{Gprg}_g^1}}$	$\underline{\underline{\mathtt{R}_g \to \mathtt{Gprf}^1}}$	$\underline{\underline{\mathtt{R}_g \to \mathtt{Gprf}^1}}$	$\underline{\underline{\mathtt{R}_g}}$
SAMPLE()	SAMPLE()	SAMPLE()	SAMPLE()
assert $y = \bot$	**assert** $y = \bot$	**assert** $y = \bot$	**assert** $y = \bot$
$y \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$		**if** $T[0^n] = \bot$ **then**	$y_\ell \leftarrow \mathsf{EVAL}(0^\lambda)$
return y		$T[0^n] \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_r \leftarrow \mathsf{EVAL}(1^\lambda)$
	$y_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_\ell \leftarrow T[0^n]$	$y \leftarrow y_\ell \| y_r$
		if $T[1^n] = \bot$ **then**	**return** y
		$T[1^n] \leftarrow\!\!\$\ \{0,1\}^\lambda$	
	$y_r \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_r \leftarrow T[1^n]$	
	$y \leftarrow y_\ell \| y_r$	$y \leftarrow y_\ell \| y_r$	
	return y	**return** y	

Remark. For an alternative way of writing this proof, see the model solutions of Exercise Sheet 2.

3 Message Authentication Codes

Definition 3.1 (Syntax of Message Authentication Code (MAC))**.** A *message authentication code* m consists of two algorithms

$$m.\mathsf{mac} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^* \qquad \text{(this algorithm is deterministic)}$$
$$m.\mathsf{ver} : \{0,1\}^\lambda \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\} \qquad \text{(this algorithm is deterministic)}$$

which have to satisfy the correctness criterion

$$\forall\, x \in \{0,1\}^*, \Pr_{k \leftarrow\!\!\$ \{0,1\}^\lambda}[m.\mathsf{ver}(k, x, m.\mathsf{mac}(k, x)) = 1] = 1$$

i.e. the verification algorithm accepts any message-tag pair produced by the tagging algorithm.

The mac algorithm $m.\mathsf{mac}$ creates a tag based on a message, and the verification algorithm $m.\mathsf{ver}$ verifies that a tag corresponds to a given message. In the following, we will state two equivalent security definitions for MACs, where the second results from outlining the Key package again.

Definition 3.2 (UNF-CMA)**.** Let the games $\mathtt{Gunf\text{-}cma}_m^0$, $\mathtt{Gunf\text{-}cma}_m^1$ be defined by

$\underline{\texttt{Gunf-cma}^0_m}$	$\underline{\texttt{Gunf-cma}^1_m}$
Package Parameters	Package Parameters
λ: security parameter	λ: security parameter
m: MAC scheme	m: MAC scheme
Package State	Package State
k: key	k: key
	$\mathcal{L}$: list
$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$
if $k = \bot$:	**if** $k = \bot$:
$k \leftarrow\!\!\$\ \{0,1\}^\lambda$	$k \leftarrow\!\!\$\ \{0,1\}^\lambda$
$t \leftarrow m.\mathsf{mac}(k, x)$	$t \leftarrow m.\mathsf{mac}(k, x)$
	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, t)\}$
return t	**return** t
$\mathsf{VERIFY}(x, t)$	$\mathsf{VERIFY}(x, t)$
assert $t \neq \bot$	**assert** $t \neq \bot$
if $k = \bot$:	**if** $(x, t) \in \mathcal{L}$:
$k \leftarrow\!\!\$\ \{0,1\}^\lambda$	**return** 1
$d \leftarrow m.\mathsf{ver}(k, x, t)$	
return d	**return** 0

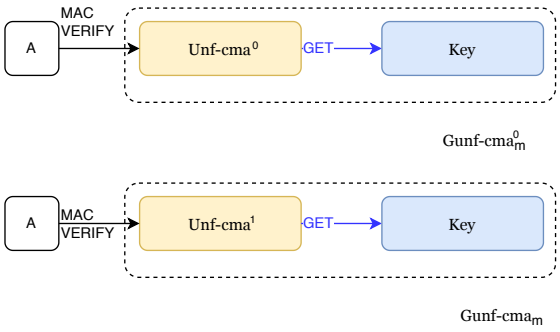
A message authentication code m is UNF-CMA-secure if the real and ideal games $\texttt{Gunf-cma}^b_m$ are computationally indistinguishable, i.e., for all PPT adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\texttt{Gunf-cma}^0_m, \texttt{Gunf-cma}^1_m}_{\mathcal{A}}(\lambda) :=$

$$\left|\Pr\left[1 = \mathcal{A} \to \texttt{Gunf-cma}^0_m\right] - \Pr\left[1 = \mathcal{A} \to \texttt{Gunf-cma}^1_m\right]\right|$$

is negligible in λ.

Definition 3.3 (UNF-CMA′)**.** Let the packages $\texttt{Unf-cma}^0$ and $\texttt{Unf-cma}^1$ be defined by

$\underline{\texttt{Unf-cma}^0}$	$\underline{\texttt{Unf-cma}^1}$
Package Parameters	Package Parameters
λ: security parameter	λ: security parameter
m: MAC scheme	m: MAC scheme
Package State	Package State
no state	$\mathcal{L}$: list
$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$
$k \leftarrow \mathsf{GET}$	$k \leftarrow \mathsf{GET}$
$t \leftarrow m.\mathsf{mac}(k, x)$	$t \leftarrow m.\mathsf{mac}(k, x)$
	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, t)\}$
return t	**return** t
$\mathsf{VERIFY}(x, t)$	$\mathsf{VERIFY}(x, t)$
assert $t \neq \bot$	**assert** $t \neq \bot$
$k \leftarrow \mathsf{GET}$	**if** $(x, t) \in \mathcal{L}$:
$d \leftarrow m.\mathsf{ver}(k, x, t)$	**return** 1
return d	**return** 0



A message authentication code m is UNF-CMA′-secure if the real game $\texttt{Unf-cma}^0 \to \texttt{Key}$ and ideal game $\texttt{Unf-cma}^1 \to \texttt{Key}$ are computationally indistinguishable, that is, for all PPT adversaries $\mathcal{A}$, the advantage

$$\begin{aligned}&\mathbf{Adv}^{\texttt{Unf-cma}^0 \to \texttt{Key}, \texttt{Unf-cma}^1 \to \texttt{Key}}_{\mathcal{A}}(\lambda)\\ &\quad := \left|\Pr\left[1 = \mathcal{A} \to \texttt{Unf-cma}^0 \to \texttt{Key}\right] - \Pr\left[1 = \mathcal{A} \to \texttt{Unf-cma}^1 \to \texttt{Key}\right]\right|\end{aligned}$$

is negligible in λ.

Claim 3. The security definitions UNF-CMA and UNF-CMA′ are equivalent, i.e., a message authentication code m is UNF-CMA secure if and only if m is UNF-CMA′-secure. In particular, the code equivalence statements

$$\texttt{Gunf-cma}^0_m \overset{\text{code}}{\equiv} \texttt{Unf-cma}^0 \to \texttt{Key}$$

$$\texttt{Gunf-cma}^1_m \overset{\text{code}}{\equiv} \texttt{Unf-cma}^1 \to \texttt{Key}$$

hold.

4 PRFs imply MACs

In this section, we show that a PRF is, essentially a MAC. I.e., if f is a PRF, then the MAC algorithm $m_f.\mathsf{mac}$ computes a MAC tag t on message x simply by running f on its key k and the message x. The verification algorithm $m_f.\mathsf{ver}$, on input the key k, the message x and some tag t^*, recomputes $t \leftarrow f(k,x)$ and returns 1 if and only if $t = t^*$. The left column of Fig. 3 describes the MAC scheme m_f in pseudo-code. The remainder of this section reduces the security of the MAC scheme m_f to the PRF security of the underlying PRF f.

$\underline{\underline{m_f}}$	$\underline{\underline{\mathtt{R}_1}}$	$\underline{\underline{\mathtt{R}_2}}$
	Pack. Par.	Pack. Par.
	no params	no params
	Pack. State	Pack. State
	no state	$\mathcal{L}$: list
$\underline{m_f.\mathsf{mac}(k,x)}$	$\underline{\mathsf{MAC}(x)}$	$\underline{\mathsf{MAC}(x)}$
$t \leftarrow f(k,x)$	$t \leftarrow \mathsf{EVAL}(x)$	$t \leftarrow \mathsf{EVAL}(x)$
		$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x,t)\}$
return t	**return** t	**return** t
$\underline{m_f.\mathsf{ver}(k,x,t)}$	$\underline{\mathsf{VERIFY}(x,t)}$	$\underline{\mathsf{VERIFY}(x,t)}$
	assert $t \neq \bot$	**assert** $t \neq \bot$
if $t = f(k,x)$:	$y \leftarrow \mathsf{EVAL}(x)$:	**if** $(x,t) \in \mathcal{L}$:
return 1	**if** $t = y$: **return** 1	**return** 1
return 0	**return** 0	**return** 0

Figure 3: Message authentication code scheme m and reductions $\mathtt{R}_1$ and $\mathtt{R}_2$. Observe that $\mathtt{R}_1$ is a very much a construction analogous reduction (CAR).

Theorem 2. If f is a secure $(*,\lambda)$-PRF, then m_f (defined in Fig. 3) is an UNF-CMA-secure MAC. In particular, there exists two PPT reductions $\mathtt{R}_1$ and $\mathtt{R}_2$ such that for all PPT adversaries $\mathcal{A}$, it holds that

$$\begin{aligned} &\mathsf{Adv}^{\mathtt{Gunf\text{-}cma}^0_{m_f},\mathtt{Gunf\text{-}cma}^1_{m_f}}_{m_f,\mathcal{A}}(\lambda) \\ &\quad \leq \mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_1}(\lambda) + \mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_2}(\lambda) + \frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda}, \end{aligned} \tag{9}$$

where $\mathsf{qry}_{\mathcal{A}}(\lambda)$ is a (polynomial) upper bound on the (polynomial) number of oracle queries which $\mathcal{A}$ makes.

4.1 Proof of Theorem 2

In this section, we provide the complete proof of Theorem 2. In Section 4.2, we provide an overview/summary of what happens in the proof conceptually. If you like to, you can read Section 4.2 first.

Proving (9) suffices. If we can provide two PPT reductions $\mathtt{R}_1$ and $\mathtt{R}_2$ such that (9) holds for all PPT adversaries $\mathcal{A}$, then Theorem 2 follows. Namely, assume towards contradiction that there exist an adversary $\mathcal{A}$ such that the advantage $\mathsf{Adv}^{\mathtt{Gunf\text{-}cma}^0_{m_f},\mathtt{Gunf\text{-}cma}^1_{m_f}}_{m_f,\mathcal{A}}(\lambda)$ is non-negligible, then by (9), at least one out of

$$\mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_1}(\lambda),\ \mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_2}(\lambda) \text{ and } \frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda}$$

must be non-negligible, since if they were all negligible, then their sum would be negligible, but by (9), their sum is non-negligible. Now, $\frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda}$ is negligible, since (i) the number of queries $\mathsf{qry}_{\mathcal{A}}(\lambda)$ which $\mathcal{A}$ makes is polynomial, (ii) $\frac{1}{2^\lambda}$ is negligible, and (iii) multiplying a negligible function by a polynomial yields a negligible function. Hence,

$$\mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_1}(\lambda) \text{ or } \mathsf{Adv}^{\mathtt{Gprf}^0,\mathtt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_2}(\lambda)$$

must be non-negligible, but since $\mathcal{A}\to\mathtt{R}_1$ and $\mathcal{A}\to\mathtt{R}_2$ are both efficient adversaries, this contradicts the security of f and we reached a contradiction. Thus, it suffices to prove that there exists two reductions $\mathtt{R}_1$ and $\mathtt{R}_2$ such that for all PPT adversaries $\mathcal{A}$, (9) holds.

Proving (10), (11) and (12) suffices. Before we define $\mathtt{R}_1$ and $\mathtt{R}_2$, we will show that in order to prove (9), it suffices to find two PPT reductions $\mathtt{R}_1$ and $\mathtt{R}_2$ such that

$$\mathtt{Gunf\text{-}cma}^0_{m_f} \overset{\text{code}}{\equiv} \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}^0_f \tag{10}$$

$$\mathtt{Gunf\text{-}cma}^1_{m_f} \overset{\text{code}}{\equiv} \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}^0_f \tag{11}$$

$$\Pr\left[1=\mathcal{A}\to\mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}^1_f\right] = \Pr\left[1=\mathcal{A}\to\mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}^1_f\right] + \frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda} \tag{12}$$

hold. This is true, because

$$
\begin{aligned}
& \mathsf{Adv}^{\texttt{Gunf-cma}^0_{m_f},\texttt{Gunf-cma}^1_{m_f}}_{m_f,\mathcal{A}}(\lambda) \\
\overset{\text{def}}{=} \quad & \left|\Pr\left[1=\mathcal{A}\to \texttt{Gunf-cma}^0_{m_f}\right] - \Pr\left[1=\mathcal{A}\to \texttt{Gunf-cma}^1_{m_f}\right]\right| \\
\overset{(10),(11)}{=} \quad & \left|\Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f\right]\right| \\
\overset{\text{adding 0}}{=} \quad & |\Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] \\
& + \Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] \\
& + \Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f\right]| \\
\overset{\text{triangle inequ.}}{\leq} \quad & \left|\Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right]\right| \\
& + \left|\Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right]\right| \\
& + \left|\Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] - \Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f\right]\right| \\
\overset{\text{def.}}{=} \quad & \mathsf{Adv}^{\texttt{Gprf}^0,\texttt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_1}(\lambda) \\
& + \left|\Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] - \Pr\left[1=\mathcal{A}\to \texttt{Gunf-cma}^1_{m_f}\right]\right| \\
& + \mathsf{Adv}^{\texttt{Gprf}^0,\texttt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_2}(\lambda) \\
\overset{(12)}{\leq} \quad & \mathsf{Adv}^{\texttt{Gprf}^0,\texttt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_1}(\lambda) + \frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda} + \mathsf{Adv}^{\texttt{Gprf}^0,\texttt{Gprf}^1}_{f,\mathcal{A}\to\mathtt{R}_2}(\lambda)
\end{aligned}
$$

Here, the two equalities labeled by *def.* use the definition of advantage. The equation labeled by *(10),(11)* replaces the $\texttt{Gunf-cma}^0_{m_f}$ and $\texttt{Gunf-cma}^1_{m_f}$ games by their code-equivalent counterpart guaranteed by (10) and (11). The inequality labeled by *(12)* plugs-in inequality (12). The equality labeled by *adding 0* adds and substracts

$$\Pr\left[1=\mathcal{A}\to \mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right]$$

and

$$\Pr\left[1=\mathcal{A}\to \mathtt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right],$$

which corresponds to adding zeroes. Finally, the step labeled by *triangle ineq.* uses the triangle inequality for absolute values, i.e. $|a+b| \leq |a|+|a|$. Since (10), (11) and (12) imply (9), all that is left to do is to define two PPT reductions $\mathtt{R}_1$ and $\mathtt{R}_2$ such that (10), (11) and (12) holds. We define $\mathtt{R}_1$ and $\mathtt{R}_2$ in the middle and right column of Fig. 3, respectively, and now prove (10), (11) and (12) each in turn.

Proof of (10). We need to show that $\texttt{Gunf-cma}^0_{m_f}$ is code-equivalent to $\mathtt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f$. This argument is quite simple, because the reduction $\mathtt{R}_1$ is analogous to the construction. Below, in the 1st column from the left, we show $\texttt{Gunf-cma}^0_{m_f}$ and from the 1st to 2nd column from the left, we inline the code of m_f, highlighted in grey. In the 2nd column from the right, we depict $\mathtt{R}_1$. From the 2nd to 3rd column from the right, we inlined the code of EVAL, highlighted

in grey. We depict the code of EVAL in the 1st column from the right. Comparing the 2nd column from the left and the 3rd column from the right, we observe that they are equal as required.

$\underline{\underline{\texttt{Gunf-cma}^0_{m_f}}}$	$\underline{\underline{\texttt{Gunf-cma}^0_{m_f}}}$	$\underline{\underline{\texttt{R}_1 \to \texttt{Gprf}^0_f}}$	$\underline{\underline{\texttt{R}_1}}$	$\underline{\underline{\texttt{Gprf}^0_f}}$
Pack. Par.	Pack. Par.	Pack. Param.	Pack. Par.	Pack. Par.
λ: sec. par.	λ: sec. par.	λ: sec. par.	no params	λ: sec. par.
m_f: MAC	f: $(*,\lambda)$-PRF	f: $(*,\lambda)$-PRF		f: $(*,\lambda)$-PRF
Pack. State	Pack. State	Pack. State	Pack. State	Pack. State
k: key	k: key	k: key	no state	k: key
$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$	$\mathsf{EVAL}(x)$
if $k = \bot$:	**if** $k = \bot$:	**if** $k = \bot$:	$t \leftarrow \mathsf{EVAL}(x)$	**if** $k = \bot$:
$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$		$k \leftarrow\$ \{0,1\}^\lambda$
$t \leftarrow m_f.\mathsf{mac}(k,x)$	$t \leftarrow f(k,x)$	$t \leftarrow f(k,x)$	**return** t	$y \leftarrow f(k,x)$
return t	**return** t	**return** t		**return** y
$\mathsf{VERIFY}(x,t)$	$\mathsf{VERIFY}(x,t)$	$\mathsf{VERIFY}(x,t)$	$\mathsf{VERIFY}(x,t)$	
assert $t \neq \bot$	**assert** $t \neq \bot$	**assert** $t \neq \bot$	**assert** $t \neq \bot$	
if $k = \bot$:	**if** $k = \bot$:	**if** $k = \bot$:	$y \leftarrow \mathsf{EVAL}(x)$	
$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$		
$d \leftarrow m_f.\mathsf{ver}(k,x,t)$	$y \leftarrow f(k,x)$	$y \leftarrow f(k,x)$		
return d	**if** $t = y$:	**if** $t = y$:	**if** $t = y$:	
	return 1	**return** 1	**return** 1	
	return 0	**return** 0	**return** 0	

Proof of (11). We need to show that $\texttt{Gunf-cma}^1_{m_f}$ is code-equivalent to $\texttt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^0_f$. This argument is conceptually more difficult to think about than the proof of (10), but technically, there are no additional complications. Below, in the 1st column from the left, we show $\texttt{Gunf-cma}^1_{m_f}$ and from the 1st to 2nd column from the left, we inline the code of m_f, highlighted in grey. In the 2nd column from the right, we depict $\texttt{R}_2$. From the 2nd to 3rd column from the right, we inlined the code of EVAL, highlighted in grey. We depict the code of EVAL in the 1st column from the right. Comparing the 2nd column from the left and the 3rd column from the right, we observe that they are equal as required.

$\underline{\underline{\texttt{Gunf-cma}^1_{m_f}}}$	$\underline{\underline{\texttt{Gunf-cma}^1_{m_f}}}$	$\underline{\underline{\texttt{R}_1 \to \texttt{Gprf}^0_f}}$	$\underline{\underline{\texttt{R}_2}}$	$\underline{\underline{\texttt{Gprf}^0_f}}$
Pack. Par.	Pack. Par.	Pack. Param.	Pack. Par.	Pack. Par.
λ: sec. par.	λ: sec. par.	λ: sec. par.	no params	λ: sec. par.
m_f: MAC	f: $(*,\lambda)$-PRF	f: $(*,\lambda)$-PRF		f: $(*,\lambda)$-PRF
Pack. State	Pack. State	Pack. State	Pack. State	Pack. State
k: key	k: key	k: key	$\mathcal{L}$: list	k: key
$\mathcal{L}$: list	$\mathcal{L}$: list	$\mathcal{L}$: list		
$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$	$\mathsf{MAC}(x)$	$\mathsf{EVAL}(x)$
if $k = \bot$:	**if** $k = \bot$:	**if** $k = \bot$:	$t \leftarrow \mathsf{EVAL}(x)$	**if** $k = \bot$:
$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$	$k \leftarrow\$ \{0,1\}^\lambda$		$k \leftarrow\$ \{0,1\}^\lambda$
$t \leftarrow m_f.\mathsf{mac}(k,x)$	$t \leftarrow f(k,x)$	$t \leftarrow f(k,x)$		$y \leftarrow f(k,x)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x,t)\}$	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x,t)\}$	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x,t)\}$	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x,t)\}$	**return** y
return t	**return** t	**return** t	**return** t	
$\mathsf{VERIFY}(x,t)$	$\mathsf{VERIFY}(x,t)$	$\mathsf{VERIFY}(x,t)$	$\mathsf{VERIFY}(x,t)$	
assert $t \neq \bot$	**assert** $t \neq \bot$	**assert** $t \neq \bot$	**assert** $t \neq \bot$	
if $(x,t) \in \mathcal{L}$:	**if** $(x,t) \in \mathcal{L}$:	**if** $(x,t) \in \mathcal{L}$:	**if** $(x,t) \in \mathcal{L}$:	
return 1	**return** 1	**return** 1	**return** 1	
return 0	**return** 0	**return** 0	**return** 0	

Proof of (12). It remains to argue that the difference

$$\left|\Pr\left[1 = \mathcal{A} \to \texttt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right] - \Pr\left[1 = \mathcal{A} \to \texttt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f\right]\right|$$

is upped bound by $\frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda}$. This argument is information-theoretic. The difference between $\texttt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$ and $\texttt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$ is that in the latter, the VERIFY oracle returns 1 only on inputs (x^*, t^*) which are in $\mathcal{L}$, i.e., come from the MAC oracle. In particular, here, VERIFY always returns 0 on inputs $(x^*, t*)$ where the adversary has never asked x^* to the MAC oracle before. In turn, in $\texttt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$, the game checks the value stored in table T and the adversary might have guesses the correct value t which is stored in T. Therefore, the two games $\texttt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$ and $\texttt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$ are not identical and, instead, have a small statistical difference.

To compute this difference, observe that each time the adversary $\mathcal{A}$ makes a $\mathsf{VERIFY}(x^*, t^*)$ query for a value x it has not queried to MAC before, then the actual tag t which is sampled and stored in the list $\mathcal{L}$ is information-theoretically hidden from the adversary *unless* the adversary uses $t^* = t$ in its query. Therefore, if the adversary makes $\mathsf{qry}(\lambda)$ many queries, the probability of guessing the correct tag t for a value x is only $\mathsf{qry}(\lambda)/2^\lambda$—if the adversary makes guesses for different x values, the probabilility is even lower. Thus, the difference between $\texttt{R}_1 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$ and $\texttt{R}_2 \overset{\mathsf{EVAL}}{\to} \texttt{Gprf}^1_f$ is upper bounded by $\frac{\mathsf{qry}_{\mathcal{A}}(\lambda)}{2^\lambda}$.

Remark in the proof of (12). The above argument is correct, but it is less detailed than most other proofs we carry out in this course. While in the research literature of cryptography, the above proof is considered sufficiently detailed, you might not be satisfied with the level of detail if you prefer the detailed style used in the most of the material of this course. I personnally would prefer a more detailed proof within the writing style of this course, but I have not found a nice way of writing it (yet). If you think you have a nice idea for how to write the proof, you can let me know :-).

4.2 Proof Summary

The high-level argument for Theorem 2 is that if it is hard to distinguish the input-output behaviour of a PRF from a uniformly random function, then it is also hard to come up with the PRF value on some value x for which one hasn't seen the PRF value yet. Conceptually, the proof proceeds by (A) replacing the PRF by a random function, (B) replacing "real" verification by "ideal" verification which only accepts strings generated by the actual MAC oracle, and (C) replacing the random function by a PRF again. (A) and (C) reduce to PRF security, and (B) holds statistically, because guessing a string of length n correctly with $\mathsf{qry}(n)$ many guesses has probability at most $\mathsf{qry}(n)/2^n$. If $\mathsf{qry}(n)$ is a polynomial in n, then $\mathsf{qry}(n)/2^n$ is negligible. We now relate this conceptual high-level overview to the formal statements in the proof provided in Section 4.1.
(10) shows that $\mathtt{R}_1$ is a suitable reduction to prove (A). I.e., $\mathtt{R}_1 \to \mathtt{Gprf}^0$ emulates the real unforgeability game $\mathtt{Gunf\text{-}cma}^0_{m_f}$ adequately, and then after applying PRF security, $\mathtt{R}_1 \to \mathtt{Gprf}$ is, essentially, the real unforgeability game, but with the real PRF replaced by a random function, as we outlined for (A).
(12) shows that $\mathtt{R}_2$ is a suitable reduction to prove (B), i.e., $\mathtt{R}_2 \to \mathtt{Gprf}^0$ emulates the *ideal* unforgeability game $\mathtt{Gunf\text{-}cma}^1_{m_f}$ adequately, i.e., when we apply the PRF security to $\mathtt{R}_2 \to \mathtt{Gprf}^1$, we can replace $\mathtt{Gprf}^1$ by $\mathtt{Gprf}^0$ and we obtain the ideal unforgeability game $\mathtt{Gunf\text{-}cma}^1_{m_f}$.
Finally, (11) directly shows (C).