CS-E4340 Cryptography: Exercise Sheet 6

—Public-key Encryption & Signature Schemes—

Submission deadline: October 31, 2022, 11:30, via MyCourses

Each exercise can give up to two participation points, 2 for a mostly correct solution and 1 point for a good attempt. Overall, the exercise sheet gives at most 4 participation points. We encourage to **choose** exercises which seem **interesting** and/or adequately challenging to you.

Exercise Sheet 6 is intended to help...

(a) ...understand the *definition* of IND-CPA security of public-key encryption (Ex. 1 & Ex.2).
(b) ...understand the *definition* of UNF-CMA security for digital signature schemes (Ex. 3 & Ex. 4).
(c) ...familiarize yourself with textbook RSA and its limitations (Ex. 2 & Ex.3).
(d) ...reflect on the relation between signature schemes and public-key encryption (Ex. 3).
(e) ...reflect on the relation between signature schemes and one-way functions (Ex. 4).

Exercise 1 (Deterministic PKE is insecure)**.** On Ex. Sheet 5, we showed that symmetric-key encryption is not IND-CPA-secure and described an attack using *two* ENC queries. Let pke_{weak} be a correct PKE where $pke_{\text{weak}}.enc$ is deterministic.
Task: Describe a PPT adversary $\mathcal{A}$ in pseudocode which breaks the IND-CPA security of pke_{weak} and only makes a GETPK and a *a single* ENC query. Analyze the success probability of your adversary and show that it is non-negligible.

Solution 1. Consider the following adversary:

```
$\mathcal{A}(1^\lambda)$
$c \leftarrow \mathsf{ENC}(1^\lambda)$
$\mathsf{pk} \leftarrow \mathsf{GETPK}$
$c' \leftarrow pke_{\text{weak}}.enc(\mathsf{pk}, 1^\lambda)$
if $c = c'$
  return 0
else
  return 1
```

Now $\Pr\left[\mathcal{A} \to \texttt{Gpind-cpa}^0_{pke_{\text{weak}}} = 1\right] = 0$ since deterministic encryption function has always the same output for the same input, so c is always equal to c'.
On the other hand, $\Pr\left[\mathcal{A} \to \texttt{Gpind-cpa}^1_{pke_{\text{weak}}} = 1\right] = 1$, since in the ideal game, the encryption oracle ENC encrypts the all zeros string with the correct public key. By the correctness criterion, the decryption algorithm must always be able to decrypt, hence, encrypting all zeros and all ones with the same public key must always lead to different output (so the if clause in the adversary's code will always be false in the ideal case).
Thus,

$$\left|\Pr\left[\mathcal{A} \to \texttt{Gpind-cpa}^0_{pke_{\text{weak}}} = 1\right] - \Pr\left[\mathcal{A} \to \texttt{Gpind-cpa}^1_{pke_{\text{weak}}} = 1\right]\right| = 1,$$

which is non-negligible.

Remark: pke_{RSA} and s_{RSA} operate on inputs from $\{1, .., N-1\}$, i.e.,

the message x, the ciphertext c and signature σ are all in $\{1, .., N-1\}$.

```
$pke_{\mathrm{RSA}}.kgen()$
    sample two big random primes $p, q$
    $N \leftarrow pq$
    $\lambda \leftarrow \mathrm{lcm}(p-1, q-1)$
    choose $e > 1$ that is coprime with $\lambda$
    $d \leftarrow e^{-1} \mod \lambda$
    $sk \leftarrow (d, N)$; $pk \leftarrow (e, N)$
    return $pk$
```

```
$s_{\mathrm{RSA}}.kgen()$
    [same as $pke_{\mathrm{RSA}}.kgen$]
```

```
$pke_{\mathrm{RSA}}.enc(pk, m)$
    $(e, N) \leftarrow pk$
    $c \leftarrow m^e \mod N$
    return $c$
```

```
$pke_{\mathrm{RSA}}.dec(sk, c)$
    $(d, N) \leftarrow sk$
    $m \leftarrow c^d \mod N$
    return $m$
```

```
$s_{\mathrm{RSA}}.sig(sk, m)$
    $(d, N) \leftarrow sk$
    $\sigma \leftarrow m^d \mod N$
    return $\sigma$
```

```
$s_{\mathrm{RSA}}.ver(pk, m, \sigma)$
    $(e, N) \leftarrow pk$
    $m' \leftarrow \sigma^e \mod N$
    return $m = m'$
```

Fig. 1: Textbook RSA (insecure)

Exercise 2 (RSA: Public-Key encryption)**.** Fig. 1 describes the RSA encryption scheme pke_{RSA} and RSA signature scheme s_{RSA} as some textbooks, discrete mathematics courses and the RSA Wikipedia article[1] (see Section 3) do. pke_{RSA} and s_{RSA} are thus called *textbook RSA*. This exercise explores why textbook RSA should not be used *as is* in practice and other confusions/misconceptions emerging from textbook RSA in popular literature.

Task 1: Compute $pke_{\mathrm{RSA}}.dec(\mathsf{sk}, c)$ for ciphertext $c = 61$ and secret-key $\mathsf{sk} = (37, 119)$. (The public-key here is $\mathsf{pk} = (13, 119)$, but it is used for encryption only, not decryption.)

Task 2: Prove that pke_{RSA} is not IND-CPA-secure by giving a PPT adversary $\mathcal{A}$ against the IND-CPA security of pke_{RSA} in pseudo-code. (You can omit the probability analysis.)

Solution 2. **Task 1:** Decryption is defined as

$$\begin{aligned} x &= c^d \mod N \\ &= 61^{37} \mod 119 \\ &= 40. \end{aligned}$$

[1] `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`

Task 2: The PKE scheme pke_{RSA} is deterministic, so the attacker from Ex. 1 works against IND-CPA security of pke_{RSA} (same analysis). (Chris: This was indeed a bit redundant as some people pointed out. Thank you for reporting. We'll merge Exercise 1 and 2 next time we use this exercise sheet.

Exercise 3 (RSA: Signing vs. Public-Key encryption)**.** As in the previous exercise, see Fig. 1 for the textbook RSA encryption scheme pke_{RSA} and textbook RSA signature scheme s_{RSA}.

Task 1: Prove that sig_{RSA} is not UNF-CMA-secure by giving a PPT adversary $\mathcal{A}$ against the UNF-CMA security of sig_{RSA} in pseudo-code. (You can omit the probability analysis.)

Task 2: Some sources describe signature schemes as the "opposite" or "inverse" of encryption. The underlying idea is that textbook RSA encryption $pke_{\mathrm{RSA}}.enc$ encrypts using the public-key, and the textbook RSA signature schemes $s_{\mathrm{RSA}}.sig$ signs using the secret-key.

Reflect whether this intuition generalizes. Can every signature scheme be transformed into a public-key encryption scheme? Justify your belief.

Solution 3. The plain RSA also fails as a signature scheme. For any two signed messages $(m_1, \sigma_1), (m_2, \sigma_2)$ the adversary can generate a new valid signed message $(m_1 m_2, \sigma_1 \sigma_2)$: if (N, e) is the public key and d is the secret key, then $\sigma_i = m_i^d \mod N$, while $m_i = \sigma_i^e \mod N$, therefore $m_1 m_2 = (\sigma_1 \sigma_2)^e \mod N$. See the precise adversary code below:

```
$\mathcal{A}$
$(e, N) \leftarrow\$ \mathrm{GETPK}()$
$m_1 \leftarrow 2$
$m_2 \leftarrow 3$
$\sigma_1 \leftarrow\$ \mathrm{SIG}(m_1)$
$\sigma_2 \leftarrow\$ \mathrm{SIG}(m_2)$
$d \leftarrow \mathrm{VERIFY}(m_1 \cdot m_2 \mod N, \sigma_1 \cdot \sigma_2 \mod N)$
return $d$
```

Now real game always accepts the verification step, but ideal game never accepts it, since $m_1 \cdot m_2$ is not in the list $\mathcal{L}$. So this adversary has advantage 1, i.e. it always succeeds in distinguishing.

Another way is to observe that if $m = 1$, then $m = m^d = \sigma = \sigma^e = 1 \mod N$ for any valid d, e, N. Thus, we can forge a valid signature $\sigma = 1$ for message $m = 1$.

```
$\mathcal{A}$
$m \leftarrow 1$
$\sigma \leftarrow 1$
return VERIFY$(m, \sigma)$
```

As before, the real game will accept and the ideal game will reject the verification, so that the advantage is again 1.

Signature schemes cannot be transformed into public-key encryption schemes, at least in a black box way. The oracle separation proof in lecture 7 is really a separation between public key encryption and the whole minicrypt. This is because everything inside the minicrypt can be transformed to each other in a black-box way, so any black-box construction of public key encryption from e.g. signature schemes would automatically give a way to transform one way functions into public key encryption, which contradicts the oracle separation result.

Exercise 4 (OWFs $\Rightarrow$ SIG)**.** Show that if f is an injective one-way function, then Lamport's signature scheme s_f is one-time UNF-CMA-secure (1-UNF-CMA) for messages of length λ. See Fig. 2 for s_f and see below for the definition of 1-UNF-CMA.

```
$\mathsf{s}_f.kgen(1^\lambda)$
for $i = 1..\lambda$
    $x_0^i \leftarrow\$ \{0,1\}^\lambda$
    $x_1^i \leftarrow\$ \{0,1\}^\lambda$
    $y_0^i \leftarrow f(x_0^i)$
    $y_1^i \leftarrow f(x_1^i)$
$\mathsf{sk} \leftarrow \begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$
$\mathsf{pk} \leftarrow \begin{pmatrix} y_0^1, .., y_0^\lambda \\ y_1^1, .., y_1^\lambda \end{pmatrix}$
return $(\mathsf{sk}, \mathsf{pk})$
```

```
$s_f.sig(\mathsf{sk}, m)$
parse sk as
$\begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$
for $i = 1..\lambda$
    $z^i \leftarrow x_{m[i]}^i$
    // m[i] is ith bit of m
$\sigma \leftarrow (z^1, .., z^\lambda)$
return $\sigma$
```

```
$s_f.ver(\mathsf{pk}, m, \sigma)$
parse pk as
$\begin{pmatrix} y_0^1, .., y_0^\lambda \\ y_1^1, .., y_1^\lambda \end{pmatrix}$
$(z^1, .., z^\lambda) \leftarrow \sigma$
for $i = 1..\lambda$
    if $f(z^i) \neq y_{m[i]}^i$ :
        return 0
return 1
```

Fig. 2: Lamport's one-time signature scheme for messages of length λ.

Public-key encryption scheme pke : $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ pke.kgen(1^\lambda)$

$$c \leftarrow\$ pke.enc(\mathsf{pk}, x)$$
$$x \leftarrow\$ pke.dec(\mathsf{sk}, c)$$

Correctness: $\forall(\mathsf{pk},\mathsf{sk}) \leftarrow\$ pke.kgen(1^\lambda)$, $\forall x \in \{0,1\}^*$: $\forall c \leftarrow\$ pke.enc(\mathsf{pk}, x)$, $pke.dec(\mathsf{sk}, c) = x$.

IND-CPA Security: $\forall$ PPT $\mathcal{A}$

$$|\Pr[1 = \mathcal{A} \to \texttt{Gind-cpa}^0_{pke}] - \Pr[1 = \mathcal{A} \to \texttt{Gind-cpa}^1_{pke}]| \text{ is negligible in } \lambda.$$

```
Gind-cpa^0_pke
═══════════════
GETPK()
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ pke.kgen(1^\lambda)$
return pk

ENC($x$)
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ pke.kgen(1^\lambda)$

$c \leftarrow\$ pke.enc(\mathsf{pk}, x)$
return $c$
```

```
Gind-cpa^1_pke
═══════════════
GETPK()
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ pke.kgen(1^\lambda)$
return pk

ENC($x$)
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ pke.kgen(1^\lambda)$
$x' \leftarrow 0^{|x|}$
$c \leftarrow\$ pke.enc(\mathsf{pk}, x')$
return $c$
```

Signature scheme s : $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$

$$\sigma \leftarrow\$ \mathsf{s}.sig(\mathsf{sk}, x)$$
$$0/1 \leftarrow\$ \mathsf{s}.ver(\mathsf{pk}, x, \sigma)$$

Correctness: $\forall(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$, $\forall x \in \{0,1\}^*$: $\forall \sigma \leftarrow\$ \mathsf{s}.sig(\mathsf{sk}, x)$, $\mathsf{s}.ver(\mathsf{pk}, x, \sigma) = 1$.

UNF-CMA Security: $\forall$ PPT $\mathcal{A}$

$$|\Pr[1 = \mathcal{A} \to \texttt{Gunf-cma}^0_{\mathsf{s}}] - \Pr[1 = \mathcal{A} \to \texttt{Gunf-cma}^1_{\mathsf{s}}]| \text{ is negligible in } \lambda.$$

1-UNF-CMA Security for Lamport: $\forall$ PPT $\mathcal{A}$

$$|\Pr[1 = \mathcal{A} \to \texttt{1-Gunf-cma}^0_{\mathsf{s}}] - \Pr[1 = \mathcal{A} \to \texttt{1-Gunf-cma}^1_{\mathsf{s}}]| \text{ is negligible in } \lambda.$$

```
Gunf-cma^0_s
═══════════════
GETPK()
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
return pk

SIG($x$)
───────────────


if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
$\sigma \leftarrow\$ \mathsf{s}.sig(\mathsf{sk}, x)$

return $\sigma$

VERIFY($x, \sigma$)
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
$d \leftarrow \mathsf{s}.ver(\mathsf{pk}, x, \sigma)$
return $d$
```

```
Gunf-cma^1_s
═══════════════
GETPK()
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
return pk

SIG($x$)
───────────────


if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
$\sigma \leftarrow\$ \mathsf{s}.sig(\mathsf{sk}, x)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, \sigma)\}$
return $\sigma$

VERIFY($x, \sigma$)
───────────────
if $(x, \sigma) \in \mathcal{L}$ :
    return 1

return 0
```

```
1-Gunf-cma^0_s
═══════════════
GETPK()
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
return pk

SIG($x$)
───────────────
assert $\sigma = \bot$
assert $|x| = \lambda$
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
$\sigma \leftarrow\$ \mathsf{s}.sig(\mathsf{sk}, x)$

return $\sigma$

VERIFY($x^*, \sigma^*$)
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
$d \leftarrow \mathsf{s}.ver(\mathsf{pk}, x^*, \sigma^*)$
return $d$
```

```
1-Gunf-cma^1_s
═══════════════
GETPK()
───────────────
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
return pk

SIG($x$)
───────────────
assert $\sigma = \bot$
assert $|x| = \lambda$
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{s}.kgen(1^\lambda)$
$\sigma \leftarrow\$ \mathsf{s}.sig(\mathsf{sk}, x)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, \sigma)\}$
return $\sigma$

VERIFY($x^*, \sigma^*$)
───────────────
if $(x^*, \sigma^*) \in \mathcal{L}$ :
    return 1

return 0
```

Solution 4. Assume towards contradiction that there is a PPT adversary $\mathcal{A}$ breaking the the 1-UNF-CMA security of Lamport's signature scheme s_f with advantage $\mathbf{Adv}_{\mathcal{A}}^{\texttt{1-Gunf-cma}^0_{\mathsf{s}_f},\texttt{1-Gunf-cma}^1_{\mathsf{s}_f}}(\lambda)$. We construct a reduction $\mathcal{R}$ which uses $\mathcal{A}$ and breaks the one-wayness of f with probability $\frac{1}{2\lambda}\mathbf{Adv}_{\mathcal{A}}^{\texttt{1-Gunf-cma}^0_{\mathsf{s}_f},\texttt{1-Gunf-cma}^1_{\mathsf{s}_f}}(\lambda)$ (which in non-negligible). Note that there is a loss factor of 2λ. We'll return to this loss later in the probability analysis. Let us now firts give a conceptual high-level overview over the reduction.

High-level description of the reduction. Conceptually, the reduction $\mathcal{R}$ receives challenge y and embeds it at a *random* position (j, b) with $j \leftarrow\!\$ \{1, .., \lambda\}$, $b \leftarrow\!\$ \{0, 1\}$ into the public-key, for example, as the third position for the 1-bit. $\mathcal{R}$ samples all other elements of the public-key uniformly at random. Then, the public-key is

$$\mathsf{pk} = \begin{pmatrix} y_0^1, & y_0^2, & y_0^3, & y_0^4, .., y_0^\lambda \\ y_1^1, & y_1^2, & y, & y_1^4, .., y_1^\lambda \end{pmatrix}.$$

Now, *if* $\mathcal{A}$ queries message m such that the 3rd bit of x is 0, then $\mathcal{R}$ can actually generate the signature. Else, $\mathcal{R}$ aborts. Such an abort happens with probability $\frac{1}{2}$, as we will analyse below. Now, for every message-signature pair (x^*, σ^*) which $\mathcal{A}$ submits to VERIFY, the reduction $\mathcal{R}$ hopes that (1) the 3rd bit of x^* is 1 (i.e., equal to b) and (2) that the signature σ^* is valid. If so, then the reduction $\mathcal{R}$ parses

$$(z^1, .., z^\lambda) \leftarrow \sigma$$

and returns z^3 which is a valid pre-image of y by definition of $s_f.ver$.

Reduction in pseudocode. We now describe the above reduction in pseudo-code (using random (j, b) and not $(3, 1)$ as in the example above).

```
$\mathcal{R}(1^\lambda, y)$
$j \leftarrow\!\$ \{1, .., \lambda\}, b \leftarrow\!\$ \{0, 1\}$
Run $\mathcal{A}^{\mathsf{GETPK},\mathsf{SIG},\mathsf{VERIFY}}$
until $\mathcal{A}$ terminates or an assert
fails, where oracle queries are
answered as specified by the
oracle code on the right, whose
state is $(y, j, b, \mathsf{sk}, \mathsf{pk}, \sigma, z)$.
return $z$
```

```
$\mathsf{GETPK}()$
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$ \mathsf{sam}(y, j, b, 1^\lambda)$
return $\mathsf{pk}$
```

```
$\mathsf{sam}(y, j, b, 1^\lambda)$
for $i = 1..\lambda$
    $x_0^i \leftarrow\!\$ \{0, 1\}^\lambda$
    $x_1^i \leftarrow\!\$ \{0, 1\}^\lambda$
    $y_0^i \leftarrow f(x_0^i)$
    $y_1^i \leftarrow f(x_1^i)$
    if $i = j$ then $y_b^j \leftarrow y$
$\mathsf{sk} \leftarrow \begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$
$\mathsf{pk} \leftarrow \begin{pmatrix} y_0^1, .., y_0^\lambda \\ y_1^1, .., y_1^\lambda \end{pmatrix}$
return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\mathsf{SIG}(x)$
assert $\sigma = \bot$
assert $|x| = \lambda$
assert $x_i \neq b$
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$ \mathsf{sam}(y, j, b, 1^\lambda)$
$\sigma \leftarrow\!\$ \mathsf{s}_f.sig(\mathsf{sk}, x)$
return $\sigma$
```

```
$\mathsf{VERIFY}(m^*, \sigma^*)$
if $\mathsf{pk} = \bot$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$ \mathsf{sam}(y, j, b, 1^\lambda)$
$d \leftarrow \mathsf{s}_f.ver(\mathsf{pk}, x, \sigma)$
if $m_i^* = b \wedge d = 1$ then
    $(z^1, .., z^\lambda) \leftarrow \sigma^*$
    $z \leftarrow z^3$
return $d$
```

Probability Analysis. First observe that the assert condition **assert** $x_i \neq b$ is triggered with probability $\frac{1}{2}$, because the position (j, b) is information-theoretically hidden from $\mathcal{A}$, and a bit agrees with a random bit with probability $\frac{1}{2}$.

$$\Pr[\textbf{assert } x_i \neq b \text{ triggers}] = \tfrac{1}{2} \tag{1}$$

Next, observe that the reduction $\mathcal{R}$ simulates the VERIFY oracle perfectly, both for the real and the ideal game—until the moment where $\mathcal{A}$ submits a successful *forgery*. That is, $\mathcal{A}$ submits a message-signature pair (x^*, σ^*) such that the real VERIFY oracle returns 1 and the ideal VERIFY oracle returns 0. Since $\mathcal{A}$ can only the real and ideal game when submitting a successful forgery, we have that

$$\Pr[\mathcal{A} \text{ submits successful forgery } (m^*, \sigma^*)] \geq \mathbf{Adv}_{\mathcal{A}}^{\texttt{1-Gunf-cma}^0_{\mathsf{s}_f}, \texttt{1-Gunf-cma}^1_{\mathsf{s}_f}}(\lambda) \tag{2}$$

Now, for the *first* forgery which $\mathcal{A}$ submits, the probability that $m_i^* = b$ is at least $\frac{1}{\lambda}$, because successful forgery means that $m^* \neq m$, because for each message, there is a *unique* valid signature, because f is injective, so if (m^*, σ^*) is valid and $(m^*, \sigma^*) \neq (m, \sigma)$, then $m^* \neq m$. Since m^* and m differ in at least one position and since j is information-theoretically hidden, this position is j with probability $\frac{1}{\lambda}$.

$$\Pr[m_i^* = b \mid \mathcal{A} \text{ submits successful forgery } (m^*, \sigma^*)] = \frac{1}{\lambda} \tag{3}$$

Putting Equations 1, 2 and 3 together, we obtain

$$\begin{aligned}
&\Pr[f(z) = y] \\
=&\Pr[\textbf{assert } x_i \neq b \text{ triggers}] \cdot \Pr[f(z) = y \mid \textbf{assert } x_i \neq b \text{ does not triggers}] \\
\geq&\tfrac{1}{2} \cdot \Pr[\mathcal{A} \text{ submits successful forgery } (m^*, \sigma^*) \wedge m_i^* = b] \\
\geq&\tfrac{1}{2} \cdot \Pr[\mathcal{A} \text{ submits successful forgery } (m^*, \sigma^*)] \\
&\qquad \cdot \Pr[m_i^* = b \mid \mathcal{A} \text{ submits successful forgery } (m^*, \sigma^*)] \\
\geq&\tfrac{1}{2} \cdot \mathbf{Adv}_{\mathcal{A}}^{\texttt{1-Gunf-cma}^0_{\mathsf{s}_f}, \texttt{1-Gunf-cma}^1_{\mathsf{s}_f}}(\lambda) \cdot \tfrac{1}{\lambda},
\end{aligned}$$

which is non-negligible.