A note on the equivalence of IND-CCA & INT-PTXT and IND-CCA & INT-CTXT

Daniel Jost, Christian Badertscher, Fabio Banfi

Department of Computer Science, ETH Zurich, Switzerland
daniel.jost@inf.ethz.ch
christian.badertscher@inf.ethz.ch
fabio.banfi@inf.ethz.ch

Abstract

The security for authenticated encryption schemes is often captured by demanding CCA security (IND-CCA) and integrity of plaintexts (INT-PTXT). In this short note, we prove that this implies in particular integrity of ciphertexts, i.e., INT-CTXT. Hence, the two sets of requirements mentioned in the title are equivalent.

1 The Security Games

We treat the stateful notions in this short note since they are the most widely used notions for authenticated encryption (the main use case is realizing a cryptographic channel). A proof for the non-stateful versions would follow along the same lines. We restate the relevant stateful notions from [BKN04] formally in Figure 1 and Figure 2. A (stateful) authenticated encryption scheme consists of a triple of algorithms $\Psi = (\mathsf{Gen}, \mathcal{E}, \mathcal{D})$ for key generation, encryption, and decryption, respectively, as defined in detail in [BKN04] (including the definitions of correctness and being stateful). The message space is denoted by $\mathcal{M}$ and the ciphertext space is denoted by $\mathcal{C}$. Our notation follows basically the notation of [BN08; BKN04].

IND-sfCCA$_\Psi$

Initialization
- $k \leftarrow \mathsf{Gen}$
- $b \twoheadleftarrow \{0,1\}$ ▷ Sampling u.a.r. from $\{0,1\}$.
- $i \leftarrow 0$
- $j \leftarrow 0$
- $\mathsf{sync} \leftarrow 1$
- $C \leftarrow \emptyset$

Oracle LR

Input: $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$
- $i \leftarrow i + 1$
- $c \leftarrow \mathcal{E}(k, m_b)$
- $C[i] \leftarrow c$
- **return** c

Oracle Dec

Input: $c \in \mathcal{C}$
- $j \leftarrow j + 1$
- $m \leftarrow \mathcal{D}(k, c)$
- **if** $j > i \lor c \neq C[j]$ **then**
  - $\mathsf{sync} \leftarrow 0$
- **if** $\mathsf{sync} = 0$ **then**
  - **return** m
- **else**
  - **return** $\perp$

Finalization

Input: $d \in \{0,1\}$
- **return** $(d = b)$

Figure 1: **IND-sfCCA$_\Psi$** security for stateful CCA security of an authenticated encryption scheme.

INT-sfCTXT$_\Psi$

Initialization
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$C \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$

Oracle Enc
Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, m)$
$C[i] \leftarrow c$
return c

Oracle VF
Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \lor \ c \neq C[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $m \neq \perp \ \land \ \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
return $(m \neq \perp)$

Finalization
return win

INT-sfPTXT$_\Psi$

Initialization
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$S \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$

Oracle Enc
Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, m)$
$S[i] \leftarrow m$
return c

Oracle VF
Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \lor \ m \neq S[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $m \neq \perp \ \land \ \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
return $(m \neq \perp)$

Finalization
return win

Figure 2: The **INT-sfPTXT$_\Psi$** and **INT-sfCTXT$_\Psi$** security games for stateful plaintext- and ciphertext-integrity, respectively, of an authenticated encryption scheme.

2 INT-PTXT & IND-CCA implies INT-CTXT

Let $\mathcal{A}$ denote an INT-sfCTXT attacker. In the following we show that

$$\mathrm{Adv}^{\textbf{IND-sfCTXT}}_{\Psi,\mathcal{A}} \leq \mathrm{Adv}^{\textbf{IND-sfCCA}}_{\Psi,\mathcal{A}_1} + \mathrm{Adv}^{\textbf{IND-sfCCA}}_{\Psi,\mathcal{A}_2} + 3\,\mathrm{Adv}^{\textbf{IND-sfPTXT}}_{\Psi,\mathcal{A}_3}$$

where $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ denote slight modifications of $\mathcal{A}$ with roughly the same efficiency.

As a first hybrid, we consider the game $\mathbf{H}_0$ depicted in Figure 3, that essentially works like **INT-sfCTXT$_\Psi$**, but initially flips a uniform random bit z, and then the encryption oracle instead of encrypting the message m encrypts $z^{|m|}$, i.e., either the all-zero or all-one bit string of the length of m. By definition of the advantage of an adversary in the forgery game, we have

$$\begin{aligned}\mathrm{Adv}^{\textbf{IND-sfCTXT}}_{\Psi,\mathcal{A}} &:= \Pr\big[\mathcal{A}^{\textbf{INT-sfCTXT}_\Psi} \Rightarrow 1\big] \\ &= \Pr\big[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\big] + \Big(\Pr\big[\mathcal{A}^{\textbf{INT-sfCTXT}_\Psi} \Rightarrow 1\big] - \Pr\big[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\big]\Big) \qquad (1)\end{aligned}$$

In the following, we will first upper bound the second term, and then proceed to upper bound $\Pr\big[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\big]$ as a second step.

2.1 Upper bounding the second term

Consider the following bit-guessing game $\mathbf{G}_0$, shown in Figure 4, that initially flips a bit b and then either behaves exactly like **INT-sfCTXT$_\Psi$**, if $b = 0$, or like $\mathbf{H}_0$ if $b = 1$, and the goal of the adversary is to guess b in the end. In addition, the adversary also gets access to an oracle **HasWon** that allows him to query the win flag of **INT-sfCTXT$_\Psi$** or $\mathbf{H}_0$, respectively.

We now define $\mathcal{A}_0$ as follows: $\mathcal{A}_0$ internally runs $\mathcal{A}$ forwarding all queries and responses to the **Enc** and **VF** oracles. Once $\mathcal{A}$ calls **Finalization**, $\mathcal{A}_0$ first queries the **HasWon** oracle, and then calls **Finalization** with $d = 0$ if **HasWon** returned true and $d = 1$ otherwise. Observe that the bit-guessing game $\mathbf{G}_0$ behaves exactly as **INT-sfCTXT$_\Psi$** if $b = 0$ and exactly like $\mathbf{H}_0$ if $b = 1$. By definition of $\mathbf{G}_0$ and $\mathcal{A}_0$ we obtain

$$\begin{aligned}\Pr\Big[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1 \ \Big|\ b = 0\Big] &= \Pr^{\mathcal{A}_0^{\mathbf{G}_0}}[d = b \mid b = 0] = \Pr^{\mathcal{A}_0^{\mathbf{G}_0}}[d = 0 \mid b = 0] \\ &= \Pr^{\mathcal{A}_0^{\mathbf{G}_0}}[\mathsf{win} = 1 \mid b = 0] = \Pr\big[\mathcal{A}^{\textbf{INT-sfCTXT}_\Psi} \Rightarrow 1\big]\end{aligned}$$

$\boxed{\textbf{INT-sfCTXT}_\Psi}$ and $\textbf{H}_0$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$C \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$\boxed{c \leftarrow \mathcal{E}(k, m)}$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $m \neq \perp \land \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
return $(m \neq \perp)$

Finalization

return win

Figure 3: The first hybrid $\mathbf{H}_0$ compared to the original $\textbf{INT-sfCTXT}_\Psi$ security game.

$\boxed{\mathbf{G}_0}$ and $\mathbf{G}_1$

Initialization

$z \twoheadleftarrow \{0,1\}$
$b \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$C \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
if $b = 0$ **then**
  $c \leftarrow \mathcal{E}(k, m)$
else
  $c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $m \neq \perp \land \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
$\boxed{\textbf{return } (m \neq \perp)}$
if $\mathsf{sync} = 0$ **then**
  return $(m \neq \perp)$
else
  return 1.

Oracle HasWon

return win

Finalization

Input: $d \in \{0,1\}$
return $d = b$

Figure 4: The bit-guessing games $\mathbf{G}_0$ and $\mathbf{G}_1$.

$\mathbf{G}_1$ and $\mathbf{G}_2$

Initialization

$z \twoheadleftarrow \{0,1\}$
$b \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$C \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
if $b = 0$ **then**
  $c \leftarrow \mathcal{E}(k, m)$
else
  $c \leftarrow \mathcal{E}(k, z^{|m|})$
$m_0 \leftarrow m$
$m_1 \leftarrow z^{|m|}$
$c \leftarrow \mathcal{E}(k, m_b)$
$C[i] \leftarrow c$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \vee \ c \neq C[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $\mathsf{sync} = 0$ **then**
  $m \leftarrow m$
else
  $m \leftarrow \perp$
if $m \neq \perp \ \wedge \ \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
if $\mathsf{sync} = 0$ **then**
  return $(m \neq \perp)$
else
  return 1.

Oracle HasWon

return win

Finalization

Input: $d \in \{0,1\}$
return $d = b$

Figure 5: The bit-guessing games $\mathbf{G}_2$ in comparison to $\mathbf{G}_1$.

and

$$\Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1 \,\middle|\, b = 1\right] = \Pr^{\mathcal{A}_0^{\mathbf{G}_0}}[d = b \mid b = 1] = \Pr^{\mathcal{A}_0^{\mathbf{G}_0}}[d = 1 \mid b = 1]$$
$$= \Pr^{\mathcal{A}_0^{\mathbf{G}_0}}[\mathsf{win} = 0 \mid b = 1] = 1 - \Pr\left[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\right],$$

which yields

$$\begin{aligned}&\Pr\left[\mathcal{A}^{\textbf{INT-sfCTXT}_\Psi} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\right] \\ &\quad= \Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1 \,\middle|\, b = 0\right] + \Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1 \,\middle|\, b = 1\right] - 1 \\ &\quad= 2\left(\Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1\right] - \frac{1}{2}\right).\end{aligned} \tag{2}$$

We now proceed by bounding $\Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1\right]$ using a sequence of simple modifications:

$\mathbf{G}_1$ The game $\mathbf{G}_1$, as depicted in Figure 4, behaves like $\mathbf{G}_0$ except that the **VF** oracle returns true instead of $(m \neq \perp)$ if $\mathsf{sync} = 1$. Since $\mathsf{sync} = 1$ implies $c = C[j]$ and $\perp \notin \mathcal{M}$, however, by correctness of the scheme this behavior is equivalent.

$\mathbf{G}_2$ Consider the game $\mathbf{G}_2$ as depicted in Figure 5. Observe that in the **Enc** oracle the same message gets encrypted as in $\mathbf{G}_1$. In the **VF** oracle it sets m to $\perp$ if $\mathsf{sync} \neq 0$. Note however, that in this case the value of m does not matter anymore. Thus, $\mathbf{G}_1$ and $\mathbf{G}_2$ behave equivalently.

It is now easy to see that winning $\mathbf{G}_2$ can be reduced to winning $\textbf{IND-sfCCA}_\Psi$ as sketched in Figure 6. For every adversary $\mathcal{A}_0$ against $\mathbf{G}_2$ we can build $\mathcal{A}_1$ against $\textbf{IND-sfCCA}_\Psi$ that works as follows: it initially flips a bit z and then internally runs $\mathcal{A}_0$ and for every query m of the **Enc** oracle it queries the **LR** oracle of $\textbf{IND-sfCCA}_\Psi$ with $m_0 = m$ and $m_1 = z^{|m|}$. In addition, $\mathcal{A}_1$ keeps track whether $\mathcal{A}_0$ is still in sync, so that on a query c to the **VF** oracle by

$\mathbf{G}_2$

Initialization

$z \twoheadleftarrow \{0,1\}$
$b \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$C \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$m_0 \leftarrow m$
$m_1 \leftarrow z^{|m|}$
$c \leftarrow \mathcal{E}(k, m_b)$
$C[i] \leftarrow c$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \vee \ c \neq C[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $\mathsf{sync} = 0$ **then**
  $m \leftarrow m$
else
  $m \leftarrow \perp$
if $m \neq \perp \ \wedge \ \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
if $\mathsf{sync} = 0$ **then**
  return $(m \neq \perp)$
else
  return 1.

Oracle HasWon

return win

Finalization

Input: $d \in \{0,1\}$
return $d = b$

Figure 6: The reduction from $\mathbf{G}_2$ to $\mathbf{IND\text{-}sfCCA}_\Psi$. The lines with the blue shade and the solid border belong to the $\mathbf{IND\text{-}sfCCA}_\Psi$ game, whereas the green shaded ones with the dashed border belong to the reduction. The uncolored lines are for bookkeeping that is replicated in both the $\mathbf{IND\text{-}sfCCA}_\Psi$ game as well as the reduction.

$\mathcal{A}_0$ it can query the decryption oracle on c and then reply correctly to $\mathcal{A}_0$. It is easy to see that $\mathcal{A}_1$ guesses b correctly if and only if $\mathcal{A}_0$ guesses b correctly by simply forwarding the guess.

Thus we obtain

$$\Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_0^{\mathbf{G}_1} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_0^{\mathbf{G}_2} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_1^{\mathbf{IND\text{-}sfCCA}_\Psi} \Rightarrow 1\right],$$

and combining this with (2) yields the desired bound

$$\begin{aligned} \Pr\left[\mathcal{A}^{\mathbf{INT\text{-}sfCTXT}_\Psi} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\right] &= 2\left(\Pr\left[\mathcal{A}_0^{\mathbf{G}_0} \Rightarrow 1\right] - \frac{1}{2}\right) \\ &= 2\left(\Pr\left[\mathcal{A}_1^{\mathbf{IND\text{-}sfCCA}_\Psi} \Rightarrow 1\right] - \frac{1}{2}\right) =: \mathrm{Adv}_{\Psi,\mathcal{A}_1}^{\mathbf{IND\text{-}sfCCA}}, \end{aligned} \tag{3}$$

where in the last step we used the definition of the (bit-guessing) advantage of a CCA adversary.

2.2 Bounding the first winning probability

In the following section we upper bound the probability $\Pr\left[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\right]$ using the hybrids $\mathbf{H}_1$ and $\mathbf{H}_2$, depicted in Figure 7 and Figure 8, respectively.

$\mathbf{H}_1$ The game $\mathbf{H}_1$ replaces the sync and the win flags of $\mathbf{H}_0$ by two pairs of flags $(\mathsf{sync}_1, \mathsf{sync}_2)$ and $(\mathsf{win}_1, \mathsf{win}_2)$, respectively. Note that sync_2 in $\mathbf{H}_1$ is defined exactly equivalent to sync of $\mathbf{H}_0$, and thus $\mathsf{win}_1 \vee \mathsf{win}_2$ is true in $\mathbf{H}_1$ if and only if win is true in $\mathbf{H}_0$. Hence, the two games behave equivalently.

$\mathbf{H}_0$ and $\mathbf{H}_1$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\mathsf{win} \leftarrow 0$
$\mathsf{win}_1 \leftarrow 0$
$\mathsf{win}_2 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync} \leftarrow 0$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync} = 0$ **then**
  $\mathsf{win} \leftarrow 1$
if $m \neq \perp \land \mathsf{sync}_1 = 0 \land \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_1 \leftarrow 1$
if $m \neq \perp \land \mathsf{sync}_1 = 1 \land \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
return $(m \neq \perp)$

Finalization

return win
return $\mathsf{win}_1 \lor \mathsf{win}_2$

Figure 7: The game $\mathbf{H}_1$ is equivalent to $\mathbf{H}_0$, which is best seen by observing that the sync_2 flag is identical to the sync one of $\mathbf{H}_0$.

$\mathbf{H}_1$ and $\mathbf{H}_2$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\mathsf{win}_1 \leftarrow 0$
$\mathsf{win}_2 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync}_1 = 0 \land \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_1 \leftarrow 1$
if $m \neq \perp \land \mathsf{sync}_1 = 0$ **then**
  $\mathsf{win}_1 \leftarrow 1$
if $m \neq \perp \land \mathsf{sync}_1 = 1 \land \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
return $(m \neq \perp)$

Finalization

return $\mathsf{win}_1 \lor \mathsf{win}_2$

Figure 8: The game $\mathbf{H}_2$ is equivalent to $\mathbf{H}_1$ as well. Observe that $\mathsf{sync}_1 = 0$ implies that $j > i$ or $m \neq S[j]$ for some j. In the latter case, the correctness of the scheme however implies that $c[j] \neq C[j]$ and thus $\mathsf{sync}_2 = 0$ as well.

$\boxed{\mathbf{P}_0}$ and $\boxed{\mathbf{C}_0}$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\mathsf{win}_1 \leftarrow 0$
$\mathsf{win}_2 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \vee \ m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \ \vee \ c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \ \wedge \ \mathsf{sync}_1 = 0$ **then**
  $\mathsf{win}_1 \leftarrow 1$
if $m \neq \perp \ \wedge \ \mathsf{sync}_1 = 1 \ \wedge \ \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
return $(m \neq \perp)$

Finalization

$\boxed{\textbf{return } \mathsf{win}_1}$
$\boxed{\textbf{return } \mathsf{win}_2}$

Figure 9: The games $\mathbf{P}_0$ and $\mathbf{C}_0$ are identical to $\mathbf{H}_2$, except that the winning condition $\mathsf{win}_1 \vee \mathsf{win}_2$ of the latter has been replaced by checking only one of the respective flags.

$\mathbf{H}_2$ The game $\mathbf{H}_2$ is equivalent to $\mathbf{H}_1$ except that the former no longer checks for $\mathsf{sync}_2 = 0$ when setting win_1 to true. Observe however that by the correctness of the scheme we have that $m \neq S[j]$ implies $c \neq C[j]$ and thus $\mathsf{sync}_1 = 0$ implies $\mathsf{sync}_2 = 0$. Hence, the two games are equivalent as well.

Now, consider the two games $\mathbf{P}_0$ and $\mathbf{C}_0$ as depicted in Figure 9. Observe that each of those games is equivalent to $\mathbf{H}_2$ except for the winning condition that only checks for win_1 or win_2, respectively, instead of $\mathsf{win}_1 \vee \mathsf{win}_2$. Using the union bound we therefore obtain

$$\Pr\left[\mathcal{A}^{\mathbf{H}_0} \Rightarrow 1\right] = \Pr\left[\mathcal{A}^{\mathbf{H}_1} \Rightarrow 1\right] = \Pr\left[\mathcal{A}^{\mathbf{H}_2} \Rightarrow 1\right] \leq \Pr\left[\mathcal{A}^{\mathbf{P}_0} \Rightarrow 1\right] + \Pr\left[\mathcal{A}^{\mathbf{C}_0} \Rightarrow 1\right]. \qquad (4)$$

We proceed by bounding those two terms separately in the next sections.

2.3 Upper bounding the advantage on $\mathbf{P}_0$

Consider the game $\mathbf{P}_1$ as shown in Figure 10, which basically corresponds to $\mathbf{P}_0$ with all code related to the two unused flags win_2 and sync_2 removed. Moreover, the **Enc**-oracle has slightly been rewritten without changing the behavior. It is now easy to reduce any adversary $\mathcal{A}$ winning $\mathbf{P}_1$ to another adversary $\mathcal{A}_3$ winning $\mathbf{INT\text{-}sfPTXT}_\Psi$, as highlighted in Figure 11: $\mathcal{A}_3$ initially flips a bit z and then whenever $\mathcal{A}$ queries the **Enc** oracle on m, $\mathcal{A}_3$ queries the actual **Enc**-oracle on $m' = z^{|m|}$. Clearly, $\mathcal{A}_3$ wins $\mathbf{INT\text{-}sfPTXT}_\Psi$ if and only if $\mathcal{A}$ wins $\mathbf{P}_1$. As a consequence, we have

$$\Pr\left[\mathcal{A}^{\mathbf{P}_0} \Rightarrow 1\right] = \Pr\left[\mathcal{A}^{\mathbf{P}_1} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_3^{\mathbf{INT\text{-}sfPTXT}_\Psi} \Rightarrow 1\right] = \mathrm{Adv}^{\mathbf{IND\text{-}sfPTXT}}_{\Psi,\mathcal{A}_3}. \qquad (5)$$

2.4 Upper bounding the advantage on $\mathbf{C}_0$

In the following section, we upper bound $\Pr\left[\mathcal{A}^{\mathbf{C}_0} \Rightarrow 1\right]$ using a sequence of hybrids $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$, $\mathbf{C}_4$, $\mathbf{C}_5$, and $\mathbf{C}_6$.

$\boxed{\mathbf{P}_0}$ and $\mathbf{P}_1$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\mathsf{win}_1 \leftarrow 0$
$\mathsf{win}_2 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
$m' \leftarrow z^{|m|}$
$c \leftarrow \mathcal{E}(k, m')$
$S[i] \leftarrow m'$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync}_1 = 0$ **then**
  $\mathsf{win}_1 \leftarrow 1$
if $m \neq \perp \land \mathsf{sync}_1 = 1 \land \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
return $(m \neq \perp)$

Finalization

return win_1

Figure 10: The game $\mathbf{P}_1$ that is equivalent to $\mathbf{P}_0$.

$\mathbf{P}_1$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$S \leftarrow \emptyset$
$\mathsf{win}_1 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$m' \leftarrow z^{|m|}$
$c \leftarrow \mathcal{E}(k, m')$
$S[i] \leftarrow m'$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync}_1 = 0$ **then**
  $\mathsf{win}_1 \leftarrow 1$
return $(m \neq \perp)$

Finalization

return win_1

Figure 11: The reduction from $\mathbf{P}_1$ to $\textbf{INT-sfPTXT}_\Psi$. The lines with the blue shade and the solid border belong to the $\textbf{INT-sfPTXT}_\Psi$ game, whereas the green shaded ones with the dashed border belong to the reduction.

$\boxed{\mathbf{C}_0}$ and $\mathbf{C}_1$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\boxed{\mathsf{win}_1 \leftarrow 0}$
$\mathsf{win}_2 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$

$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$

$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \vee m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \vee c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
$\boxed{\textbf{if } m \neq \bot \wedge \mathsf{sync}_1 = 0 \textbf{ then } \mathsf{win}_1 \leftarrow 1}$
if $m \neq \bot \wedge \mathsf{sync}_1 = 1 \wedge \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
return $(m \neq \bot)$

Finalization

return win_2

Figure 12: The game $\mathbf{C}_1$ that is equivalent to $\mathbf{C}_0$.

$\mathbf{C}_1$ The game $\mathbf{C}_1$, as depicted in Figure 12, corresponds to $\mathbf{C}_0$ with all code related to the unused flag win_1 removed. Hence, the two games behave obviously equivalent.

$\mathbf{C}_2$ The game $\mathbf{C}_2$ corresponds to $\mathbf{C}_1$ with the winning flag win_2 replaced by a variable d guessing z. It is depicted in Figure 13. Note that $\mathsf{sync}_1 = 1$ implies $m = S[j]$, and thus $m = z^\ell$ for some length $\ell > 0$ (we use here that the empty bit-string is not in the message space). Hence, setting d to the first bit of m implies that the game is won, and is thus equivalent to setting the winning flag in $\mathbf{C}_1$.

$\mathbf{C}_3$ The game $\mathbf{C}_3$, as depicted in Figure 14, corresponds to $\mathbf{C}_2$ but with d initialized to 0 instead of $\bot$ giving an adversary a fifty percent chance of winning the game without setting the win_2 flag. This makes $\mathbf{C}_3$ a bit-guessing game. Observe that

$$\Pr^{\mathcal{A}^{\mathbf{C}_3}}[\mathsf{win}_2 = 1] = \Pr^{\mathcal{A}^{\mathbf{C}_2}}[\mathsf{win}_2 = 1] = \Pr\big[\mathcal{A}^{\mathbf{C}_2} \Rightarrow 1\big]$$

and

$$\Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z \wedge \mathsf{win}_2 = 1] = \Pr^{\mathcal{A}^{\mathbf{C}_3}}[\mathsf{win}_2 = 1],$$

yielding

$$\begin{aligned}\Pr\big[\mathcal{A}^{\mathbf{C}_3} \Rightarrow 1\big] &= \Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z] \\ &= \Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z \wedge \mathsf{win}_2 = 1] + \Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z \wedge \mathsf{win}_2 = 0] \\ &= \Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z \wedge \mathsf{win}_2 = 1] + \Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z \mid \mathsf{win}_2 = 0]\Pr^{\mathcal{A}^{\mathbf{C}_3}}[\mathsf{win}_2 = 0] \\ &= \Pr^{\mathcal{A}^{\mathbf{C}_3}}[d = z \wedge \mathsf{win}_2 = 1] + \frac{1}{2}\Big(1 - \Pr^{\mathcal{A}^{\mathbf{C}_3}}[\mathsf{win}_2 = 1]\Big) \\ &= \Pr\big[\mathcal{A}^{\mathbf{C}_2} \Rightarrow 1\big] + \frac{1}{2}\Big(1 - \Pr\big[\mathcal{A}^{\mathbf{C}_2} \Rightarrow 1\big]\Big).\end{aligned}$$

Rewriting the last equation we obtain

$$\Pr\big[\mathcal{A}^{\mathbf{C}_2} \Rightarrow 1\big] = 2\Big(\Pr\big[\mathcal{A}^{\mathbf{C}_3} \Rightarrow 1\big] - \frac{1}{2}\Big). \tag{6}$$

$\boxed{\mathrm{C}_1}$ and $\boxed{\mathrm{C}_2}$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\mathsf{win}_2 \leftarrow 0$
$\boxed{d \leftarrow \perp}$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \vee m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \vee c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \wedge \mathsf{sync}_1 = 1 \wedge \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
  $\boxed{d \leftarrow m(1)}$
return $(m \neq \perp)$

Finalization

$\boxed{\textbf{return } \mathsf{win}_2}$
$\boxed{\textbf{return } (d = z)}$

Figure 13: The game $\mathbf{C}_2$, where $m(1)$ denotes the first bit of m. Note that $\mathsf{sync} = 1$ implies $m = S[j] = z^\ell$ for some $\ell > 0$ (since $\lambda \notin \mathcal{M}$). Hence, we have $\mathsf{win}_2 = 1$ iff $d = z$.

$\boxed{\mathrm{C}_2}$ and $\boxed{\mathrm{C}_3}$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$\mathsf{win}_2 \leftarrow 0$
$\boxed{d \leftarrow \perp}$
$\boxed{d \leftarrow 0}$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \vee m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \vee c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \wedge \mathsf{sync}_1 = 1 \wedge \mathsf{sync}_2 = 0$ **then**
  $\mathsf{win}_2 \leftarrow 1$
  $d \leftarrow m(1)$
return $(m \neq \perp)$

Finalization

return $(d = z)$

Figure 14: The bit-guessing game $\mathbf{C}_3$. Observe that in comparison to $\mathbf{C}_2$, the adversary has a fifty percent chance of winning the game without managing to set the win_2 flag.

$\boxed{\mathbf{C}_3}$ and $\boxed{\mathbf{C}_4}$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$d \leftarrow 0$
$\boxed{\mathsf{bad} \leftarrow 0}$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
$\boxed{\textbf{if } m \neq \bot \land \mathsf{sync}_1 = 1 \land \mathsf{sync}_2 = 0 \textbf{ then } d \leftarrow m(1)}$
if $m \neq \bot \land \mathsf{sync}_2 = 0$ **then**
  if $\mathsf{sync}_1 = 1$ **then**
    $d \leftarrow m(1)$
  else
    $\mathsf{bad} \leftarrow 1$
return $(m \neq \bot)$

Finalization

return $(d = z)$

Figure 15: The game $\mathbf{C}_4$, that introduces the bad flag. It behaves equivalent to $\mathbf{C}_3$, however, since bad is an internal variable only.

$\mathbf{C}_4$ The game $\mathbf{C}_4$, as depicted in Figure 15, corresponds to $\mathbf{C}_3$ with a bad flag introduced. The two games behave obviously equivalent.

$\mathbf{C}_5$ The game $\mathbf{C}_5$ is depicted in Figure 16 and is *identical until bad* to $\mathbf{C}_5$. Hence, by the Fundamental Lemma of game-playing we have

$$\Pr\left[\mathcal{A}^{\mathbf{C}_4} \Rightarrow 1\right] \leq \Pr\left[\mathcal{A}^{\mathbf{C}_5} \Rightarrow 1\right] + \Pr\left[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\right]. \tag{7}$$

We defer bounding the probability of bad being set to the end of the proof and continue bounding $\Pr\left[\mathcal{A}^{\mathbf{C}_5} \Rightarrow 1\right]$.

$\mathbf{C}_6$ The game $\mathbf{C}_6$, as depicted in Figure 17, is a version of $\mathbf{C}_5$ with the internal bad flag removed. This, in addition, allows removing all code related to the sync_1 flag without altering the behavior.

$\mathbf{C}_7$ The game $\mathbf{C}_7$ is depicted in Figure 18. First, compared to $\mathbf{C}_6$ the **Enc**-oracle has slightly been rewritten without modifying the behavior. Then, in the **VF**-oracle, in case of $\mathsf{sync}_2 = 1$ we no longer return $(m \neq \bot)$ but true. Since $\mathsf{sync}_2 = 1$ implies $c = C[j]$, however, we have by correctness that $m \in \mathcal{M}$ and thus $m \neq \bot$. Moreover, if $\mathsf{sync}_2 = 1$, we then reset m to $\bot$ without affecting the behavior. Hence, $\mathbf{C}_7$ and $\mathbf{C}_6$ behave equivalently.

Now, observe that $\mathbf{C}_7$ can be easily reduced to $\textbf{IND-sfCCA}_\Psi$, as shown in Figure 19. For every adversary $\mathcal{A}$ against $\mathbf{C}_7$ we can build $\mathcal{A}_2$ against $\textbf{IND-sfCCA}_\Psi$ that works as follows: it internally runs $\mathcal{A}$ and for every query m of the **Enc** oracle it queries the **LR** oracle of $\textbf{IND-sfCCA}_\Psi$ with $m_0 = 0^{|m|}$ and $m_1 = 1^{|m|}$. In addition, $\mathcal{A}_2$ keeps track whether $\mathcal{A}$ is still in sync, so that on a query c to the **VF** oracle by $\mathcal{A}$ it queries the decryption oracle on c and then replies correctly to $\mathcal{A}$. Moreover, once it detects that $\mathcal{A}$ is out of sync and the ciphertext decrypted to a valid ciphertext, it uses the first bit of the decrypted message as the guess of z. It is now easy to see that

$$\Pr\left[\mathcal{A}^{\mathbf{C}_7} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_2^{\textbf{IND-sfCCA}_\Psi} \Rightarrow 1\right]. \tag{8}$$

C_4 and C_5

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$S \leftarrow \emptyset$
$d \leftarrow 0$
$\mathsf{bad} \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$S[i] \leftarrow z^{|m|}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync}_2 = 0$ **then**
  if $\mathsf{sync}_1 = 1$ **then**
    $d \leftarrow m(1)$
  else
    $\mathsf{bad} \leftarrow 1$
    $\boxed{d \leftarrow m(1)}$
return $(m \neq \perp)$

Finalization

return $(d = z)$

Figure 16: The game $\mathbf{C}_4$ that is *identical until bad* to the game $\mathbf{C}_5$.

$\boxed{\mathsf{C}_5}$ and C_6

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\boxed{\mathsf{sync}_1 \leftarrow 1}$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$\boxed{S \leftarrow \emptyset}$
$d \leftarrow 0$
$\boxed{\mathsf{bad} \leftarrow 0}$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$c \leftarrow \mathcal{E}(k, z^{|m|})$
$C[i] \leftarrow c$
$\boxed{S[i] \leftarrow z^{|m|}}$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $j > i \lor c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync}_2 = 0$ **then**
  if $\mathsf{sync}_1 = 1$ **then**
    $d \leftarrow m(1)$
  else
    $\mathsf{bad} \leftarrow 1$
    $d \leftarrow m(1)$
  $\boxed{d \leftarrow m(1)}$
return $(m \neq \perp)$

Finalization

return $(d = z)$

Figure 17: The game $\mathbf{C}_6$. Since bad is an internal variable only, removing this flag and all then unused code related to setting it does not affect the behavior.

$\boxed{\mathrm{C}_6}$ and C_7

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$d \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$\boxed{c \leftarrow \mathcal{E}(k, z^{|m|})}$
$m_0 \leftarrow 0^{|m|}$
$m_1 \leftarrow 1^{|m|}$
$c \leftarrow \mathcal{E}(k, m_z)$
$C[i] \leftarrow c$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \vee \ c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $\mathsf{sync}_2 = 0$ **then**
  $m \leftarrow m$
else
  $m \leftarrow \perp$
if $m \neq \perp \ \wedge \ \mathsf{sync}_2 = 0$ **then**
  $d \leftarrow m(1)$
$\boxed{\textbf{return } (m \neq \perp)}$
if $\mathsf{sync}_2 = 0$ **then**
  return $(m \neq \perp)$
else
  return 1

Finalization

return $(d = z)$

Figure 18: The game $\mathbf{C}_7$. Observe that in the **VF** oracle, if $\mathsf{sync}_2 = 1$, then we have $c = C[j]$, which by correctness in turn implies that the cyphertext decrypts to the original message that is not equal to $\perp$. Moreover, if $\mathsf{sync}_2 = 1$, then m is unused for the rest of the oracle call.

C_7

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_2 \leftarrow 1$
$C \leftarrow \emptyset$
$d \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$m_0 \leftarrow 0^{|m|}$
$m_1 \leftarrow 1^{|m|}$
$c \leftarrow \mathcal{E}(k, m_z)$
$C[i] \leftarrow c$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \ \vee \ c \neq C[j]$ **then**
  $\mathsf{sync}_2 \leftarrow 0$
if $\mathsf{sync}_2 = 0$ **then**
  $m \leftarrow m$
else
  $m \leftarrow \perp$
if $m \neq \perp \ \wedge \ \mathsf{sync}_2 = 0$ **then**
  $d \leftarrow m(1)$
if $\mathsf{sync}_2 = 0$ **then**
  return $(m \neq \perp)$
else
  return 1

Finalization

return $(d = z)$

Figure 19: The reduction from $\mathbf{C}_7$ to $\mathbf{IND\text{-}sfCCA}_{\Psi}$. The lines with the blue shade and the solid border belong to the $\mathbf{IND\text{-}sfCCA}_{\Psi}$ game, whereas the green shaded ones with the dashed border belong to the reductions. The uncolored lines are for bookkeeping that is replicated in both the $\mathbf{IND\text{-}sfCCA}_{\Psi}$ game as well as the reduction.

Putting all together – especially (6), (7), and (8) – we obtain

$$\begin{aligned}
\Pr\big[\mathcal{A}^{\mathbf{C}_0} \Rightarrow 1\big] &= \Pr\big[\mathcal{A}^{\mathbf{C}_1} \Rightarrow 1\big] \\
&= \Pr\big[\mathcal{A}^{\mathbf{C}_2} \Rightarrow 1\big] \\
&= 2\Big(\Pr\big[\mathcal{A}^{\mathbf{C}_3} \Rightarrow 1\big] - \frac{1}{2}\Big) \\
&= 2\Big(\Pr\big[\mathcal{A}^{\mathbf{C}_4} \Rightarrow 1\big] - \frac{1}{2}\Big) \\
&\leq 2\Big(\Pr\big[\mathcal{A}^{\mathbf{C}_5} \Rightarrow 1\big] + \Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big] - \frac{1}{2}\Big) \\
&= 2\Big(\Pr\big[\mathcal{A}^{\mathbf{C}_6} \Rightarrow 1\big] + \Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big] - \frac{1}{2}\Big) \\
&= 2\Big(\Pr\big[\mathcal{A}^{\mathbf{C}_7} \Rightarrow 1\big] + \Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big] - \frac{1}{2}\Big) \\
&= 2\Big(\Pr\Big[\mathcal{A}_2^{\mathbf{IND\text{-}sfCCA}_\Psi} \Rightarrow 1\Big] + \Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big] - \frac{1}{2}\Big) \\
&= \mathrm{Adv}_{\Psi,\mathcal{A}_2}^{\mathbf{IND\text{-}sfCCA}} + 2\Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big].
\end{aligned} \tag{9}$$

It remains to bound $\Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big]$. To this end, consider the game $\mathbf{B}_0$, depicted in Figure 20, which is identical to $\mathbf{C}_4$ except that the winning condition is no longer win being set, but bad being set. Hence, by definition we have

$$\Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big] = \Pr\big[\mathcal{A}^{\mathbf{B}_0} \Rightarrow 1\big],$$

and moreover, it is easy to see that both $\mathbf{B}_1$ and $\mathbf{B}_2$ behaves equivalently as well, as seen in Figures 20 and 21. Finally, observe that $\mathbf{B}_2$ is almost identical to the game $\mathbf{P}_1$ defined above, as shown in Figure 22. Thus, using (5) we obtain

$$\Pr\big[\mathcal{A}^{\mathbf{C}_4} \text{ sets } \mathsf{bad}\big] = \Pr\Big[\mathcal{A}_3^{\mathbf{INT\text{-}sfPTXT}_\Psi} \Rightarrow 1\Big]. \tag{10}$$

Combining (1), (3), (4), (5), (9), and (10) concludes the proof.

$\boxed{\mathrm{B}_0}$ and B_1 (dashed box)

```
Initialization
  z ←$ {0,1}
  k ← Gen
  i ← 0
  j ← 0
  sync_1 ← 1
  sync_2 ← 1
  C ← ∅
  S ← ∅
  [d ← 0]
  bad ← 0

Oracle Enc
Input: m ∈ M
  i ← i + 1
  c ← E(k, z^|m|)
  C[i] ← c
  S[i] ← z^|m|
  return c

Oracle VF
Input: c ∈ C
  j ← j + 1
  m ← D(k, c)
  if j > i ∨ m ≠ S[j] then
    sync_1 ← 0
  if j > i ∨ c ≠ C[j] then
    sync_2 ← 0
  if m ≠ ⊥ ∧ sync_2 = 0 then
    [if sync_1 = 1 then
       d ← m(1)
     else
       bad ← 1]
    [if sync_1 = 0 then
       bad ← 1]   (dashed)
  return (m ≠ ⊥)

Finalization
  return bad
```

Figure 20: The games $\mathbf{B}_0$ and $\mathbf{B}_1$. The former is identical to $\mathbf{C}_4$ except that in the finalization now the bad flag gets checked. Moreover, $\mathbf{B}_1$ behaves equivalent to $\mathbf{B}_0$, since d is unused.

$\boxed{\mathrm{B}_1}$ and B_2 (dashed box)

```
Initialization
  z ←$ {0,1}
  k ← Gen
  i ← 0
  j ← 0
  sync_1 ← 1
  [sync_2 ← 1
   C ← ∅]
  S ← ∅
  bad ← 0

Oracle Enc
Input: m ∈ M
  i ← i + 1
  c ← E(k, z^|m|)
  [C[i] ← c]
  S[i] ← z^|m|
  return c

Oracle VF
Input: c ∈ C
  j ← j + 1
  m ← D(k, c)
  if j > i ∨ m ≠ S[j] then
    sync_1 ← 0
  [if j > i ∨ c ≠ C[j] then
     sync_2 ← 0]
  [if m ≠ ⊥ ∧ sync_2 = 0 then
     if sync_1 = 0 then
       bad ← 1]
  [if m ≠ ⊥ ∧ sync_1 = 0 then
     bad ← 1]   (dashed)
  return (m ≠ ⊥)

Finalization
  return bad
```

Figure 21: The game $\mathbf{P}_2$. Note that by correctness $\mathsf{sync}_1 = 0$ implies $\mathsf{sync}_2 = 0$, and thus removing the former check does not change the behavior.

$\boxed{\mathbf{B}_2}$ and $\mathbf{P}_1$

Initialization

$z \twoheadleftarrow \{0,1\}$
$k \leftarrow \mathsf{Gen}$
$i \leftarrow 0$
$j \leftarrow 0$
$\mathsf{sync}_1 \leftarrow 1$
$S \leftarrow \emptyset$
$\boxed{\mathsf{bad} \leftarrow 0}$
$\mathsf{win}_1 \leftarrow 0$

Oracle Enc

Input: $m \in \mathcal{M}$
$i \leftarrow i + 1$
$\boxed{c \leftarrow \mathcal{E}(k, z^{|m|})}$
$\boxed{S[i] \leftarrow z^{|m|}}$
$m' \leftarrow z^{|m|}$
$c \leftarrow \mathcal{E}(k, m')$
$S[i] \leftarrow m'$
return c

Oracle VF

Input: $c \in \mathcal{C}$
$j \leftarrow j + 1$
$m \leftarrow \mathcal{D}(k, c)$
if $j > i \lor m \neq S[j]$ **then**
  $\mathsf{sync}_1 \leftarrow 0$
if $m \neq \perp \land \mathsf{sync}_1 = 0$ **then**
  $\boxed{\mathsf{bad} \leftarrow 1}$
  $\mathsf{win}_1 \leftarrow 1$
return $(m \neq \perp)$

Finalization

$\boxed{\textbf{return } \mathsf{bad}}$
return win_1

Figure 22: It is easy to verify that $\mathbf{B}_2$ is equivalent to the game $\mathbf{P}_1$ that has already been defined above.

References

[BKN04] M. Bellare, T. Kohno, and C. Namprempre, “Breaking and provably repairing the SSH authenticated encryption scheme”, *ACM Transactions on Information and System Security*, vol. 7, no. 2, pp. 206–241, 2004. DOI: 10.1145/996943.996945. [Online]. Available: https://eprint.iacr.org/2002/078.pdf.

[BN08] M. Bellare and C. Namprempre, “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm”, *Journal of Cryptology*, vol. 21, no. 4, pp. 469–491, 2008. DOI: 10.1007/s00145-008-9026-x. [Online]. Available: https://cseweb.ucsd.edu/%7B~%7Dmihir/papers/oem.pdf.