Applied Cryptography, FS2023
Mar 30–31, 2023 // Version 1.2

Exercise Sheet 6

Prof. Kenny Paterson, F. Günther
M. Backendal, M. Filić

Discussion Questions (Confidentiality and integrity).

(a) In Exercise Sheet 4, we discussed why IND-CPA security of symmetric encryption does not capture integrity. Why do the notions for UF-CMA security of MACs or INT-CTXT security of symmetric encryption not capture confidentiality?

(b) What is the difference between integrity of plaintexts (INT-PTXT) and integrity of ciphertexts (INT-CTXT) for authenticated encryption? Which of the two do you think is preferable?

Suggested focus. Attempt these problems
before class: Problem 2.
in class: Problems 1 and 4.
in your own time: Problem 3.

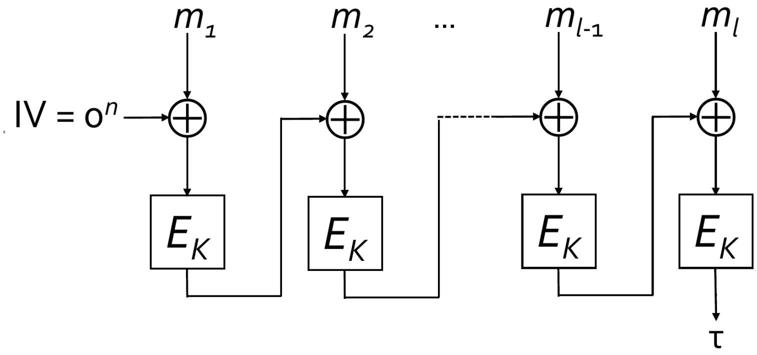
Suggested reading. Reading the following sections in the Boneh-Shoup book [1] might help with the problems on this exercise sheet: Sections 6.2, 6.4, 6.7–6.10 (message integrity), 9.1, and 9.4 (AE).

Problem 1 (CBC-MAC and insecure variants). The CBC mode of operation you saw in the lectures for encryption can also be used to construct a MAC scheme, called CBC-MAC. In this problem we will explore the basic, as well as several insecure, variants of CBC-MAC. In each case a slight change in the specification of CBC-MAC leads to a complete loss of security, meaning it becomes easy to forge tags. Most of these changes look innocent to non-experts, and could end up being implemented. (For further discussion of CBC-MAC and standardized variants, see, e.g., the Boneh-Shoup book [1] Sections 6.4 and 6.7–6.10.)

(a) Basic CBC-MAC. Let $\mathsf{E}\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. The basic CBC-MAC scheme $\mathcal{I}_{\mathsf{basic}} = (\mathrm{KGen}, \mathrm{Tag}, \mathrm{Vfy})$ is defined by KGen sampling a uniformly random key $\mathsf{K} \leftarrow_\$ \{0,1\}^k$ for E and the following Tag algorithm:

```
Tag(K, m)
$m_1 \| \ldots \| m_\ell \leftarrow m$   // s.t. $\forall i\colon |m_i| = n$
$\mathsf{IV} \leftarrow 0^n$ ; $c_0 \leftarrow \mathsf{IV}$
For $i = 1, \ldots, \ell$ do $c_i \leftarrow \mathsf{E}(\mathsf{K}, m_i \oplus c_{i-1})$
Return $c_\ell$
```

Here is a pictorial representation of Tag, which runs CBC-mode encryption with a zero IV, and outputs the final CBC block c_ℓ as the tag τ.



1) $\mathcal{I}_{\mathsf{basic}}$ is an example of a deterministic MAC scheme with a single tag per message. Write down the verification algorithm Vfy for $\mathcal{I}_{\mathsf{basic}}$.

```
Vfy(K, m, τ)
...
```

2) The basic CBC-MAC scheme is provably SUF-CMA secure (under the assumption that the used block cipher is PRP secure) if messages have the same *fixed* length, e.g., if the message space is $\mathcal{M} = \{0,1\}^{n \cdot z}$ for an arbitrary fixed $z \in \mathbb{N}$.

We now show that the scheme is *insecure* if arbitrary-length messages are allowed. Let $\mathcal{M}$ contain all messages whose length is a multiple of n, meaning $\mathcal{M} = (\{0,1\}^n)^*$. Define an adversary $\mathcal{A}_{\mathsf{basic}}$ that breaks the SUF-CMA security of $\mathcal{I}_{\mathsf{basic}}$ using only one tag query.

Hint: Call tag with an arbitrary single-block message as input.

```
Adversary $\mathcal{A}^{\mathsf{tag}}_{\mathsf{basic}}$
...



Return $(m, \tau)$
```

3) Now assume that an appropriate message space is used, meaning $\mathcal{M} = \{0,1\}^{n \cdot z}$ for an arbitrary fixed $z \in \mathbb{N}$. Show that the basic CBC-MAC becomes insecure if the Tag algorithm uses a random IV (as opposed to a fixed string 0^n). Let $\mathcal{I}_{\mathsf{random\text{-}iv}}$ be the resulting MAC scheme with the Tag algorithm defined below. Define an adversary $\mathcal{A}_{\mathsf{random\text{-}iv}}$ that breaks SUF-CMA security of $\mathcal{I}_{\mathsf{random\text{-}iv}}$, again using only one tag query.

```
Tag(K, m)
$m_1 \| \ldots \| m_\ell \leftarrow m$   // s.t. $\forall i\colon |m_i| = n$
$\mathsf{IV} \leftarrow_{\$} \{0,1\}^n$ ; $c_0 \leftarrow \mathsf{IV}$
For $i = 1, \ldots, \ell$ do $c_i \leftarrow \mathsf{E}(\mathsf{K}, m_i \oplus c_{i-1})$
Return $(\mathsf{IV}, c_\ell)$
```

```
Adversary $\mathcal{A}^{\mathsf{tag}}_{\mathsf{random\text{-}iv}}$
...




Return $(m, \tau)$
```

(b) CBC-MAC with prepended message length. In order to allow messages of variable length (in the number of blocks), the basic CBC-MAC can be tweaked to always prepend the message length (in the number of blocks) to the message that needs to be tagged. This results in a MAC scheme $\mathcal{I}_{\mathsf{prepend}}$ that is based on the following Tag algorithm, requiring that $\mathcal{M} = (\{0,1\}^n)^{\leq 2^n}$ contains full-block messages with at most $2^n - 1$ blocks total. This scheme is SUF-CMA. Here $\mathsf{bin}_n : \{0, \ldots, 2^n - 1\} \rightarrow \{0,1\}^n$ is a function that on input a non-negative integer $i \leq 2^n - 1$ returns the binary n-bit representation of i.

```
Tag(K, m)
$m_2 \| \ldots \| m_\ell \leftarrow m$   // s.t. $\forall i$: $|m_i| = n$
$m_1 \leftarrow \mathsf{bin}_n(\ell - 1)$   // prepend length of $m$
$\mathsf{IV} \leftarrow 0^n$ ; $c_0 \leftarrow \mathsf{IV}$
For $i = 1, \ldots, \ell$ do $c_i \leftarrow \mathsf{E}(\mathsf{K}, m_i \oplus c_{i-1})$
Return $c_\ell$
```

1) Informally motivate why the attack on the basic scheme from part (a) 2) doesn't work on $\mathcal{I}_{\mathsf{prepend}}$.

2) Now let $\mathcal{M} = \{0,1\}^*$, such that $\mathcal{I}_{\mathsf{prepend}}$ needs to be modified to pad the messages prior to processing them. Show that 0-padding leads to an attack against SUF-CMA. Let $\mathcal{I}_{\mathsf{zero\text{-}pad}}$ be a MAC scheme based on the Tag algorithm defined below. Define an adversary $\mathcal{A}_{\mathsf{zero\text{-}pad}}$ that breaks the SUF-CMA security of $\mathcal{I}_{\mathsf{zero\text{-}pad}}$, again using only one tag query.

```
Tag(K, m)
$m_2 \| \ldots \| m_\ell \leftarrow m$   // s.t. $\forall i < \ell$: $|m_i| = n, |m_\ell| \leq n$
$m_1 \leftarrow \mathsf{bin}_n(\ell - 1)$   // prepend block length of $m$
$m_\ell \leftarrow m_\ell \| 0^{n-|m_\ell|}$   // pad with 0s
$\mathsf{IV} \leftarrow 0^n$ ; $c_0 \leftarrow \mathsf{IV}$
For $i = 1, \ldots, \ell$ do $c_i \leftarrow \mathsf{E}(\mathsf{K}, m_i \oplus c_{i-1})$
Return $c_\ell$
```

```
Adversary $\mathcal{A}^{\mathsf{tag}}_{\mathsf{zero\text{-}pad}}$
...


Return $(m, \tau)$
```

(c) Output truncation. Another approach to making CBC-MAC secure for variable-length messages is to not output the full last block c_ℓ as tag, but to return a truncated version of it. Briefly and informally, how does this help to prevent the "chaining" attack we saw on the basic CBC-MAC scheme in part (a) 2)?

Problem 2 (Do not use MACs as hash functions). A MAC is a keyed primitive, whereas a hash function is not keyed. While it may be tempting to build hash functions from secure MACs by picking an arbitrary MAC key and using it as a constant, the example below shows that this approach does not guarantee that the resulting hash function will be collision resistant.

The MAC scheme $\mathcal{I}_{\mathsf{prepend}} = (\mathrm{KGen}, \mathrm{Tag}, \mathrm{Vfy})$ from Problem 1 (b) is SUF-CMA. Let $\mathsf{K} \in \{0,1\}^n$ be an arbitrary fixed key for $\mathcal{I}_{\mathsf{prepend}}$. Define a hash function $\mathsf{H}(x) := \mathrm{Tag}(\mathsf{K}, x)$ for all $x \in (\{0,1\}^n)^*$.

Construct an adversary $\mathcal{A}$ that breaks the collision-resistance of H regardless of which key K is used.
Hint: The hash function H is *not* a keyed primitive, so K is a part of the public description of H and known to $\mathcal{A}$.

Problem 3 (MAC combiner). To obtain cryptographic constructions that are more resilient to break-downs of individual primitives, one sometimes combines two different constructions $\mathcal{I}_1, \mathcal{I}_2$ of the same primitive (e.g., MAC) to build a new construction $\mathcal{I}$ in such a way that $\mathcal{I}$ remains secure as long as one of $\mathcal{I}_1, \mathcal{I}_2$ is secure. This is called a "combiner" construction.

In this problem, we define such a construction for a MAC scheme $\mathcal{I}$ from deterministic MAC schemes $\mathcal{I}_1 = (\mathrm{KGen}_1, \mathrm{Tag}_1, \mathrm{Vfy}_1)$ and $\mathcal{I}_2 = (\mathrm{KGen}_2, \mathrm{Tag}_2, \mathrm{Vfy}_2)$. Your assignment is to prove that the construction remains SUF-CMA secure as long as one of the underlying MAC schemes is SUF-CMA secure. For your proof, you should use the SUF-CMA security model with no verification queries. Assume $\mathrm{Vfy}_1, \mathrm{Vfy}_2$ are built using $\mathrm{Tag}_1, \mathrm{Tag}_2$ as follows:

```
$\underline{\mathrm{Vfy}_i(\mathsf{K}, m, \tau)}$
$\tau' \leftarrow \mathrm{Tag}_i(\mathsf{K}, m)$
If $\tau = \tau'$ then return 1 else return 0
```

Let $\mathcal{I} = (\mathrm{KGen}, \mathrm{Tag}, \mathrm{Vfy})$ be a deterministic MAC scheme such that

$$\begin{aligned}
\mathrm{KGen}() &:= (\mathrm{KGen}_1(), \mathrm{KGen}_2())\\
\mathrm{Tag}((k_1, k_2), m) &:= (\mathrm{Tag}_1(k_1, m), \mathrm{Tag}_2(k_2, m))\\
\mathrm{Vfy}((k_1, k_2), m, (\tau_1, \tau_2)) &:= \text{If } \mathrm{Vfy}_1(k_1, m, \tau_1) \wedge \mathrm{Vfy}_2(k_2, m, \tau_2) \text{ then return 1 else return 0.}
\end{aligned}$$

Problem 4 (Secure authenticated encryption). Let $\mathrm{SE} = (\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ be a symmetric encryption scheme that provides authenticated encryption (IND-CPA + INT-CTXT). Let SE be defined for key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Which of the following symmetric encryption schemes provide authenticated encryption? For those that do, give a short proof. For those that do not, present an attack that breaks either the IND-CPA or the INT-CTXT security of the new scheme.

You can assume that $\mathcal{M}$ and $\mathcal{C}$ are efficiently samplable, and that $|\mathcal{M}| > 1$ and $|\mathcal{C}| > 1$ (this means that you can sample distinct elements from either set, if necessary). Each of your adversaries should make at most 1 query to the encryption oracle enc (regardless of whether it attacks IND-CPA or INT-CTXT). Each of the new schemes we construct inherits the key generation algorithm KGen from the initial scheme.

(a) Let $\mathrm{SE}_1 = (\mathrm{KGen}, \mathrm{Enc}_1, \mathrm{Dec}_1)$, where algorithms Enc_1 and Dec_1 are defined as follows:

```
$\underline{\mathrm{Enc}_1(k, m)}$                     $\underline{\mathrm{Dec}_1(k, c)}$
$c_0 \leftarrow\!\$\ \mathrm{Enc}(k, m)$          $(c_0, c_1) \leftarrow c$
$c_1 \leftarrow\!\$\ \mathrm{Enc}(k, m)$          $m_0 \leftarrow \mathrm{Dec}(k, c_0)$
Return $(c_0, c_1)$                $m_1 \leftarrow \mathrm{Dec}(k, c_1)$
                                   If $m_0 = m_1$ then return $m_0$
                                   Else return $\perp$
```

Hint: Note that $\mathrm{Enc}(k, \cdot)$ is randomized; running it twice on the same input will result in different outputs with high probability.

(b) Let $\mathrm{SE}_2 = (\mathrm{KGen}, \mathrm{Enc}_2, \mathrm{Dec}_2)$, where algorithms Enc_2 and Dec_2 are defined as follows:

```
$\underline{\mathrm{Enc}_2(k, m)}$
$c \leftarrow\!\$\ \mathrm{Enc}(k, m)$
Return $(c, c)$
```

```
$\underline{\mathrm{Dec}_2(k, c)}$
$(c_0, c_1) \leftarrow c$
$m_0 \leftarrow \mathrm{Dec}(k, c_0)$
If $c_0 = c_1$ then return $m_0$
Else return $\perp$
```

(c) Let $\mathrm{SE}_3 = (\mathrm{KGen}, \mathrm{Enc}_3, \mathrm{Dec}_3)$, where algorithms Enc_3 and Dec_3 are defined as follows:

```
$\underline{\mathrm{Enc}_3(k, m)}$
$c \leftarrow\!\$\ \mathrm{Enc}(k, m)$
Return $(c, c)$
```

```
$\underline{\mathrm{Dec}_3(k, c)}$
$(c_0, c_1) \leftarrow c$
$m_0 \leftarrow \mathrm{Dec}(k, c_0)$
Return $m_0$
```

(d) Let $\mathsf{H}_1 \colon \mathcal{M} \to \{0,1\}^n$ be a collision-resistant hash function for any $n \in \mathbb{N}$. Let $\mathrm{SE}_4 = (\mathrm{KGen}, \mathrm{Enc}_4, \mathrm{Dec}_4)$, where algorithms Enc_4 and Dec_4 are defined as follows:

```
$\underline{\mathrm{Enc}_4(k, m)}$
$c \leftarrow\!\$\ \mathrm{Enc}(k, m)$
$h \leftarrow \mathsf{H}_1(m)$
Return $(c, h)$
```

```
$\underline{\mathrm{Dec}_4(k, (c, h))}$
$m \leftarrow \mathrm{Dec}(k, c)$
$h' \leftarrow \mathsf{H}_1(m)$
If $h' = h$ then return $m$
Else return $\perp$
```

(e) Let $\mathsf{H}_2 \colon \mathcal{C} \to \{0,1\}^n$ be a collision-resistant hash function for any $n \in \mathbb{N}$. Let $\mathrm{SE}_5 = (\mathrm{KGen}, \mathrm{Enc}_5, \mathrm{Dec}_5)$, where algorithms Enc_5 and Dec_5 are defined as follows:

```
$\underline{\mathrm{Enc}_5(k, m)}$
$c \leftarrow\!\$\ \mathrm{Enc}(k, m)$
$h \leftarrow \mathsf{H}_2(c)$
Return $(c, h)$
```

```
$\underline{\mathrm{Dec}_5(k, (c, h))}$
$m \leftarrow \mathrm{Dec}(k, c)$
$h' \leftarrow \mathsf{H}_2(c)$
If $h' = h$ then return $m$
Else return $\perp$
```

(f) Now answer part (e) again, assuming only that SE is IND-CPA secure (but *not* AE secure).

Acknowledgements. This exercise sheet is in part inspired by (and adapted from) problems by Mihir Bellare and Dan Boneh, as well as the books "A Graduate Course in Applied Cryptography" by Dan Boneh and Victor Shoup, and "Cryptography: Theory and Practice" by Douglas Stinson.

References

[1] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography.* Online, version 0.6 edition, Jan. 2023.