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.

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 $\{1, .., N-1\}$.

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

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

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

$\underline{s_{\text{RSA}}.kgen()}$
[same as $pke_{\text{RSA}}.kgen$]

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

$\underline{s_{\text{RSA}}.ver(pk, m, \sigma)}$
$(e, N) \leftarrow pk$
$m' \leftarrow \sigma^e \mod N$
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

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

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 $pke_{\text{RSA}}.dec(\mathsf{sk}, c)$ 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.)

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 $pke_{\text{RSA}}.enc$ encrypts using the public-key, and the textbook RSA signature schemes $s_{\text{RSA}}.sig$ 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.

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.

$\underline{s_{\mathsf{f}}.kgen(1^\lambda)}$

for $i = 1..\lambda$

$\quad x_0^i \leftarrow\!\!\$\ \{0,1\}^\lambda$

$\quad x_1^i \leftarrow\!\!\$\ \{0,1\}^\lambda$

$\quad y_0^i \leftarrow f(x_0^i)$

$\quad y_0^i \leftarrow f(x_1^i)$

$\mathsf{sk} \leftarrow \begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$

$\mathsf{pk} \leftarrow \begin{pmatrix} y_0^1, .., y_0^\lambda \\ y_1^1, .., y_1^\lambda \end{pmatrix}$

return (sk,pk)

$\underline{s_f.sig(\mathsf{sk}, m)}$

parse sk as $\begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$

for $i = 1..\lambda$

$\quad z^i \leftarrow x^i_{m[i]}$

$\quad$ // m[i] is ith bit of m

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

return σ

$\underline{s_f.ver(\mathsf{pk}, m, \sigma)}$

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 $i = 1..\lambda$

$\quad$ **if** $f(z^i) \neq y^i_{m[i]}$:

$\quad\quad$ **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 $\mathcal{A}$

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

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 $\mathcal{A}$

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

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

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

$\texttt{Gind-cpa}^0_{pke}$

GETPK()
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, pke.kgen(1^\lambda)$
return pk

ENC(x)
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, pke.kgen(1^\lambda)$
$c \leftarrow\!\!\$\, pke.enc(\mathsf{pk}, x)$
return c

$\texttt{Gind-cpa}^1_{pke}$

GETPK()
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, pke.kgen(1^\lambda)$
return pk

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

$\texttt{Gunf-cma}^0_{\mathsf{s}}$

GETPK()
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, \mathsf{s}.kgen(1^\lambda)$
return pk

SIG(x)
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, \mathsf{s}.kgen(1^\lambda)$
$\sigma \leftarrow\!\!\$\, \mathsf{s}.sig(\mathsf{sk}, x)$
return σ

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

$\texttt{Gunf-cma}^1_{\mathsf{s}}$

GETPK()
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, \mathsf{s}.kgen(1^\lambda)$
return pk

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

VERIFY(x, σ)
if $(x, \sigma) \in \mathcal{L}$:
  return 1
return 0

$\texttt{1-Gunf-cma}^0_{\mathsf{s}}$

GETPK()
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, \mathsf{s}.kgen(1^\lambda)$
return pk

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

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

$\texttt{1-Gunf-cma}^1_{\mathsf{s}}$

GETPK()
if $\mathsf{pk} = \perp$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, \mathsf{s}.kgen(1^\lambda)$
return pk

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

VERIFY(x, σ)
if $(x, \sigma) \in \mathcal{L}$:
  return 1
return 0