Index of Security Definitions

One-time secrecy for symmetric-key encryption (Definition 2.6):

$\mathcal{L}^{\Sigma}_{\text{ots-L}}$

QUERY($m_L, m_R \in \Sigma.\mathcal{M}$):
  $k \leftarrow \Sigma.\text{KeyGen}$
  $c \leftarrow \Sigma.\text{Enc}(k, m_L)$
  return c

$\mathcal{L}^{\Sigma}_{\text{ots-R}}$

QUERY($m_L, m_R \in \Sigma.\mathcal{M}$):
  $k \leftarrow \Sigma.\text{KeyGen}$
  $c \leftarrow \Sigma.\text{Enc}(k, m_R)$
  return c

t-out-of-n secret sharing (Definition 3.3):

$\mathcal{L}^{\Sigma}_{\text{tsss-L}}$

QUERY($m_L, m_R \in \Sigma.\mathcal{M}, U$):
  if $|U| \geqslant \Sigma.t$: return err
  $s \leftarrow \Sigma.\text{Share}(m_L)$
  return $\{s_i \mid i \in U\}$

$\mathcal{L}^{\Sigma}_{\text{tsss-R}}$

QUERY($m_L, m_R \in \Sigma.\mathcal{M}, U$):
  if $|U| \geqslant \Sigma.t$: return err
  $s \leftarrow \Sigma.\text{Share}(m_R)$
  return $\{s_i \mid i \in U\}$

Pseudorandom generator (Definition 5.1):

$\mathcal{L}^{G}_{\text{prg-real}}$

QUERY():
  $s \leftarrow \{0,1\}^{\lambda}$
  return $G(s)$

$\mathcal{L}^{G}_{\text{prg-rand}}$

QUERY():
  $z \leftarrow \{0,1\}^{\lambda+\ell}$
  return z

Pseudorandom function (Definition 6.1):

$\mathcal{L}^{F}_{\text{prf-real}}$

$k \leftarrow \{0,1\}^{\lambda}$

QUERY($x \in \{0,1\}^{in}$):
  return $F(k, x)$

$\mathcal{L}^{F}_{\text{prf-rand}}$

$T :=$ empty assoc. array

QUERY($x \in \{0,1\}^{in}$):
  if $T[x]$ undefined:
    $T[x] \leftarrow \{0,1\}^{out}$
  return $T[x]$

Pseudorandom permutation (Definition 7.2):

$\mathcal{L}^F_{\text{prp-real}}$

```
k ← {0,1}^λ

QUERY(x ∈ {0,1}^blen):
  return F(k,x)
```

$\mathcal{L}^F_{\text{prp-rand}}$

```
T := empty assoc. array

QUERY(x ∈ {0,1}^blen):
  if T[x] undefined:
    T[x] ← {0,1}^blen \ range(T)
  return T[x]
```

Strong pseudorandom permutation (Definition 7.6):

$\mathcal{L}^F_{\text{sprp-real}}$

```
k ← {0,1}^λ

QUERY(x ∈ {0,1}^blen):
  return F(k,x)

INVQUERY(y ∈ {0,1}^blen):
  return F^{-1}(k,y)
```

$\mathcal{L}^F_{\text{sprp-real}}$

```
T, T^{-1} := empty assoc. arrays

QUERY(x ∈ {0,1}^blen):
  if T[x] undefined:
    y ← {0,1}^blen \ range(T)
    T[x] := y; T^{-1}[y] := x
  return T[x]

INVQUERY(y ∈ {0,1}^blen):
  if T^{-1}[y] undefined:
    x ← {0,1}^blen \ range(T^{-1})
    T^{-1}[y] := x; T[x] := y
  return T^{-1}[y]
```

CPA security for symmetric-key encryption (Definition 8.1, Section 9.2):

$\mathcal{L}^\Sigma_{\text{cpa-L}}$

```
k ← Σ.KeyGen

CHALLENGE(m_L, m_R ∈ Σ.M):
  if |m_L| ≠ |m_R| return null
  c := Σ.Enc(k, m_L)
  return c
```

$\mathcal{L}^\Sigma_{\text{cpa-R}}$

```
k ← Σ.KeyGen

CHALLENGE(m_L, m_R ∈ Σ.M):
  if |m_L| ≠ |m_R| return null
  c := Σ.Enc(k, m_R)
  return c
```

CPA$ security for symmetric-key encryption (Definition 8.2, Section 9.2):

$\mathcal{L}^\Sigma_{\text{cpa\$-real}}$

```
k ← Σ.KeyGen

CHALLENGE(m ∈ Σ.M):
  c := Σ.Enc(k, m)
  return c
```

$\mathcal{L}^\Sigma_{\text{cpa\$-rand}}$

```
CHALLENGE(m ∈ Σ.M):
  c ← Σ.C(|m|)
  return c
```

CCA security for symmetric-key encryption (Definition 10.1):

$\mathcal{L}^{\Sigma}_{\text{cca-L}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$
$\mathcal{S} := \emptyset$

$\underline{\textsc{challenge}(m_L, m_R \in \Sigma.\mathcal{M})}$:
  if $|m_L| \neq |m_R|$ return null
  $c := \Sigma.\mathsf{Enc}(k, m_L)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

$\underline{\textsc{dec}(c \in \Sigma.\mathcal{C})}$:
  if $c \in \mathcal{S}$ return null
  return $\Sigma.\mathsf{Dec}(k, c)$

$\mathcal{L}^{\Sigma}_{\text{cca-R}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$
$\mathcal{S} := \emptyset$

$\underline{\textsc{challenge}(m_L, m_R \in \Sigma.\mathcal{M})}$:
  if $|m_L| \neq |m_R|$ return null
  $c := \Sigma.\mathsf{Enc}(k, m_R)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

$\underline{\textsc{dec}(c \in \Sigma.\mathcal{C})}$:
  if $c \in \mathcal{S}$ return null
  return $\Sigma.\mathsf{Dec}(k, c)$

CCA$ security for symmetric-key encryption (Definition 10.2):

$\mathcal{L}^{\Sigma}_{\text{cca\$-real}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$
$\mathcal{S} := \emptyset$

$\underline{\textsc{challenge}(m \in \Sigma.\mathcal{M})}$:
  $c := \Sigma.\mathsf{Enc}(k, m)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

$\underline{\textsc{dec}(c \in \Sigma.\mathcal{C})}$:
  if $c \in \mathcal{S}$ return null
  return $\Sigma.\mathsf{Dec}(k, c)$

$\mathcal{L}^{\Sigma}_{\text{cca\$-rand}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$
$\mathcal{S} := \emptyset$

$\underline{\textsc{challenge}(m \in \Sigma.\mathcal{M})}$:
  $c \leftarrow \Sigma.\mathcal{C}(|m|)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

$\underline{\textsc{dec}(c \in \Sigma.\mathcal{C})}$:
  if $c \in \mathcal{S}$ return null
  return $\Sigma.\mathsf{Dec}(k, c)$

MAC (Definition 11.2):

$\mathcal{L}^{\Sigma}_{\text{mac-real}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$

$\underline{\textsc{getmac}(m \in \Sigma.\mathcal{M})}$:
  return $\Sigma.\mathsf{MAC}(k, m)$

$\underline{\textsc{ver}(m \in \Sigma.\mathcal{M}, t)}$:
  return $t \stackrel{?}{=} \Sigma.\mathsf{MAC}(k, m)$

$\mathcal{L}^{\Sigma}_{\text{mac-fake}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$
$\mathcal{T} := \emptyset$

$\underline{\textsc{getmac}(m \in \Sigma.\mathcal{M})}$:
  $t := \Sigma.\mathsf{MAC}(k, m)$
  $\mathcal{T} := \mathcal{T} \cup \{(m, t)\}$
  return t

$\underline{\textsc{ver}(m \in \Sigma.\mathcal{M}, t)}$:
  return $(m, t) \stackrel{?}{\in} \mathcal{T}$

Collision resistance (Definition 12.1):

$\mathcal{L}^{\mathcal{H}}_{\text{cr-real}}$

```
H ← 𝓗

GETH():
  return H

HASH(x ∈ {0,1}*):
  y := H(x)
  return y
```

$\mathcal{L}^{\mathcal{H}}_{\text{cr-fake}}$

```
H ← 𝓗
H⁻¹ := empty assoc. array

GETH():
  return H

HASH(x ∈ {0,1}*):
  y := H(x)
  if H⁻¹[y] defined and H⁻¹[y] ≠ x:
    self destruct
  H⁻¹[y] := x
  return y
```

Key agreement (Definition 14.4):

$\mathcal{L}^{\Sigma}_{\text{ka-real}}$

```
QUERY():
  (t,K) ← EXECPROT(Σ)
  return (t,K)
```

$\mathcal{L}^{\Sigma}_{\text{ka-rand}}$

```
QUERY():
  (t,K) ← EXECPROT(Σ)
  K' ← Σ.𝒦
  return (t,K')
```

Decisional Diffie-Hellman assumption (Definition 14.5):

$\mathcal{L}^{\mathbb{G}}_{\text{dh-real}}$

```
QUERY():
  a,b ← ℤ_n
  return (g^a, g^b, g^ab)
```

$\mathcal{L}^{\mathbb{G}}_{\text{dh-rand}}$

```
QUERY():
  a,b,c ← ℤ_n
  return (g^a, g^b, g^c)
```

CPA security for public-key encryption (Definition 15.1):

$\mathcal{L}^{\Sigma}_{\text{pk-cpa-L}}$

```
(pk,sk) ← Σ.KeyGen

GETPK():
  return pk

CHALLENGE(m_L, m_R ∈ Σ.𝓜):
  return Σ.Enc(pk, m_L)
```

$\mathcal{L}^{\Sigma}_{\text{pk-cpa-R}}$

```
(pk,sk) ← Σ.KeyGen

GETPK():
  return pk

CHALLENGE(m_L, m_R ∈ Σ.𝓜):
  return Σ.Enc(pk, m_R)
```

CPA\$ security for public-key encryption (Definition 15.2):

$\mathcal{L}^{\Sigma}_{\text{pk-cpa\$-real}}$
$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$
<u>GETPK()</u>:
return pk
<u>CHALLENGE($m \in \Sigma.\mathcal{M}$)</u>:
return $\Sigma.\text{Enc}(pk, m)$

$\mathcal{L}^{\Sigma}_{\text{pk-cpa\$-rand}}$
$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$
<u>GETPK()</u>:
return pk
<u>CHALLENGE($m \in \Sigma.\mathcal{M}$)</u>:
$c \leftarrow \Sigma.\mathcal{C}$
return c

One-time secrecy for public-key encryption (Definition 15.4):

$\mathcal{L}^{\Sigma}_{\text{pk-ots-L}}$
$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$
$count := 0$
<u>GETPK()</u>:
return pk
<u>CHALLENGE($m_L, m_R \in \Sigma.\mathcal{M}$)</u>:
$count := count + 1$
if $count > 1$: return null
return $\Sigma.\text{Enc}(pk, m_L)$

$\mathcal{L}^{\Sigma}_{\text{pk-ots-R}}$
$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$
$count := 0$
<u>GETPK()</u>:
return pk
<u>CHALLENGE($m_L, m_R \in \Sigma.\mathcal{M}$)</u>:
$count := count + 1$
if $count > 1$: return null
return $\Sigma.\text{Enc}(pk, m_R)$