Index of Security Definitions

One-time uniform ciphertexts for symmetric-key encryption (Definition 2.7):

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

$\textsc{ctxt}(m \in \Sigma.\mathcal{M})$:
$k \leftarrow \Sigma.\text{KeyGen}$
$c \leftarrow \Sigma.\text{Enc}(k, m)$
return c

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

$\textsc{ctxt}(m \in \Sigma.\mathcal{M})$:
$c \leftarrow \Sigma.C$
return c

One-time secrecy for symmetric-key encryption (Definition 2.8):

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

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

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

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

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

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

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

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

$\textsc{share}(m_L, m_R \in \Sigma.\mathcal{M}, U)$:
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):

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

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

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

$\textsc{query}()$:
$r \leftarrow \{0,1\}^{\lambda+\ell}$
return r

Pseudorandom function (Definition 6.1):

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

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

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

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

$T :=$ empty assoc. array

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

Pseudorandom permutation (Definition 6.6):

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

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

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

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

$T :=$ empty assoc. array

LOOKUP($x \in \{0,1\}^{blen}$):
  if $T[x]$ undefined:
    $T[x] \leftarrow \{0,1\}^{blen} \setminus T$.values
  return $T[x]$

Strong pseudorandom permutation (Definition 6.13):

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

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

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

INVLOOKUP($y \in \{0,1\}^{blen}$):
  return $F^{-1}(k,y)$

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

$T, T_{inv} :=$ empty assoc. arrays

LOOKUP($x \in \{0,1\}^{blen}$):
  if $T[x]$ undefined:
    $y \leftarrow \{0,1\}^{blen} \setminus T$.values
    $T[x] := y$; $T_{inv}[y] := x$
  return $T[x]$

INVLOOKUP($y \in \{0,1\}^{blen}$):
  if $T_{inv}[y]$ undefined:
    $x \leftarrow \{0,1\}^{blen} \setminus T_{inv}$.values
    $T_{inv}[y] := x$; $T[x] := y$
  return $T_{inv}[y]$

CPA security for symmetric-key encryption (Definition 7.1, Section 8.2):

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

$k \leftarrow \Sigma$.KeyGen

EAVESDROP($m_L, m_R \in \Sigma.\mathcal{M}$):
  if $|m_L| \neq |m_R|$ return err
  $c := \Sigma.\text{Enc}(k, m_L)$
  return c

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

$k \leftarrow \Sigma$.KeyGen

EAVESDROP($m_L, m_R \in \Sigma.\mathcal{M}$):
  if $|m_L| \neq |m_R|$ return err
  $c := \Sigma.\text{Enc}(k, m_R)$
  return c

CPA$ security for symmetric-key encryption (Definition 7.2, Section 8.2):

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

$k \leftarrow \Sigma$.KeyGen

CHALLENGE($m \in \Sigma.\mathcal{M}$):
  $c := \Sigma.\text{Enc}(k, m)$
  return c

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

CHALLENGE($m \in \Sigma.\mathcal{M}$):
  $c \leftarrow \Sigma.C(|m|)$
  return c

CCA security for symmetric-key encryption (Definition 9.1):

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

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

EAVESDROP$(m_L, m_R \in \Sigma.\mathcal{M})$:
  if $|m_L| \neq |m_R|$ return err
  $c := \Sigma.\text{Enc}(k, m_L)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

DECRYPT$(c \in \Sigma.\mathcal{C})$:
  if $c \in \mathcal{S}$ return err
  return $\Sigma.\text{Dec}(k, c)$

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

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

EAVESDROP$(m_L, m_R \in \Sigma.\mathcal{M})$:
  if $|m_L| \neq |m_R|$ return err
  $c := \Sigma.\text{Enc}(k, m_R)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

DECRYPT$(c \in \Sigma.\mathcal{C})$:
  if $c \in \mathcal{S}$ return err
  return $\Sigma.\text{Dec}(k, c)$

CCA$ security for symmetric-key encryption (Definition 9.2):

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

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

CTXT$(m \in \Sigma.\mathcal{M})$:
  $c := \Sigma.\text{Enc}(k, m)$
  $\mathcal{S} := \mathcal{S} \cup \{c\}$
  return c

DECRYPT$(c \in \Sigma.\mathcal{C})$:
  if $c \in \mathcal{S}$ return err
  return $\Sigma.\text{Dec}(k, c)$

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

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

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

DECRYPT$(c \in \Sigma.\mathcal{C})$:
  if $c \in \mathcal{S}$ return err
  return $\Sigma.\text{Dec}(k, c)$

MAC (Definition 10.2):

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

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

GETTAG$(m \in \Sigma.\mathcal{M})$:
  return $\Sigma.\text{MAC}(k, m)$

CHECKTAG$(m \in \Sigma.\mathcal{M}, t)$:
  return $t \stackrel{?}{=} \Sigma.\text{MAC}(k, m)$

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

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

GETTAG$(m \in \Sigma.\mathcal{M})$:
  $t := \Sigma.\text{MAC}(k, m)$
  $\mathcal{T} := \mathcal{T} \cup \{(m, t)\}$
  return t

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

Collision resistance (Definition 11.1):

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

$H \leftarrow \mathcal{H}$

GETH():
  return H

HASH($x \in \{0,1\}^*$):
  $y := H(x)$
  return y

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

$H \leftarrow \mathcal{H}$
H^{-1} := empty assoc. array

GETH():
  return H

HASH($x \in \{0,1\}^*$):
  $y := H(x)$
  if $H^{-1}[y]$ defined and $H^{-1}[y] \neq x$:
    self destruct
  $H^{-1}[y] := x$
  return y

Key agreement (Definition 13.4):

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

QUERY():
  $(t, K) \leftarrow$ EXECPROT(Σ)
  return (t, K)

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

QUERY():
  $(t, K) \leftarrow$ EXECPROT(Σ)
  $K' \leftarrow \Sigma.\mathcal{K}$
  return (t, K')

Decisional Diffie-Hellman assumption (Definition 13.5):

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

QUERY():
  $a, b \leftarrow \mathbb{Z}_n$
  return (g^a, g^b, g^{ab})

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

QUERY():
  $a, b, c \leftarrow \mathbb{Z}_n$
  return (g^a, g^b, g^c)

CPA security for public-key encryption (Definition 14.1):

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

$(pk, sk) \leftarrow \Sigma$.KeyGen

GETPK():
  return pk

CHALLENGE($m_L, m_R \in \Sigma.\mathcal{M}$):
  return Σ.Enc(pk, m_L)

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

$(pk, sk) \leftarrow \Sigma$.KeyGen

GETPK():
  return pk

CHALLENGE($m_L, m_R \in \Sigma.\mathcal{M}$):
  return Σ.Enc(pk, m_R)

CPA$ security for public-key encryption (Definition 14.2):

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

$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$

GETPK():
  return pk

CHALLENGE($m \in \Sigma.\mathcal{M}$):
  return $\Sigma.\text{Enc}(pk, m)$

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

$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$

GETPK():
  return pk

CHALLENGE($m \in \Sigma.\mathcal{M}$):
  $c \leftarrow \Sigma.C$
  return c

One-time secrecy for public-key encryption (Definition 14.4):

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

$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$
$count := 0$

GETPK():
  return pk

CHALLENGE($m_L, m_R \in \Sigma.\mathcal{M}$):
  $count := count + 1$
  if $count > 1$: return null
  return $\Sigma.\text{Enc}(pk, m_L)$

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

$(pk, sk) \leftarrow \Sigma.\text{KeyGen}$
$count := 0$

GETPK():
  return pk

CHALLENGE($m_L, m_R \in \Sigma.\mathcal{M}$):
  $count := count + 1$
  if $count > 1$: return null
  return $\Sigma.\text{Enc}(pk, m_R)$