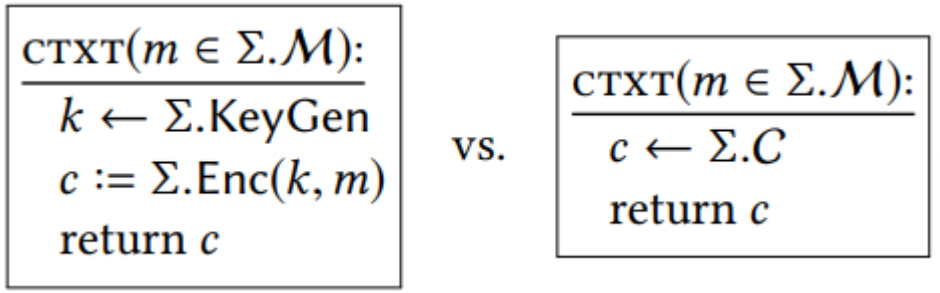
CSE 539: Applied Cryptography Pseudorandom Generator/Function

Ni Trieu (ASU)

Reading: https://joyofcryptography.com/pdf/chap5.pdf

Recap: Provable Security

- "Real-vs-Random" Style of Security Definition



CPA: secure if Adversary chooses plaintext

- Cares about m ---> c direction

CCA: secure if Adversary gets all of Dec(ctxt)

- Cares about c ---> m direction

Pseudorandom Generator

- Suppose Alice & Bob share only a short 1-bit secret k, but they want to encrypt a 2-bit plaintext m using OTP

$$\underline{\text{KeyGen:}}\quad k \leftarrow \{0,1\}^\lambda \quad \text{return } k \qquad \underline{\text{Enc}(k, m \in \{0,1\}^\lambda):}\quad \text{return } k \oplus m \qquad \underline{\text{Dec}(k, c \in \{0,1\}^\lambda):}\quad \text{return } k \oplus c$$

Pseudorandom Generator

- Suppose Alice & Bob share only a short λ-bit secret s, but they want to encrypt a 2λ-bit plaintext m using OTP

$\underline{\text{KeyGen:}}$
$k \leftarrow \{0,1\}^\lambda$
return k

$\underline{\text{Enc}(k, m \in \{0,1\}^\lambda):}$
return $k \oplus m$

$\underline{\text{Dec}(k, c \in \{0,1\}^\lambda):}$
return $k \oplus c$

- Q: Can we transform a short random string into a long string that looks random?

Pseudorandom Generators

- Suppose Alice & Bob share only a short λ-bit secret s, but they want to encrypt a 2λ-bit plaintext m using OTP

KeyGen:	$\text{Enc}(k, m \in \{0,1\}^\lambda)$:	$\text{Dec}(k, c \in \{0,1\}^\lambda)$:
$k \leftarrow \{0,1\}^\lambda$	return $k \oplus m$	return $k \oplus c$
return k		

- Q: Can we transform a short random string into a long string that looks random?

- How to obtain the key k from the shared seed s => PRG

Pseudorandom Generator (PRG)

- Definition: A pseudorandom generator (PRG) is a deterministic function G whose outputs are longer than its inputs. When the input to G is chosen uniformly at random, it induces a certain distribution over the possible output.
- **A PRG is a function** $\mathbf{G}: \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{\lambda}} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{\lambda}+\boldsymbol{\ell}}$

Pseudorandom Generator (PRG)

- **A PRG is a function** $\mathbf{G}: \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{\lambda}} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{\lambda}+\boldsymbol{\ell}}$
- Security:

Pseudorandom Generator (PRG)

- A PRG is a function $\mathbf{G}: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+\ell}$
- Security:

Let $G : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+\ell}$ *be a deterministic function with* $\ell > 0$. *We say that* G *is a* ***secure pseudorandom generator (PRG)*** *if* $\mathcal{L}^{G}_{\text{prg-real}} \approx \mathcal{L}^{G}_{\text{prg-rand}}$, *where:*

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

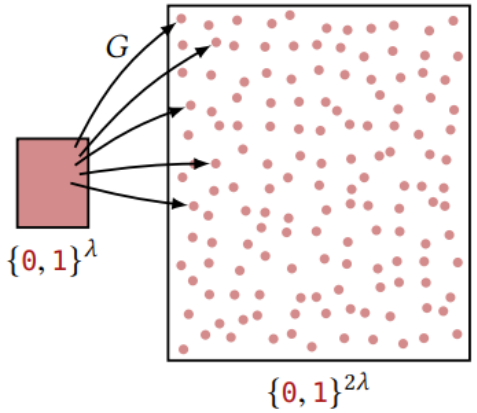
QUERY():
$s \leftarrow \{0,1\}^{\lambda}$
return $G(s)$

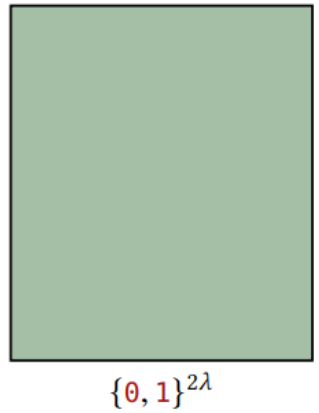
$\mathcal{L}^{G}_{\text{prg-rand}}$

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

Pseudorandom Generator

- If G is a deterministic function, then there are only 2^{λ} possible outputs of G, so the distribution of G(k) cannot be uniform in $\{0,1\}^{\lambda}$





Pseudorandom Generators

- How to build a PRG?

$G(s)$:
return $s \| s$

Pseudorandom Generator

- Quiz Sample: Is the below function a secure PRG?
  - $G(s) = \bar{s}||s$

Pseudorandom Generator

- Quiz Sample: Is the below function a secure PRG?
  - $G(s) = f(s)||f(f(s))$ where f is the secure PRG

Pseudorandom Generator

- How to build a PRG?
  - From block cipher

Pseudorandom Function

- A PRF is a function F: $\{\mathbf{0},\mathbf{1}\}^{\boldsymbol{\lambda}} \times \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{in}} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{out}}$

Pseudorandom Function

Definition 6.1 (PRF security)

Let $F : \{0,1\}^\lambda \times \{0,1\}^{in} \to \{0,1\}^{out}$ *be a deterministic function. We say that* F *is a secure* ***pseudorandom function (PRF)*** *if* $\mathcal{L}^F_{\text{prf-real}} \approx \mathcal{L}^F_{\text{prf-rand}}$, *where:*

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

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

LOOKUP($x \in \{0,1\}^{in}$):

return $F(k, x)$

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

$T :=$ empty assoc. array

LOOKUP($x \in \{0,1\}^{in}$):

if $T[x]$ undefined:

$T[x] \leftarrow \{0,1\}^{out}$

return $T[x]$