CPA-secure encryption from a PRF:

$\Sigma[F]$:

$\mathcal{K} = \{0,1\}^\lambda$
$\mathcal{M} = \{0,1\}^{out}$
$\mathcal{C} = \{0,1\}^\lambda \times \{0,1\}^{out}$

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

Enc(k, m):
- $r \leftarrow \{0,1\}^\lambda$
- $x := F(k, r) \oplus m$
- return (r, x)

Dec$(k, (r, x))$:
- $m := F(k, r) \oplus x$
- return m

Claim:

If F is a secure PRF (with in $= \lambda$) then Σ is a CPA\$-secure encryption scheme. That is, $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$.

Slides for CS427 at Oregon State University. eecs.orst.edu/~rosulekm/crypto. Copyright Mike Rosulek.

Overview:

Want to show:

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$
$k \leftarrow \{0,1\}^\lambda$
$\underline{\text{CHALLENGE}(m):}$
$r \leftarrow \{0,1\}^\lambda$
$x := F(k, r) \oplus m$
return (r, x)

$\approx$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$
$\underline{\text{CHALLENGE}(m):}$
$c \leftarrow \{0,1\}^{\lambda+\text{out}}$
return c

The proof will **use** the fact F is a secure PRF. In other words,

$\mathcal{L}^{F}_{\text{prf-real}}$
$k \leftarrow \{0,1\}^\lambda$
$\underline{\text{QUERY}(r):}$
return $F(k, r)$

$\approx$

$\mathcal{L}^{F}_{\text{prf-rand}}$
$T :=$ empty
$\underline{\text{QUERY}(r):}$
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

Security proof

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

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

$\underline{\text{CHALLENGE}(m)\text{:}}$

$r \leftarrow \{0,1\}^\lambda$
$x := F(k,r) \oplus m$
return (r, x)

Starting point is $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$.

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

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

$\underline{\textsc{challenge}(m):}$

$r \leftarrow \{0,1\}^\lambda$
$x := F(k,r) \oplus m$
return (r, x)

Starting point is $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$. Factor out call to F.

CHALLENGE(m):
$r \leftarrow \{0,1\}^\lambda$
$z := \text{QUERY}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-real}}$
$k \leftarrow \{0,1\}^\lambda$
QUERY(r):
return $F(k, r)$

Starting point is $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$. Factor out call to F.

Security proof

$\textsc{challenge}(m)$:
$r \leftarrow \{0,1\}^\lambda$
$z := \textsc{query}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-real}}$
$k \leftarrow \{0,1\}^\lambda$
$\textsc{query}(r)$:
return $F(k, r)$

Starting point is $\mathcal{L}^\Sigma_{\text{cpa\$-real}}$. Factor out call to F.

$$\begin{array}{|l|}\hline \underline{\textsc{challenge}(m):} \\ \quad r \leftarrow \{0,1\}^\lambda \\ \quad z := \textsc{query}(r) \\ \quad x := z \oplus m \\ \quad \text{return } (r, x) \\ \hline \end{array} \diamond \begin{array}{|l|}\hline \mathcal{L}^F_{\text{prf-rand}} \\ \hline T := \text{empty} \\ \\ \underline{\textsc{query}(r):} \\ \quad \text{if } T[r] \text{ undefined:} \\ \qquad T[r] \leftarrow \{0,1\}^{\text{out}} \\ \quad \text{return } T[r] \\ \hline \end{array}$$

Apply security of F: replace $\mathcal{L}_{\text{prf-real}}$ with $\mathcal{L}_{\text{prf-rand}}$.

$\textsc{challenge}(m)$:
$r \leftarrow \{0,1\}^\lambda$
$z := \textsc{query}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
$T :=$ empty
$\textsc{query}(r)$:
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

Apply security of F: replace $\mathcal{L}_{\text{prf-real}}$ with $\mathcal{L}_{\text{prf-rand}}$. **Are we done?**

CHALLENGE(m):
$r \leftarrow \{0,1\}^\lambda$
$z :=$ QUERY(r)
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
$T :=$ empty
QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

If r happens to repeat (which is possible), one-time pad z is reused!

CHALLENGE(m):
$r \leftarrow \{0,1\}^\lambda$
$z :=$ QUERY(r)
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
$T :=$ empty
QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

Must use fact that r is unlikely to repeat (when chosen this way)

Security proof

CHALLENGE(m):
$r := \textsc{samp}()$
$z := \textsc{query}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
$T :=$ empty
QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

$\diamond$

$\mathcal{L}_{\text{samp-L}}$
SAMP():
$r \leftarrow \{0,1\}^{\lambda}$
return r

Isolate sampling of r.

Security proof

CHALLENGE(m):
$r :=$ SAMP()
$z :=$ QUERY(r)
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
$T :=$ empty
QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

$\diamond$

$\mathcal{L}_{\text{samp-L}}$
SAMP():
$r \leftarrow \{0,1\}^{\lambda}$
return r

Isolate sampling of r.

$\textsc{challenge}(m)$:
$r := \textsc{samp}()$
$z := \textsc{query}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$
$T :=$ empty
$\textsc{query}(r)$:
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
$\textsc{samp}()$:
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Sample r without replacement (change $\mathcal{L}_{\text{samp-L}}$ to $\mathcal{L}_{\text{samp-R}}$).

CHALLENGE(m):
$r :=$ SAMP()
$z :=$ QUERY(r)
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
$T :=$ empty
QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{\text{out}}$
return $T[r]$

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
SAMP():
$r \leftarrow \{0,1\}^\lambda \setminus R$
$R := R \cup \{r\}$
return r

Sample r without replacement (change $\mathcal{L}_{\text{samp-L}}$ to $\mathcal{L}_{\text{samp-R}}$).

CHALLENGE(m):	$\diamond$	$\mathcal{L}^{F}_{\text{prf-rand}}$	$\diamond$	$\mathcal{L}_{\text{samp-R}}$
$r :=$ SAMP() $z :=$ QUERY(r) $x := z \oplus m$ return (r, x)		$T :=$ empty QUERY(r): if $T[r]$ undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return $T[r]$		$R := \emptyset$ SAMP(): $r \leftarrow \{0,1\}^{\lambda} \setminus R$ $R := R \cup \{r\}$ return r

Now r values are **guaranteed** to never repeat.

Security proof

$\text{CHALLENGE}(m)$:	$\diamond$	$\mathcal{L}^{F}_{\text{prf-rand}}$	$\diamond$	$\mathcal{L}_{\text{samp-R}}$
$r := \text{SAMP}()$ $z := \text{QUERY}(r)$ $x := z \oplus m$ return (r, x)		$T :=$ empty $\text{QUERY}(r)$: if $T[r]$ undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return $T[r]$		$R := \emptyset$ $\text{SAMP}()$: $r \leftarrow \{0,1\}^{\lambda} \setminus R$ $R := R \cup \{r\}$ return r

If-statement is always taken.

$\textsc{challenge}(m)$:
- $r := \textsc{samp}()$
- $z := \textsc{query}(r)$
- $x := z \oplus m$
- return (r, x)

$\diamond$

$\textsc{query}(r)$:
- $z \leftarrow \{0,1\}^{\text{out}}$
- return z

$\diamond$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

$\textsc{samp}()$:
- $r \leftarrow \{0,1\}^{\lambda} \setminus R$
- $R := R \cup \{r\}$
- return r

Middle library can therefore be simplified.

CHALLENGE(m):		QUERY(r):		$\mathcal{L}_{\text{samp-R}}$
$r :=$ SAMP()	$\diamond$	$z \leftarrow \{0,1\}^{\text{out}}$	$\diamond$	$R := \emptyset$
$z :=$ QUERY(r)		return z		SAMP():
$x := z \oplus m$				$r \leftarrow \{0,1\}^\lambda \setminus R$
return (r, x)				$R := R \cup \{r\}$
				return r

Middle library can therefore be simplified.

CHALLENGE(m):
$r :=$ SAMP$()$
$z :=$ QUERY(r)
$x := z \oplus m$
return (r, x)

$\diamond$

QUERY(r):
$z \leftarrow \{0,1\}^{\text{out}}$
return z

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
SAMP$()$:
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Inline call to QUERY.

$\textsc{challenge}(m)$:
$r := \textsc{samp}()$
$z \leftarrow \{0,1\}^{\text{out}}$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
$\textsc{samp}()$:
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Inline call to QUERY.

CHALLENGE(m):
$r := \textsc{samp}()$
$z \leftarrow \{0,1\}^{out}$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
SAMP():
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Inline call to QUERY.

CHALLENGE(m):
$r := \textsc{samp}()$
$z \leftarrow \{0,1\}^{\text{out}}$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

SAMP():
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Can apply the "one-time pad rule" (since mask z is uniform each time)

Security proof

$$\begin{array}{|l|}\hline \underline{\textsc{challenge}(m):} \\ \quad r := \textsc{samp}() \\ \quad x \leftarrow \{0,1\}^{out} \\ \quad \text{return } (r, x) \\ \hline \end{array} \diamond \begin{array}{|l|}\hline \mathcal{L}_{\text{samp-R}} \\ \hline R := \emptyset \\ \\ \underline{\textsc{samp}():} \\ \quad r \leftarrow \{0,1\}^\lambda \setminus R \\ \quad R := R \cup \{r\} \\ \quad \text{return } r \\ \hline \end{array}$$

Can apply the "one-time pad rule" (since mask z is uniform each time)

Security proof

CHALLENGE(m):
$r :=$ SAMP()
$x \leftarrow \{0,1\}^{\text{out}}$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
SAMP():
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Can apply the "one-time pad rule" (since mask z is uniform each time)

Security proof

$\textsc{challenge}(m)$:
$r := \textsc{samp}()$
$x \leftarrow \{0,1\}^{out}$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-L}}$
$\textsc{samp}()$:
$r \leftarrow \{0,1\}^{\lambda}$
return r

Replace $\mathcal{L}_{\text{samp-L}}$ with $\mathcal{L}_{\text{samp-R}}$.

CHALLENGE(m):
r := SAMP()
$x \leftarrow \{0,1\}^{out}$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-L}}$
SAMP():
$r \leftarrow \{0,1\}^{\lambda}$
return r

Replace $\mathcal{L}_{\text{samp-L}}$ with $\mathcal{L}_{\text{samp-R}}$.

CHALLENGE(m):
$r :=$ SAMP()
$x \leftarrow \{0,1\}^{\text{out}}$
return (r, x)

$\diamond$

$\mathcal{L}_{\text{samp-L}}$
SAMP():
$r \leftarrow \{0,1\}^{\lambda}$
return r

Inline call to SAMP.

```
CHALLENGE(m):
  r ← {0,1}^λ
  x ← {0,1}^out
  return (r, x)
```

Inline call to SAMP.

$\textsc{challenge}(m)$:
$r \leftarrow \{0,1\}^\lambda$
$x \leftarrow \{0,1\}^{\text{out}}$
return (r, x)

Inline call to $\textsc{samp}$.

Security proof

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$
$\underline{\text{CHALLENGE}(m):}$ $r \leftarrow \{0,1\}^{\lambda}$ $x \leftarrow \{0,1\}^{\text{out}}$ return (r, x)

But every response is chosen uniformly: This is just $\mathcal{L}_{\text{cpa\$-rand}}$.

Summary

We showed:

$$\begin{array}{|l|}\hline \mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \\ \hline k \leftarrow \{0,1\}^{\lambda} \\ \\ \underline{\textsc{challenge}(m):} \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad x := F(k,r) \oplus m \\ \quad \text{return } (r,x) \\ \hline \end{array} \approx \begin{array}{|l|}\hline \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}} \\ \hline \underline{\textsc{challenge}(m):} \\ \quad c \leftarrow \{0,1\}^{\lambda+\text{out}} \\ \quad \text{return } c \\ \hline \end{array}$$

So our scheme is a CPA$-secure encryption scheme when F is a secure PRF.