Cryptography: HW4

Trevor Bramwell

Professor Mike Rosulek
February 5, 2015

1a Consider a variant of the CPA definition for CBC encryption in which an adversary is able to see the next IV before choosing a plaintext.

$k \leftarrow \Sigma.\text{KeyGen}$ $\underline{\text{CHALLENGE}(m_L, m_R)\text{:}}$ return null if $\lvert m_L \rvert \neq \lvert m_R \rvert$ $c_0 c_1 \cdots c_\ell := \Sigma.\textsc{Enc}(k, m_L)$ return $c_0 c_1 \cdots c_\ell$ $\underline{\text{VEC:}}$ return iv_{next}	$k \leftarrow \Sigma.\text{KeyGen}$ $\underline{\text{CHALLENGE}(m_L, m_R)\text{:}}$ return null if $\lvert m_L \rvert \neq \lvert m_R \rvert$ $c_0 c_1 \cdots c_\ell := \Sigma.\textsc{Enc}(k, m_R)$ return $c_0 c_1 \cdots c_\ell$ $\underline{\text{VEC:}}$ return iv_{next}

1b This modification breaks CPA security for CBC encryption. This can be seen by the following distinguisher A.

$\underline{\text{A:}}$
$iv_1 \leftarrow \text{VEC}$
$iv_0 ca_1 \cdots ca_\ell := \text{CHALLENGE}(m_1 m_2 \cdots m_\ell)$
$iv_1 cb_1 \cdots cb_\ell := \text{CHALLENGE}((iv_0 \oplus iv_1 \oplus m_1) m_2 \cdots m_\ell)$
return $ca_{1-\ell} \stackrel{?}{=} cb_{1-\ell}$

Assume k does not change for each call to CHALLENGE. In the first call to CHALLENGE, F_k is called with $(iv_0 \oplus m_1)$. In the second call to

CHALLENGE, we know the iv ahead of time. This allows us to replace it with the previously used iv_0.

$$iv_0 \oplus m_1 = iv_1 \oplus (iv_0 \oplus iv_1 \oplus m_1)$$

This results in the same ciphertext being output twice. The bias for A is then:

$$\begin{aligned}
\text{bias}(A, \mathcal{L}^{CBC}_{\textsf{cpa-real}}, \mathcal{L}^{CBC}_{\textsf{cpa-rand}}) &= |\Pr[A \diamond \mathcal{L}^{CBC}_{\textsf{cpa-real}} \text{ outputs } 1] - \Pr[A \diamond \mathcal{L}^{CBC}_{\textsf{cpa-rand}} \text{ outputs } 1]| \\
&= |1 - \frac{1}{2^n}| \\
&= \text{Non-negligible}
\end{aligned}$$

2 Consider a variant of CPA\$ security in which the adversary is allowed to *choose* but not *re-use* IVs. We know "standard" CBC encryption (with a random IV) is CPA\$-secure.

(a)

$k \leftarrow \{0,1\}^{2n}$
$iv \leftarrow$ Adversary Choice

<u>CHALLENGE($m_1 \cdots m_\ell$):</u>
  $iv' \leftarrow F(k_{left}, iv)$
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

(b)

$k_{left} \leftarrow \{0,1\}^n$
$k_{right} \leftarrow \{0,1\}^n$
$iv \leftarrow$Adversary Choice

$\underline{\text{CHALLENGE}(m_1 \cdots m_\ell):}$
  $iv' \leftarrow F(k_{left}, iv)$
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

(c)

$k_{right} \leftarrow \{0,1\}^n$
$iv \leftarrow$Adversary Choice

$\underline{\text{CHALLENGE}(m_1 \cdots m_\ell):}$
  $iv' \leftarrow \text{QUERY}(iv)$
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

$k_{left} \leftarrow \{0,1\}^n$

$\underline{\text{QUERY}(iv):}$
  return $F(k_{left}, iv)$

(d)

$k_{right} \leftarrow \{0,1\}^n$
$iv \leftarrow$Adversary Choice

$\underline{\text{CHALLENGE}(m_1 \cdots m_\ell):}$
  $iv' \leftarrow \text{QUERY}(iv)$
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

$T := \textit{empty}$

$\underline{\text{QUERY}(iv):}$
  if $T[iv]$ undefined:
    $T[iv] \leftarrow \{0,1\}^n$
  return $T[iv]$

(e)

$k_{right} \leftarrow \{0,1\}^n$
$iv \leftarrow$Adversary Choice

CHALLENGE($m_1 \cdots m_\ell$):
  $iv' \leftarrow$ QUERY(iv)
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

$T := empty$

QUERY(iv):
  if $T[iv]$ undefined:
    $T[iv] :=$ SAMP()
  return $T[iv]$

$\diamond$

SAMP():
  $s \leftarrow \{0,1\}^n$
  return s

(f)

$k_{right} \leftarrow \{0,1\}^n$
$iv \leftarrow$Adversary Choice

CHALLENGE($m_1 \cdots m_\ell$):
  $iv' \leftarrow$ QUERY(iv)
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

$T := empty$

QUERY(iv):
  if $T[iv]$ undefined:
    $T[iv] :=$ SAMP()
  return $T[iv]$

$\diamond$

$S := \emptyset$

SAMP():
  $s \leftarrow \{0,1\}^n \setminus S$
  $S := S \cup \{s\}$
  return s

(g)

$k_{right} \leftarrow \{0,1\}^n$
$iv \leftarrow$Adversary Choice

CHALLENGE($m_1 \cdots m_\ell$):
  $iv' \leftarrow$ QUERY(iv)
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

QUERY(iv):
  $t :=$ SAMP()
  return t

$\diamond$

$S := \emptyset$

SAMP():
  $s \leftarrow \{0,1\}^n \setminus S$
  $S := S \cup \{s\}$
  return s

(h)

$k_{right} \leftarrow \{0,1\}^n$

CHALLENGE($m_1 \cdots m_\ell$):
  $iv' :=$ SAMP()
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

$S := \emptyset$

SAMP():
  $s \leftarrow \{0,1\}^n \setminus S$
  $S := S \cup \{s\}$
  return s

(i)

$k_{right} \leftarrow \{0,1\}^n$

CHALLENGE($m_1 \cdots m_\ell$):
  $iv' :=$ SAMP()
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

$\diamond$

SAMP():
  $s \leftarrow \{0,1\}^n$
  return s

(j)

$k_{right} \leftarrow \{0,1\}^n$

CHALLENGE($m_1 \cdots m_\ell$):
  $iv' \leftarrow \{0,1\}^n$
  $c :=$ CBC encryption of $m_1 \cdots m_\ell$
    under key k_{right}
    using initialization vector iv'
  return $c_1 \cdots c_\ell$

The CPA$ security relies on iv's being pseudorandom. Since each iv' is a result from the call to the PRF, the outputs will only be pseudorandom if the inputs to the PRF are distinct. Because this variant does not allow for ivs to be repeated, they are guaranteed to be distinct.

Justification of steps:

- (a) This is the inlining of the encryption scheme to the CPA$ security variant.
- (a)⇒(b) k is split into it's two halves. No affect on the program output distribution.
- (b)⇒(c) The PRF call is factored out into it's own function along with k_{left}. No affect on the program.
- (c)⇒(d) $\mathcal{L}^F_{\textsf{prf-real}}$ replaced with $\mathcal{L}^F_{\textsf{prf-rand}}$. Program output distribution changes by a negligible amount.
- (d)⇒(e) Factoring out the sampling of random blocks. No effect on program.
- (e)⇒(f) Sampling without replacement instead of with replacement. Negligible effect to output distribution.
- (g) Because iv are not allowed to be reused, QUERY is only called with a non-defined $T[iv]$. Removing the if-statement does not affect the output distribution.
- (h) iv is now never used, so it is removed from the definition. QUERY merely calls SAMP, so this redundancy is removed.
- (h)⇒(i) Sampling with replacement instead of sample without replacement. Negligible effect on program's output distribution.
- (i)⇒(j) Call to SAMP inlined. No effect on program.

Program (j) now has pseudorandom ciphertexts under chosen plaintext and *chosen-but-never-repeated-IV* attacks.

3 A distinguisher showing Σ^2 does not have CCA security is as follows:

$$\boxed{\begin{array}{l} \underline{\text{A:}} \\ \quad c_1, c_2 = \text{CHALLENGE}(m_L, m_R) \\ \quad m = \text{DEC}(k, (c_2, c_1)) \\ \quad \text{return } m_{right} || m_{left} \stackrel{?}{=} m_L \end{array}}$$

By switching c_1 and c_2 when calling DEC, the output will be reversed.
The bias of A is as follows:

$$\begin{aligned} \text{bias}(A, \mathcal{L}^{\Sigma^2}_{\textsf{cca-real}}, \mathcal{L}^{\Sigma^2}_{\textsf{cca-rand}}) &= |\Pr[A \diamond \mathcal{L}^{\Sigma^2}_{\textsf{cca-real}} \text{ outputs } 1] - \Pr[A \diamond \mathcal{L}^{\Sigma^2}_{\textsf{cca-rand}} \text{ outputs } 1]| \\ &= |1 - \frac{1}{2^{2n}}| \\ &= \text{Non-negligible} \end{aligned}$$