Problem Set 5

Alex Miller

November 11, 2021

Abstract

Collaborators: Elizabeth Coble, Lucy Li

1

1.1

(a) **There exists such an adversary:**

Algorithm 1: $A(\Pi_1)$

```
m_0 ← {0}^256
m_1 ← {1}^255||0 // This is equivalent to 2^128 − 1||2^128 − 2
c ← O(m_0, m_1)
c[0]||c[1]||c[2]||c[3] ← c
if c[1] ≠ c[2] then
  | return 0
end
return 1
```

A is an efficient adversary; it submits one query. It had a high advantage.

Consider an encryption of $m_0 = m_0[1]||m_0[2] = \{0\}^{128}||\{0\}^{128}$ versus one of $m_1 = m_1[1]||m_2[2] = 2^{128} - 1||2^{128} - 2$, given a random seed r and a key k. Let $pad(m)$ denote the bit string concatenated by Enc_1's padding function to some input string m:

$c_0 = Enc_1(k, m_0, r)$
$c_0 = c_0[0]||c_0[1]||c_0[2]||c_0[3]$
$c_0 = r||AES(k, r + 1 + m_0[1])||AES(k, r + 2 + m_0[2])||AES(k, r + 3 + pad(m_0))$
$c_0 = r||AES(k, r + 1 + 0)||AES(k, r + 2 + 0)||AES(k, r + 3 + pad(m_0))$
$c_0 = r||AES(k, r + 1)||AES(k, r + 2)||AES(k, r + 3 + pad(m_0))$

$c_1 = Enc_1(k, m_1, r)$
$c_1 = c_1[0]||c_1[1]||c_1[2]||c_1[3]$
$c_1 = r||AES(k, r + 1 + m_0[1])||AES(k, r + 2 + m_0[2])||AES(k, r + 3 + pad(m_1))$
$c_1 = r||AES(k, r + 1 + 2^{128} - 1)||AES(k, r + 2 + 2^{128} - 2)||AES(k, r + 3 + pad(m_1))$
$c_1 = r||AES(k, r)||AES(k, r)||AES(k, r + 3 + pad(m_1))$

Under $AES_{k,r}$ $c_0[1] \neq c_0[2]$ always holds and $c_1[1] = c_1[2]$ always holds. Our adversary queries the messages m_0, m_1 and tests the response from the oracle in the manner described above. Therefore A will return $\hat{b} = 0$ if O encrypted m_0 and $\hat{b} = 1$ if it encrypted m_1.

Therefore $Pr[Expt^{cpa}_{\Pi_1}(A) = 1] = 1$, so

$Adv^{cpa}_{\Pi_1}(A) = |Pr[Expt^{cpa}_{\Pi_1}(A) = 1] - \frac{1}{2}| = \frac{1}{2}$

(b) **There exists such an adversary:**

We've shown there's an efficient adversary A such that $Adv^{cpa}_{\Pi_1}(A)$ is high and A issues exactly one query. Consider an Adversary A' s.t. A' works like A but queries the oracle O some trivial request that it doesn't need to decide what to output. Such an adversary A' would be an efficient adversary that issues more than one oracle query s.t. $Adv_{\Pi_1}^{cpa}(A')$ is high.

(c) **There exists such an adversary:**

We've shown there's an efficient adversary A such that $Adv_{\Pi_1}^{cpa}(A)$ is high and A issues more than one query. Consider an Adversary A' s.t. A' works like A, even though A' also has access to a decryption oracle. Such an adversary A' would be an efficient adversary that issues more than one oracle query s.t. $Adv_{\Pi_1}^{cca}(A')$ is high.

1.2

(a) **there exists no such feasible adversary A:**

The resulting ciphertext from an invocation of of Enc_2 is too opaque to break in of itself; effectively, the cipher uses a perfectly secret one time pad generated from random input. The weakness of the encryption scheme lies in the fact that the pad generated by any random seed value r is recoverable by feeding Dec_2 a constructed cipher text $c = r||\{0\}^*$ (ignoring padding for now), and reading the resulting message, which would contain the pad. This would allow an adversary to decrypt any message m encrypted by given k and r under $Enc_2(k, m, r)$. However that only becomes possible if an Adversary has access to a Decryption oracle, which it does not have under CPA security analysis. Therefore there should not exist an Adversary s.t. A is efficient and $Adv_{\Pi_2}^{cpa}(A)$ is high.

(b) **there exists no such feasible adversary A:**

Refer to answer for part (a)

(c) **there exists such an adversary A:**

Algorithm 2: $A(\Pi_2)$

$m_0 \leftarrow \{0\}^{256}$
$m_1 \leftarrow \{1\}^{256}$
$c \leftarrow LR_{k,b}(m_0, m_1)$
$c[0]||c[1]||c[2]||c[3] \leftarrow c$
$m \leftarrow Dec_k(c[0]||m_0||c[3])$
$m[1]||m[2] \leftarrow m$
$m_0[1]||m_0[2] \leftarrow m_0$
if $c[1] \oplus m[1] == m_0[1]$ **then**
  return 0
end
return 1

A is an efficient adversary that issues more than one oracle query. It has a high advantage:

Consider the encryption of some block aligned message $m \in \{0,1\}^{256} = m[1]||m[2]$, given a random seed r and a key k. Let $pad(m)$ denote the bit string concatenated by Enc_2's padding function to some input string m:

If the $LR_{k,b}$ oracle encrypts m, it will return a cipher text c s.t.:

$c = Enc_2(k, m, r)$
$c = c[0]||c[1]||c[2]||c[3]$
$c = r||AES(k, r) \oplus m[1]||c[3]||AES(k, r) \oplus pad(m)$

Say we were to query our Dec_k oracle the fake cipher text $c[0]||0^{256}||c[3]$. It would return an m' s.t.:

$m' = m'[1]||m'[2] = AES(k, r) \oplus 0^{128}||m'[2] = AES(k, r)||m'[2]$

This is true by the nature of our query, but also because we provided the oracle appropriate encrypted padding in the form of $c[3]$; m was block aligned, and so is m'. Moreover $c[3]$ was encrypted by the same random see used to decrypt our forged cipher text into m'. So when $c[3]$ is decrypted it is also a valid pad for m'.

We observe that value contained in $m'[1]$ is that of the pad used to encrypt $m[1]$. Therefore we can say that $c[1] \oplus m'[1] = m[1]$

This relationship will hold for any such cipher text m 128 bits or longer (so long as m is block aligned) since it effectively recovers the random block $AES(k, r)$ used to pad the first 128 bits of m.

Therefore, between two block aligned message texts of equal length m_0, m_1, such that the first block of either message text is distinct from the other, if $LR_{k,b}(m_0, m_1)$ encrypted m_0, then for the resulting cipher text c and our previous construction of m', the following will be true:

$c[1] \oplus m'[1] = m_0[1]$

If that does not hold then that means that the oracle encrypted m_1 instead.

Our adversary A constructs two such messages m_0, m_1 and tests them in the manner described above. Therefore A will return $\hat{b} = 0$ if $LR_{k,b}$ encrypted m_0 and $\hat{b} = 1$ if it encrypted m_1.

Therefore $Pr[Expt^{cca}_{\Pi_2}(A) = 1] = 1$, so

$Adv^{cca}_{\Pi_2}(A) = |Pr[Expt^{cca}_{\Pi_2}(A) = 1] - \frac{1}{2}| = \frac{1}{2}$

1.3

(a) There exists such an adversary:

Algorithm 3: $A(\Pi_3)$

$m_0 \leftarrow \{0\}^{256}$
$m_1 \leftarrow \{1\}^{256}$
$c \leftarrow O(m_0, m_1)$
$c[1]||c[2] \leftarrow c$
if $c[1] \neq c[2]$ **then**
  return 0
end
return 1

A is an efficient adversary; it submits one query. It has a high advantage:

Consider an encryption of $m_0 = \{0\}^{256} = m_0[1]||m_0[2]$ versus one of $m_1 = \{1\}^{256} = m_1[1]||m_2[2]$, under some given values of k and r. Let $pad(m)$ denote the bit string concatenated by Enc_3's padding function to some input string m:

$c_0 = Enc_3(k, m_0, r)$
$c_0 = c_0[1]||c_0[2]||c_0[3]$
$c_0 = AES(k, \{1\}^{128} \oplus m_0[1])||AES(k, m_0[1] \oplus m_0[2])||AES(k, m_0[2] \oplus pad(m_0))$
$c_0 = AES(k, \{1\}^{128} \oplus \{0\}^{128})||AES(k, \{0\}^{128} \oplus \{0\}^{128})||AES(k, m_0[2] \oplus pad(m_0))$
$c_0 = AES(k, \{1\}^{128})||AES(k, \{0\}^{128})||AES(k, m_0[2] \oplus pad(m_0))$

$c_1 = Enc_3(k, m_1, r)$
$c_1 = c_1[1]||c_1[2]||c_1[3]$
$c_1 = AES(k, \{1\}^{128} \oplus m_1[1])||AES(k, m_1[1] \oplus m_1[2])||AES(k, m_1[2] \oplus pad(m_1))$
$c_1 = AES(k, \{1\}^{128} \oplus \{1\}^{128})||AES(k, \{1\}^{128} \oplus \{1\}^{128})||AES(k, m_1[2] \oplus pad(m_1))$
$c_1 = AES(k, \{0\}^{128}||AES(k, \{0\}^{128}||AES(k, m_1[2] \oplus pad(m_1))$

Under $AES_{k,r}$ $c_0[1] \neq c_0[2]$ always holds and $c_1[1] = c_1[2]$ always holds. Our adversary A queries the messages m_0, m_1 and tests the response from the oracle in the manner described above. Therefore A will return $\hat{b} = 0$ if O encrypted m_0 and $\hat{b} = 1$ if it encrypted m_1.
Therefore $Pr[Expt^{cpa}_{\Pi_3}(A) = 1] = 1$

$Adv^{cpa}_{\Pi_3}(A) = |Pr[Expt^{cpa}_{\Pi_3}(A) = 1] - \frac{1}{2}| = \frac{1}{2}$

(b) We've shown there's an efficient adversary A such that $Adv^{cpa}_{\Pi_3}(A)$ is high and A issues exactly one query. Consider an

Adversary A' s.t. A' works like A but queries the oracle O some trivial request that it doesn't need to decide what to output. Such an adversary A' would be an efficient adversary that issues more than one oracle query s.t. $Adv^{cpa}_{\Pi_3}(A')$ is high.

(c) We've shown there's an efficient adversary A such that $Adv^{cpa}_{\Pi_3}(A)$ is high and A issues more than one query. Consider an Adversary A' s.t. A' works like A, even though A' also has access to a decryption oracle. Such an adversary A' would be an efficient adversary that issues more than one oracle query s.t. $Adv^{cca}_{\Pi_3}(A')$ is high.

1.4

(a) **there exists no such feasible adversary A:**

The resulting ciphertext from an invocation of of Enc_4 is too opaque to break in of itself; effectively, the cipher uses a perfectly secret one time pad generated from random input. The weakness of the encryption scheme lies in the fact that the pad generated by any random seed value r is recoverable by feeding Dec_4 a constructed cipher text $c = c'||d = r||\{0\}^*||d$, s.t. d is a good Mac for c', and reading the resulting message, which would contain the pad. However that only becomes possible if an Adversary has access to a Decryption oracle, which it does not have under CPA security analysis. Therefore there should not exist an Adversary s.t. A is efficient and $Adv^{cpa}_{\Pi_4}(A)$ is high.

(b) **there exists no such feasible adversary A:**

Refer to answer for part (a)

(c) **there exists no such adversary A:** As stated above, an efficient adversary A should be able to recover the pad produced by some random seed value r by feeding Dec_4 a constructed cipher text $c = c'||d = r||\{0\}^*||d$, s.t. d is a good Mac for c', and reading the resulting message, which would contain the pad. This is allowed under CCA security, however we cannot forge a d s.t. d is a good Mac for c' because a well-implemented CBC-MAC is unforgeable – any CCA attack that tried to forge a Mac d without knowledge of k' would only yield error messages from the Dec_4 oracle, revealing nothing about the pad used to encrypt any given message. Therefore, there are no additional vulnerabilities entailed with an adversary having access to Dec_4 oracle. Since there are also no lesser CPA based adversaries, there should not exist an Adversary s.t. A is efficient and $Adv^{cca}_{\Pi_4}(A)$ is high.

2

(a) Consider an Adversary A with oracle access to Mac_k and $Vrfy_k$:

Algorithm 4: $A(Mac)$

```
m_0 ← a // Where a is ∈ {0,1}^256, or some other multiple of blocks
m_1 ← a||b // Where b is ∈ {0,1}^64, one block
m_2 ← c // Where c is ∈ {0,1}^256
t_0 ← Mac_k(m_0)
t_1 ← Mac_k(m_1
t_2 ← Mac_k(m_2
Vrfy_k(t_0 ⊕ t_1 ⊕ t_2, m_2||b)
return
```

A is an efficient adversary; it submits a total of four queries. It has a high advantage.

Consider two distinct messages of $t < 2^{64} - 1$ 64-bit blocks, a, c. Consider a third message comprised of one 64-bit block b. Now consider the Macs generated on the messages $m_0 = a, m_1 = a||b$,and $m_2 = c$:

$Mac(k, m_0) = t_0$, $Mac(k, m_1) = t_1$, and $Mac(k, m_2) = t_2$.

We claim that $Mac(k, m_2||b) = t_0 \oplus t_1 \oplus t_2$. This follows from observing that, since m_1 is just m_0 concatenated with one more block b, $t_1 = t_0 \oplus AES(k, (t+1) \oplus b)$. Moreover $x = AES(k, (t+1) \oplus b) = t_0 \oplus t_1$; this value is equivalent to that generated by Mac_k for a block that contains b at position $t+1$ in an input string, in order to be used as a pad. Therefore we can generate a new for a novel string $c||b$ and know that $Mac(k, c||b) = Mac(k, c) \oplus x$, since c is the same length as a.

Since our adversary A submits equivalent queries to a, b, and c, and crafts a novel message $\hat{m} = c||b$ not previously queried to the Mac_k oracle with a valid Mac $\hat{t} = t_0 \oplus t_1 \oplus t_2$, it always succeeds in its experiment. Therefore:

$Adv^{uf}_{Mac}(A) = Pr[Expt^{uf}_{Mac}(A) = 1] = 1.$

(b) Consider an Adversary A with oracle access to Mac_k and $Vrfy_k$:

Algorithm 5: $A(Mac)$

$a \leftarrow \$(\{0,1\}^{128})^l, 0 < l < 2^{128} - 3$ *//pick a random message of l blocks, bounding t to prevent overflow*
$b \leftarrow \$(\{0,1\}^{64})^l$ *//pick a random message of l blocks, s.t. $a \neq b$*

$m_0 \leftarrow a$
$t_0 \leftarrow Mac_k(m_0)$
$m'_0 \leftarrow a||\langle l\rangle_{128}||t_0$
$t'_0 \leftarrow Mac_k(m'_0)$

$m_1 \leftarrow b$
$t_1 \leftarrow Mac_k(m_1)$
$m'_1 \leftarrow b||\langle l\rangle_{128}||t_1$

$Vrfy_k(t'_0, m'_1)$
return

A is an efficient adversary; it submits a total of four queries. It has a high advantage:

Consider any two messages of l 128-bit blocks $a, b \in \cup_{t=1}^{2^{128}-3}(\{0,1\}^{128})^t, a \neq b$. We claim that:

$t'_a = Mac_k(a||\langle l\rangle_{128}||Mac_k(a)) = Mac_k(b||\langle l\rangle_{128}||Mac_k(b)) = t'_b$

First observe that $Mac_k(a) = AES_k(x \oplus \langle l\rangle_{128}) = t_a$, where x is the last intermediate value of $Mac_k(a)$, and $Mac_k(b) = AES_k(y \oplus \langle l\rangle_{128}) = t_a$, where y is the last intermediate value of $Mac_k(b)$

Notice that:

$t'_a = Mac_k(a||\langle l\rangle_{128}||Mac_k(a))$
$t'_a = Mac_k(a||\langle l\rangle_{128}||t_a)$
$t'_a = AES_k(AES_k(AES_k(x \oplus \langle l\rangle_{128}) \oplus t_a) \oplus \langle l+2\rangle_{128})$
$t'_a = AES_k(AES_k(t_a \oplus t_a) \oplus \langle l+2\rangle_{128})$
$t'_a = AES_k(AES_k(0) \oplus \langle l+2\rangle_{128})$

and

$t'_b = Mac_k(b||\langle l\rangle_{128}||Mac_k(b))$
$t'_b = Mac_k(b||\langle l\rangle_{128}||t_b)$
$t'_b = AES_k(AES_k(AES_k(y \oplus \langle l\rangle_{128}) \oplus t_b) \oplus \langle l+2\rangle_{128})$
$t'_b = AES_k(AES_k(t_b \oplus t_b) \oplus \langle l+2\rangle_{128})$
$t'_b = AES_k(AES_k(0) \oplus \langle l+2\rangle_{128})$

Therefore $t'_a = t'_b$; furthermore, we can use this relationship to forge tags for novel strings: if we had the values of $a, b, t_a = Mac_k(a), t_b = Mac_k(b)$, and $t'_a = Mac_k(a||\langle l\rangle_{128}||t_a)$, we could forge that correct tag t'_b for the novel message $b||\langle l\rangle_{128}||t_b$ since $t'_a = t'_b$ and we already have t'_a.

Since our adversary A submits Mac queries such that it builds and verifies a tag and message pair $\hat{t}, \hat{m}$ s.t. $\hat{t} = t'_a$ and $\hat{m} = b||\langle l\rangle_{128}||t_b$, and since our adversary never queries $\hat{m}$ to the Mac_k oracle, it always succeeds in its experiment. Therefore:

$Adv^{uf}_{Mac}(A) = Pr[Expt^{uf}_{Mac}(A) = 1] = 1.$

(c) Consider an Adversary A with oracle access to Mac_k and $Vrfy_k$:

Algorithm 6: $A(Mac)$

$m[1] \leftarrow \{0,1\}^{128}$
$t \leftarrow Mac_k(m[1]||m[1])$
$t[1]||t[2] \leftarrow t$
$Vfry_k(t[2]||t[2], t[1]||m[1])$
return

A is an efficient adversary; it submits a total of two queries. It has a high advantage.

Consider any message $m = m[1]||m[2] \in \{0,1\}^{256}$ s.t. $m[1] = m[2]$. Observe that for some random key k, the result of $Mac_k(m) = t = t[1]||t[2] = AES_k(m[1])||AES_k(AES_k(m[2]))$ can be used to construct a valid Mac for some novel message m'. We set $m' = m'[1]||m'[2] = t[1]||m[2]$ and observe that:

$Mac_k(m') = t' = AES_k(t[1])||AES_k(AES_K(m[2]))$. Accounting for the fact that $m[1] = m[2]$,
$t' = AES_k(t[1])||AES_k(AES_K(m[1]))$, and the value of $t[1]$,
$t' = AES_k(AES_K(m[1]))||AES_k(AES_K(m[1]))$, which is equivalent to
$t' = t[2]||t[2]$

Since our adversary A submits a Mac query with a message m s.t. $m[1] = m[2]$, and crafts a novel message $\hat{m} = t[1]||m[1]$ not previously queried to the Mac_k oracle with a valid Mac $\hat{t} = t[2]||t[2]$, it always succeeds in its experiment. Therefore:

$Adv^{uf}_{Mac}(A) = Pr[Expt^{uf}_{Mac}(A) = 1] = 1.$

3

Consider an Adversary A with oracle access to Mac_k and $Vrfy_k$:

Algorithm 7: $A(Mac)$

$m \leftarrow \$M$ // *Pick a random message from our message space M to make a 'tag' for*
$t \leftarrow \$T$ // *Pick a random value in the tag space T*
$Vrfy_k(t, m)$
return

A issues as few queries as possible (one query). Assuming Mac has uniform output across the tag space $T = \{0,1\}^t$, it has $Adv^{uf}_{Mac}(A) = Pr[Expt^{uf}_{Mac}(A) = 1] = 2^{-t}$ since it draws a tag at random for a random message and the probability of drawing the correct tag t for m at random is $\frac{1}{|T|} = \frac{1}{2^t}$.