Technische Universität München (I7)
Dr. M. Luttenberger / S. Jaax

Winter term 2017/2018
2019/01/05

Solution

Cryptography – Homework 3

Discussed on Wednesday, 28th of November, 2018.

Exercise 3.1

Let F be a PRP.

(a) Show that F-rCBC is not CCA-secure.

(b) Show that F-CBC-CIV (with chained IV—see lecture slides) is *not* CPA-secure.

Solution:

(a) A possible CCA-attack:

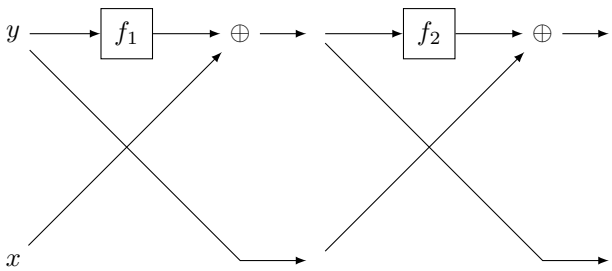
- Generate two messages $m_0 = 0^n$ and $m_1 = 1^n$ and send them to Alice/Bob.
- Receive a ciphertext $c = c^{(0)}||c^{(1)}$ (where $c^{(0)} = IV$) and compute $\tilde{c} = \overline{c^{(0)}}||c^{(1)}$ (just negate the first n bits).
- Since $\tilde{c} \neq c$ we are allowed to query the decryption oracle Dec_k on $\tilde{c}$ and this yields $\mathsf{Dec}_k(\tilde{c}) = m_b \oplus IV \oplus \overline{IV} = m_b \oplus 1^n = \overline{m_b}$. Hence, $\overline{\mathsf{Dec}_k(\tilde{c})} = m_b$.
- If $m_b = 1^n$ output "1" else output "0".

This attack is able to recover m_b and hence succeeds with probability 1.

(b) We describe a CPA-attack:

- Before crafting our two messages we "set up" the Enc_k Algorithm with two queries (i.e. we force it into some state where we can predict the next IV it will use)
- First query Enc_k on 0^n and get back $\mathsf{Enc}_k(0^n) = IV||\underbrace{F_k(IV \oplus 0^n)}_{=IV'} = IV||\underbrace{F_k(IV)}_{=IV'}$
- Then query Enc_k on IV' and receive $\mathsf{Enc}_k(IV') = IV'||\underbrace{F_k(IV' \oplus IV')}_{=IV''} = IV'||\underbrace{F_k(0^n)}_{=IV''}$
- Now we build our two messages: $m_0 = IV''$ and $m_1 = \overline{IV''}$ and send them to Alice& Bob.
- Encrypting m_0 now yields $IV''||F_k(IV'' \oplus m_0) = IV''||F_k(0^n) = IV''||IV''$, while encrypting m_1 yields $IV''||F_k(1^n)$.
- As F is a PRP we trivially have $F_k(1^n) \neq F_k(0^n)$, so we can win with probability 1 by outputting $r = 0$ iff $c = IV''||IV''$.

Exercise 3.2



Let $f : \{0,1\}^* \to \{0,1\}^*$ be some function s.t. $|f(x)| = |x|$ for all $x \in \{0,1\}^*$. A *single-round Feistel network* FN_f is defined by

$$\mathsf{FN}_f(x||y) := y||x \oplus f(y) \text{ for all } x, y \in \{0,1\}^* \text{ with } |x| = |y| .$$

Similarly, given functions $f_1, \ldots, f_j$ a *j-round Feistel network* is inductively defined by

$$\mathsf{FN}_{f_1,f_2,\ldots,f_j}(x||y) := \mathsf{FN}_{f_j}(\mathsf{FN}_{f_1,f_2\ldots,f_{j-1}}(x||y))$$

(a) Show that independent of the choice of $f_1, \ldots, f_j$ the function $\mathsf{FN}_{f_1,\ldots,f_j}$ is invertible if $f_1, \ldots, f_j$ are known.

(b) Let F be a PRF of key and block length n and $P_{k_1,k_2}(x||y) := \mathrm{FN}_{F_{k_1},F_{k_2}}(x||y)$ be a two-round Feistel network using F.

i) Compute $P_{k_1,k_2}(0^n||y)$ and $P_{k_1,k_2}(F_{k_1}(0^n) \oplus z||0^n)$.

ii) Show that PPT-Eve can compute $P^{-1}_{k_1,k_2}$ when given oracle access to P_{k_1,k_2}.

(c) Is $\mathrm{FN}_{F_{k_1},F_{k_2},F_{k_3}}$ with three independent keys $k_1, k_2, k_3 \xleftarrow{u} \{0,1\}^n$ a PRP? Is it a PRF? (y/n)

Solution:

(a) Set $(x', y') := \mathsf{FN}_f(x, y) = (y, x \oplus f(y))$.

Note: $x' = y$, so we immediately know y and thus can compute $f(y) = f(x')$

Hence: $x = y' \oplus f(x') = (x \oplus f(y)) \oplus f(y)$

In other words: $(y, x) = \mathsf{FN}_f(y', x')$

Let $\mathsf{swap}(x, y) := (y, x)$. Then:

$$\mathsf{FN}_f^{-1}(x', y') = \mathsf{swap} \circ \mathsf{FN}_f \circ \mathsf{swap}(x', y')$$

As $\mathsf{swap} \circ \mathsf{swap} = \mathsf{id}$:

$$\begin{aligned} & \mathsf{FN}^{-1}_{f_1,f_2,\ldots,f_j}(x, y) \\ = \; & (\mathsf{FN}_{f_j} \circ \ldots \circ \mathsf{FN}_{f_2} \circ \mathsf{FN}_{f_1})^{-1}(x, y) \\ = \; & \mathsf{FN}^{-1}_{f_1} \circ \mathsf{FN}^{-1}_{f_2} \circ \ldots \circ \mathsf{FN}^{-1}_{f_j}(x, y) \\ = \; & \mathsf{swap} \circ \mathsf{FN}_{f_1} \circ \mathsf{FN}_{f_2} \circ \ldots \circ \mathsf{FN}_{f_j} \circ \mathsf{swap}(x, y) \\ = \; & \mathsf{swap} \circ \mathsf{FN}_{f_j,\ldots,f_2,f_1} \circ \mathsf{swap}(x, y) \end{aligned}$$

(b) See the slides for an illustration:

Result of first round: $(y, F_{k_1}(y) \oplus x)$.

Result of second round: $(F_{k_1}(y) \oplus x, F_{k_2}(F_{k_1}(y) \oplus x) \oplus y)$.

i) $P_{k_1,k_2}(0^n, y) = (F_{k_1}(y), F_{k_2}(F_{k_1}(y)) \oplus y)$.

$P_{k_1,k_2}(F_{k_1}(0^n) \oplus z, 0^n) = (z, F_{k_2}(z))$.

ii) By the preceding result, Eve can compute F_{k_1}, F_{k_2} by quering her oracle at most twice. Any Feistel network can be efficiently inverted if the round functions can be efficiently computed.

(Note that Eve is not given access to k so the important observation is that she can trick the oracle into supplying the required information.)

(c) i) Yes (see the result regarding FNs in the slides).

ii) Yes (see the result that any PRP is also a PRF).

Exercise 3.3 MAC or no MAC?

(a) Does rOFB mode yield a secure MAC?

(b) Show that if the IV in the CBC-MAC-Algorithm is not fixed (but chosen randomly and pre-pended to the CBC-output), the MAC becomes insecure.

Solution:

(a) No: Query 0^n and receive $c^{(0)}||c^{(1)}$, then compute $\mathtt{tag} = c^{(0)}||\overline{c^{(1)}}$ which is a tag for the message 1^n.

(b) Query a tag for $m_0 = 0^1$ which is padded to $\lfloor 1 \rceil||0^n$, and receive $t_0 = \rho||t$. We will now forge a tag for the message $m_1 = 0^2$: First we compute the "difference" $\delta = \lfloor 1 \rceil \oplus \lfloor 2 \rceil$. Then we add this difference to ρ to get $\rho' = \rho \oplus \delta$ and produce a valid tag for m_1 by $t_1 = \rho'||t$.

Exercise 3.4 **MACs using hash-functions done wrong**

Before NMAC and HMAC, several ad-hoc solutions for constructing MACs were used. For instance, given a (hash) function $H\colon \{0,1\}^* \to \{0,1\}^l$, the tag was defined to be $\mathsf{Mac}_k(m) := H(k||m)$, i.e. the outer encryption used in NMAC and HMAC is missing.

Assume a PRF F with (for simplicity) $n = l_{\text{in}}(n) = l_{\text{out}}(n)$. Using the padding function $\mathsf{pad}(m) := m||10^p||\lfloor|m|\rceil$, set $\mathsf{Mac}_k(m) := H(k||m) := F_k^*(\mathsf{pad}(m))$ for $k \in \{0,1\}^n$.

Show that $\mathsf{Mac}_k(m)$ is not secure.

Hint: Recall that the outer encryption used by NMAC and HMAC is to restrict the adversary to prefix-free queries.

Solution: We first query a tag for 0^{n-1} which is padded to $0^{n-1}1\lfloor n-1\rceil$ and we receive $k_2 = F_{k_1}(\lfloor n-1\rceil)$ where $k_1 = F_k(0^{n-1}1)$. Now we can forge a tag for the new message $0^{n-1}1\lfloor n-1\rceil$ (which will be padded to $0^{n-1}1\lfloor n-1\rceil 10^{n-1}\lfloor 2n\rceil$) simply by computing $k_3 = F_{k_2}(10^{n-1})$ (we know k_2!!) and then $\texttt{tag} = F_{k_3}(\lfloor 2n\rceil)$. It is important to remember that F_k is easily computable if we know the key k and that the *insecure* "cascading F" construction without the "sealing-off" outer encryption can leak the intermediate keys and therefore allows us to continue the computation.

Remark: The construction can be made secure by using a prefix-free padding (see lecture slides for a reference that the cascading construction is secure if no prefix-queries are allowed).

However, for practical purposes it is often inefficient to scan the whole message first to get its length and prepend it in the padding and therefore the above MD-padding is used instead in combination with outer encryption (NMAC/HMAC).

Exercise 3.5

Let F be some secure block cipher with key and block length n (think of AES-128).

Consider the following deterministic MAC:

- Gen: as usual, in input 1^n, output $k \xleftarrow{u} \{0,1\}^n$.
- Mac: given $m \in \{0,1\}^+$ and k,

  first pad m to a multiple of n by appending a minimial number of 0,

  then break the padded message into n-bit blocks $m^{(i)}$.

  Starting with $k^{(0)} := 0^n$, compute $k^{(i)} = F_{k^{(i-1)}}(m^{(i)})$ for i from 1 to d where $d = \frac{|m|}{n}$.

  Finally, output $t := F_{k^{(n)}}(k)$.

  (Draw a picture! Note that the key is appended in this case.)
- Vrf: given m, t, and k, check that $\mathsf{Mac}_k(m) = t$.

Is this MAC secure?

Solution: This MAC is not secure, as the padding function is not injective, (e.g. 0 and 00 will - independently of the key - obtain the same tag).

Moreover, the attacker has the possibility to restore the key:

Query 0^n and receive $t_0 = F_{F_{0^n}(0^n)}(k)$. We can simply compute $F_{0^n}(0^n)$ and thus obtain $F^{-1}_{F_{0^n}(0^n)}(t_0) = k$.

Note that $F_k(x)$ is of course efficiently computable if we know the key k!