Advanced Cryptography — Final Exam

Serge Vaudenay

22.6.2017

- duration: 3h
- any document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **not** answer any technical question during the exam
- readability and style of writing will be part of the grade

1 Breaking AES Reduced to 4 Rounds

In this exercise, we consider a block cipher $\mathsf{AES4}$ which is a reduced version of AES. The block cipher $\mathsf{AES4}$ takes as input a key K and a plaintext block X and returns a ciphertext block Y. The key K consists of a sequence of five blocks $K_0, K_1, \ldots, K_4$. A block (like X, Y, or the K_r) is a 4×4 matrix of bytes. (A byte is a bitstring of length 8, i.e. an element of $\{0,1\}^8$.) We let $K_{r,i,j}$ be the byte at position (i,j) in K_r, $0 \leq i,j \leq 3$. The bitwise exclusive OR of blocks (bitwise, component-wise) is denoted with $\oplus$. We define $\mathsf{AES4}$ as follows:

$\mathsf{AES4}(K,X)$:

$S \leftarrow X \oplus K_0$
for $r = 1$ to 4 **do**
  $S \leftarrow \mathsf{SubBytes}(S)$
  $S \leftarrow \mathsf{ShiftRows}(S)$
  $S \leftarrow \mathsf{MixColumns}(S)$
  $S \leftarrow S \oplus K_r$
end for
return S

The $V = \mathsf{SubBytes}(U)$ function is defined by $V_{i,j} = S(U_{i,j})$ for all (i,j), where S is a bijective operation on the set of bytes which is defined by a table. The $V = \mathsf{ShiftRows}(U)$ function is defined by $V_{i,j} = U_{i,j-i \bmod 4}$ for all (i,j). The $V = \mathsf{MixColumns}(U)$ function is defined by $V_{\cdot,j} = \mathcal{L}(U_{\cdot,j})$ for all j, where $U_{\cdot,j}$ denotes the vector formed by the j-th column of U, and $\mathcal{L}$ is an invertible linear transform on the set of vectors of four bytes. (It is linear in the sense of the $\oplus$ operation.) All these functions are known by the adversary. Only K is unknown. We want to construct an adversary who will do a key recovery attack with chosen plaintexts or known plaintexts. We denote $N = 256$.

Given a block B, let $\mathcal{S}_B$ be the set of all blocks X such that $X_{i,j} = B_{i,j}$ for all (i,j) such that $i + j \bmod 4 \neq 0$. (So, only $X_{0,0}, X_{1,3}, X_{2,2}, X_{3,1}$ change.)

We also define the set $\mathcal{Z}$ of all blocks X such that $X_{i,j} = 0$ for all (i,j) such that $(i,j) \neq (0,0)$.

Q.1 Given B, we pick $X, X' \in \mathcal{S}_B$ at random. We denote by Z_r resp. Z'_r the state of encryption after round r. I.e., $Z_0 = X \oplus K_0$, $Z'_0 = X' \oplus K_0$, and

$$Z_r = \mathsf{MixColumns}(\mathsf{ShiftRows}(\mathsf{SubBytes}(Z_{r-1}))) \oplus K_r$$
$$Z'_r = \mathsf{MixColumns}(\mathsf{ShiftRows}(\mathsf{SubBytes}(Z'_{r-1}))) \oplus K_r$$

for $r = 1, \ldots, 4$. What is the probability that $Z_1 \oplus Z'_1 \in \mathcal{Z}$? We let E denote this event in what follows.

Q.2 If E occurs, what does $Z_2 \oplus Z'_2$ look like?

Q.3 The set of column vectors is a vector space of dimension 32 when considered over $\mathbf{Z}_2$, and dimension 4, when considered over $\mathsf{GF}(N)$. Define four linear subspaces $\mathcal{L}_j$ of dimension 8 (over $\mathbf{Z}_2$), or 1 (over $\mathsf{GF}(N)$) such that if E occurs, then $Z_{3,.,j} \oplus Z'_{3,.,j} \in \mathcal{L}_j$ for all j.

Q.4 Give an algorithm which recovers a set of about N^4 possible values in which K_4 belongs to with probability $1/N^3$, with a time complexity equivalent to N^4 encryptions, and two chosen plaintexts. Explain why the attack works and justify the complexity.
HINT: recover $\mathsf{ShiftRows}^{-1}(\mathsf{MicColumns}^{-1}(K_4))$ by chunks of four bytes.

Q.5 Deduce an attack to recover K_4 with good probability, using as little complexity as possible.
CHALLENGE: obtain an attack using $\sqrt{2}N^{\frac{3}{2}}$ chosen plaintexts, time complexity $\mathcal{O}(N^4)$, and memory complexity $\mathcal{O}(N^4)$.

Q.6 Design an attack to recover K_4 with good probability, using $\sqrt{2}N^{\frac{15}{2}}$ known plaintexts, time complexity $\mathcal{O}(N^8)$, and memory complexity $\mathcal{O}(N^4)$.

2 ZKPoK from Sigma

We consider a relation $R(x, w)$ defining a language for which we have a Σ protocol (P, V) over a challenge set $\{0,1\}^t$ with accepting predicate $V(x, a, e, z)$, Σ-simulator S, and Σ-extractor E. We define a relation $R'((x, a), (e, z))$ to hold on instance (x, a) with witness (e, z) if $V(x, a, e, z)$ is accepting. We assume that R' also has a Σ protocol (P', V') over the same challenge set $\{0,1\}^t$ with accepting predicate $V'(x, a, a', e', z')$, Σ-simulator S', and Σ-extractor E'. We consider the following protocol:

Prover		**Verifier**
w	x	
		pick $e \in_U \{0,1\}^t$, $(a, e, z) \leftarrow S(x, e)$
	$\xleftarrow{a, a'}$	pick ρ_V, $a' \leftarrow P'((x, a), (e, z); \rho_V)$
pick $e' \in_U \{0,1\}^t$	$\xrightarrow{e'}$	
unless $V'(x, a, a', e', z')$, abort	$\xleftarrow{z'}$	$z' \leftarrow P'((x, a), (e, z), e'; \rho_V)$
pick $e_2 \in_U \{0,1\}^t$, $(a'_P, e_2, z'_P) \leftarrow S'((x, a), e_2)$		
pick ρ_P, $a_P \leftarrow P(x, w; \rho_P)$	$\xrightarrow{a_P, a'_P}$	
	$\xleftarrow{e_V}$	pick $e_V \in_U \{0,1\}^t$
$e_1 \leftarrow e_V \oplus e_2$		
$z_P \leftarrow P(x, w, e_1; \rho_P)$	$\xrightarrow{z_P, z'_P, e_2}$	$e_1 \leftarrow e_V \oplus e_2$
		$V(x, a_P, e_1, z_P) \wedge V'(x, a, a'_P, e_2, z'_P)$

Q.1 In the first part of the protocol, recognize and isolate a commitment on the value e and a proof of knowledge of a valid opening of this commitment. Fully describe the commitment scheme. Fully describe the proof of knowledge.

Q.2 In the second part of the protocol, recognize a proof of knowledge of either w for $(R(x, w)$ or (e, z) for $R'((x, a), (e, z))$.

Q.3 Show that the protocol is complete and runs in polynomial time $\mathsf{poly}(t, |x|)$ (where $|x|$ is the length of x) for the verifier.

Q.4 Show that the protocol is zero-knowledge by constructing a black-box simulator.

Q.5 Construct a knowledge extractor for this protocol to prove that it is a zero-knowledge proof of knowledge for R.

3 PRP versus Left-or-Right

Given a security parameter (which is implicit and omitted from notations for better readability), we consider a pair $(\mathsf{Enc}, \mathsf{Dec})$ of functions from $\{0,1\}^k \times \{0,1\}^n$ to $\{0,1\}^n$ (k and n are functions of the security parameter). These functions are such that for all K and X, we have

$$\mathsf{Dec}(K, \mathsf{Enc}(K, X)) = X$$

It is assumed that there are implementations which can evaluate both functions in polynomial time complexity (in terms of the security parameter). We define several security notions.

PRP. We say that this pair is a *pseudorandom permutation* (PRP) if there exists a negligible function negl such that for all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, we have $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, 1) \to 1] \leq \mathsf{negl}$, where $\Gamma^{\mathsf{PRP}}(\mathcal{A}, b)$ is the PRP game defined as follows:

$\Gamma^{\mathsf{PRP}}(\mathcal{A}, b)$:
- initialize a list $\mathcal{L}$ to empty
- pick $K \in \{0,1\}^k$ uniformly at random
- pick a permutation Π over $\{0,1\}^n$ uniformly at random
- run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
- return b'

subroutine $\mathcal{O}(x)$:
- if $x \in \mathcal{L}$ abort
- insert x in $\mathcal{L}$
- **if** $b = 0$ **then**
  - return $\mathsf{Enc}(K, x)$
- **else**
  - return $\Pi(x)$
- **end if**

LoR. We say that this pair is *LoR-secure* if there exists a negligible function negl such that for all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, we have $\Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, 1) \to 1] \leq \mathsf{negl}$, where $\Gamma^{\mathsf{LoR}}(\mathcal{A}, b)$ is the left-or-right game defined as follows:

$\Gamma^{\mathsf{LoR}}(\mathcal{A}, b)$:
- initialize two lists $\mathcal{L}_l$ and $\mathcal{L}_r$ to empty
- pick $K \in \{0,1\}^k$ uniformly at random
- run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
- return b'

subroutine $\mathcal{O}(x_l, x_r)$:
- if $x_l \in \mathcal{L}_l$ or $x_r \in \mathcal{L}_r$, abort
- insert x_l in $\mathcal{L}_l$ and x_r in $\mathcal{L}_r$
- **if** $b = 0$ **then**
  - return $\mathsf{Enc}(K, x_l)$

else
  return $\mathsf{Enc}(K, x_r)$
end if

We want to show the equivalence between these notions.

Q.1 Is the list management important in each security definition (or: what happens with modified definitions in which we remove the lists)? Justify your answer.

Q.2 We consider the following hybrid game:

$\Gamma^{\mathsf{hyb}}(\mathcal{A}, b)$:
initialize a list $\mathcal{L}$ to empty
pick $K \in \{0,1\}^k$ uniformly at random
pick a permutation Π over $\{0,1\}^n$ uniformly at random
run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
return b'
subroutine $\mathcal{O}(x)$:
if $x \in \mathcal{L}$ abort
insert x in $\mathcal{L}$
if $b = 0$ **then**
  return $\mathsf{Enc}(K, x)$
else
  return $\mathsf{Enc}(K, \Pi(x))$
end if

Show that for all $\mathcal{A}$ playing the PRP game and any b, we have $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{hyb}}(\mathcal{A}, b) \to 1]$.

Q.3 Given $\mathcal{A}$ playing the PRP game, we define $\mathcal{B}$ playing the LoR game as follows:

$\mathcal{B}^{\mathcal{O}}$:
pick a permutation Π over $\{0,1\}^n$ uniformly at random
run $\mathcal{A}$
when $\mathcal{A}$ makes a query x to its oracle, answer by $\mathcal{O}(x, \Pi(x))$
return the same output as $\mathcal{A}$

Show that $\Pr[\Gamma^{\mathsf{hyb}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{B}, b) \to 1]$ for any b.

Q.4 Deduce that LoR-security implies PRP.
CAUTION: adversaries must be PPT.

Q.5 Using the following game, show that PRP security implies LoR security. Give a precise proof with the reductions.

$\Gamma^{\mathsf{generic}}(\mathcal{A}, b, c)$:
initialize two lists $\mathcal{L}_l$ and $\mathcal{L}_r$ to empty
pick $K \in \{0,1\}^k$ uniformly at random
pick a permutation Π over $\{0,1\}^n$ uniformly at random
run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
return b'
subroutine $\mathcal{O}(x_l, x_r)$:

if $x_l \in \mathcal{L}_l$ or $x_r \in \mathcal{L}_r$, abort
insert x_l in $\mathcal{L}_l$ and x_r in $\mathcal{L}_r$
if $b = 0$ **then**
    if $c = 0$ **then**
        return $\mathsf{Enc}(K, x_l)$
    else
        return $\Pi(x_l)$
    end if
else
    if $c = 0$ **then**
        return $\mathsf{Enc}(K, x_r)$
    else
        return $\Pi(x_r)$
    end if
end if