Indian Institute of Technology Kharagpur

SPRING Semester, 2023
COMPUTER SCIENCE AND ENGINEERING

CS60004: Hardware Security

End Semester Examination

Full Marks: 50

3 hour

1. Consider the Advanced Encryption Standard (AES) SBox operation in $GF((2^4)^2)$. Let the irreducible polynomial of an element in $GF((2^4)^2)$ be $r(Y) = \gamma^2 + \gamma + \mu$. Let $\gamma = \gamma_1 Y + \gamma_0$ be the input to the circuit and $\delta = (\gamma_1 Y + \gamma_0)^{-1} = (\delta_1 Y + \delta_0) mod (Y^2 + Y + \mu)$ be the output. The inverse can be expressed by the following equations

$$\delta_1 = \gamma_1(\gamma_1^2\mu + \gamma_1\gamma_0 + \gamma_0^2)^{-1} = \gamma_1.d'$$
$$\delta_0 = (\gamma_0 + \gamma_1)(\gamma_1^2\mu + \gamma_1\gamma_0 + \gamma_0^2)^{-1} = (\gamma_0 + \gamma_1).d'$$

where $d' = d^{-1}$ and $d = \gamma_1^2\mu + \gamma_1\gamma_0 + \gamma_0^2$. Your task is to perform masking of the AES SBox operation. For that, we need to mask the above operations such that final outputs are also masked by random values. Let m_h and m_l be the input masks for γ_1 and γ_0 respectively. Hence the masked input is $(\gamma_1 + m_h)Y + (\gamma_0 + m_l)$. Let the output masks be m'_h, m'_l, m_d, m'_d and masked outputs be $(\delta_1 + m'_h), (\delta_0 + m'_l), (d + m_d), (d' + m'_d)$. Proceed step by step by answering the following questions.

(a) Consider the masking of term δ_1 with m'_h. The desired result is $\gamma_1 d' + m'_h$. However, due to masking, the input and the outputs are masked. You have to propose the steps to compute this masking using inputs $(\gamma_1 + m_h), (d' + m'_d), m_h, m'_h, m'_d$. State the steps. (5 marks)

(b) Since we have obtained the masked expression for $\gamma_1 d'$ with m'_h, now compute the masking for δ_0 in terms of $(\delta_1 + m'_h), (\gamma_0 + m_l), (d' + m'_d), m_l, m'_h, m'_l, m'_d$. (5 marks)

2. Bob performs first-order masking for AND gate as shown in Fig. 1. Let a and b be the inputs to an AND gate and let m_a and m_b be the masks of a and b respectively. Thus a_m and b_m are masked values corresponding to a and b respectively, i.e. $a_m = a \oplus m_a$ and $b_m = b \oplus m_b$. However, Prof. Alice points out that this AND gate is vulnerable to first-order leakages due to glitches. In other words, the glitch power is correlated with the unmasked input. Prove or disprove Prof. Alice's statement considering a glitch in input a_m and b_m and m_b are 1 as depicted in Fig. 1. Assume that all the AND gates and the XOR gates have the same delay. (5 marks)

3. Bob knows that AND gate leaks information about the inputs. In order to hide the information about the input he tried to mask them using multiplicative masking, as described below: Let, $a_m = a.m_a$, $b_m = b.m_b$, $y_m = a_m.b_m$ and $m_y = m_a.m_b$, then $y = y_m.m_y^{-1}$ where $a, b \in GF(2)$ are the unmasked inputs to the AND gate, m_a, m_b are input masks chosen uniformly at random and a_m and b_m are the masked inputs of a and b respectively.

(a) Prove the correctness of the equation.

(b) Even after implementing the masked AND gate, Bob figured out that there is a certain amount of leakage about the input values. He tried to quantify the leakage by computing the probability of $a = 0$ given that $y_m = 0$. Compute this probability. (5 marks)

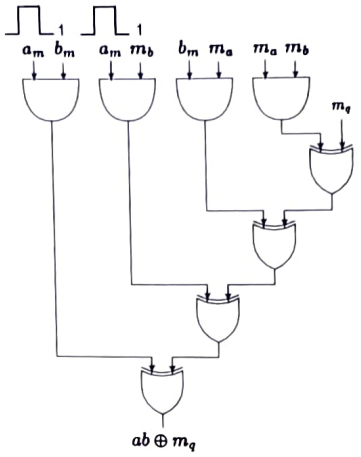

Figure 1: Masked AND Gate

4. Consider the Advanced Encryption Standard (AES)'s Key scheduling as shown in Fig. 2. In this figure, the keys of 8^{th}, 9^{th}, and 10^{th} rounds of AES-128 are depicted, wherein $K^r_{i,j}$, refers to the $(i,j)^{th}$ byte of the round key for r^{th} round, where $r \in \{8, 9, 10\}$. The round keys are represented by a 4×4 matrix. For instance, the first row of the 10^{th} round key K^{10} is represented as $\langle K^{10}_{0,0}, K^{10}_{0,1}, K^{10}_{0,2}, K^{10}_{0,3} \rangle$.

Here, SubWord operation refers to a substitution operation using the AES SBox (denoted by S) and RotWord refers to rotation operation. $Rcon_8, Rcon_9$ and $Rcon_{10}$ denote the round constants corresponding to the $8^{th}, 9^{th}$ and 10^{th} round respectively. To illustrate their effect, consider the last column of the 8^{th} round key is $\langle K^8_{0,3}, K^8_{1,3}, K^8_{2,3}, K^8_{3,3} \rangle$. Here the output of SubWord and RotWord when operated on this column is $\langle S(K^8_{1,3}), S(K^8_{2,3}), S(K^8_{3,3}), S(K^8_{0,3})) \rangle$.

An attacker wants to retrieve the AES last round key by inducing a byte fault in the 8^{th} round of the AES-128 key scheduling algorithm as depicted in Fig. 3. Note unlike as studied in the class, here the fault is in the 8^{th} round key and not on the state matrix. Let p denote the byte fault value induced by the attacker and q, r denote the fault values spread due to the operations in the key scheduling algorithm, as marked in the Fig. 3.

(a) Derive the relationship between the fault values p and q in terms of K^{10} bytes. (2 marks)

(b) Derive the relationship between the fault values q and r in terms of K^{10} bytes. (2 marks)

(c) The fault induced in the 8^{th} round key corrupts the first row in the state matrix in the 9^{th} round of AES with the fault value $\{p, p, p, p\}$. The corrupted values are further spread to other rows after the MixColumn operation. Fig. 4 depicts the propagation of the fault differentials through the last 3 rounds of AES-128 in such a scenario. For your convenience, we have marked some of the fault differential bytes. Your task is to complete the remaining blocks in state matrices S_1, S_2, S_3 and key matrices K_8, K_9, K_{10} in Fig. 4. Showing the state matrices in the answer is sufficient.

(6 marks)

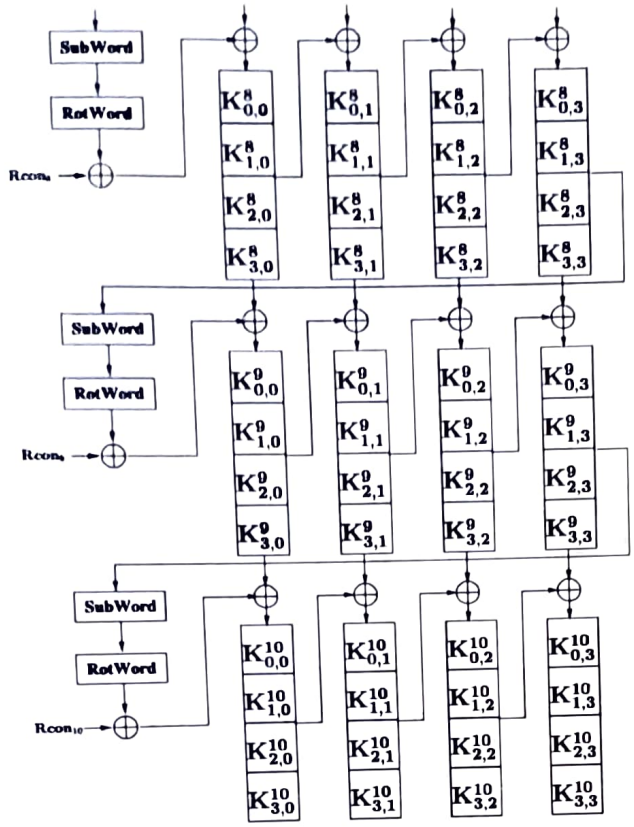

Figure 2: Last 3 rounds of AES-128 Key Scheduling

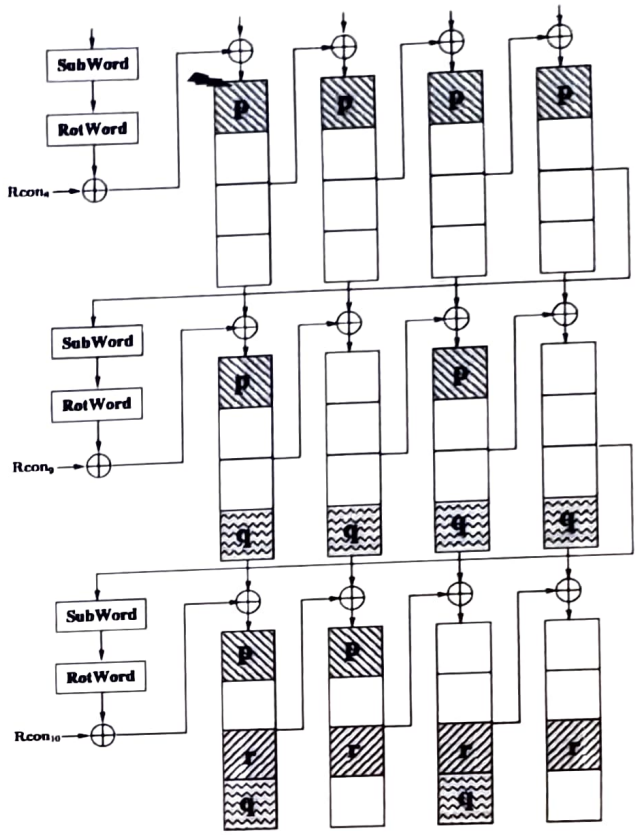

Figure 3: Byte Fault induced in the 8^{th} round key of AES-128 Key Scheduling

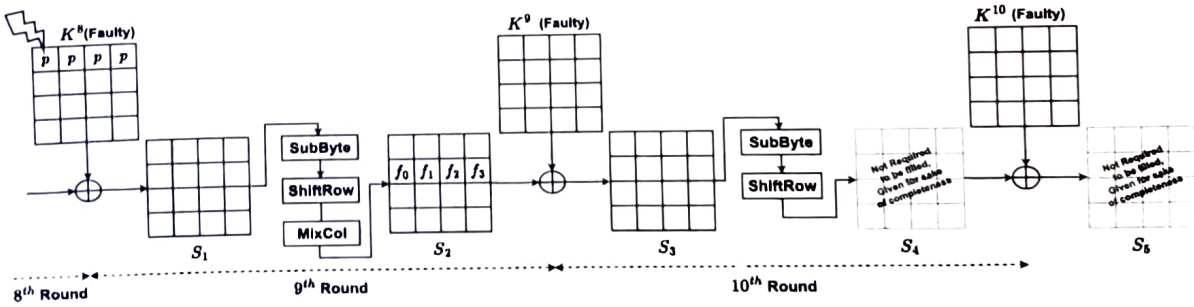

Figure 4: Differential fault propagation through last three rounds of AES-128

5. Consider the following code for the computation of exponentiation $s = y^x$ starting from LSB. In the function below, y is the base, x is the exponent, s stores the intermediate values that lead to the final result.

```
double power(int y, int x, int from, int to)
{   int i, bit; long double s = 1;
    for(i=from; i<=to; ++i)
    {
        bit = (x >> i) & 1;
        if (bit == 1)
        { s = s * y;}
        y = y * y;
    }
    return s;
}
```

Perform Template attack on Algorithm given above by considering $x = 7$, while the value of $from = 0$ and $to = 2$. Note that the MSB is 1 by definition. The attacker builds templates on a known device for the power consumption due to the computation of the exponentiation. The template consists of the tuple $\langle T_0, T_1, T_2 \rangle$, where each T_i denotes the power consumption while processing the i^{th} iteration. In the table, the column heads indicate the various 3-bit key choices starting from bit positions 0 to 2. The template for the unknown exponent of interest is $T_0 = 2.21, T_1 = 2.85, T_2 = 3.95$. Find the 3-bit exponent by performing the Least Mean Square Test using the template below (refer Tab. 1).

Hint: The formula for Mean Square Test: $[(t_1 - m_1)^2 + (t_2 - m_2)^2 + (t_3 - m_3)^2]^{1/2}$

Table 1: Power Template for 3-bit Exponent

Iterations	000	001	010	011	100	101	110	111
T_0	2.70	3.13	2.61	2.42	2.22	3.19	2.12	2.34
T_1	2.79	2.34	2.31	2.88	2.10	2.70	2.27	3.06
T_2	2.36	3.06	2.28	2.24	2.55	3.13	2.88	3.34

(10 marks)

6. Mr. Thompson wants to protect a nonlinear function $A = XY \oplus Z$ using Threshold Implementation (TI) where X is shared as (X_1, X_2, X_3) such that $X = X_1 \oplus X_2 \oplus X_3$. Likewise, Y and Z are shared into three shares. He has come up with the following shares:
$A_1 = ((X_2 \oplus X_3)(Y_2 \oplus Y_3)) \oplus Z_2$,
$A_2 = (X_1Y_3 \oplus Y_1X_3 \oplus X_1Y_1) \oplus Z_3$,
$A_3 = (X_1Y_2 \oplus Y_1X_2) \oplus Z_1$
Can you help him verify whether his sharing has the following properties?

(a) Correctness

(b) Non-Completeness

(c) Uniformity - *It is sufficient to argue the uniformity for a specific case of* $x = 0, y = 0, z = 0$.

(3+2+5 marks)