HWSec: exam

Renaud Pacalet

18 June 2021

You can use any document you need. Please number the different pages of your work and indicate on each page your first and last names. Write your answers in French or in English, as you wish, but avoid mixing the languages. If some extra information or hypotheses are missing to answer a question or solve a problem, decide by yourself and write down the added hypotheses or information. If you consider a question as absurd and thus decide not to answer, explain why. If you do not have time to answer a question or solve a problem but know how to, briefly explain your ideas.

Important advice #1: quickly go through the document and answer first the easy parts.

Important advice #2: copying verbatim the slides of the lectures or any other provided material is not considered a valid answer.

The 5 questions are worth 2 points each. The problem is worth 10 points.

1 Questions

1.1. Power attacks: we consider the DES hardware implementation represented on figure 1. Each encryption takes 16 clock cycles and the 16 successive states $L_0R_0, L_1R_1, \ldots, L_{15}R_{15}$ are stored in the LR 64-bits register.

In the notations we use the same bit numbering as in the DES standard: the leftmost bit of any bit-string is numbered 1. Consider the following basic DPA attack against $K_1[1 \ldots 6]$, the 6 leftmost bits of the first round key:

1: \text{best\_guess} \leftarrow 0; \text{best\_score} \leftarrow 0;
2: \textbf{for} guess = 0 to 63 \textbf{do}
3: \quad T_0 \leftarrow ZERO; n_0 \leftarrow 0; T_1 \leftarrow ZERO; n_1 \leftarrow 0;
4: \quad \textbf{for} each pair of power trace and plaintext \{T, p\} \textbf{do}
5: \quad \quad \text{compute decision bit} \ d(guess, p);
6: \quad \quad \textbf{if} d(guess, p) = 0 \textbf{then}
7: \quad \quad \quad T_0 \leftarrow T_0 + T; n_0 \leftarrow n_0 + 1;
8: \quad \quad \textbf{else}
9: \quad \quad \quad T_1 \leftarrow T_1 + T; n_1 \leftarrow n_1 + 1;
10: \quad \quad \textbf{end if}
11: \quad \textbf{end for}
12: \quad \text{score} \leftarrow \max(|T_0/n_0 - T_1/n_1|);
13: \quad \textbf{if} score > \text{best\_score} \textbf{then}
14: \quad \quad \text{best\_score} \leftarrow score; \text{best\_guess} \leftarrow guess;
15: \quad \textbf{end if}
16: \textbf{end for}
17: \textbf{return} best\_guess;
Figure 1: DES hardware implementation
Scalar names are lowercase, vector names are uppercase. \( Z E R O \) denotes the all-zeros vector. The decision bit is \( R_1[9] \), one of the four bits of \( R_1 \) that depend on \( K_1[1 \ldots 6] \) (the three others are \( R_1[17], R_1[23] \) and \( R_1[31] \)). \( d(guess,p) \) is the value that this decision bit would take for plaintext \( p \), assuming \( K_1[1 \ldots 6] = guess \).

\( \max(T) \) is the maximum component of vector \( T \); it is a scalar. Addition and difference of vectors are component-wise. The division of a vector by a scalar is component-wise. \( |T| \) is the component-wise absolute value of vector \( T \); it is a vector.

Enumerate several optimizations that can be used to improve this attack. Explain why each optimization helps reducing the number of required power traces. In your opinion which optimization is the most promising, why? Under which hypotheses is it the most promising?

1.2. Side-channel attacks: what makes timing attacks usually more practical than power attacks? What makes them also frequently less efficient?

1.3. Timing attacks: consider a software implementation of some cryptographic algorithm. A countermeasure against timing attacks consists in careful design of each if-then-else statement such that the time spent in the if-then branch is the same as the time spent in the else branch. What are the advantages and drawbacks of this solution? Under what hypothesis can it be considered as efficient?

1.4. Fault attacks: an attacker injects faults in a target device that computes cryptographic signatures, and observes the effects. The device is equipped with an error detection mechanism and never returns the results of erroneous computations. Instead it sends an error message, always the same. After a fault injection the attacker thus receives the error message if the fault caused a computation error, else the correct cryptographic signature.

Give an example of a situation where this simple fact that the fault caused an error or not can be exploited by the attacker to retrieve a valuable secret.

1.5. Probing attacks: assume you are asked to mount a probing attack against the DES hardware implementation represented on figure \( \text{figure} \). Your goal is to retrieve the embedded secret key. You can use only one probe, you can put it only on one output bit of a register and you cannot move it afterwards. But you can monitor your probe during as many encryptions as you wish. For each encryption you have access to the ciphertexts but not to the plaintexts.

What will you probe and how will you proceed to retrieve the secret key? How many encryptions will you need? If your attack does not reveal the full secret key how many bits does it reveal?

2 Problem: timing attack against DES

To solve this problem you will need a global understanding of the DES decryption algorithm. Figures-2, 3 and 4 should be sufficient. \( IP \) and \( FP \) are 64 to 64 bits permutations. \( \oplus \) is the symbol used to represented the bitwise exclusive or of two bitstrings. \( E \) is a 32 to 48 bits expansion-permutation. \( P \) is a 32 to 32 bits permutation.
$S_1, S_2, \ldots, S_8$ are 8 different, 6 to 4 bits, non-linear, substitution functions. $PC_1$ is a 64 to 56 bits selection-permutation. $PC_2$ is a 56 to 48 bits selection-permutation. $RS$ is a 28 to 28 bits rotation by one or two positions to the right, depending on the round number. All these primitive functions are perfectly defined in the DES standard.

Assume you’re in charge of attacking a software implementation of the DES decryption. You can pass it a chosen ciphertext and it will return you the corresponding plaintext. Your target is the unknown but constant secret key. The software implementation suffers a flaw: its computation time has been optimized and depends on the input ciphertext and on the secret key. This dependency is in the $E$ expansion-permutation only and it is very easy to model: the number of cycles to compute $E$ is linearly dependent on the Hamming weight\(^\text{1}\) of its 32 bits input. That is:

$$t_E(A) = \alpha + \beta \times HW(A)$$

where $A$ is the 32 bits input, $t_E(A)$ is the number of cycles taken by the microprocessor to apply permutation $E$ on $A$ and $HW(A)$ is the Hamming weight of $A$. $\alpha$ and $\beta$ are two natural, positive integers. You can send as many ciphertexts as you wish and you can measure the computation time for each of them. The timing measurements are not perfect, they are noisy:

$$\hat{t} = t + e$$

where $\hat{t}$ is the measurement, $t$ the actual computation time and $e$ a measurement error.

Explain what your attack will be. What will you do, in what order and why? Clearly and carefully define your notations. Try to express your attack algorithm in a semi-formal way (pseudo-language) so that it is both complete and non ambiguous.

What is your opinion about this flaw? Is it critical? Is your attack practical? Does its efficiency depend on some parameters? If yes, what parameters?

Finally, assume it is not $E$ that has a data dependent timing but $IP$. Would your attack still work? Why? If it does not, could you adapt it to this new case?

What about the other permutations of the DES algorithm ($P$, $PC_1$, $PC_2$, $FP$)?

\(^{1}\)Remember that the Hamming weight of a bitstring is its number of ones
Figure 2: DES decryption
Figure 3: DES Feistel function

Figure 4: DES decryption key schedule