Malicious Key Emission via Hardware Trojan Against Encryption System

David Hély
Grenoble INP, LCIS
Valence, France
david.hely@grenoble-inp.fr

Maurin Augagneur, Yves Clauzel and Jérémy Dubeuf
Grenoble INP-Esisar
Valence, France
firstname.lastname@ grenoble-inp.org

Abstract—**In this work, we propose a hardware Trojan within a given encryption platform. This malicious hardware aims at leaking the secret key used for encryption without perturbing the system so that the user does not notice it. We propose a hardware Trojan which detects any new encryption start and then transmit the used expended key on the system serial link. This hardware Trojan does not require any processor modification. The paper presents the way the hardware Trojan has been developed according the given platform and the associated information. This work has been developed by Grenoble INP students (M. Augagneur, Y. Clauzel and J. Dubeuf) during the secure IC design labs of the Grenoble INP Esisar and was then presented for the Embedded System Challenge during the Cyber Security Awareness Week (CSAW) 2011 organized by Polytechnic Institute of New York University.**

Keywords-hardware trojan, secure hardware

I. INTRODUCTION

Hardware Trojans [1], [2] are malicious hardware components embedded within the chip by an adversary in order to disable or destroy a system at some future time or in order to leak confidential information. The embedded System Challenge 2011 proposes to university teams to design a hardware Trojan for a dedicated system. This system is an FPGA based encryption platform which performs RC5 encryption. The RC5 operations are calculated by an 8051 processor while different RAM memories are used to store separately the software code and the data. The communication between the end user and the processing platform is done via a serial link (i.e; an Universal Asynchronous Receiver Transmitter). Hardware Trojan designers have full access to the vhdl code of the platform as well to the assembler code used for the encryption. The first step when designing a hardware Trojan is to decide which will be the effect of the malicious circuit. Our approach was to leak secret information of the encryption platform without disturbing the system. If we refer to [3] where a hardware Trojan classification is given, the proposed Trojan of this work fits with the functional class. The strategy chosen by the team was to not modify the processor core but to add malicious components at the interfaces between the processor and the external memories and the communication interface. As proposed in [4], this Trojan main purpose is to quietly leak piece of secret information on the communication link without disturbing the normal communication. Nevertheless, while in [4] the hardware Trojan is always on, the proposed Trojan is only active under certain consitions. Such hardware Trojan may require more circuit modifications but are very difficult to detect for the user since neither the system functionalities nor the system performances are decreased. In order to perform such an illegal activity, it is necessary to first detect the moment the interesting data are processed by the processor then to store the targeted data in a dedicated memory and finally to communicate this data to the external world. The designed Trojan is based on three main components which respectively:

1. Trigger the Trojan activity
2. Store the data to be leaked in a given memory place
3. Leak the eavesdropped data on the communication interface

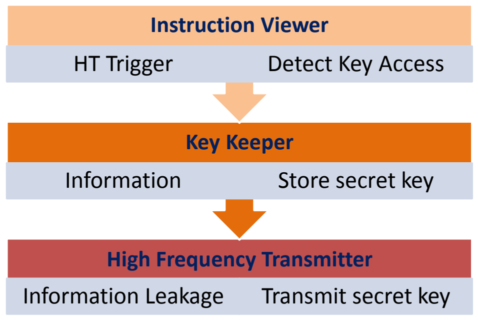

Figure 1. Trojan process

The methodology chosen by the team was to design and validate each block separately since they have different and independent functions. In this paper we propose then to detail each blocks separately before to discuss the whole hardware Trojan. This paper is then organized as follows. The next section provides a description of the Trojan trigger. Section III provides the details of the second element of the Trojan which capture the secret data to be leaked. The description of the high frequency transmitter is given in section IV. The evaluation of the complete Trojan against the encryption platform is given in section V and finally the conclusions are discussed in section VI.

II. THE TROJAN TRIGGER

Since our Trojan purpose is to leak pieces of the secret information, a trigger is necessary to active the Trojan upon the secret information is processed by the processor. The solution we decided to design was a kind of "instruction viewer", which should be able to detect any manipulation of secret subkeys by the processor. A dedicated component has thus been designed in order to spy data exchanges between the instruction RAM and the processor which would be able to detect any access to the secret subkeys done during RC5 encryption. The RC5 encryption used subkeys which are generated from the original key. It has been decided to target subkeys of the RC5. The encryption detection is based on the fact that when looking at the RC5 algorithm, the extended key is always used as an operand of an addition. Figure 2 gives an overview of the RC5 encryption pseudo-code where S[i] is the ith element of the subkey and A and B are the two registers of the input block.

```
$A = A \boxed{+ S[0];}$
$B = B \boxed{+ S[1];}$
for $i = 1$ to $r$ do
    $A = ((A \oplus B) \lll B) \boxed{+ S[2 * i];}$
    $B = ((B \oplus A) \lll A) \boxed{+ S[2 * i + 1];}$
```

Figure 2. RC5 encryption pseudo-code

We can deduce from this that any access to the external memory followed by an addition should allow us to detect an access to the extended key. Studying the 8051 assembly code, we can find out the hexadecimal values corresponding to the sequences we are looking for. Specifically, we can note that the instruction bytes 0xE0, which is the move of a byte from the external memory pointed by the data pointer to the accumulator, followed by 0x36, which is the addition of the content of the accumulator and a register correspond to the these data. By watching these two bytes successive on the data bus, we can detect on any access to any bytes of the extended key. The so call instruction viewer is then a dedicated small circuit which spies the address bus and the data bus. Upon a change on the address bus, the data bus is read if the data corresponds to the expected bytes then the trigger is on in order to indicate to the rest of Trojan that secret data (i.e. a byte from the subkey) will be placed on the data bus. In order to validate the instruction viewer, a simple module was design in order to count how many time the trigger has been activated for an encryption. In the given configuration, the extend key is 104 byte long, we then experimentally validate on the board that the trigger is activated 104 times.

III. Storing sensitive data

Upon detection of an access to the subkey it is not possible to directly send data to the serial link without perturbing the system. The chosen strategy consists then in storing the sensitive data within the internal memory before to communicate them to the external world. In addition to the "*instruction viewer*" module, a second module has then been designed in order to write in a given place of the memory the targeted data (i.e. the one being accessed at the moment the instruction viewer has detected an access to the extended key). It could has been a strategy to not write the data in a dedicated memory emplacement nevertheless according to code compilation or memory organization, the sensitive data is not always stored at the same place from a system to another. By storing the attacked data to a known place, it will be then easier to later access it to leak it on the communication link. The so called "data writer" acts just like a direct memory access circuit. It intercepts the data being read by the processor and writes it back in memory.

In order to reduce the area impact of this "writing" module, the data being captured are not stored in flip-flop registers but are stored in memory system of the given 8051 platform. Since the whole memory system is not full, it is possible to reserve a dedicated place which will be used to store the captured data.

IV. Information leakage

The given encryption platform uses an RS232 communication line to transmit encrypted data to the external world. As proposed in [6], It has been decided to use this link in order to leak secret information eavesdropped by the Trojan. The main goal of this Trojan part is to leak information as quietly as possible, it is then important that it does not perturb the communication. In [6] the authors propose to exploit the message structure to encode the stolen data or to play with the protocol to leak dedicated data at a different baud rate than the expected one. We propose here a Trojan which sends a high frequency signal which contains the data to be leaked using a dedicated protocol. As proposed in [4], the electric signal of the transmission is modified in order to add information without disturbing the original transmission. In [4], the authors chose to encode data by tuning the original data symbol. In this work, we propose to add new symbols which should not be detected by the original receiver.

First, it is necessary to define a dedicated encoding, in this case the 0 data is encoded by a null signal while a 1 data is encoded by a 10 ns pulse as depicted on Figure 3. Moreover a preamble is defined in order to detect a communication start; it is defined by a data 1 symbol.

Data-0
Data-1
10ns

Figure 3. High Frequency data encoding

The high frequency signal containing the eavesdropped information is xored with the original signal transmitted by the UART. The serial link operates at a rate of 9600 baud, initially the same rate was used for the high frequency communication. Nevertheless such a configuration perturbs the UART communication which fails when the Trojan is on. The data rate of the high frequency signal has then been reduced to 1200 bits per second in order to not disturb the original signal. With this configuration, the UART link is completely safe and thus the Trojan hardly detectable. Figure 4 shows an UART transmission when the Trojan is active.

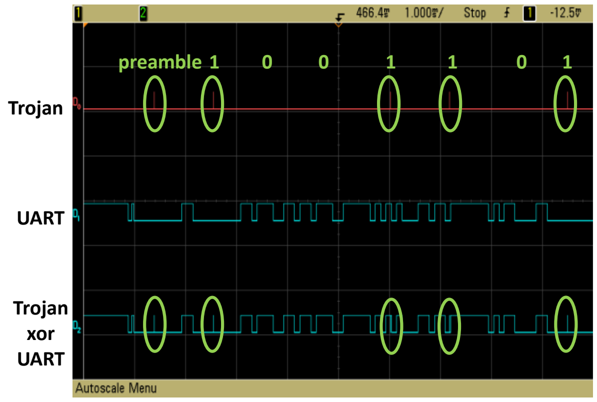

Figure 4. High frequency transmission on UART

The first chronogram shows the high frequency link with eavesdropped data, the second one the original UART line and the third one modified UART line containing both the high frequency signal xored with the UART line. In order to retrieve the data a hacker having knowledge of the Trojan will just probe the UART line and using a high pass filter will retrieve the secret information.

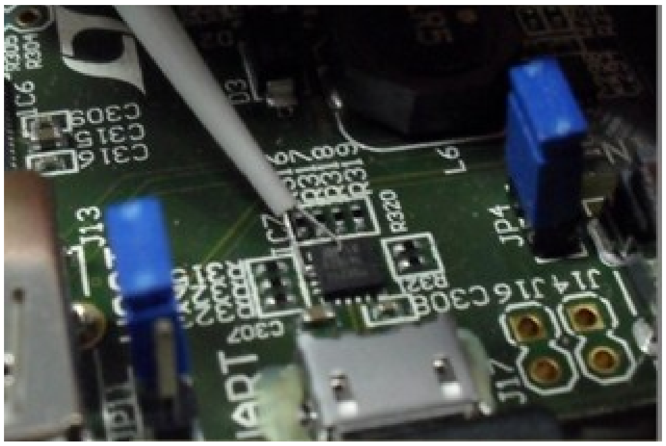
Figure 5. Probing the UART line

Such a Trojan presents the advantage of not disturbing the original system by mixing the secret information with the trusted one without damaging the communication link. On the other hand, it can only work at a lower data rate than the original one to be quiet and will require storing the secret information to be leaked before transmission.

V. THE GLOBAL TROJAN

Figure 6 illustrates the final hardware Trojan within the initial architecture. This Trojan is based on three components:

- The instruction viewer (which acts as a trigger)
- The writing system (to store the confidential data)
- The high frequency system to transmit the data

First, the Trojan detects an access to a byte of the subkey used for encryption thanks to the instruction viewer. Then the subkey byte is written back in memory. This will be done 104 times since the subkey is 104 byte long. Moreover, the Trojan was modified so that information will be transmitted only once until a new key is used. Then during some transmission, the subkey written in memory will be transmitted on the serial link using the high frequency transmitter. This Trojan has been validated experimentally using a known key on the dedicated FPGA board, probing the UART line as illustrated on Figure 5 we retrieve the targeted key.

As depicted on Figure 6, the Trojan is split within the system. Neither the processor core nor the memories have been modified. The malicious circuitry is placed at the interfaces between the several components.

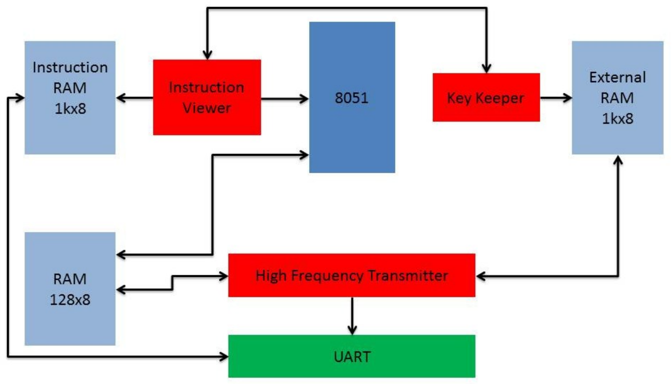

Figure 6. The Final hardware Trojan architecture

Such a Trojan architecture presents several advantages. First, the detection is not trivial since none of the main cores are modified, functional verification and validation may not trivially find out the malicious modifications. Usual detection methods aiming at detecting any abnormal behavior are thus not efficient against this Trojan. Moreover due to the Trojan distribution, one can detect suspicious blocks; nevertheless each block analyzed independently may not appear as a critical part at a first glance.

The Trojan is not always active, thanks to the trigger “instruction viewer”, it will be activated only when an encryption starts limiting also its visibility. This Trojan does not modify any functional parameters of the system. Particularly, the encryption time is not increased when the Trojan is on since this one works in parallel of the encryption process. The data leakage perturbations are also very limited. The proposed communication does not perturb the communication link. Analyzing the FPGA UART output, a test engineer may detect some high frequency peaks, nevertheless these ones could be misinterpreted as an electrical noise.

Finally the area overhead due to the Trojan is limited since the targeted key is stored in an unused but mapped memory. The area overhead of the Trojan is around 5% of the digital logic part of the system. This area overhead may at a first glance appear high, but it is to be noted that in modern system on chip, such processor design are combined with complex mixed design and then represents only a minor part of the design area.

VI. Conclusions

In this work we propose a hardware Trojan which can leak the secret key of an encryption platform without modifying the system functionalities. We show the different steps and the methodologies followed for the design of a malicious circuit. The Trojan has been validated on the FPGA platform and demonstrated during CSAW 2011 where the Grenoble INP students who entirely specify and design it won the second place. The participation to this contest was also a good opportunity to offer a real case study during hardware security course within Grenoble INP. The proposed design will serve for the coming class to perform experimentation and get further results for its characterization.

Acknowledgment

This work was part of the Grenoble INP-Esisar master degree cursus. Student participation to the ESC contest was sponsored by Grenoble INP, Grenoble INP alumni and A2I Conseil. The FPGA platforms for the contest were donated by Xilinx.

References

[1] [Kar10] “Trustworthy hardware: Identifying and classifying hardware Trojans”Karri, R., Rajendran, J., Rosenfeld, K. & Tehranipoor, M. , IEEE Computer 43(2010), 39 –46.

[2] [Lov11]“ Enhancing security via provably trustworthy hardware intellectual property” Love, E., Jin, Y. & Makris, Y, in IEEE International Symposium on Hardware-Oriented Security and Trust 2011

[3] [Raj10]“Towards a comprehensive and systematic classification of hardware Trojans” Rajendran, J., Gavas, E., Jimenez, J., Padman, V. & Karri, R. (2010), in Circuits and Systems(ISCAS), Proceedings of 2010 IEEE International Symposium on, pp. 1871 –1874.

[4] [Jin11]Hardware trojans in wireless cryptographic integrated circuits Jin, Y. & Makris, Y, Design Test of Computers, 2009, IEEE PP

[5] R.L. Rivest, "The RC5 Encryption Algorithm," Fast Software Encryption: Second International Workshop, Leuven, Belgium, December 1994, Proceedings, Springer-Verlag, 1994, pp. 86-96.

[6] A. Baumgarten, M. Steffen, M. Clausman and J. Zambreno, "A Case Study in Hardware Trojan Design and Implementation", International Journal of Information Security (IJIS), vol. 10, no. 1, pp. 1-14, 2011