Is there an ultimate use of cryptography?
(Extended Abstract)

Yvo Desmedt

Katholieke Universiteit Leuven, ESAT, Belgium†

current address:
Université de Montréal, Dépt. IRO, Case postale 6128, succursale A,
Montréal (Québec), H3C 3J7 Canada

ABSTRACT

Cryptography can increase the security of computers and modern telecommunication systems. Software viruses and hardware trapdoors are aspects of computer security. Based on a combination of these two aspects, an attack on computer security is presented. The complexity of finding such an attack is discussed. A new open problem is: can cryptography prevent such an attack.

1. Introduction

The problem of authenticity plays a very important role in the security of modern telecommunication and computer systems. Cryptography can be used to control access to chips used in these systems and to verify the authenticity of the commands and/or messages and sender. Today such chips are being introduced, e.g. Keyproms. Cryptography can evidently also protect the privacy of the information going around in these systems. In other words, the use of cryptography on each (VLSI) chip in the system increases the computer security, e.g., by protecting against eavesdropping and/or active eavesdropping on the bus of the computer.

In this paper we show that the above solution is not enough to obtain secure computer systems. Let us first introduce the notion of software virus and briefly discuss the hardware aspects of computer security.

A software virus is a subprogram that contains some undesired commands (e.g. time-bomb) and has the capability to copy itself into executables. As a consequence one can completely control a computer (or several computers) using such a software virus. For example one can give somebody a computer game containing such a software virus. The virus copies itself into all executables of the owner of the computer game, while he is playing. If the owner shares some of its executables with other users, the virus affects all executables of the others, and so finally affects the operating system.

Let us briefly discuss some of the hardware aspects of computer security. PCB (printed circuit boards) are sometimes changed during maintenance and it is not excluded that the new PCB is intentionally slightly different from the previous. For example if a PCB controls the access to files, the new PCB may allow a trapdoor access.

Before explaining our attack, we introduce (in Section 2) hardware trapdoors and similar hardware frauds (in chips), which we will call hardware viruses. The name of this term will be clear at the end of this paper. Our attack is presented in Section 3. The attack and hardware viruses would

† This research started while the author was aangesteld navorser NFWO at the Katholieke Universiteit Leuven

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 459-463, 1987.

not be important if they would be easy to detect. In Section 4 we prove that the computational complexity of proving that no hardware virus exists in a chip is at least **NP**-hard. In Section 5 the dangers of such hardware viruses are briefly discussed, and more advanced attacks are overviewed in Section 6. Section 7 is related to a new open problem in cryptography.

2. A hardware virus

For a chip it is possible that non-specified commands exist. These non-specified commands may be (or were) used for test purposes of the chip, or will be used in new versions of the chip. However it is also possible that these non-specified commands allow the chip to act completely differently than specified, e.g., block (halt) itself, or destroy itself, or affect all following commands and/or inputs/outputs and performance of the chip. It is possible that these non-specified commands are only known by the VLSI designer, who put them in for fun, or other reasons. Let us now briefly discuss some variations to implement these fraudulent non-specified commands. The non-specified commands can be given using the pins used for commands, or can be introduced through the input. In the last case a 64 bit pattern is given at the input (e.g. using other non-command pins). This 64 bit pattern is recognized by the chip as being the start sequence of non-specified commands. Evidently in a VLSI chip which also performs encryption, this pattern can be signed, and/or hardware trapdoors can be used to bypass the encryption part. If a (VLSI) chip contains such fraudulent non-specified commands, we will call it affected by a *hardware virus*. We remark that the effect of the hardware virus can be much worse than that of a simple trapdoor access to the chip. Another variant is a hardware virus which is activated at random by the chip itself, for example if during the calculations in the chip some pattern of 46 bits appears, the chip starts to do crazy things. The length of this pattern can be chosen by taking into consideration: the speed of the chip and the "desired" frequency (or probability) of the fraudulent effect of the virus. In this case no external command is in fact necessary. The chip performs however differently than specified. We will also say in this case that the chip is affected by a *hardware virus*. Instead of the virus being activated at a random moment, the time can be used to trigger the effect of the virus.

It is evident that such a virus will be easy to detect if the complexity of the VLSI chip is not high (e.g. a lot of repetition). In other words in such cases the designer will probably not introduce a hardware virus. Remark that the harder it is to reverse engineer a chip, the more difficult it becomes to find the hardware virus. The complexity of finding a hardware virus is discussed in Section 4.

A solution against hardware viruses is to make only chips which are straightforward to reverse engineer. However this allows other companies to copy the bright ideas (e.g. new faster algorithms) and use them for their own purposes in different applications. So this solution has important drawbacks. Another approach to limit the probability that the designer introduces a hardware virus is a clearance procedure to check the designer. This solution never excludes the possibility of fraud. A double check method can be used by having two designers instead of one.

In the next section we explain that the last two methods are not sufficient.

3. Software viruses creating hardware viruses

In previous section it was the designer who intentionally introduced the hardware virus in the chip. In this section we discuss how hardware viruses can be introduced by a CAD (Computer Aided Design) program, without the knowledge of the VLSI designer and/or without the knowledge of the CAD designer.

CAD is more and more used to design complex VLSI chips. CAD being (mainly) software suffers from all its inconveniences. In particular a software virus can exist in a CAD system. An awful case of a software virus living in a CAD program is when the virus contains information to create a hardware virus. Let us briefly explain how this works.

Chips contain easily detectable patterns as transistors. These patterns (and others) are used by the software virus to first check if it is in a CAD program. Once confirmed a flag can be set to avoid rechecking and losing time. Then during the execution of the CAD to make the VLSI chip, the software virus checks the complexity of the chip. So it checks if the same block is used over and over again. If this is not the case and if the software virus concludes that the complexity of the chip is high, it introduces a hardware virus.

The hardware viruses introduced by a software virus through a CAD can be very trivial ones. An example of the effect of the hardware virus is to create some electrical shortcut after a fixed time of use of the chip. More complicated hardware viruses can be introduced, using more advanced software viruses. These software viruses can use artificial intelligence and/or software libraries. This last idea is explained in more detail in Section 6.

We remember that a software virus can be introduced through a game. It is possible that the software virus was injected in the CAD on the computer of the designers of the CAD, or at the computer of the users of it. As a consequence it is *very difficult to find it in the CAD, certainly if the length of CAD programs is taken into consideration.* Antibody programs (detecting and removing software viruses) can be ineffective, if the virus is introduced immediately after the antibody program checked the CAD program.

4. Complexity of finding hardware viruses

A hardware virus is only dangerous if it is very hard to detect it. Let us focus on the problem of proving that no hardware virus exists in a chip. In order to prove that this problem is **NP**-hard, we restrict ourselves to the chips which are nothing else than a Boolean function. Proving that no hardware virus exists in this case means proving that for all possible inputs, the Boolean function f corresponding with the specs is equal to the Boolean function g corresponding with the chip. If for *all* inputs $f = g$, then there is no hardware virus. This problem is the complement of the satisfiability problem, which is **NP**-complete. This is trivial to understand by considering the function h, which is defined as the exclusive or of f and g. So the problem is **CO-NP**-complete. If one drops the restriction on the chips and considers more general cases than Boolean functions (e.g. with feedback) we can say that the problem of proving that no hardware virus is in the chip is at least **NP**-hard. In order to make the proof correct from a mathematical point of view, we allow chips with enormous number of pins, transistors and so on.

We remark that the above reasoning is very similar to a well known problem in testing of VLSI chips.

5. Dangers of hardware viruses

In order to estimate the dangers and the impact of software viruses making hardware viruses, it is very important to remember that chips are not only used in computers. Chips are used today in: telecommunication systems, instruments, controllers (e.g. controlling the temperature of a process in a chemical plant), consumer electronics, security systems (e.g. detecting burglary), medical electronics, industrial electronics (as in robots), cameras, cars, trains, aviation, military applications, space and so on. It is not difficult to imagine the consequences of hardware viruses in these applications, certainly

if the hardware virus is more complex (see Section 6).

It is evident that such hardware viruses (certainly when introduced without the knowledge of the chip designer, see Section 3) affect not only the user, the consumer, the industry using the chips, but also the whole economy based on chip technology and national security.

It is even worse if one takes into consideration that the modern methods of fault tolerant computing are not adequate to protect against such an attack. Indeed the hardware virus in one chip can trigger other ones, and/or the event can be planned from the moment of the design of the virus.

If license fees have to be paid for chips made by the CAD program, the above attack can be used in a more positive way.

6. More advanced and variant CAD attacks

If artificial intelligence continues to develop, it is not excluded that in the future more advanced attacks will become possible. Indeed the software virus in the CAD could also test what kind of chip is under design. Once it figures out what the purpose of the chip is, it selects an adequate hardware virus from a secret software library. If it is for example a disk controller, the hardware virus can be designed to destroy intentionally information on the disk. In the case of a microprocessor (or other part of a computer) the hardware virus contains information to start on the computer (which is under development) a software virus. The opinion of the author related to the last example is that it will only be realistic if VLSI chips contain many more transistors.

Gate arrays are very frequently used in industry. The design is done by the client (some company). The translation from gates to transistors and optimization of these gates is done by another company. After delivery it is even difficult for the designer to reverse engineer the chip. Here a hardware virus can be introduced at several levels: gate level and transistor level. In general it is possible that the introduction of the virus is done just before processing the chip. There are enough steps in the process of making a chip (and certainly a gate array), where a hardware virus can be introduced by a CAD or other program.

7. Open problem

A few years ago several chips were used to implement a cryptographic algorithm. Recently one chip became sufficient. Now cryptographic algorithms can be a part of a VLSI chip. During this process of miniaturization, cryptography is beginning to be used to check access to chips (see Section 1). In other words enlarging its application domain from protecting the system (e.g. against eavesdropping) to protecting parts of the system (access to chips). An open problem is if cryptography can be used on a sub-chip level in order to make virus-free chips, or prove that the chips are virus-free, without affecting the privacy of the chip design, in other words without making chips easily reverse engineerable. From this point of view we can wonder *if there is an ultimate use of cryptography*.

The author thinks that it will be very difficult for modern cryptography to protect against the described attack, without imposing severe restrictions on the chip design methods. This personal idea finds its grounds in the fact that proving that a chip is virus-free is **NP**-hard (see Section 4).

8. Conclusions

The problem of hardware viruses introduced by a software virus coming from a computer game through a CAD program affects several aspects of computer security and security in general. It is also an interesting open problem if cryptography can move into the sub-chip level, or if its use is limited to the system and chip level.

Nevertheless the author did not really used such an attack against a CAD program, several arguments are given related to the feasibility of such an attack. It would be worth to construct such a software virus in order to obtain more information about practical problems and related aspects.

Acknowledgement

The author wants to thank Gilles Brassard (Univ. de Montréal, Canada) for the discussions about this paper and more specially Section 4, Frank Hoornaert (IMEC and ESAT, Belgium) and William Rowan (Univ. Cal Berkeley, USA) for suggesting Section 6 and Jean-Jacques Quisquater (Philips Research Brussels, Belgium) for his discussions in particular related to Section 1.