System-Level Methods to Prevent Reverse-Engineering, Cloning, and Trojan Insertion

Sylvain Guilley, Jean-Luc Danger, Robert Nguyen, Philippe Nguyen

System-Level Methods to Prevent Reverse-Engineering, Cloning, and Trojan Insertion

— Extended Version of [12] —

Sylvain Guilley[1], Jean-Luc Danger[1]
Robert Nguyen[2] and Philippe Nguyen[2].

[1] TELECOM-ParisTech, COMELEC departement, SEN group,
(CNRS LTCI, UMR 5141), 46 rue Barrault, 75 013 Paris, FRANCE.
[2] Secure-IC S.A.S., 80 avenue des Buttes de Coësmes, 35 700 Rennes, FRANCE.

Corresponding author: ⟨ sylvain.guilley@TELECOM-ParisTech.fr ⟩

Abstract. The reverse-engineering (RE) is a real threat on high-value electronic circuits. Many unitary solutions have been proposed to make RE difficult. Most of them are low-level, and thus costly to design and to implement. In this paper, we investigate alternative solutions that attempt to deny the possibility of RE using high-level methods, at virtually no added cost.

Keywords: Reverse-engineering (RE), Cloning, Hardware Trojans Insertion.

1 Introduction

The field of protection against reverse-engineering (RE) has developed recently, especially in the case of software (where it is called digital forensic [11]). Those techniques have extended to hardware, because the design of circuits is more and more outsourced (for fabrication, packaging, assembly, *etc.*). Some governments have additionally enforced anti-RE laws, to force industrialists take this risk into consideration. This article is a position paper about non-technical ways to resist RE and derived attacks that span from mere cloning to Trojan insertion.

The rest of this paper is structured as follows. A detail analysis of the threats categories and of the typical attack protections against them is covered in Sec. 2. Some technologies, mainly proposed by the academia, suitable to fight RE, have emerged. They are reviewed in Sec. 3. Solutions that aim at preventing RE and related attacks at a system-level are discussed in Sec. 4. Eventually, the conclusions are given in Sec. 5.

2 Threats

Reverse-engineering consists in recovering the design by analyzing its implementation. It can serve *per se* to gain illegitimate information. The motivation is

typically acquiring the knowledge about industrial design secrets embedded in a system, or cloning a design by extracting its masks or its netlist, in a view to producing compatible chips (albeit without paying for the research/development cost). Also, it can be a first step for Trojan [18] insertion. A Trojan (or a backdoor) is a piece of software or of hardware that is added to the circuit. It is designed not to interfere with its nominal operation, and to allow for the leakage of some secrets when activated by a secret and unpredictable sequence of inputs. Sometimes, the Trojan is an added functionality, that will not be detectable by the test; sometimes, it can alter one functionality. For instance, the "bug attack" of Shamir [3] can be seen as an exploitation of a Trojan. The idea of this Trojan is that the integer multiplier of the circuit returns one incorrect value, which allows the attacker that knows the data that causes this error to craft a message that will be signed (in practice: raised to the power of the secret key) properly modulo p (resp. q) but incorrectly modulo q (resp. p), which makes it possible to carry out a remote Bellcore attack on CRT-RSA [5].

Accessing the design can be obtained by several means. The first one consists in simply getting the whole design, if the source is accessible or can be easily stolen. To avoid this risk, recommended design practices consist in concealing the design, by good practices of development. For instance, the class `ALC`[3] of the common criteria formally addresses [1] this point.

But even when such organizational measures are enforced, the design can still be reversed. Indeed, the binaries can be extracted from memories, and subsequently decompiled to recover its functionality. Nonetheless, as this threat is very pregnant, the designers usually encrypt the memory contents. This doctrine is commonplace in the military objects, where different kinds of data exist:

1. insensitive data, called black, that are either public or encrypted variables,
2. sensitive data, called red, that consist in secret material manipulated unencrypted (*i.e.* plain).

Thus, as a countermeasure, the data that is intended to be stored in easy dumpable memories is encrypted (*i.e.* turned to black). We talk about a secure bus architecture [10], *a.k.a.* a protection about the so-called bus probing attacks. The attacker can target the bus encryption mechanism, that can be for instance a side-channel attack or a fault attack (corruption of a data when writing to memory). The side-channel attack is perfectly suitable for this scenario, because it can work as well on read and write operations. Indeed, they work in a context of known-plaintext or know-ciphertext only. For instance, such an attack has been conducted successfully in a know-ciphertext context of FPGA bitstream decryption [19,20]. The fault injection attack can only be applied in some conditions. For instance, in some cases, it is useless in read mode: indeed, the data that is faulted is subsequently processed within the CPU, and might not be known by the attacker. Indeed, at best, the faulted read data can make the

[3] In version $\geqslant$ 3.1 of the CC, `ALC` details the requirements associated with the developer's site.

programme crash. But the system can resist the fault injection – but nonetheless without leaving the opportunity for the attacker to recover any information about the injection. Typically, the bitstream decryption in Xilinx FPGAs is sent to the fabric (thus becoming non-functional), but the attacker cannot know which fault has been triggered (it remains internal to the FPGA). Also, if a mode of operation is used in way secure way (an initialization vector is refreshed each time), then it is difficult to observe both a correct and a faulty ciphertext; this is the root of the fault injection resilience countermeasure introduced in [14].

Now, secrets can be recovered from a circuit by various means:

- Non-invasive means: for instance, RE using side-channel analysis attacks, *aka* SCARE [8,16];
- Semi-invasive means [25]; for instance, RE using fault injection attacks, *aka* FIRE [23];
- Invasive means; for instance, the live probing of signals or the delayering of circuits.

3 State-of-the-Art Technical Solutions

The protection of circuits against RE has been appropriated by the scientific community (CHES, FDTC, *etc.*). Many technical solutions, that most of the time assume *optimistically* that an attacker is omnipotent, have been put forward and studied. The most relevant are detailed in the next subsections.

3.1 Physically Unclonable Functions (PUFs)

Physically Unclonable Functions (PUFs [26]) consist in devices that produce a stable yet unique per chip output. They can be used to deter RE by producing internally a key. This key is known by nobody, since it is intrinsic to the electronic circuit; thus there is no possibility that it is compromised by any kind of unwanted disclosure (unintentional copy or theft, *i.e.* human errors or attacks).

Also, as the PUF is unique per chip, it does not prevent to extract the encrypted data (common to a product line), but it prevents from the alteration (by a remote Trojan patch) of all the product line. Indeed, an attacker can only forge the data for the device whose key (PUF result) is known.

3.2 Obfuscation

Obfuscation consists in turning a straightforward implementation into a complex description, that is hard to unravel. It has been proved that there is no universal obfuscation technique, that can apply to any algorithm [2]. Nonetheless, in theory, a concrete obfuscator for RSA, AES, *etc.* can exist. This method exists for both the pure hardware modules as well as for the pure software.

At the hardware-level, the typical impediments are:

– Covering of the chip with a top-level metal to prevent seeing the layers underneath, and to fight probing;
– Spaghetti routing, for evident control wires not to clearly stand out;
– Standard cells scattering, which comes down to the previous item (except that the countermeasure is implemented by constraining the placement and not the interconnection).

At the software-level, the typical impediments are:

– Adding an intermediate virtual machine operating with unknown opcodes;
– Call graph re-engineering;
– Addition of florid tautologies, randomly and abundantly scattered everywhere in the source code.

3.3 Whitebox Cryptography

Whitebox cryptography is a scientific domain that aims at providing a description of an algorithm that embeds a secret (for instance an AES with a constant key), but from which the key cannot be recovered. Some examples exist:

– A PIN code or password verification algorithm, in which the PIN code or the password is stored hashed;
– A block cipher that is implemented as its codebook (hence as a huge memory, that is unrealistic in practice).

Concrete implementations of whitebox cryptography make of secret tables much smaller than the codebook [7]. This solution is attractive in hardware also, since EEPROM memories are much more difficult to read by invasive means than RAM or ROM.

3.4 Backend-Level Decoiling

Circuits can be reversed by the analysis of their layout. Advanced techniques exist for state-of-the-art circuits [27]. But on various occasions, older designs have been exposed with much lower technology tools. For instance,

– CRYPTO1, the encryption algorithm of the NXP MyFare card [21], and
– DSC, the encryption algorithm of the DECT [22],

have between disclosed by the observation of the chip and the reconstruction of its netlist (open source software tools exist for this final step [24]). Incidentally, those algorithms have been cryptanalysed almost as soon as they were known, which is a brilliant example of the danger of security by obscurity mechanisms.

Optical dissimulation against delayering consists in making the analysis of the images difficult, by:

– adding dummy wires, dummy interconnections, dummy gates, *etc.*,
– scrambling the memories [6],
– using custom cells that resemble despite their function differ; For instance the SecLib cells are all derived from a common template, and are customized by the addition of vias [13].

4 More "Holistic" Solutions

In this section, we promote prospective solutions, most of which do not rely on a technical mechanism. The primary goal of the electronic design industry is to sell products that accomplish a given function. Preventing RE is thus usually only a secondary goal. Thus, the anti-RE mechanisms shall be as less costly as possible. This means that both:

1. the design of the countermeasure (if devised internally) shall be as cheap as possible (or the cost of the IP, if bought from a third-party shall be low), and
2. its overhead must not be too large (in terms of silicon area, code size), and it must be discrete enough not to demand for further developments (*e.g.* because the clock frequency is not high enough, *etc.*)

Therefore, the state-of-the-art techniques listed in Sec. 3 are often not suitable. Indeed, they answer (fully or partially) to the issue of preventing RE, but are definitely a cost center. In addition, it can be noticed that they are *ad hoc* solutions that attempt to fix a local problem.

Now, the goal of the protection against RE is to protect the whole circuit, and not only some parts in a unitary manner. Therefore, global solutions can be thought about. The next subsection details some recommendations. They might not cover all the problems of RE, but still already provide some hints for good design practices at low cost.

4.1 Leaving No Secret in the Weakest Link

In some communication systems, one party is weaker than the other. For instance, in the case of a smartcard talking to a terminal, the smartcard is the easiest party to attack. Indeed, it is depends on the environment for its power supply and time reference, and it is cost-constrained.

Thus, it shall contain as few secrets as possible. This is indeed possible using public-key cryptography [9], for instance. The terminal generates a signature, and the smartcard verifies it. The verification requires no secret, thus there is no reason to attack the smartcard. This technique allows typically to deny virused code (*e.g.* containing a Trojan) to be executed on the smartcard.

4.2 Providing a Minimalist Application Programming Interface (API)

The more rich the API, the more attack scenarios an attacker can write. At the opposite, if the API is minimalist, most of the operations are realized internally, at an unknown pace. They leave fewer control for the attacker to understand what the circuit is actually computing.

4.3 Randomizing the Protocols

If the protocols are deterministic, then they can be replayed, which enables some training. Also, differential attacks (such as DFA [4]) require a correct and at least one [28] faulty cryptogram. This approach is similar to resilient countermeasures: the secret is made volatile [17], as well as the data [15]. We emphasize that for this class of countermeasure to be efficient, a tamper-resistant TRNG (True Random Number Generator) shall be available.

5 Conclusions

Fighting RE can be done by dedicated means that have been widely discussed in the scientific literature. Nonetheless, they are costly in general, because they consider a very powerful attacker that is able to play with the target at her will. The solutions we sketch in this article do not have this drawback. They are designed not to give the opportunity for the attacker to be in a favourable position, for instance by ascertaining the attack is impossible to setup. The resulting countermeasures, being only "organizational", are thus also low cost, hence acceptable in an industrial context. Nonetheless, they require to define new specific high-level communication protocols that are low-level security oriented.

Acknowledgments

This work has been partially conducted under the framework of the **MARSHAL+** (*Mechanisms Against Reverse-engineering for Secure Hardware and Algorithms*) research project, subsidized by FUI #12, and co-sponsored by the competitivity clusters System@tic (IdF region) and SCS (PACA region).

References

1. Common Criteria (*aka* CC) for Information Technology Security Evaluation (ISO/IEC 15408). Website: http://www.commoncriteriaportal.org/.
2. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.
3. Eli Biham, Yaniv Carmeli, and Adi Shamir. Bug attacks. In *CRYPTO*, volume 5157 of *LNCS*, pages 221–240. Springer, 2008. Santa Barbara, CA, USA.
4. Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO*, volume 1294 of *LNCS*, pages 513–525. Springer, August 1997. Santa Barbara, California, USA. DOI: 10.1007/BFb0052259.
5. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Proceedings of Eurocrypt'97*, volume 1233 of *LNCS*, pages 37–51. Springer, May 11-15 1997. Konstanz, Germany.

6. Éric Brier, Helena Handschuh, and Christophe Tymen. Fast Primitives for Internal Data Scrambling in Tamper Resistant Hardware. In *CHES*, volume 2162 of *LNCS*, pages 16–27. Springer, May 14-16 2001. Paris, France.
7. Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *LNCS*, pages 250–270. Springer, 2002.
8. Christophe Clavier. An Improved SCARE Cryptanalysis Against a Secret A3/A8 GSM Algorithm. In *ICISS*, volume 4812 of *LNCS*, pages 143–155. Springer, 2007. Delhi, India. DOI: 10.1007/978-3-540-77086-2_11.
9. Whitfield Diffie and Martin Edward Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.
10. Reouven Elbaz, David Champagne, Catherine H. Gebotys, Ruby B. Lee, Nachiketh R. Potlapally, and Lionel Torres. Hardware Mechanisms for Memory Authentication: A Survey of Existing Techniques and Engines. *Transactions on Computational Science*, 4:1–22, 2009.
11. Simson Garfinkel. Anti-Forensics: Techniques, Detection and Countermeasures. In *ICIW, 2nd International Conference on i-Warfare and Security*, pages 77–84, 8-9 March 2007. Naval Postgraduate School, Monterey, California, USA.
12. Sylvain Guilley, Jean-Luc Danger, Robert Nguyen, and Philippe Nguyen. System-Level Methods to Prevent Reverse-Engineering, Cloning, and Trojan Insertion. In Sumeet Dua, Aryya Gangopadhyay, Parimala Thulasiraman, Umberto Straccia, Michael A. Shepherd, and Benno Stein, editors, *ICISTM (PPREW workshop)*, volume 285 of *Communications in Computer and Information Science*, pages 433–438. Springer, 2012.
13. Sylvain Guilley, Florent Flament, Yves Mathieu, and Renaud Pacalet. Security Evaluation of a Balanced Quasi-Delay Insensitive Library. In *DCIS*, Grenoble, France, nov 2008. IEEE. Session 5D – Reliable and Secure Architectures, ISBN: 978-2-84813-124-5. http://hal.archives-ouvertes.fr/hal-00283405/en/.
14. Sylvain Guilley, Laurent Sauvage, Jean-Luc Danger, and Nidhal Selmane. Fault Injection Resilience. In *FDTC*, pages 51–65. IEEE Computer Society, August 21 2010. Santa Barbara, CA, USA. DOI: 10.1109/FDTC.2010.15; Complete version: http://hal.archives-ouvertes.fr/hal-00482194/en/.
15. Sylvain Guilley, Laurent Sauvage, Jean-Luc Danger, and Nidhal Selmane. Fault Injection Resilience. In *FDTC*, pages 51–65. IEEE Computer Society, August 21 2010. Santa Barbara, CA, USA. DOI: 10.1109/FDTC.2010.15.
16. Sylvain Guilley, Laurent Sauvage, Julien Micolod, Denis Réal, and Frédéric Valette. Defeating Any Secret Cryptography with SCARE Attacks. In *LatinCrypt*, volume 6212 of *LNCS*, pages 273–293. Springer, August 8-11 2010. Puebla, México, DOI: 10.1007/978-3-642-14712-8_17.
17. Paul C. Kocher. Leak-resistant cryptographic indexed key update, March 25 2003. United States Patent 6,539,092 filed on July 2nd, 1999 at San Francisco, CA, USA.
18. Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 382–395. Springer, September 6–9 2009. Lausanne, Switzerland.
19. Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks — Extracting Keys from Xilinx Virtex-II FPGAs. Cryptology ePrint Archive, Report 2011/390, 2011. http://eprint.iacr.org/2011/390/.

20. Amir Moradi, Markus Kasper, and Christof Paar. On the Portability of Side-Channel Attacks — An Analysis of the Xilinx Virtex 4 and Virtex 5 Bitstream Encryption Mechanism. Cryptology ePrint Archive, Report 2011/391, 2011. http://eprint.iacr.org/2011/391/.
21. Karsten Nohl, David Evans Starbug, and Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, pages 185–193, July 31 2008. San Jose, CA, USA.
22. Karsten Nohl, Erik Tews, and Ralf-Philipp Weinmann. Cryptanalysis of the DECT Standard Cipher. In *FSE*, Lecture Notes in Computer Science. Springer, February 7-10 2010. Seoul, South Korea.
23. Manuel San Pedro, Soos Mate, and Sylvain Guilley. FIRE: Fault Injection for Reverse Engineering. In *WISTP*, volume 6633 of *LNCS*, pages 280–293. Springer, June 1-3 2011. Heraklion, Greece. DOI: 10.1007/978-3-642-21040-2_20.
24. Martin Schobert. GNU software DEGATE. Webpage: http://www.degate.org/.
25. Sergei P. Skorobogatov. *Semi-Invasive Attacks — A new approach to hardware security analysis*. PhD thesis, Cambridge University / Computer Laboratory, Security Group, TAMPER laboratory, April 2005. Technical Report UCAM-CL-TR-630, http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf.
26. G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14, 2007.
27. Randy Torrance and Dick James. The State-of-the-Art in IC Reverse Engineering. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 363–381. Springer, September 6-9 2009. Lausanne, Switzerland.
28. Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In Claudio Agostino Ardagna and Jianying Zhou, editors, *WISTP*, volume 6633 of *Lecture Notes in Computer Science*, pages 224–233. Springer, 2011.