

Compromising device security via NVM controller vulnerability

Sergei Skorobogatov
Dept of Computer Science and
Technology
University of Cambridge
Cambridge, UK
sps32@cam.ac.uk

Abstract—This paper introduces a new vulnerability found in a low-cost secure authentication IC that stores its security settings in non-volatile memory (NVM). Such devices are widely used for prevention of counterfeiting in consumable products and accessories (printer cartridges, batteries etc.) and for aftermarket control. The particular device targeted here uses hardwired application logic and lacks a microprocessor, however, more sophisticated security devices used in medical and banking applications could similarly be vulnerable. The newly discovered self-induced fault attacks exploit implications of the use of error-correction codes inside modern embedded NVM blocks and their associated control logic, which can leave the application vulnerable to early termination of NVM write operation. This could potentially be used to change the security settings of a device in a way that bypasses the intended state machine controlling access and allow reverting the stored hardware security level back to the factory test/debug mode. The paper also outlines some measures that could substantially reduce the chances of a successful attack.

Keywords—hardware security, counterfeit prevention, embedded EEPROM and Flash NVM, backdoors, fault attacks, reverse engineering

I. INTRODUCTION

Modern systems rely on hardware security to prevent all sorts of attacks [1,2]. In order to comply with these demands modern semiconductor chips are designed with hardware security in mind. This helps in designing a secure system by combining the knowledge of existing attack technologies with state-of-the-art defence technologies. Without awareness of modern attack methods it would be impossible to design an adequate secure system. Existing countermeasures could help in assisting the design process. However, for economical and convenience reasons it is not always possible to incorporate all countermeasures. Hence, the job of the knowledgeable hardware designer is to choose the right solutions. Nevertheless, some oversights in the hardware design could open new attack vectors later like it happened with modern CPUs [3,4]. This paper exposes another area of possible vulnerability that could potentially affect the security of semiconductor devices. It is related to the embedded non-volatile memory (NVM) and the way it is controlled by an on-chip logic.

Counterfeit prevention and aftermarket control has long been a concern for automotive, medical, entertainment and printing industries. In the old days this was achieved by using semiconductor devices with simple functions, like serial ID numbers or proprietary access protocols. From the late 1990s, the demand for devices with some cryptographic functionality has grown substantially. This was caused by the widespread use of low-cost attack methods [5]. Such devices started using cryptographically strong message-authentication functions

such as HMAC-SHA-1 to verify that an authentication device shares a secret key or password. In order to prevent eavesdropping attacks, the host normally sends a random challenge to the device. Then a response computed from the hash of the key and the challenge is sent back for verification. In the host, the secret key can be derived from a master key, to avoid sharing the same key between multiple devices.

Modern authentication devices are usually based on asymmetric cryptography and incorporate a wide range of countermeasures against many known attacks. That way the host device does not have to hold any secrets. Instead only a certificate's public key and verification algorithm are stored. This prevents host reverse-engineering attacks common for authentication devices based on symmetric cryptography. That is where the secret keys are either directly extracted from the host or the host is forced to calculate the derived secret keys. The communication between the host and the device is normally performed with a random challenge and randomised response to prevent eavesdropping attacks.

II. BACKGROUND

Attacks on semiconductor devices can be split into several categories. Non-invasive attacks are usually low-cost and involve observations of the device operation or manipulations of external signals. They require only moderately sophisticated equipment and knowledge to implement. They do not physically harm the chip and often leave no trace. Invasive attacks, in contrast, are expensive and require sophisticated equipment and knowledgeable attackers. However, they offer almost unlimited capabilities to extract information from chips and understand their functionality. As these attacks involve contacting the intra-chip circuitry, they always leave traces and often destroy the chip. Semi-invasive attacks fill the gap between non-invasive and invasive attacks and are more affordable to many attackers. For these attacks the chip still needs to be de-packaged, but the internal structure remains intact. Although these attacks often leave traces, in most cases the chip remains fully operational.

Tools used for carrying out non-invasive attacks are usually available at most electronics engineering labs. These include digital multimeter, IC soldering/desoldering station, universal programmer, oscilloscope, logic analyser, signal generator, power supply, PC and prototyping boards. Non-invasive attacks can be divided into side-channel attacks (timing [6], power analysis [7], emission analysis [8]), data remanence [9], data mirroring [10], fault injection (glitching [11], bumping [12]) and brute forcing [13].

Tools used for carrying out invasive attacks involve a simple chemistry lab, high-resolution optical microscope, wire bonding machine, laser cutting system, microprobing station, oscilloscope, logic analyser, signal generator, PC,

prototyping board, scanning electron microscope (SEM) and focused ion beam (FIB) workstation. Invasive attacks can be divided into sample preparation [14], imaging [15], direct memory extraction [16], reverse engineering [17], microprobing [18], fault injection [19] and chip modification [20].

Tools used for carrying out semi-invasive attacks involve a simple chemistry lab, high-resolution optical microscope, UV light source, lasers, oscilloscope, logic analyser, signal generator, PC and prototyping boards. Semi-invasive attacks can be divided into imaging [21], laser scanning [22], optical fault injection [23], optical emission analysis [24] and combined attacks [25].

Modern authentication devices usually support a product life cycle that steps through multiple levels of security protection. When the device is initialised at the factory, it typically starts in its lowest security level, which allows full access to its hardware resources. After the factory testing is passed, the device is personalised with some essential data, before being shipped to a customer (ID, slot number, access password, encryption keys etc.). This places it into the next security level, with restricted access. The customer may then program the device with customer-specific parameters, data and keys, before restricting its security level further. Then it is placed into a product and shipped to an end user. Of course, there could be plenty of variations, but one aspect remains unchanged – the hardware security protection level goes from the lowest to the highest on the way between fabrication and end user.

In the past, multiple attacks were carried out on authentication devices. Those ranged from a simple write unprotection of an EEPROM and memory corruption [5] to more sophisticated optical fault injection attacks [8,26].

Another attack vector that became successful in some cases is exploiting backdoors present in many semiconductor devices. They normally exist in the form of undocumented features implemented by hardware designers for assisting post-fabrication and factory testing. In some cases, these allow to completely circumvent the security protection and gain full access to on-chip memory and low-level hardware control features [27]. There had been some speculation about the covert nature of such insertions being hardware Trojans, however, no real proof of this has been found.

III. PREVIOUS RESEARCH

Our target device for demonstrating the new fault injection attack is the Infineon Optiga™ Trust B SLE95250 [28], which implements a very basic authentication protocol based on elliptic-curve cryptography (ECC). As there was very little information provided by the manufacturer on this device without NDA, we had to obtain all the information necessary to operate it first through the Internet search and reverse engineering its Evaluation Kit [29]. This led to our nearly full understanding of its communication protocol, which turned out to be based on a protocol by Braun/Hess/Meyer [30], and the discovery of the backdoor that allowed full access to the on-chip NVM and successful extraction of secrets [31].

While we were able to order SLE95250 engineering samples from several distributors, availability of the Evaluation Kit was limited to only a few. The information on the Infineon website about Optiga™ Trust B products [32] was limited to a brief public datasheet on SLE95250 [33] with

very limited information – primarily package layout, pinout and connections, electrical characteristics. It only mentioned that the device communicates via a proprietary “SWI” single-wire interface, without any information on it at all. Neither the modes of operation were described, nor the authentication protocol or ECC parameters, apart from them involving a 131-bit engine for authentication and 163-bit certificates. Essential information about communication waveforms, bit encoding, protocols and commands, NVM access, usage of Life Span counter, signature verification and authentication details including the Message Authentication Code (MAC) function was not publically available. However, some searches over the Internet revealed one Infineon’s patent [34] and an IEC62700-committee standards proposal [35] that shed some light on the SWI interface and the basic communication protocol, which turned out to be related to the physical interface used in the MIPI Battery Interface (BIF) specification.

A. Reverse Engineering of the Evaluation Kit

The Optiga™ Trust B Evaluation Kit [29] is a USB dongle that comes with Windows GUI software. Not much information was provided on the hardware, not even a simple circuit diagram. Nevertheless, it did not take much time to eavesdrop on the communication between the SLE95250 device and the main microcontroller of the kit, an Infineon XMC4500. We reconstructed the full circuit diagram of the kit using computed-tomography slices prepared with an X-ray microscope. This helped us to trace the connections with the USB chip and debug points. Finally, we reconstructed the complete communication from USB down to Optiga™ Trust B device with the help of a logic analyser and an improvised SWI-to-UART converter (Figure 1).

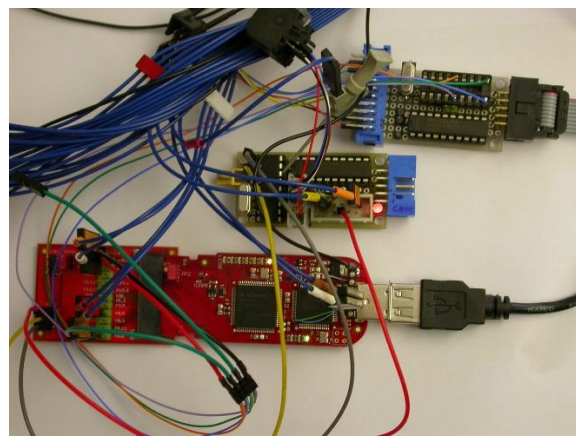


Fig. 1. Eavesdropping on the internal communication in the Evaluation Kit.

The firmware of the Evaluation Kit stored inside XMC4500 microcontroller was not read protected. It was possible to download the whole image of the internal Flash using a standard ARM debugger (J-Link) via the single-wire debug (SWD) interface. We then decompiled and annotated the firmware into readable C code using Ghidra [36]. A similar approach was attempted to the GUI program using a .NET decompiler, however, it turned out that the authentication algorithm is performed entirely inside the XMC4500 and the Windows software is only used for visualisation. Still, this exercise revealed the names of the functions called via the USB interface and helped to better understand the firmware. Finally, we implemented the SWI communication with the SLE95250 on an XMC4500 Relax Lite Kit [37], and the ECC authentication in Python.

B. SWI communication and NVM access

SWI communication is performed through commands. Most operations are performed by accessing registers. Some of those registers act as a communication buffer, some initiate NVM operations or decrement a Life Span counter. Table 1 summarises commonly used commands and operations.

TABLE I. SWI COMMANDS AND OPERATIONS

Operation	SWI communication		
	Command	Address	Data
Reset	800	—	—
Read 1 byte	820, 850	5HH, 7LL	7XX
Write 1 byte	820, 850	5HH, 6LL	7XX
Register read	820, 851	5HH, 7LL	7XX
Register write	820, 851	5HH, 6LL	7XX
ID search	830, 83X, 83Y	—	—
ID select	830, 83Y	—	—
ECC run	8C1	—	—
Status query	810	—	—
Select device	9HH, ALL	—	—
NVM read	820, 851	502, 674	4LL
	820, 851	502, 672	4HH
	820, 851	5XX, 7YY	7XX
NVM write	820, 851	5XX, 7YY	4XX
	820, 851	502, 674	4LL
	820, 851	502, 672	4HH
Life Span Counter decr	820, 851	502, 674	420
	820, 851	502, 672	489

NVM read and write operations can only access 256 bytes of memory, as the address is specified by 8 bits in SWI registers. However, 64 bytes within the NVM are read protected and writing is only allowed into a user area of 104 bytes. The Public key and certificate can be read, but are write protected. Life Span counter can be overwritten in engineering samples but can be made write protected using register 0x26F.

Our initial assumption was that the Private key is stored somewhere within the read protected area of the NVM. However, in order to access that area the hardware security protection of the chip would have to be circumvented.

C. Optical Fault injection attack

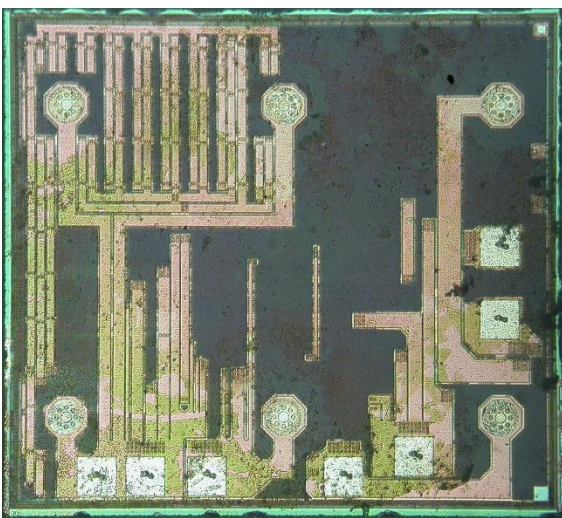


Fig. 2. Optical image of the die.

Our first successful attack performed on the chip was via laser fault injection. However, prior to that, the location of the NVM array and a way to deliver the laser beam had to be

found. Figure 2 shows an optical image of decapsulated die under a microscope. Fabricated with an advanced process, it leaves no visible gaps passing through the metal layers, hence, the only practical approach for laser attacks is from the rear side, through the bulk silicon.

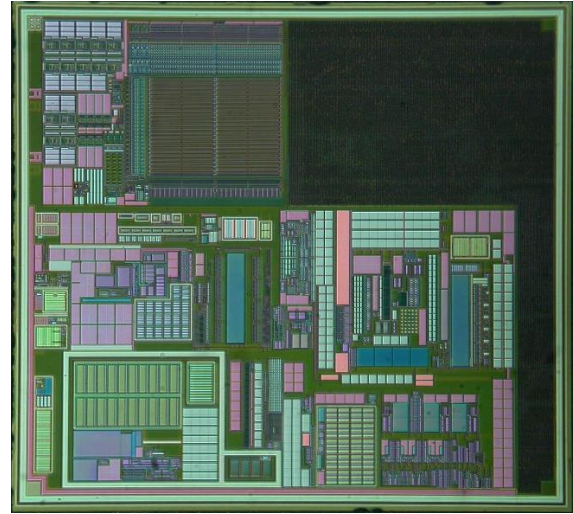


Fig. 3. Optical image of the rear-side deprocessed die.

Figure 3 shows an optical image of rear-side of a deprocessed die under a microscope. The NVM area at the top left corner is easy to spot by its regular structure. However, optical fault injection attacks require a fully operational device. It is quite challenging to open a 1.5mm × 1.1mm device while keeping it still working. Therefore, the package was first reinforced with epoxy before polishing the silicon substrate (Figure 4).

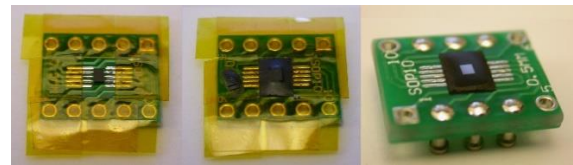


Fig. 4. Sample preparation for optical fault injection attacks.

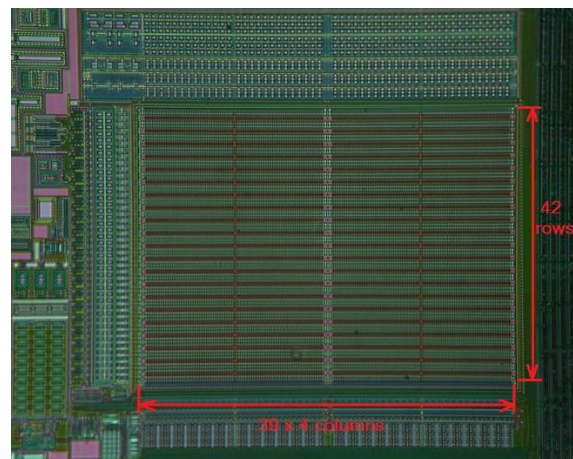


Fig. 5. Optical image of the NVM physical array.

Focusing a 1064nm laser with more than 40mW power at the NVM region caused temporary data corruption in a corresponding memory region. However, the chip retrieved its security settings from NVM only during power-on reset. By performing an exhaustive search, it was possible to find the

correct location and laser settings to cause the NVM read protection to be disabled. As a result, all 256 bytes of NVM became readable. However, there were no interesting secrets in there – just device ID, ECC curve parameter and some constants. At the same time a closer look at the NVM array under a microscope revealed that it contains 6552 bits of information – far more than the 256 bytes accessible in the known way (Figure 5). Therefore, a new way of obtaining full access to the stored data was required, such as exploiting a backdoor implemented for factory testing and debugging purposes.

D. Quest for Backdoors to gain full NVM access

The process of finding the registers that could assist with extended NVM access started with scanning the whole register space to identify all readable and writable registers. However, none of them appeared to behave similar to NVM related registers, apart from the already known ones. Then the same scanning operation was performed with disabled security. This revealed a small number of registers which became active after disabling the security. Fuzzing those registers identified two interesting candidates. Register 0x266 was behaving similar to the known register 0x274 in respect of initiating NVM read and write operations. Register 0x264 was setting some parameters for NVM access. Unlike previously known NVM access commands those extended NVM access operations applied to a whole row in the physical NVM array. That is 156 bits at a time – 4 sets of 32-bit data with 7-bit error correction. The full address space of 1024 bytes was limited to 672 bytes of real data physically present on the die. Error correction bits were not only always readable, but were writable when permitted by settings in register 0x270.

Although it was now possible to read all 6552 bits of the extended NVM array, this was still not enough to extract the Private key. The memory below address 0x200 contained already known information, but everything above appeared to be encrypted.

E. Memory Decryption and Private Key extraction

It turned out that another set of registers needed to be identified and correctly set before the readout from the extended NVM appeared unencrypted. The scrambling key turned out to be only 8 bits long and was stored at address 0xA0 in normal NVM, though read protected by default. We verified the correctness of the 131-bit elliptic-curve Private key that became readable that way by multiplying it with the base point. The result matched the Public key of the device.

The error correction codes turned out to be standard Hamming codes and were reconstructed by programming the memory with a single-bit-set test pattern. The memory encryption function appeared to merely XOR memory content function with a 64-bit key-dependant table repeated over the whole range. In addition to data scrambling the chip also applied address permutation, however, this did not affect the error correction bits at all. Therefore, it was relatively easy to find the correct address by matching the 16-byte data blocks to the sequence of four-byte error correction words.

If we modified the Private key, this blocked ECC computation is not performed. It turned out that a 32-bit CRC value of the key is stored inside the extended NVM and must match the computed one. However, any Private key with correct CRC can be programmed into any device and will be

accepted. The CRC was easy to identify as a linear function, i.e. $Key_1 \text{ xor } Key_2 = Key_3$ implies $CRC_1 \text{ xor } CRC_2 = CRC_3$,

Table 2 shows the data extracted from one compromised device.

TABLE II. DATA FROM ONE COMPROMISED DEVICE

Parameter	Value
curve	$y^2 + xy = x^3 + ax^2 + b$
\sqrt{b}	00e4808f8949d33c69e070a5f82c3633d9
$f(x)$	$x^{131} + x^8 + x^3 + x^2 + 1$
G_x (base point)	03651a4282ae22c4fc6c20c9b7281ec1f5
Q_x (Public key)	06d046e3bf7bb34479bd3aad1301f14cbd
Q_x^* (padded Q_x)	1dd6d046e3bf7bb34479bd3aad1301f14cbd
Device ID	07203c0210c4981a8d68
Signature R	001c8f15507787ba50c293427d0794f447e899c150
Signature S	0016733472325207c535908434ac0563548dbaa1d
q (Private key)	d861429f79fed9f8090ae83df804970
q* (padded q)	0d861429f79fed9f8090ae83df804970

The above-mentioned attack, like most semi-invasive attacks, has certain limitations and is unlikely to present a real threat to the security of real-world systems using the Optiga™ Trust B. Firstly, laser fault injection attacks require specialised and expensive equipment. Secondly, samples must be prepared in a special way to allow the laser beam to be focused from the rear side of the chip. Finally, the chip must still be fully operational. Achieving reliable sample preparation that involves substrate thinning while maintaining full functionality of the chip is a quite challenging, tedious and time-consuming process, especially for such a small device with a silicon die area of less than 1mm². Since each device holds a unique Private/Public key pair an attacker will have to extract them from hundreds or even thousands of samples to avoid key revocation or ID blocking. Therefore, a more efficient and reliable non-invasive attack is required in order to become the real threat to the security of Optiga™ Trust B devices.

IV. FINDINGS AND IMPLEMENTATION

The embedded NVM array is the only reprogrammable non-volatile data storage present on the SLE95250 die. This means that any changes in the security settings will be managed by the same hardware control logic. The reason why many chip manufacturers go for a single physical NVM is the size penalty for embedding extra physical arrays. Cost-optimised sub-mm² chips usually share the same NVM array for multiple purposes: device ID or serial number, constants, factory data, user data, secret key or password. Theoretically, faults induced into the chip operation could force it to write into a wrong area and influence the value of the security-control settings.

A. NVM challenges

Many modern chips are built with deep submicron fabrication process. As the memory cells shrink, their reliability, data retention time and maximum number of cycles become worse. In order to improve fabrication yield, prolong the device lifetime and improve characteristics, chip manufacturers choose NVM blocks with error correction. Very often this feature comes as a complete, independent and proprietary design block. As a result, even the low-level hardware design engineers do not have any control over the memory operation and only manipulate input and output connections. Hence, changes between any intermediate states for each memory row during write operation are determined by the memory-block IP designer at a fab, rather than the

ASIC designer. The same applies to the handling of error correction functions and read-modify-write buffers. Neither can the timing of the internal operations be controlled by the chip designer, and often is fixed in the memory-block library provided by the fab.

One good practice in the design of security-sensitive devices can be to only allow changes in security level in one direction – from lowest to highest. In older devices, based on classic EEPROM and Flash arrays, this was achieved by permitting bit changes only in one direction – from erased to programmed state. However, the introduction of error correction thwarted the inherent one-way-programming nature of NVM cells and became embedded into the low-level design of the NVM controller logic. Chip designers who rely on fabless manufacturing or outsourcing fabrication often cannot control such internal functionality of the NVM controller.

B. Fault injection during NVM writing

In order to understand the process of memory writing, we supplied power to the SLE95250 target device directly from an I/O pin of our microcontroller test board, to be able to precisely power down the target, synchronized with events on the SWI bus. The result of a memory write operation terminated after different time intervals is presented in Table 3. In this example, the write operation overwrites an initial value of 5Ah with A5h value. As the table shows, changes to that byte do not happen simultaneously for all bits. Also, there is a 750 μ s long delay between the apparent completion of erasure and the first signs of the programming cycle.

TABLE III. NVM WRITE RESULT AFTER DELAY

t μ s	0	58	77	78	79	830	831	832	833	999
N	5A	5A	7B	FB	FF	FF	F7	A7	A5	A5

The embedded memory write operation can be terminated by switching off the power supply of the chip and forcing it to GND level. By controlling the delay time, it is possible to perform the change bit by bit, until the desired bit is affected. Both EEPROM and Flash memory store the data in the form of a charge inside their cells. If the normal write operation is terminated early, there is a chance that internal charge will stay at an intermediate level. Then it would be possible to influence the values read from of the memory by changing the power supply voltage [38]. The overall time required for disabling the security protection in an OptigaTM Trust B device this way is less than 0.1 seconds.

C. Self-induced fault injection

The presence of other protection features could also affect the susceptibility to the above type of fault injection attacks making them easier to perform. For example, in the SLE95250 both the security-protection register 0x26F and user-NVM write-protection register 0x275 are physically located in the same row of the NVM array. This increases the number of attempts to find the correct fault injection settings from 4 (bits in 0x26F register) to 12 (bits in 0x26F and 0x275 registers).

Although the introduced attack, with fault injection into the power supply line during NVM write, is completely non-invasive and very fast, it still requires some additional preparation time. At least physical access to the chip's power supply line is required. If the chip is used in a configuration where the power supply is delivered through the SWI interface

line, this can present an additional challenge, requiring desoldering of the chip and placing it onto a test bench. Hence, the attack time could be much longer than the actual 0.1 seconds required to amend the security register.

A more powerful alternative was found through exploiting a feature in control register 0x270. Apart from enabling the Life Span counter, NVM bulk 0s writing and error correction overwriting it also has a bit that terminates the device operation. This acts as some kind of a soft kill switch. The effect from it is very similar to the one achieved by switching off the power supply to the device. Hence, the already very successful NVM-write-terminate attack was improved into a purely software-controlled fault attack, by aborting the write operation with the help of this soft-kill switch feature. Yet another backdoor on the device.

V. CONCLUSION

The attacks presented above are capable of inducing faults into the NVM memory through exploiting the overwriting mechanism. The process of updating the memory content is internally controlled by logic hardwired into silicon. This is imposed by the use of error-correction logic. It does not allow to change a single bit of memory directly from erased to programmed state without rewriting the whole memory row to also update the error-correction bits. Such a row consists in case of the SLE95250 of 128 data bits and 28 error-correction bits. Even a single-bit change operation starts with fetching the row into hardware buffer, modifying it there, then erasing the row followed by writing the modified data from the buffer back into it. If this flow of events is terminated early, either during the row-erase or row-write operation, it results in incorrect data stored in NVM. In some cases, the memory content became unstable and gave a slightly different reading each time. This can be further exploited in a controllable way due to a power-supply voltage dependency of these fluctuations. As a demonstration of the power of this attack we cloned one OptigaTM Trust B chip with user data completely into a blank chip in less than 1 second. The two chips were indistinguishable and shared the same Device ID, Private key, Certificate, Security settings and user data. Both devices successfully passed the authentication-protocol challenge in the Evaluation Kit.

There are several mitigation techniques against fault attacks. Firstly, the physical design and functionality of the NVM block could be reviewed. Even if it was provided as a black box, it would still be possible to reverse engineer and simulate its operations. Secondly, the actual security register could rely on data from multiple memory rows, thus minimising the risk of errors and faults. Third, the process of changing the security from one level to another could be made more robust with extra redundancy and CRC checks.

A serious new hardware-security vulnerability was revealed by the research described in this paper. Formal security evaluation is unlikely to spot such flaws due to the restricted nature of low-level IP blocks. Unless appropriate testing is introduced modern embedded NVM blocks could pose a serious threat to the security of semiconductor devices.

ACKNOWLEDGMENT

I would like to thank my colleagues Markus G. Kuhn and Shih-Chun You for their many suggestions and help in understanding the Infineon OptigaTM Trust B devices and the cryptography used during their authentication process.

REFERENCES

- [1] F. Rahman, M. Farmani, M. Tehranipoor, and Y. Jin, "Hardware-assisted Cybersecurity for IoT Devices," 18th International Workshop on Microprocessor and SOC Test and Verification, Austin, USA, December 2017
- [2] S. Skorobogatov, "Physical Attacks and Tamper Resistance," Chapter 7 in *Introduction to Hardware Security and Trust*, Eds: Mohammad Tehranipoor and Cliff Wang, Springer, September 2011
- [3] M. Lipp et al., "Meltdown: Reading Kernel Memory from User Space," in *USENIX Security Symposium*, 2018
- [4] P. Kocher et al., "Spectre Attacks: Exploiting Speculative Execution," 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019, pp. 1-19
- [5] R. Anderson, M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," Security Protocol Workshop, April 1997
- [6] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *Advances in Cryptology: Proceedings of CRYPTO'96*, Springer-Verlag, August 1996
- [7] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," In M. Wiener, editor, *Advances in Cryptology-crypto'99*, vol. 1666 of *Lecture Notes in Computer Science*, Springer-Verlag, 1999
- [8] S. Skorobogatov, "Semi-invasive attacks – A new approach to hardware security analysis," Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005
- [9] S. Skorobogatov, "Data Remanence in Flash Memory Devices," *Cryptographic Hardware and Embedded Systems Workshop (CHES-2005)*, August-September 2005, LNCS 3659, Springer
- [10] S. Skorobogatov, "The bumpy road towards iPhone 5c NAND mirroring," arXiv:1609.04327, September 2016
- [11] K. Gandolfi, C. Mourtel and F. Olivier, "Electromagnetic Attacks: Concrete Results," In *Proc. Workshop on Cryptographic Hardware and Embedded Systems*, Paris, France, May 2001
- [12] S. Skorobogatov, "Flash Memory 'Bumping' Attacks," *CHES-2010*, August 2010, LNCS 6225, Springer
- [13] C. Paar, J. Pelzl, and B. Preneel, "Understanding Cryptography: A Textbook for Students and Practitioners," Springer 2010
- [14] F. Beck, "Integrated Circuit Failure Analysis : A Guide to Preparation Techniques," John Wiley and Sons Ltd, March 1998
- [15] F. Courbon, S. Skorobogatov, and C. Woods, "Reverse engineering flash EEPROM memories using scanning electron microscopy," In: Lemke-Rust, K., Tunstall, M. (eds.) *CARDIS 2016*. LNCS, vol. 10146
- [16] S. Skorobogatov, "Deep dip teardown of tubeless insulin pump," arXiv:1709.06026, September 2017
- [17] R. Torrance, and D. James, "The State-of-the-Art in IC Reverse Engineering," *CHES 2009*, LNCS, vol. 5747, Springer, Heidelberg, 2009
- [18] S. Skorobogatov, "How microprobing can attack encrypted memory," In *Proceedings of Euromicro Conference on Digital System Design, AHSA 2017 Special Session*, Vienna, Austria. IEEE Computer Society, 2017
- [19] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert, "Fault attacks on RSA with CRT: concrete results and practical countermeasures," *CHES 2002*, CA, USA, August 2002
- [20] O. Kömmerling, and M.G. Kuhn, "Design principles for tamper-resistant smartcard processors," *USENIX Workshop on Smartcard Technology*, Chicago, Illinois, USA, May 1999
- [21] S. Skorobogatov, "Video Imaging of Silicon Chips," University of Cambridge, poster, March 2004
- [22] K. Ueda, "Backside OBIC Scanner," *LSI Testing Symposium in Japan*, 1996
- [23] S. Skorobogatov, "Optical Fault Induction Attacks," *CHES-2002*, August 2002, LNCS 2523, Springer-Verlag
- [24] S. Skorobogatov, "Using Optical Emission Analysis for Estimating Contribution to Power Analysis," *FDTC 2009*, September 2009, Lausanne, Switzerland. IEEE-CS Press
- [25] S. Skorobogatov, "Optically Enhanced Position-Locked Power Analysis," *CHES-2006*, October 2006, LNCS 4249, Springer
- [26] S. Skorobogatov, "Optical Fault Masking Attacks," 7th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2010), 21 August 2010, Santa Barbara, USA. IEEE-CS Press, ISBN 978-0-7695-4169-3, pp.23-29
- [27] S. Skorobogatov, C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," *Cryptographic Hardware and Embedded Systems Workshop (CHES-2012)*, 9-12 September 2012, Leuven, Belgium, LNCS 7428, Springer, ISBN 978-3-642-33026-1, pp.23-40
- [28] OPTIGA™ Trust B SLE95250: Authentication solution for improved security and reduced system costs, Product Brief, Infineon 2017
- [29] OPTIGA™ Trust B SLE95250 Evaluation Kit User Guide, Infineon 2017. https://www.mouser.co.uk/datasheet/2/196/Infineon-OPTIGA_Trust_B_SLE95250_Evaluation_Kit_Us-1379946.pdf
- [30] M. Braun, E. Hess, B. Meyer, "Using Elliptic Curves on RFID Tags," *IJCSNS International Journal of Computer Science and Network Security*, 8(2):1-9, 2008.
- [31] *HardwearIO 2020 Netherland*, Online Hardware Security Conference. <https://hardwear.io/netherlands-2020/>
- [32] OPTIGA™ Trust Products, Infineon 2020. <https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-trust/>
- [33] SLE95250 OPTIGA™ Trust B Authentication IC, Datasheet Rev.1.01, Infineon 2017. <https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-trust/optiga-trust-b-sle-95250/>
- [34] Electronic system and method for sending or receiving a signal, Patent US7636806, Infineon Technologies AG, 2007
- [35] Wolfgang Furtner, "Proposal for IEC 62700 Identification and Communication Method for Notebook Computer supporting Class 1-3 ID," Infineon Technologies AG, 2019. http://www.y-adagio.com/public/committees/iec_tc100_agts/meetings/35/100ags566.pdf
- [36] Ghidra: A software reverse engineering (SRE) suite of tools developed by NSA's Research Directorate in support of the Cybersecurity mission. <https://ghidra-sre.org/>
- [37] Infineon XMC4500 Relax Lite Kit, Infineon, 2014. <https://www.infineon.com/cms/en/product/evaluation-boards/>
- [38] S. Skorobogatov, "Data Remanence in Flash Memory Devices," *Cryptographic Hardware and Embedded Systems Workshop (CHES-2005)*, 30 August - 1 September 2005, LNCS 3659, Springer, ISBN 3-540-28474-5, pp.339-353