A Secure Hardware-Software Solution Based on RISC-V, Logic Locking and Microkernel

Dominik Šišejković
RWTH Aachen University, Germany
sisejkovic@ice.rwth-aachen.de

Farhad Merchant
RWTH Aachen University, Germany
merchantf@ice.rwth-aachen.de

Lennart M. Reimann
RWTH Aachen University, Germany
reimannl@ice.rwth-aachen.de

Rainer Leupers
RWTH Aachen University, Germany
leupers@ice.rwth-aachen.de

Massimiliano Giacometti
Hensoldt Cyber GmbH, Germany
massimiliano.giacometti@hensoldt-cyber.com

Sascha Kegreiß
Hensoldt Cyber GmbH, Germany
sascha.kegreiss@hensoldt-cyber.com

ABSTRACT

In this paper we present the first generation of a secure platform developed by following a security-by-design approach. The security of the platform is built on top of two pillars: a secured hardware design flow and a secure microkernel. The hardware design is protected against the insertion of hardware Trojans during the production phase through netlist obfuscation provided by logic locking. The software stack is based on a trustworthy and verified microkernel. Moreover, the system is expected to work in an environment which does not allow physical access to the device. Therefore, on-the-field attacks are only possible via software. We present a solution whose security has been achieved by relying on simple and open hardware and software solutions, namely a RISC-V processor core, open-source peripherals and an seL4-based operating system.

CCS CONCEPTS

• **Security and privacy → Security in hardware**; **Embedded systems security**;

KEYWORDS

RISC-V, secure OS, hardware security, logic locking, seL4

ACM Reference Format:
Dominik Šišejković, Farhad Merchant, Lennart M. Reimann, Rainer Leupers, Massimiliano Giacometti, and Sascha Kegreiß. 2020. A Secure Hardware-Software Solution Based on RISC-V, Logic Locking and Microkernel. In *23rd International Workshop on Software and Compilers for Embedded Systems (SCOPES '20), May 25–26, 2020, Sankt Goar, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3378678.3391886

1 INTRODUCTION

Nowadays, being connected everywhere and all the time has become the norm. Consequently, cyber crime has emerged as a flourishing activity. Security breaches increased by 67% from 2014 to 2018 [1], resulting in an average cost of $3.92 million in 2019 [8]; the total damage is projected to reach $6 trillion by 2021 [13], despite the budget for cyber security increased by 141% between 2010 and 2018, reaching $124 billion worldwide in 2019 [17]. These numbers demonstrate the need to rethink the traditional approach followed in cyber security. As acknowledged by the US Defense Advanced Research Projects Agency [14], relying solely on a "patch and pray" approach cannot be the answer. Add-on solutions to security flaws can only help curing the symptoms after they arise, but they do not really fix the underlying vulnerability, and leave untouched weaknesses that are still waiting to be discovered by a clever attacker.

The need to protect against cyber attacks does not start the moment the system becomes operational, as the hardware can be compromised also prior or during manufacturing. The integrated circuit (IC) flow is mainly relying on outsourcing fabrication and design services as well as the active usage of third party intellectual property (IP). This business model ensures cost efficiency and short time-to-market, thereby facilitating the competitiveness of semiconductor companies. Nonetheless, this model introduces severe security back doors, as it enables untrusted third parties to have full access to a particular hardware component during the design and production cycle. This has led to a diverse palette of security concerns, ranging from malicious hardware modifications (known as hardware Trojans) to IP piracy [3, 21]. One promising technology for ensuring the integrity of hardware designs is known as logic locking [2, 5]. This scheme introduces functional and structural changes to a gate-level netlist, thus binding the correct functionality of the design to a secret key. The induced obfuscation conceals the design while passing through the hands of third parties, therefore mitigating the insertion of malicious modifications.

Contribution: Under the motto "Secure IT, not IT security", we present the first generation of our platform that provides a trusted computing base for security critical applications. Hereby, our efforts are directed at enforcing security by design of both hardware (HW) and software (SW). The HW platform is based on a trustworthy implementation, whose (uncontrollable) production chain has been protected against the insertion of hardware Trojans through the extensive application of functional and structural hardware obfuscation introduced by logic locking. On top of the protected HW runs an operating system based on the trustworthy, mathematically proven seL4 microkernel.

SCOPES '20, May 25–26, 2020, Sankt Goar, Germany

ACM ISBN 978-1-4503-7131-5/20/05…$15.00
https://doi.org/10.1145/3378678.3391886

The rest of the paper is organized as follows. In Section 2, we present the background knowledge regarding HW and SW security topics relevant for this paper. Section 3 depicts the strategies followed during hardware development and the concepts behind the implementation of a suitable operating system. Finally, Section 4 concludes the paper and discusses future activities to test the security solutions in place.

2 PRELIMINARIES

2.1 Logic Locking

Logic locking typically operates on a gate-level netlist by introducing a set of key-controlled gates known as key gates [18]. On the application of a correct activation key, the key gates ensure that the obfuscated design functions as intended. Otherwise, an incorrect key implies incorrect and faulty functional behavior. To understand the principle, let us consider the random logic locking scheme known as EPIC [16]. This scheme disseminates XOR and XNOR (XOR + inverter) gates at random locations in the design. An XOR gate buffers its second input when its first input is set to a fixed value of 0. Similarly, XNOR gates buffer for a fixed input value 1. In case the values are inverted, the gates will act as inverters for their second input. With this property, the disseminated key gates preserve the original functionality only for a correct key. Otherwise, incorrect functional behavior is induced.

HW Threat Model: Binding the correct functionality of a HW design to a specific key protects against the insertion of hardware Trojans by third parties, e.g., external design houses and the foundry. The assumption is that a successful hardware Trojan insertion is only viable if the adversary is able to fully reverse engineer and understand the design at hand. Only then, a powerful, stealthy and targeted hardware Trojan attack can be implemented, bypassing any verification and testing procedures. If logic locking is implied, the first step towards unlocking the design is finding the correct activation key. Typically, it is assumed that the adversary has access to an instance of the obfuscated netlist as well as to an activated IC to use as an oracle (acquired from the semiconductor market). However, recent advances have shown that it is possible to gain access to the key even in the presence of a tamper-proof memory, without having to attack the locking scheme through oracle-guided attacks [9, 15]. Therefore, we consider the security implications of logic locking only in the first production round, where no activated IC with the identical locking scheme and activation key exists on the market.

2.2 Microkernels and Security

A monolithic kernel is an operating system architecture where all the functionalities (process, memory and device management, file system, networking and others) are running in kernel space. Examples of this architecture are Linux, Unix and MS-DOS. The code base of this kind of systems is typically large given all the features that must be provided. Consequently, this results in a high number of bugs (estimates vary from 0.5 to 6 bugs per kLOC [4]).

On the other hand, microkernels implement the bare minimum of software required to develop an operating system. All additional services (file system, networking, device drivers and others) run in user space. In terms of code size, microkernels are much leaner than monolithic ones (10kLOC in seL4 against the tens of MLOC of Linux), resulting in a dramatically lower number of potential attack points.

SW Threat Model: In a monolithic OS, the security of an application depends on the security of each of the services running in kernel space: therefore the trusted computing base (TCB) is constituted by the whole (huge, thus highly insecure) kernel. This is not the case when using microkernels, where most of the OS services are running in user space: the TCB is thus minimized, being composed only of the application's required services and the lean kernel. The effectiveness of microkernel-based systems has been demonstrated by the Secure Mathematically-Assured Composition of Control Models (SMACCM) project [6]. An attacker has been able to exploit a vulnerability in the Linux VM (running on an unverified kernel) to get control of the system's critical assets; the same attack on an equivalent seL4-based system has not been possible: the effects of the attack remained confined within the VM, thanks to the isolation provided by the microkernel, and the critical components remained untouched. As explained in [4]: monolithic OS design is inherently flawed in terms of security.

3 THE SECURE HW/SW SOLUTION

3.1 Secure Hardware Platform

HW Implementation: The core of our processor is the 64-bit 6-stage, in-order, single-issue Ariane core [22]. This core implements the open-source RISC-V instruction set architecture (ISA) [20]. Being an open and relatively simple ISA, it can benefit from not having any backward compatibility issue and the errors made in older ISAs; it is therefore a good candidate for a secure system. Moreover, the open-source RTL offers multiple advantages: it is provided with substantial testing facilities and it enables the inspection and validation of the implementation as well as the application of logic locking. For the rest of the system we have also included open-source components (when possible) for our peripherals.

Secure HW Design Flow: One way to protect against the corruption of a device is inspection. Each device must be physically analyzed, e.g., with a scanning electron microscope, and the scanned image compared with the expected layout of the chip. This procedure is extremely costly and challenging. Other techniques involve testing, but it can happen that the applied stimuli does not trigger the Trojan functionality, and the malicious circuitry remains undetected. Because of these difficulties, our approach to protect against Trojans is oriented towards prevention instead of detection.

To ensure the integrity of the design and mitigate potential malicious hardware Trojan insertion by third parties, we implemented a secure design flow with logic locking at its backbone. The flow is presented in Figure 1. It consists of two major parts: the trusted and untrusted regime. Here we assume the involvement of an external design house for performing the layout synthesis. In the trusted regime, the processor core on RTL is further logically synthesized by, e.g., using the Synopsys Design Compiler. Note that the tools and the personnel in this regime are also trusted. After the gate-level netlists are generated, the next step includes the application of logic locking to all selected components of the core. This generates two results: locked gate-level netlists and a key. At this point, a verification and testing phase is performed. The verification is

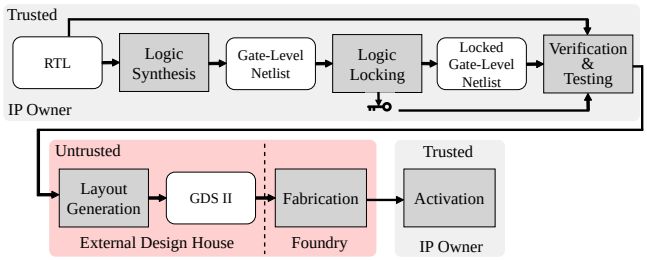

Figure 1: Logic Locking in the IC Design Flow

done by utilizing the Synopsys Formality tool to formally check the equivalence between the original RTL design and the locked core with the correct key applied. Moreover, since the technology library and the key are available, we perform a set of unit tests to check for functional faults in the core. This includes the open-source asm, benchmark and torture tests [11, 22]. Once the locked core is tested, it can proceed in the untrusted regime, i.e., the layout generation and the final fabrication. Once the IC is fabricated, it is returned back to the trusted party to be activated. Further checks can be performed to verify that the silicon and the locked netlist match, by applying the same incorrect key to both the netlist in order to prove that the two systems fail in the same way.

Locking Scheme: It is not necessary, even if possible, to apply logic locking to the whole system. The reason behind this is that meaningful hardware Trojans need some intelligence (e.g., a specific program sequence) to be activated [3, 19]. Therefore, as a first step, we carefully selected all the critical processor modules that should be included in the locking procedure (e.g., pc select, decoder, ALU and others). Afterwards, we comprehensively applied an XOR/XNOR-based logic locking scheme using the Inter-Lock framework and a 1024-bit key [23]. This framework is based on the principle of exploiting existing functional dependencies in the design for security. Therefore, a set of critical modules of the processor core is selected and a joint locking of all components is performed. More details about the framework can be found in [23]. As final note, it is important to remember that logic locking has an impact on the size of the device and on the delay of the obfuscated paths. The length of the key must be selected so that it has a negligible impact on the chip size and the critical path of the design.

3.2 Secure Software Platform

Secure OS: seL4 [10] is a third generation microkernel designed from scratch by NICTA (now part of CSIRO's Data61) starting from 2006. It is meant to provide the basis for highly secure systems by guaranteeing functional correctness, enforcing the confidentiality of the critical data and providing data and temporal integrity [7]. Confidentiality means that data cannot be intercepted by parts of the system which have not been explicitly given access. Data integrity means that a memory location cannot be written without explicit permission. Temporal integrity means that untrusted code must not be able to prevent real-time components from meeting their deadlines.

All these features have undergone a rigorous process of formal verification, including a machine-checked mathematical proof that any possible implementation behavior is allowed by the specification of the kernel's functionality and that the specification provides the right mechanisms to enforce the given security properties [10]. Such guarantees are only possible through formal methods (mathematical proof techniques). Traditional verification based on testing and code inspection are never exhaustive and can leave critical sequences unverified. Formal methods, on the other hand, analyze all possible behaviors of a system an can therefore provide a clear answer about its correctness.

seL4 has been designed following the two principles which also guided the development of the hardware: the designers included only the concepts which are necessary to implement the system's functionality (as per definition of a microkernel [12]) and released it under GPLv2 license. It also supports the RISC-V architecture (even though formal verification is still ongoing), making it the ideal candidate as founding element in our secure HW+SW stack.

seL4 does not provide all the high-level features required by a modern operating system, but it gives the foundation to build a trustworthy one. By guaranteeing integrity and confidentiality, it is possible to use it as foundation for more complex applications (file system, cryptography, networking and others) which together constitute a secure operating system. These extra components are not guaranteed to function correctly; if not formally verified, they are assumed to contain bugs and vulnerabilities. By using seL4 as underlying layer, we have the guarantee that a corrupted module cannot affect the whole system (as it happens in monolithic operating systems like Windows and Linux).

Boot Code: Given that the formal verification process is a highly specialized and time consuming activity, it is not recommended to modify (not even intentionally) the seL4 source code: it is expected to be immutable. To ensure that the microkernel running on our secure chip is the one which has been officially released by Data61, we store it in ROM and make it part of the boot code, after having verified in-house that all the proofs still hold. The immutable ROM content is verified both during the chip's validation process and by a self-test task started every time the system boots.

3.3 Physical Attacks

Once the chip is activated and operational, the system still remains under threat. However, for the first generation we assume that an attacker has no physical access to the device.

4 CONCLUSION

This paper presents a novel secure HW/SW solution. The hardware implementation is based on trustworthy RISC-V-based RTL that is protected against malicious modifications through logic locking. The software stack is backed by seL4; a secure OS based on the most trustworthy microkernel available. The combination of these two elements results in a trustworthy platform suitable for security critical applications. Moreover, the HW and SW building blocks have their roots deep in the open-source community, providing a huge audience that is able to monitor, test and fix these components.

To further strengthen the security claims of the presented HW/SW platform, together with external partners and research institutions, we have started analyzing the overall security of our system, its tolerance to fault injections and the strength of the logic locking

scheme. The outcome of these studies will constitute the base for the next generation of our stack.

REFERENCES

[1] Accenture and Ponemon Institute. 2019. Ninth Annual Cost of Cybercrime Study. *https://www.accenture.com/us-en/insights/security/cost-cybercrime-study* (mar 2019).

[2] Sarah Amir, Bicky Shakya, Domenic Forte, Mark Tehranipoor, and Swarup Bhunia. 2017. Comparative Analysis of Hardware Obfuscation for IP Protection. In *Proceedings of the on Great Lakes Symposium on VLSI 2017* (Banff, Alberta, Canada) *(GLSVLSI '17)*. Association for Computing Machinery, New York, NY, USA, 363–368. https://doi.org/10.1145/3060403.3060495

[3] Swarup Bhunia and Mark M. Tehranipoor. 2017. *The Hardware Trojan War: Attacks, Myths, and Defenses* (1st ed.). Springer Publishing Company, Incorporated.

[4] Simon Biggs, Damon Lee, and Gernot Heiser. 2018. The Jury Is In: Monolithic OS Design Is Flawed: Microkernel-based Designs Improve Security. 1–7. https://doi.org/10.1145/3265723.3265733

[5] A. Chakraborty, N. G. Jayasankaran, Y. Liu, J. Rajendran, O. Sinanoglu, A. Srivastava, Y. Xie, M. Yasin, and M. Zuzak. 2019. Keynote: A Disquisition on Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019), 1–1. https://doi.org/10.1109/TCAD.2019.2944586

[6] Darren D. Cofer, John D. Backes, Andrew Gacek, Daniel daCosta, Michael W. Whalen, Ihor Kuz, Gerwin Klein, Gernot Heiser, Lee Pike, Adam Foltzer, Michal Podhradský, Douglas A. Stuart, Jason Grahan, and Brett Wilson. 2017. Secure Mathematically-Assured Composition of Control Models. *HACMS Final Report* (sep 2017).

[7] Gernot Heiser, Toby Murray, and Gerwin Klein. 2012. It's Time for Trustworthy Systems. *IEEE Security and Privacy* 10 (03 2012), 67–70. https://doi.org/10.1109/MSP.2012.41

[8] IBM. 2020. How much would a data breach cost your business? *https://www.ibm.com/security/data-breach* (jan 2020).

[9] Ayush Jain, Ziqi Zhou, and Ujjwal Guin. 2019. TAAL: Tampering Attack on Any Key-based Logic Locked Circuits, In IEEE T VLSI SYST. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

[10] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, and Michael Norrish. 2009. seL4: Formal Verification of an OS Kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles* (Big Sky, Montana, USA) *(SOSP '09)*. Association for Computing Machinery, New York, NY, USA, 207–220. https://doi.org/10.1145/1629575.1629596

[11] Yunsup Lee and Henry Cook. 2012. RISC-V Torture Test Generator. https://github.com/ucb-bar/riscv-torture.

[12] Jochen Liedtke. 1995. On micro-Kernel Construction. In *Proceedings of the Fifteenth ACM Symposium on Operating System Principles, SOSP 1995, Copper Mountain Resort, Colorado, USA, December 3-6, 1995*. 237–250. https://doi.org/10.1145/224056.224075

[13] Cybercrime Magazine. 2020. Global Cybercrime Damages Predicted To Reach $6 Trillion Annually By 2021. *https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/* (jan 2020).

[14] Network World Michael Cooney. 2017. DARPA to eliminate "patch & pray" by baking chips with cybersecurity fortification. *https://www.varonis.com/blog/cybersecurity-budget/* (apr 2017).

[15] Mir Tanjidur Rahman, Shahin Tajik, M. Sazadur Rahman, Mark Tehranipoor, and Navid Asadizanjani. 2019. The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes. Cryptology ePrint Archive, Report 2019/719. https://eprint.iacr.org/2019/719.

[16] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *2008 Design, Automation and Test in Europe*. 1069–1074. https://doi.org/10.1109/DATE.2008.4484823

[17] Sarah Hospelhorn Varonis. 2019. The Future of Cybersecurity Budgeting. *https://www.varonis.com/blog/cybersecurity-budget/* (may 2019).

[18] Dominik Šišejković, Rainer Leupers, Gerd Ascheid, and Simon Metzner. 2018. A Unifying Logic Encryption Security Metric. In *Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation* (Pythagorion, Greece) *(SAMOS '18)*. Association for Computing Machinery, New York, NY, USA, 179–186. https://doi.org/10.1145/3229631.3229636

[19] Dominik Šišejković, Farhad Merchant, Rainer Leupers, Gerd Ascheid, and Sascha Kegreiss. 2019. Control-Lock: Securing Processor Cores Against Software-Controlled Hardware Trojans. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI* (Tysons Corner, VA, USA) *(GLSVLSI '19)*. Association for Computing Machinery, New York, NY, USA, 27–32. https://doi.org/10.1145/3299874.3317983

[20] Andrew Waterman, Yunsup Lee, David Patterson, and Krste Asanovic. 2014. The RISC-V instruction set manual. *volume I: User-level ISA, version 2.0, Tech. Rep. UCB/EECS-2014-54* (2014).

[21] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mark Mohammad Tehranipoor. 2016. Hardware Trojans: Lessons Learned after One Decade of Research. *ACM Trans. Design Autom. Electr. Syst.* 22, 1 (2016), 6:1–6:23. https://doi.org/10.1145/2906147

[22] F. Zaruba and L. Benini. 2019. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 11 (Nov 2019), 2629–2640. https://doi.org/10.1109/TVLSI.2019.2926114

[23] D. Šišejkovic, F. Merchant, R. Leupers, G. Ascheid, and S. Kegreiß. 2019. Inter-Lock: Logic Encryption for Processor Cores Beyond Module Boundaries. In *2019 IEEE European Test Symposium (ETS)*. 1–6. https://doi.org/10.1109/ETS.2019.8791528