An End-to-End Bitstream Tamper Attack Against Flip-Chip FPGAs

Fahim Rahman, Farimah Farahmandi, and Mark Tehranipoor
Florida Institute for Cybersecurity Research
University of Florida
Gainesville, United States
{fahimrahman, farimah, tehranipoor}@ece.ufl.edu

***Abstract*—FPGA bitstream encryption and authentication can be defeated by various techniques and it is critical to understand how these vulnerabilities enable extraction and tampering of commercial FPGA bitstreams. We exploit the physical vulnerability of bitstream encryption keys to readout using failure analysis equipment and conduct an end-to-end bitstream tamper attack. Our work underscores the feasibility of supply chain bitstream tampering and the necessity of guarding against such attacks in critical systems.**

I. Introduction

Bitstream encryption and authentication secure FPGAs against counterfeiting, intellectual property (IP) theft, and bitstream tampering attacks [1]. However, architectural flaws, side-channel analysis, and failure analysis techniques can undermine bitstream protection schemes [2]. Considered together with bitstream editing [3]–[5] and reverse engineering tools [6]–[8], these weaknesses suggest that supply chain tampering attacks on FPGAs are feasible. Recent publications affirm this threat: bitstream tampering has been able to sabotage cryptographic engines [9] and bypass the system-level root-of-trust in Cisco routers [10].

We perform an end-to-end attack to study how an attacker with temporary physical access to an FPGA system could tamper its bitstream. Modern FPGAs include hashing algorithms in bitstream loading circuitry to verify bitstream authenticity. The security of such a scheme is rooted in an on-device symmetric key that can only be read by bitstream loading logic and is hardened against side-channels, but can be extracted with failure analysis techniques. Possession of this key enables decryption and modification of a target bitstream.

We extract the key from a commercial FPGA and insert a trojan circuit in empty space in the decrypted bitstreams. Our trojan could give attackers a remotely-accessible foothold in otherwise secure systems. We re-package the tampered bitstream and show that the FPGA loads it without complaint.

II. Background

A. FPGA bitstream protection

Field programmable gate arrays (FPGAs) are composed of reconfigurable logic elements that get "programmed" by a **bitstream**. Many FPGAs do not have internal non-volatile memory (NVM), so bitstreams are stored in external NVM and loaded during boot [1]. However, external NVM can be written or read by other components in a system (e.g., application processors) or by physical adversaries, so bitstreams must be *encrypted* to prevent intellectual property theft and cloning and *authenticated* to prevent tampering and Trojan insertion [1].

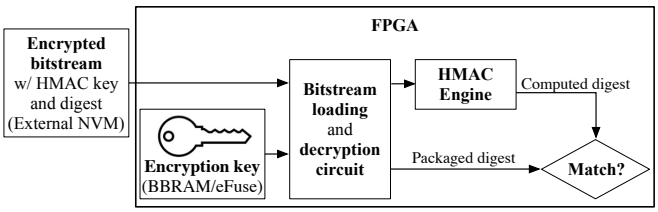

Fig. 1. Overview of FPGA secure boot process. If the packaged and computed digests match, the bitstream is deemed authentic and boot proceeds. If the digests do not match, boot is aborted.

Xilinx's 7-series bitstream protection (Fig. 1) is typical of modern FPGAs. An HMAC digest is generated for a bitstream and the HMAC key is packaged in the encrypted portion of the bitstream. The encryption key is then written to FPGA eFuse ROM or battery-backed RAM (BBRAM) [11]. On boot, the FPGA decrypts the bitstream, extracts the HMAC key and verifies the HMAC digest using dedicated bitstream loading circuitry. If HMAC verification fails, the boot is aborted.

Unfortunately, researchers have found and exploited several vulnerabilities that undermine this protocol [12]–[14]. Several compromise *both* confidentiality and integrity by learning the bitstream encryption key; since the encryption key protects the authentication key as shown in Fig. 2, knowledge of the former implies knowledge of the latter.

B. Key extraction via thermal laser stimulation

Laser stimulation is a class of contactless fault isolation techniques that measures the effect of laser radiation on a device under test (DUT). The laser's effect, and suitability for different attacks, depends on its photon energy. In ***thermal* laser stimulation** (TLS) the photon energy is below the silicon band gap and thus cannot generate photocarriers. This ensures, for example, that SRAM contents can be imaged *without disturbing stored values*.

To generate an image with TLS, a small voltage is applied to supply pins and current flow is monitored while a laser is scanned over silicon substrate in a raster pattern. The laser penetrates bulk silicon and causes supply current fluctuations

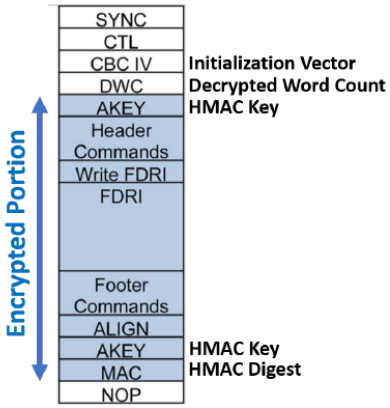

Fig. 2. Xilinx 6- and 7-series bitstream format. The encrypted portion of the bitstream is decrypted using a key written to special FPGA memory when the bitstream is programmed. Each FPGA can receive a unique encryption key to prevent cloning. The HMAC authentication key (AKEY) is stored in the encrypted portion of the bitstream. Thus, so long as the encryption key remains secret, the bitstream is protected from reverse engineering, cloning, and tampering.

due to the Seebeck effect [15] and Optical Beam Induced Resistance Change (OBIRCH) [16]. For a detailed explanation of TLS and TLS imaging of SRAM, we refer the reader to the excellent work of Lohrke et al. [13].

TLS can extract bitstream encryption keys from FPGA BBRAM in mere minutes and many modern FPGAs are believed vulnerable [13]. We demonstrate TLS key extraction in this study to draw attention to this potent and widespread threat. FPGA BBRAM is an *ideal* target for TLS attacks for several reasons:

- Many modern FPGAs are packaged as bare-die or lidded flip-chips [16], obviating the need for tedious and damaging depackaging steps. Previous experiments [13] have extracted FPGA BBRAM without *any* sample preparation, including silicon bulk thinning. This significantly reduces attack complexity and time requirements.
- BBRAM contents are maintained using a coin-cell battery and separate power terminals. This creates an optimal low-noise TLS signal when imaging BBRAM.
- BBRAM cells use relatively large transistors (approximately ten times minimum cell size) to increase reliability and stretch coin cell battery life [13]. This makes them easier to image using a laser spot larger than the minimum feature width in an FPGA [17]. Non-standard BBRAM cell sizes may also help adversaries to more quickly localize BBRAM.

C. Bitstream tampering attacks

Even rudimentary bitstream editing capabilities can facilitate critical exploits against real targets: Kataria et al. demonstrate a 15-byte bitstream modification that bypasses Cisco's FPGA-based secure boot [10], and Swierczynski et al. undermine a FIPS-certified USB device by modifying AES S-boxes stored in block RAM [9]. Tools like TORC [18] and BITMAN [19] enable such direct bitstream editing.

Manipulation of encrypted bitstreams has also been explored. Swierczynski et al. proved that tampering encrypted bitstreams can enable key extraction from AES engines [3]. However, bitstream authentication effectively prevents this attack.

D. Other bitstream protection weaknesses

Several methods besides TLS can extract bitstream encryption keys and enable tampering. We summarize these here to emphasize that the threat we demonstrate impacts many FPGAs and applies in various threat models.

1) Side-channel bitstream key extraction: Researchers have successfully extracted encryption keys from multiple Xilinx [20]–[22] and Intel [12], [23] device families using power and electromagnetic (EM) side channels. Decryption cores in newer FPGAs are hardened against side channel attacks [24], but many older FPGA families are still widely used in safety-critical systems [14] and are trivially exploitable.

2) Architectural flaws: An alleged "backdoor" was found in the JTAG circuit of one Microsemi FPGA. The flaw enabled bitstream readback, key readout, tamper attacks, and changes to low-level silicon configuration [25].

More recently, Ender et al. disclosed an architectural flaw affecting 7-series and Virtex 6 devices [14]. The attack abuses a configuration register that is not cleared on reset to turn the FPGA into a decryption oracle, then use the decryption oracle to encrypt an arbitrary bitstream and bypass authentication. An end-to-end exploit of this vulnerability would require 6-12 hours depending on bitstream size.

3) Failure analysis techniques: TLS is not the only failure analysis technique capable of undermining bitstream protections. Encryption key readout using a focused ion beam (FIB) is suggested in [26], though such an attack has not been demonstrated. Tajik et al. extract a plaintext bitstream using electro-optical probing [17]. Their analysis enables IP theft and cloning but does not enable bitstream tampering. [27] discusses how several contactless imaging techniques could be applied to SRAM readout and demonstrates that Laser Logic State Imaging (LLSI) based on Electro-Optical Frequency Mapping (EOFM) can read BBRAM keys.

III. Threat Model

We study a system containing an SRAM FPGA. Such systems are used in critical infrastructure, aviation, defense, etc. and their security is of great consequence. Bitstream encryption and authentication are enabled but can be undermined as discussed in Sections II-B and II-D.

Our adversary has temporary physical access to the system and aims to tamper its bitstream, e.g., to cause denial of service [5], undermine cryptography [28], create software-exploitable hardware flaws [29], create bit flips to aid in reverse engineering activities [3], etc. Such an attack is possible at any point after bitstream and encryption key installation. Electronic systems pass through many hands between assembly and end-user and the feasibility of supply chain "interdiction" attacks has been demonstrated [9], [30].

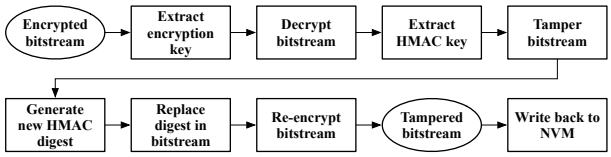

Fig. 3. Stages in hard-HMAC bitstream tamper attack.

For TLS key extraction, an adversary would prepare for their attack by localizing a target device's BBRAM and writing image comparison programs [13] so that at attack time, the target can be imaged quickly and images can be automatically processed to recover the encryption key. The necessary failure analysis equipment could be rented for an estimated $300/hr [13] or a makeshift laser scanning setup could be constructed for an estimated $100k [31].

We assume our adversary is capable of basic bitstream manipulation using tools like BITMAN [27].

IV. Methods

The steps in an end-to-end attack against hard authentication are summarized in Fig. 3. We follow the procedure outlined by Lohrke et al. [13] to extract the bitstream key from BBRAM using TLS, then use standard encryption and hashing algorithms to decrypt and re-package the bitstream.

A. Experimental setup

We target a Xilinx XC7K70T, a 7-series bare-die flip chip that implements a hard authentication core and AES-CBC encryption. We write a random, unknown encryption key to BBRAM. The legitimate bitstream contains an Ethernet PHY and a serial module for intra-system communication.

Our TLS setup uses a PHEMOS-1000 laser scanning microscope with a 1300 nm wavelength and 50x magnification lens. To read the BBRAM key, we transfer BBRAM to a 1.5 V benchtop power supply with current monitoring capability.

B. Key extraction

Emulating an adversary's preparation, we used publicly-available documents, information from Xilinx's layout planner, and images published in prior works [13] to localize our target's BBRAM. We verified that the logical-spatial mapping of BBRAM bits matched prior literature [13], then imaged an all-zero key to obtain a reference for image analysis algorithms as suggested by [13].

Then, we programmed the FPGA with an unknown random key. BBRAM was transferred from battery power to our current-monitoring power supply and a TLS image of the FPGA substrate was generated as described in Section II-B. The captured image was compared against our all-zero reference to extract the key.

C. Bitstream decryption

We use the extracted key and a standard AES-CBC implementation to decrypt the bitstream. The unencrypted header is parsed to obtain the decrypted word count (dwc) and AES-CBC initialization vector (IV). Then, for each 16-byte block in B_E, the block is bitwise-mirrored then decrypted using AES-CBC.

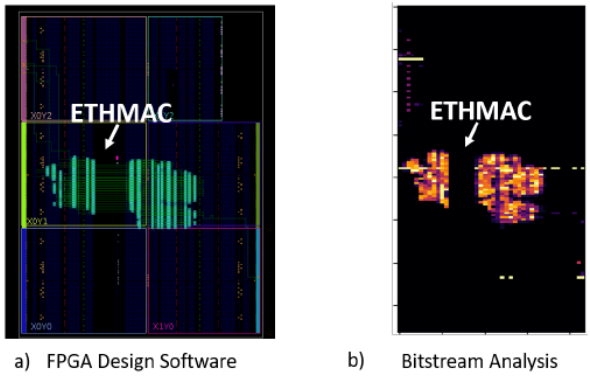

Fig. 4. a) Initial ETHMAC design place and route view in Vivado FPGA design software. b) Binary configuration analysis plotting row and column resource usage to identify unused areas for Trojan insertion in a bitstream.

Algorithm 1 Insert trojan circuit into bitstream

Input: Decrypted bitstream B_D, Required trojan area A_{Target}, Trojan bitstream T_j
Output: Tampered bitstream B_{DM}
procedure MODIFY(B_D, A_{Target})
  $A_{Unused} \leftarrow 0$
  while $A_{Unused} < A_{Target}$ **do**
    scan B_D for unused resources
    $A_{Unused} \leftarrow$ Union(A_{Unused}, resources)
  $T_j \leftarrow$ trojan circuit design
  Constrain T_j to A_{Unused}
  $B_{DTj} \leftarrow$ GenerateBitstream(T_j)
  $B_{DM} \leftarrow B_D \cup B_{DTj}$
  Disable CRC in bitstream header
  Return: B_{DM}

D. Bitstream tampering

We parse B_D according to vendor documentation [32] and note the row, column, and minor address of populated bitstream frames. Contiguous areas of unused fabric are candidate sites for our trojan circuit. Unused frame addresses are used as placement constraints in CAD tools. Algorithm 1 summarizes this process. Fig. 4 illustrates our utilization analysis. For our proof of concept, we create necessary trojan routing and placement constraints in the Vivado GUI so that in-depth reverse engineering of switch matrix connections is not required. We generate a bitstream containing only our trojan circuit, then logically OR frame write commands in the trojan bitstream with the original.

Our malicious circuit, "PadLeech", connects to an FPGA pin (in this case, a UART interface) as shown in Fig. 5 and to the FPGA's Internal Configuration Access Port (ICAP). The circuit snoops UART signals until it receives its trigger sequence, then forwards all received data to the ICAP. Such a trojan would be useful in a system where the UART is connected to a CPU: an adversary could use software exploits to gain code execution on the CPU, then control the UART interface and remotely reprogram the FPGA.

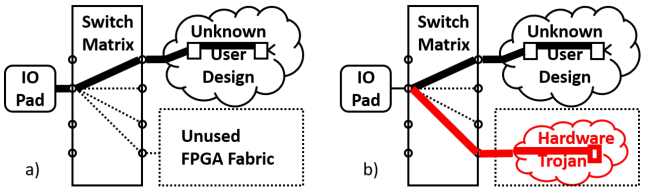

Fig. 5. PadLeech attack. a) Design as exists in initial bitstream, b) Design after PadLeech bitstream modification to insert hardware Trojan remotely controlled using existing design I/O.

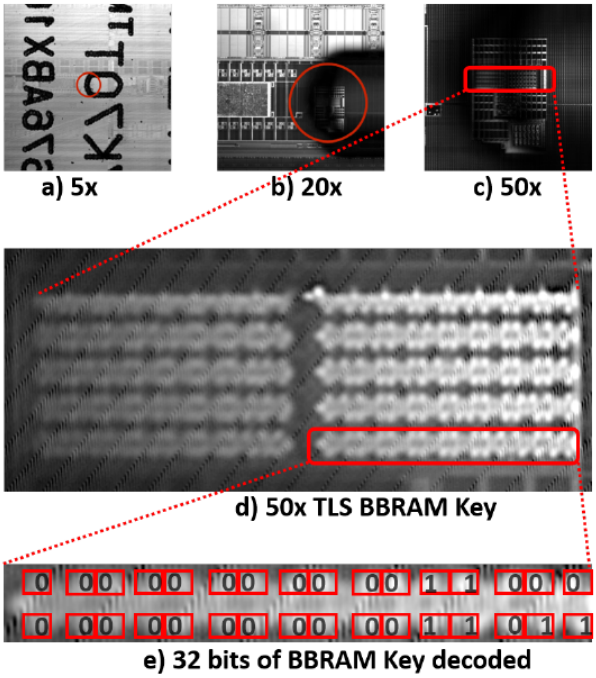

Fig. 6. Xilinx XCK70T-676 FPGA BBRAM localization microscope images at a) 5x, b) 20x, c) 50x objectives. d) BBRAM key extraction using thermal laser stimulation technique from [13]. e) BBRAM key decoding of a 32-bit section.

E. Bitstream re-packaging

To complete the end-to-end attack, *dwc* is updated to accommodate changes in bitstream length and a new HMAC digest is generated for the tampered bitstream. The tampered bitstream is assembled according to Fig. 2 with a new HMAC and HMAC key (which need not equal the legitimate key), and the bitstream body is re-encrypted using the key extracted from BBRAM. The tampered bitstream is written to NVM and we verify that the the FPGA boots the tampered bitstream without error.

V. Results

A. TLS images

Images of the BBRAM cells at magnifications of 5x to 50x are shown in Fig. 6(a)-(c). The random unknown key is depicted in Fig. 6(d), and Fig. 6(e) illustrates a decoded 32-bit key segment.

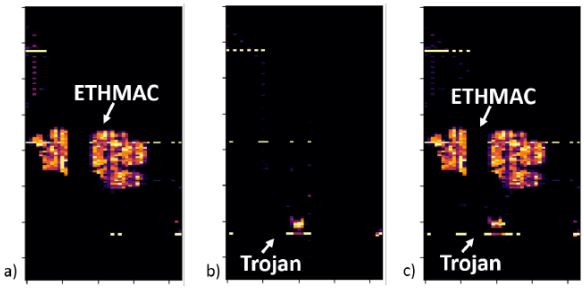

Fig. 7. Bitstream analysis of a) ETHMAC design before tampering, b) Trojan circuit connected to a UART pin, and c) Tampered ETHMAC design. Regions with all zeros (i.e., unused resources) are shown in black. Colored areas are used FPGA resources and brightness indicates the intensity of configuration bits.

B. Bitstream tampering

Fig. 7 compares resource utilization of the untampered design with the trojan-inserted design. The Trojan-only bitstream is shown in Fig. 7(b) and the tampered bitstream is shown in Fig. 7(c).

We validated our trojan circuit by transmitting its trigger sequence to the relevant UART port and sending commands to the ICAP that reconfigured GPIO connections on the board. The GPIO ports were reconfigured as expected.

VI. Traditional Countermeasures

A. Bitstream Obfuscation

[33], [34] logic locked the original FPGA implementations through modifications on the unused portions of LUTs so that the circuitry cannot behave correctly until being unlocked by receiving the expected obfuscation key [35]. The obfuscation key could be generated on the FPGA fabric using PUFs or stored in an external tamper- and read-proof memory [36]–[38]. Given cases are also vulnerable to the demonstrated attack if the key is stored in the memory.

B. Run-time Bitstream Authentication

Many modern FPGAs incorporate the primitives such as ICAP and PCAP for the dynamic partial configuration and readout capabilities [39]. This can be used for checking bitstream integrity at run-time. [40] proposes a protocol for secure key initialization and the general idea behind is utilized in [41] for run-time bitstream authentication. However, the proposed method in [41] requires security primitives like PUF [42], [43] and cryptographic ciphers and hash functions [44] for hardware and bitstream authentication by trusted entities.

VII. Acknowledgments

The authors would like to acknowledge the contributions made by Adib Nahiyan, Tanjid Rahman, Shahin Tajik, Nitin Varshney, Tao Zhang, and Navid Asadi.

VIII. Conclusion

We demonstrate the feasibility of an end-to-end tampering attack on modern FPGAs. We used TLS to image a bitstream encryption key in Kintex Ultrascale BBRAM, then processed the images to extract the key. We used the key to decrypt the bitstream and used basic bitstream manipulation to insert a trojan circuit. We verified that the FPGA loads our tampered bitstream and that the trojan works as expected. Our work highlights a significant threat to FPGA-based systems used in critical real-world applications.

References

[1] S. M. Trimberger and J. J. Moore, "FPGA Security: Motivations, Features, and Applications," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1248–1265, Aug. 2014.

[2] A. Duncan, F. Rahman, A. Lukefahr, F. Farahmandi, and M. Tehranipoor, "FPGA Bitstream Security: A Day in the Life," in *2019 IEEE International Test Conference (ITC)*. IEEE, 2019, pp. 1–10.

[3] P. Swierczynski, G. T. Becker, A. Moradi, and C. Paar, "Bitstream Fault Injections (BiFI)–Automated Fault Attacks Against SRAM-Based FPGAs," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 348–360, 2018.

[4] D. Ziener, J. Pirkl, and J. Teich, "Configuration Tampering of BRAM-based AES Implementations on FPGAs," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2018, pp. 1–7.

[5] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik, "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream," *IEEE Design & Test*, vol. 30, no. 2, pp. 45–54, 2013.

[6] T. Zhang, J. Wang, S. Guo, and Z. Chen, "A Comprehensive FPGA Reverse Engineering Tool-chain: From Bitstream to RTL code," *IEEE Access*, vol. 7, pp. 38 379–38 389, 2019.

[7] F. Benz, A. Seffrin, and S. A. Huss, "BIL: A Tool-chain for Bitstream Reverse-engineering," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2012, pp. 735–738.

[8] M. Ender, P. Swierczynski, S. Wallat, M. Wilhelm, P. M. Knopp, and C. Paar, "Insights into the Mind of a Trojan Designer: the Challenge to Integrate a Trojan into the Bitstream," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 112–119.

[9] P. Swierczynski, M. Fyrbiak, P. Koppe, A. Moradi, and C. Paar, "Interdiction in practice—Hardware Trojan against a high-security USB flash drive," *Journal of Cryptographic Engineering*, vol. 7, no. 3, pp. 199–211, Sep. 2017. [Online]. Available: https://doi.org/10.1007/s13389-016-0132-7

[10] J. Kataria, R. Housley, J. Pantoga, and A. Cui, "Defeating Cisco Trust Anchor: A Case-Study of Recent Advancements in Direct FPGA Bitstream Manipulation," in *13th USENIX Workshop on Offensive Technologies (WOOT 19)*. Santa Clara, CA: USENIX Association, Aug. 2019. [Online]. Available: https://www.usenix.org/conference/woot19/presentation/kataria

[11] Xilinx, "Using Encryption to Secure a 7 Series FPGA Bitstream," 2021, app note #1239.

[12] P. Swierczynski, A. Moradi, D. Oswald, and C. Paar, "Physical Security Evaluation of the Bitstream Encryption Mechanism of Altera Stratix II and Stratix III FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 4, Dec. 2014, place: New York, NY, USA Publisher: Association for Computing Machinery. [Online]. Available: https://doi.org/10.1145/2629462

[13] H. Lohrke, S. Tajik, T. Krachenfels, C. Boit, and J.-P. Seifert, "Key Extraction Using Thermal Laser Stimulation: A Case Study on Xilinx Ultrascale FPGAs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 573–595, Aug. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7287

[14] M. Ender, A. Moradi, and C. Paar, "The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series FPGAs," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1803–1819. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/ender

[15] C. Boit, C. Helfmeier, D. Nedospasov, and A. Fox, "Ultra high precision circuit diagnosis through seebeck generation and charge monitoring," in *Proceedings of the 20th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, 2013, pp. 17–21.

[16] Wai Mun Yee, M. Paniccia, T. Eiles, and V. Rao, "Laser voltage probe (LVP): a novel optical probing technology for flip-chip packaged microprocessors," in *Proceedings of the 1999 7th International Symposium on the Physical and Failure Analysis of Integrated Circuits (Cat. No.99TH8394)*, Jul. 1999, pp. 15–20, journal Abbreviation: Proceedings of the 1999 7th International Symposium on the Physical and Failure Analysis of Integrated Circuits (Cat. No.99TH8394).

[17] S. Tajik, H. Lohrke, J.-P. Seifert, and C. Boit, "On the Power of Optical Contactless Probing: Attacking Bitstream Encryption of FPGAs," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1661–1674, event-place: Dallas, Texas, USA. [Online]. Available: https://doi.org/10.1145/3133956.3134039

[18] N. Steiner, A. Wood, H. Shojaei, J. Couch, P. Athanas, and M. French, "Torc: Towards an Open-Source Tool Flow," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 41–44, event-place: Monterey, CA, USA. [Online]. Available: https://doi.org/10.1145/1950413.1950425

[19] K. Dang Pham, E. Horta, and D. Koch, "BITMAN: A tool and API for FPGA bitstream manipulations," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, Mar. 2017, pp. 894–897, journal Abbreviation: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017.

[20] A. Moradi and T. Schneider, "Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series," in *Constructive Side-Channel Analysis and Secure Design*, F.-X. Standaert and E. Oswald, Eds. Cham: Springer International Publishing, 2016, pp. 71–87.

[21] A. Moradi, M. Kasper, and C. Paar, "Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures," in *Topics in Cryptology – CT-RSA 2012*, O. Dunkelman, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–18.

[22] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II FPGAs," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 111–124, event-place: Chicago, Illinois, USA. [Online]. Available: https://doi.org/10.1145/2046707.2046722

[23] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side-Channel Attacks on the Bitstream Encryption Mechanism of Altera Stratix II: Facilitating Black-Box Analysis Using Software Reverse-Engineering," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 91–100, event-place: Monterey, California, USA. [Online]. Available: https://doi.org/10.1145/2435264.2435282

[24] Xilinx, "Using Encryption and Authentication to Secure an UltraScale/UltraScale+ FPGA Bitstream," 2021, app note #1267.

[25] S. Skorobogatov and C. Woods, "Breakthrough Silicon Scanning Discovers Backdoor in Military Chip," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 23–40.

[26] B. Linke, "Protect Your FPGA Against Piracy: Cost-Effective Authentication Scheme Protects IP in SRAM-Based FPGA Designs," 2009, app Note #4594.

[27] C. Boit, T. Kiyan, T. Krachenfels, and J.-P. Seifert, "Logic State Imaging From FA Techniques for Special Applications to One of the Most Powerful Hardware Security Side-Channel Threats," in *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, 2020, pp. 1–7.

[28] M. Ender, P. Swierczynski, S. Wallat, M. Wilhelm, P. M. Knopp, and C. Paar, "Insights into the Mind of a Trojan Designer: The Challenge to Integrate a Trojan into the Bitstream," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 112–119, event-place: Tokyo, Japan. [Online]. Available: https://doi.org/10.1145/3287624.3288742

[29] G. Dessouky, D. Gens, P. Haney, G. Persyn, A. Kanuparthi, H. Khattri, J. M. Fung, A.-R. Sadeghi, and J. Rajendran, "HardFails: Insights into Software-Exploitable Hardware Bugs," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 213–230. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/dessouky

[30] "NSA ANT Catalog." [Online]. Available: https://www.eff.org/document/20131230-appelbaum-nsa-ant-catalog

[31] T. Krachenfels, H. Lohrke, J.-P. Seifert, E. Dietz, S. Frohmann, and H.-W. Hübers, "Evaluation of Low-Cost Thermal Laser Stimulation for Data Extraction and Key Readout," *Journal of Hardware and Systems Security*, vol. 4, no. 1, pp. 24–33, Mar. 2020. [Online]. Available: https://doi.org/10.1007/s41635-019-00083-9

[32] Xilinx, "7 Series FPGAs Configuration," 2018, uG470.

[33] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "Robust Bitstream Protection in FPGA-based Systems through Low-overhead Obfuscation," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2016, pp. 1–8.

[34] B. Olney and R. Karam, "Tunable FPGA Bitstream Obfuscation with Boolean Satisfiability Attack Countermeasure," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 25, no. 2, pp. 1–22, 2020.

[35] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-lock: A Security-corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 175–202, 2020.

[36] P. Mishra, S. Bhunia, and M. Tehranipoor, *Hardware IP Security and Trust*. Springer, 2017.

[37] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani, "The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 262–272.

[38] M. T. Rahman, M. S. Rahman, H. Wang, S. Tajik, W. Khalil, F. Farahmandi, D. Forte, N. Asadizanjani, and M. Tehranipoor, "Defense-in-depth: A Recipe for Logic Locking to Prevail," *Integration*, vol. 72, pp. 39–57, 2020.

[39] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer Science & Business Media, 2011.

[40] A. Duncan, A. Nahiyan, F. Rahman, G. Skipper, M. Swany, A. Lukefahr, F. Farahmandi, and M. Tehranipoor, "SeRFI: Secure Remote FPGA Initialization in an Untrusted Environment," in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–6.

[41] T. Zhang, F. Rahman, F. Farahmandi, and M. Tehranipoor, "FPGA-Chain: Enabling Holistic Protection of FPGA Supply Chain with Blockchain Technology," in *IEEE Workshop on Silicon Lifecycle Management (SLM)*. IEEE, 2021.

[42] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor, "An Aging-resistant RO-PUF for Reliable Key Generation," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 3, pp. 335–348, 2015.

[43] N. N. Anandakumar, M. S. Hashmi, and M. Tehranipoor, "FPGA-based Physical Unclonable Functions: A Comprehensive Overview of Theory and Architectures," *Integration*, 2021.

[44] L. Wu, X. Wang, X. Zhao, Y. Cheng, D. Su, A. Chen, Q. Shi, and M. Tehranipoor, "AES Design Improvement Towards Information Safety," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 1706–1709.