FPGA Bitstream Encryption

Martin Krenn

November 29th, 2023

Agenda

- Introduction
    - What is the bitstream?
    - Why is bitstream encryption needed?
    - How does bitstream encryption work?
- Attacks on bitstream encryption
- "Case Study": The Unpatchable Silicon by Ender, Moradi, and Paar [2]
- Lessons learned

Introduction

What is the bitstream?

The “binary” of the FPGA [1], [2]

- The “fabric data”
  - “Logic”: Content of look-up tables, ...
  - “Wiring”: Configuration of switch-boxes, ...
  - Block-RAM configuration and initial values
- General configuration of the FPGA
  - Instructions for the configuration engine

Why is bitstream encryption needed?

- Bitstream contains confidential information
    - Intellectual Property (IP)
    - Keys
- Most FPGAs are SRAM-based
    - Bitstream stored in non-volatile memory (NVM) at rest
    - Bitstream loaded to SRAM upon startup
- Threats
    - Cloning
    - Reverse engineering
    - Tampering (e.g. insert Trojans)

How does bitstream encryption work?

- Keys securely stored in FPGA
  - BBRAM/eFuse
- Encrypted bitstream transferred & stored on NVM
- Configuration engine of FPGA

$$\left.\begin{array}{l}\text{Loads}\\ \text{Authenticates}\\ \text{Decrypts}\end{array}\right\}\text{ the bitstream}$$

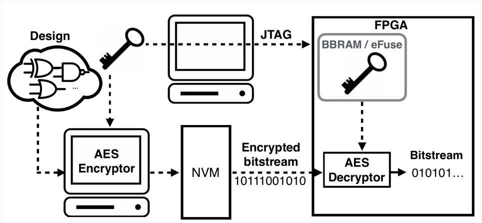

Figure 1: Simplified bitstream encryption [3]

Attacks on bitstream encryption

Attacks on bitstream encryption

- Side-channel attacks (e.g. differential power analysis (DPA))
- Fault attacks (e.g. power glitching)
- Protocol attacks

“Case Study”: The Unpatchable Silicon by Ender, Moradi, and Paar [2]

Adversary model

- Access to the encrypted bitstream
- Limited knowledge about plaintext
- FPGA with loaded AES-key
- Access to configuration interface

Figure 2: The adversary [4]

Bitstream encryption of the Xilinx 7-Series

- On-chip decryption engine
    - CBC-AES-256 for confidentiality
    - HMAC-SHA-256 for authenticity
- AES key is set using configuration interface (e.g. JTAG)

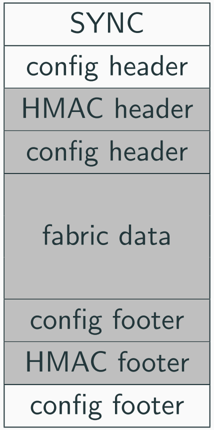

Figure 3: Simplified bitstream format of the Xilinx 7-series - gray parts encrypted [2], [5]

Attacking the encryption

Two vulnerabilities:

1. CBC-mode is malleable during decryption
    - Flipped bits Δ in C_i results in garbled P_i'
    - but in the same bits flipped in $P_{i+1} \oplus \Delta$

    $\rightarrow$ Used for known plaintext attack to insert instructions into the bitstream

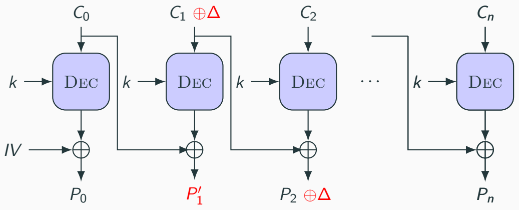

Figure 4: CBC malleability illustrated [2], [6]

Attacking the encryption

Two vulnerabilities:

1. CBC-mode is malleable during decryption
2. HMAC is last to be checked
    - Forged bitstream gets decrypted
    - Instructions are executed
    - Authenticity check fails afterwards
    - Reset is triggered

→ Use FPGA as decryption oracle

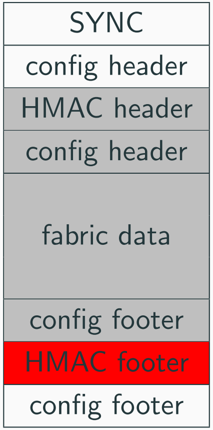

Figure 5: Simplified bitstream format - red part contains authentication-tag [2], [5]

The final ingredient

How do we read-back the decrypted data?

- Exploit a special configuration register:
  WBSTAR (warm-boot start-address register)
    - Used for MultiBoot [7]
    - Not cleared during reset

Put all pieces together

1. Craft a malicious bitstream containing
   - The fabric block to decrypt
   - An instruction to write to the WBSTAR
2. Download malicious bitstream
   - FPGA decrypts & executes the bitstream
     and eventually resets due to authentication error
3. Use a “readout” bitstream to obtain the content of WBSTAR
4. Rinse
5. Repeat

What else

The attack can be used to break authenticity as well

- Take arbitrary C_n, C_{n-1}, results in quasi-random $P_n = DEC_K(C_n) \oplus C_{n-1}$
- Find C'_{n-1} which generates the desired P'_n inside the FPGA
  $\rightarrow$ Set $C'_{n-1} = P_n \oplus C_{n-1} \oplus P'_n$ (acc. CBC malleability)
- Repeat steps with arbitrary C_{n-2} and obtained C'_{n-1}, etc.
- Set IV to C'_0 in the end

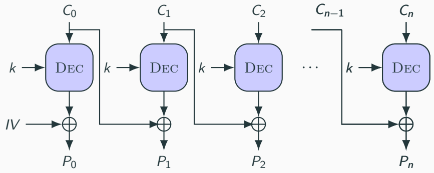

Figure 6: CBC mode of operation [6]

Lessons learned

Lessons learned

- Use state-of-the-art protocols/crypto
    - Authenticate well before use
- Minimize the unpatchable part and use reconfigurable logic for the rest [8]

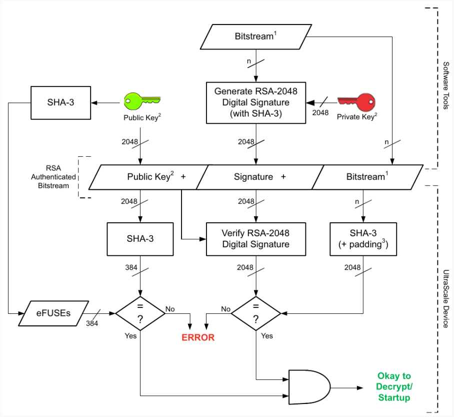

Figure 7: Xilinx UltraScale(+) bitstream encryption using RSA [9]

Thank you for your attention

Questions?

References i

References

[1] A. Duncan, F. Rahman, A. Lukefahr, F. Farahmandi, and M. M. Tehranipoor, FPGA bitstream security: A day in the life, in IEEE International Test Conference, ITC 2019, Washington, DC, USA, November 9-15, 2019, IEEE, 2019, pp. 1–10.

[2] 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 2020, August 12-14, 2020, S. Capkun and F. Roesner, Eds., USENIX Association, 2020, pp. 1803–1819.

[3] 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, Cryptology ePrint Archive, Paper 2018/717, https://eprint.iacr.org/2018/717, 2018. [Online]. Available: https://eprint.iacr.org/2018/717.

References ii

[4] CC BY-NC 4.0. [Online]. Available: https://pngimg.com.

[5] S. M. Trimberger and J. J. Moore, FPGA Security: Motivations, Features, and Applications, Proceedings of the IEEE, vol. 102, no. 8, pp. 1248–1265, 2014. DOI: 10.1109/JPROC.2014.2331672.

[6] J. Jean, TikZ for Cryptographers, https://www.iacr.org/authors/tikz/, 2016.

[7] Xilinx Inc., MultiBoot with 7 Series FPGAs and SPI,, 2017. [Online]. Available: https://docs.xilinx.com/v/u/en-US/xapp1247-multiboot-spi.

[8] F. Unterstein, N. Jacob, N. Hanley, C. Gu, and J. Heyszl, Sca secure and updatable crypto engines for fpga soc bitstream decryption: Extended version, Journal of Cryptographic Engineering, vol. 11, no. 3, pp. 257–272, 2021. [Online]. Available: https://doi.org/10.1007/s13389-020-00247-2.

References iii

[9] Xilinx Inc., Developing Tamper-Resistant Designs with UltraScale and UltraScale+ FPGAs,, 2021. [Online]. Available: https://docs.xilinx.com/v/u/en-US/xapp1098-tamper-resist-designs.