Introduction to Hardware Security and Trust
ECE-GY_9453
Project Milestone 3 Report

1st Ali Rasteh
Tandon School of Engineering
New York University
New York City, USA
ar7655@nyu.edu

2nd Sayam Dhingra
Tandon School of Engineering
New York University
New York City, USA
sd5292@nyu.edu

***Abstract*—This document is the report for project milestone 3 of the Introduction to Hardware Security and Trust course report and was compiled using Overleaf and LaTeX.**

I. M1-Security Asset Analysis

To identify security-relevant assets and properties, we need to understand the design specification of the stm32f042 nucleo board in detail. The security-relevant assets and properties that need to be considered for the stm32f042 nucleo board include:

1) Confidentiality: The board should ensure the confidentiality of the data being processed and transmitted.
2) Integrity: The board should ensure the integrity of the data being processed and transmitted, the data should not be tampered with or modified in any way without detection.
3) Availability: The board should ensure the availability of the data being processed and transmitted, the system should be available when needed and able to handle the expected workload.
4) Authentication: The board should ensure the authenticity of the data being processed and transmitted.
5) Non-Repudiation: The board should ensure that the parties involved in the communication cannot deny their actions or the data being transmitted.

II. Part of design important for security

The following parts of the design of the stm32f042 nucleo board are important for security,

1) Hardware Security: The board should use secure hardware components like a secure boot, secure random number generator, secure key storage, and tamper-resistant components to protect sensitive data and prevent unauthorized access to the board. More specifically access to the UART port is very important and should be appropriately protected in this design.
2) Encryption Algorithms: The implementation of the algorithms should follow industry-standard practices and be properly configured to ensure the confidentiality and integrity of the data being processed.
3) Key Management: The board should have a secure key management system that includes key generation, storage, and distribution. The keys should be protected from unauthorized access, and their integrity should be verified to ensure they have not been tampered with.
4) Secure Communication: The board should use secure communication protocols like SSL/TLS to prevent unauthorized access and ensure the CIA triad.

III. Features that can enhance or weaken security

Features that can enhance security include:

1) Tamper-resistant components: Using tamper-resistant components, such as sensors that detect physical tampering, can enhance security by making it harder for attackers to compromise the hardware.
2) Encryption: Strong encryption algorithms and key management practices can enhance security by ensuring the confidentiality and integrity of the data being processed and transmitted.

Features that can weaken security include:

1) Insecure password practices: Using weak passwords or not enforcing password complexity can weaken security by making it easier for attackers to guess or crack passwords.
2) Lack of encryption: Failing to use encryption or using weak encryption algorithms can weaken security by exposing sensitive data to attackers.

IV. Potential adversaries

Potential adversaries who may be interested in compromising the security of the stm32f042 nucleo board include:

1) Hackers: Hackers may be interested in gaining unauthorized access to the board to steal sensitive data or use it as a platform for launching attacks against other systems.

2) Competitors: Competitors may be interested in stealing proprietary information or intellectual property from the board to gain a competitive advantage.
3) Nation-states: Nation-states may be interested in compromising the board's security to gather intelligence or disrupt critical infrastructure.
4) Insiders: Insiders with authorized access to the board may also pose a security threat if they abuse their privileges or engage in malicious activities.
5) Generally anybody with appropriate access to the hardware and more specifically the UART port could be the potential adversary.

V. Threat Model

The threat model for the stm32f042 nucleo board should consider the potential threats and attacks that the board could face:

1) Physical attacks: The board could be subject to physical attacks, such as tampering or theft, which could compromise its security.
2) Malware attacks: The board could be targeted by malware attacks, such as viruses, worms, or Trojans, which could compromise its security or steal sensitive data.
3) Brute-force attacks: The board could be subject to brute-force attacks, where an attacker attempts to guess a password or encryption key by trying many different combinations until the correct one is found.
4) Cryptographic attacks: The board could be subject to cryptographic attacks, such as side-channel attacks, which attempt to exploit weaknesses in the encryption algorithm or key management practices. Threat actors could be hackers, cybercriminals, or insiders with malicious intent.
5) Software vulnerabilities: The board could be vulnerable to software vulnerabilities, such as buffer overflows, that could be exploited to gain unauthorized access or execute malicious code.
6) Supply chain attacks: The board could be subject to supply chain attacks, where an attacker inserts malicious code or hardware into the board during manufacturing or shipping. Threat actors could be nation-state attackers, cybercriminals, or insiders with malicious intent.
7) Physical/environmental vulnerabilities: The board could be vulnerable to physical/environmental vulnerabilities, such as power surges or electromagnetic interference, that could damage or disrupt its operation. Threat actors could be accidental or intentional.
8) Fault attacks: The board could be subject to fault attacks where an attacker uses fault injection in the UART streaming decryption procedure to extract secret data or useful information by comparison between the faulty and healthy data steams.
9) Scan chain attacks: The boards could be subject to scan chain attacks where the scan chain is used to read the content of the internal registers and extract secret information using some algorithms.

VI. XOR attack and code hardening

To get the XOR keys we analyzed the binary files via Ghidra. We imported the binaries into Ghidra where we could see the decompiled functions and their values. After further analysis, we could find the XOR function and we could see which data variable was used to store the XOR Keys. We found XOR function call when we reviewed the main function check for input value as 0x20 which was the input value for doing XOR encryption. We can observe in figure 1 the function call and the XOR key being sent to the function. Diving deeper into this function we can confirm that this is in fact the XOR function that does the encryption/decryption with the XOR key, as observed in figure 2.

```
else {
  if (CheckFunc != 0x20) goto LAB_08002070;
  *DAT_0800213c = 1;
  FUN_08002a24(0x48000000,0x80,1);
  XOR_Func(&Input,DAT_08002140,&Output);
  FUN_08002a24(0x48000000,0x80,0);
}
```

XOR Key

Fig. 1. XOR Function Call

```
1
2 void XOR_Func(int Input,int XOR_Key,int Output)
3
4 {
5   int Iter;
6
7   for (Iter = 0; Iter < 0x10; Iter = Iter + 1) {
8     *(byte *)(Output + Iter) = *(byte *)(XOR_Key + Iter) ^ *(byte *)(Input + Iter);
9   }
10  return;
11}
12
```

Fig. 2. XOR Function

Later, if we trace the data value of where the key is stored we can see it at memory locations highlighted in the figure 3. We repeated this method for all binary files.

```
080045d8 a6    ??    A6h
080045d9 81    ??    81h
080045da 75    ??    75h   u
080045db 54    ??    54h   T
080045dc 13    ??    13h
080045dd b9    ??    B9h
080045de bc    ??    BCh
080045df 45    ??    45h   E
080045e0 1c    ??    1Ch
080045e1 06    ??    06h
080045e2 8f    ??    8Fh
080045e3 99    ??    99h
080045e4 5f    ??    5Fh   _
080045e5 fb    ??    FBh
080045e6 59    ??    59h   Y
080045e7 98    ??    98h
```

Fig. 3. XOR Key

We also used the chipwhisperer to perform cryptanalysis on the output of the XOR encryption to determine and brute-force the XOR keys with cryptanalysis.

A. XOR attack code hardening

One possible method of securing the keys against this type of attack would be to use **multi-byte XOR**.

Multi-byte XOR is a cryptographic operation that involves performing an XOR (exclusive or) operation on two or more blocks of data, where each block consists of multiple bytes. In this operation, the corresponding bytes in each block are XORed together to produce a new block of data.

This method is commonly used in cryptography for encrypting data. It can provide confidentiality by producing ciphertext that is difficult to decrypt without the correct key. To decrypt the ciphertext, the same key is used to perform the XOR operation in reverse, which recovers the original plaintext. This will therefore make it more difficult to perform cryptanalysis on the XOR encryption. But this method is still not foolproof.

Another solution to implement would be to use **key hardening**.

Key hardening is the process of making encryption keys more resistant to attacks by applying various techniques to increase their strength and reduce the likelihood of successful attacks. The goal of key hardening is to make it as difficult as possible for an attacker to guess or discover the key, even if they have access to large amounts of encrypted data.

We can implement hardening techniques like Key stretching, Key diversification, Salting, and Key rotation. With this method, it would be more difficult to derive the key manually using reversing tools such as Ghidra and even make cryptanalysis more difficult as the final key would be longer and more complicated.

VII. DES ATTACK AND CODE HARDENING

For attacking the DES algorithm and getting its keys we used a similar method to the XOR attack method. We analyzed the binary files in Ghidra and discovered the function where we were checking for the input value 0x30 which was to do DES decryption. This way we discovered the DES function and we could see the input and output values along with the key input we were sending to the function. We can see this in figure 4. In this figure, we can see that the key is in the location "DAT_08002144" as highlighted in the green box. Further going into this function we can confirm that this is in fact the key and this is the DES decrypt function.

```
else if (CheckFunc < 0x41) {
  if (CheckFunc == 0x30) {
    *DAT_0800213c = 2;
    FUN_08002a24(0x48000000,0x80,1);
    DES_Decrypt(&Input,DAT_08002144,&Output);
    FUN_08002a24(0x48000000,0x80,0);
  }
  else {
```

Fig. 4. DES Function

Now if we follow this data location we can see the DES key at the data locations in the figure 5

```
080045ef 00    ??    00h
080045f0 27    ??    27h    '
080045f1 6e    ??    6Eh    n
080045f2 df    ??    DFh
080045f3 58    ??    58h    X
080045f4 bd    ??    BDh
080045f5 43    ??    43h    C
080045f6 80    ??    80h
080045f7 c0    ??    C0h
080045f8 09    ??    09h
```

Fig. 5. DES Key

A. Possible solutions to DES attack

A possible solution for this type of attack would be to use **obfuscation**. We can use code obfuscation techniques to make it more difficult for attackers to reverse-engineer the code and understand how the encryption algorithm works. This can include techniques such as renaming variables, code splitting, or code obfuscation tools. This method can be used together with **Key Hardening** techniques such as, deriving the key before encryption using multiple stored values instead of storing the key in a single data location as we can see in figure 5. Another Key hardening technique is Key Rotation. Keys should be rotated periodically to reduce the risk of them being compromised. The frequency of rotation should be based on the sensitivity of the data being protected. We can also split the DES function among multiple other functions to perform different tasks to make the DES function more difficult to find. This method being used in tandem with key derivation would make it much more difficult to reverse the code and derive the key. Another possible solution similar to code obfuscation would be to **use a niche programming language** which would be more difficult to reverse, for example, go or java [5].

VIII. AES ATTACK AND CODE HARDENING

A. AES attack method

For the AES attack as the project wanted, we used CPA or Correlation Power Analysis to detect the AES attacks via exploration of the correlation of power the chip power traces with the operation needed for doing AES encryption. Four binaries were provided by the instructors which were doing AES encryption on input data from UART and returned the encrypted data via UART. For doing the attack we should explore the correlation between the guess of power consumption for different guessed keys and the real power traces and compute the correlation coefficient between these two. Our objective is to compute the following:

$$r_{i,j} = \frac{\sum_{d=1}^{D}[(h_{d,i} - \bar{h_i})(t_{d,j} - \bar{t_j})]}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \bar{h_i})^2 \sum_{d=1}^{D}(t_{d,j} - \bar{t_j})^2}}$$

In the previous formula "d", and "i" are trace numbers, and subkey guesses respectively. "j" is the sample point in the power trace. "h" is the guess for power consumption, and "t" is the real power trace. The code is written to compute this formula and the next step is to see which guessed key

makes this value maximum and that's the best key guess for the provided binaries.

B. AES attack code hardening

You can see the power traces for binary1.hex in Figure 6. Also, the power traces for one of the other implementations by students could be seen in Figure 7. As you can see the power traces for the binary1.hex is sparsed in comparison to the other and the difference between different part of the power trace is more obvious compared to the other implementation which seems periodic in the time. These properties of the power trace make breaking the AES key for binary1.hex simpler and more straightforward compared to the implementations of the students.

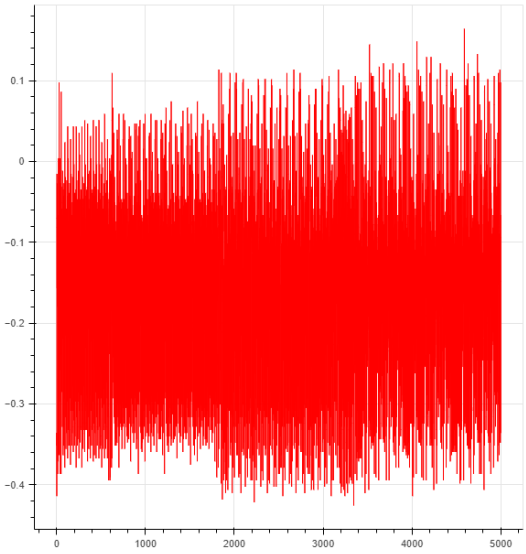

Fig. 6. Binary1.hex AES power traces

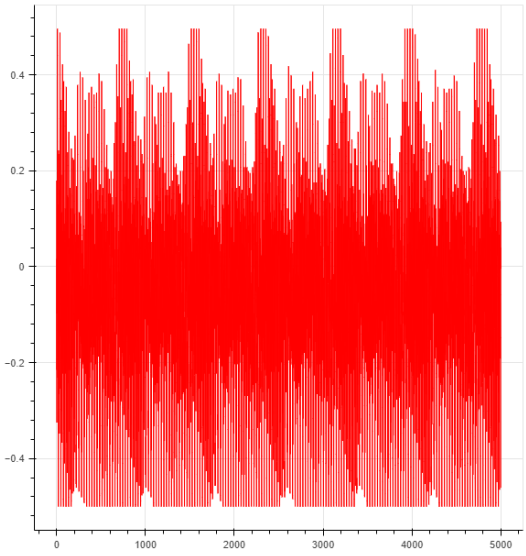

Fig. 7. fu262.hex AES power traces

For the security hardening against the CPA attack on AES, the most important point is to create background operations during the encryption and decryption performances. By doing this we can make sure that there is no easy way to infer the key via CPA attack. Actually, the CPA attack entirely depends on the correlation between the power traces and the actual mathematical operations for doing the AES encryption and decryption. If we make background operations beside the AES encryption and decryption operations then this correspondence and correlation is destroyed because the power traces no longer show just the AES encryption/decryption operations and are related to a lot of other operations as well. We can create multiple threads in the code or other methods to do this. Also if we add random arbitrary mathematical operations between different parts of the encryption/decryption in each round this correspondence could be removed at a very high rate.

IX. Discussion on practical and realistic threats

In this section, we want to discuss that based on our experimental data which of the threats in the threat model that we previously defined were realistic and which were not.

- Physical attack: This type of attack is realistic because the Chipwhisperer HW is entirely open to tampering but in this project, we didn't use it.
- Malware attacks: This type of attack doesn't seem very realistic in this project because stealing data by malware seems very difficult.
- Brute-force attacks: This type of attack is not possible in this project because the time needed for sending and receiving data on UART prohibits using of this type of attack.
- Cryptographic attacks: This type of attack is the main type that is used in all the parts of this project and because of that it was completely realistic. We used Cryptanalysis, and CPA to attack the binaries in this project. The main defense in front of the CPA attack was to create background operations to destroy the correlation analysis for AES, and also for the XOR attack we can't do anything to make it secure but if we return all-zero data in the case of failure in decryption it forces the attacker to do a brute-force at the first byte of the key and makes the attack a little more difficult.
- Software vulnerabilities: As the projects are majorly written using open-source well-written codes of encryption methods finding software vulnerabilities seems a little difficult and unrealistic in this project.
- Supply chain attacks: As there is no supply chain in this project it seems that this item was not very realistic.
- Physical/environmental vulnerabilities: As the environment for using Chipwhisperer is just a Laptop and USB port this type of attack doesn't seem very realistic.
- Fault attacks: For XOR, doing the fault attack is exactly what we need for getting the key and in the previous parts we explained how we can make it more difficult to do. But for AES, and DES techniques we don't have the needed facilities to do the fault attack on the last rounds

of operations, and because of that it doesn't seem very possible in this project!
- Scan chain attacks: As there is no scan chain in this project, it seems impossible to use this type of attack in this project, and not very realistic.

X. Insights about common weaknesses

During the course of this project, we discovered some common weaknesses and vulnerabilities in hardware devices. Some common issues and weaknesses that need to be addressed are,

- Side-channel attacks: Attacks that extract sensitive data by targeting physical characteristics of the hardware.
- Supply chain attacks: Compromising the hardware supply chain to insert malicious components.
- Hardware Trojans: Malicious components inserted during the design or manufacturing process.

To address these issues and weaknesses, there are several ways that hardware security specifications could be improved moving forward. There are now hardware security modules (HSMs) that provide secure storage and management of cryptographic keys, as well as hardware-based encryption and authentication solutions. In recent research by Mavrovouniotis et al., [6], they mention that an HSM is essentially a cryptographic engine and it serves no useful purpose if secret or private cryptographic keys are exposed to an attacker during command processing. Hence, such keys must never appear in plain form outside the secure confines of the HSM. This would ensure protection against reversing attacks we used against the DES and XOR algorithms.
Furthermore, H. Wang et al. [**?**] suggest a method to mitigate side-channel attacks using randomization. Randomizing the execution of the algorithm can help to thwart side-channel attacks. This involves introducing random delays or adding random values to the input and output data.
Ultimately, we can surmise that it is important to continue to invest in research and development of hardware security solutions and to promote standards and best practices for hardware security. This will require collaboration among manufacturers, researchers, and regulators to prioritize hardware security and address challenges and opportunities in the field.

XI. Strengths and weaknesses of Power Analysis Attack

In this section, we want to talk a little about the strengths and weaknesses of the power analysis attack. The main strength of the CPA attack is that it doesn't need very special and expensive equipment to do. Actually, you can do it easily by monitoring the hardware power supply and by using relatively simple equipment. Moreover, there is no need for tampering or special operations on the target hardware or software which makes this method simpler compared to the other methods. On the other hand, one of the weaknesses of this method is that it's not easy to do by correlation analysis and actually there could be a lot of parameters that affect the power consumption and prevent you from doing the attack as we just saw in this project for the binaries provided by the students. Moreover, another weakness is that you need a complete set of information about the software architecture and code to make this attack possible because the power consumption mainly is the result of the running code. The latter is not always possible and could make this type of attack very challenging. For the Chipwhisperer hardware it seems that it's very sensitive to the background operations and a very special condition is needed to gain the keys with this method.

XII. discussion on the identification and mitigation of side channel analysis and fault attacks throughout a product's life-cycle

Throughout a product life-cycle, there is a lot of work needed to do to assure that the product is safe against side channel analysis and fault attacks. First, it's important to make the hardware immune to tampering because a very initial stage for both types of attacks seems to access the hardware and without this access, attackers have much tougher work to do. The next thing is to make sure to destroy the correlation between the side channel characteristics of the hardware and the cryptographic operations in the hardware and it needs a lot of testing on the hardware before going for production. The next item that could help is to make sure that the hardware is not changed at the production and fabrication steps to add something to make the side channel analysis simpler. The next thing to make fault attacks more difficult is to use countermeasure techniques for fault attacks like execution redundancy, checksums on data transfers, and randomized execution.

References

[1] https://github.com/dhuertas/DES

[2] https://github.com/kokke/tiny-AES-c

[3] M. Rostami, F. Koushanfar, J. Rajendran and R. Karri, "Hardware security: Threat models and metrics," 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2013, pp. 819-823, doi: 10.1109/ICCAD.2013.6691207.

[4] J. Da Rolt, G. Di Natale, M. -L. Flottes and B. Rouzeyre, "New security threats against chips containing scan chain structures," 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, San Diego, CA, USA, 2011, pp. 110-110, doi: 10.1109/HST.2011.5955005.

[5] https://enoumen.com/2022/10/04/which-programming-language-produces-binaries-that-are-the-most-difficult-to-reverse-engineer/

[6] Mavrovouniotis, S., Ganley, M. (2014). Hardware Security Modules. In: Markantonakis, K., Mayes, K. (eds) Secure Smart Embedded Devices, Platforms and Applications. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-7915-4_17