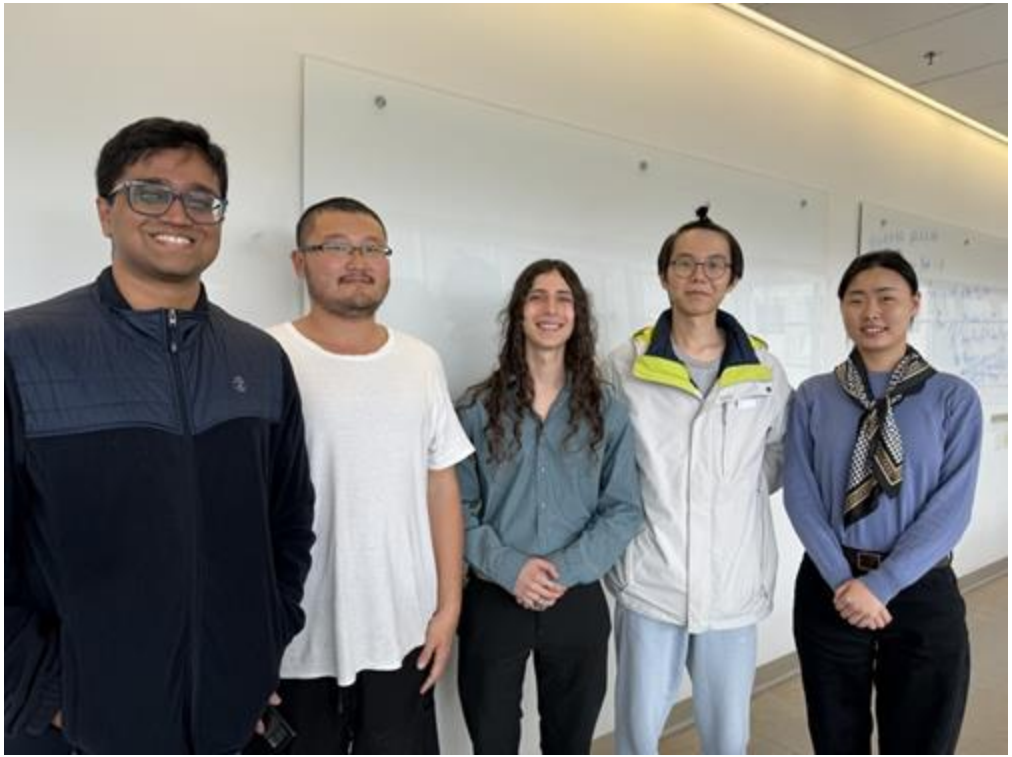
Team Cacti University at Buffalo



Team Cacti

University at Buffalo

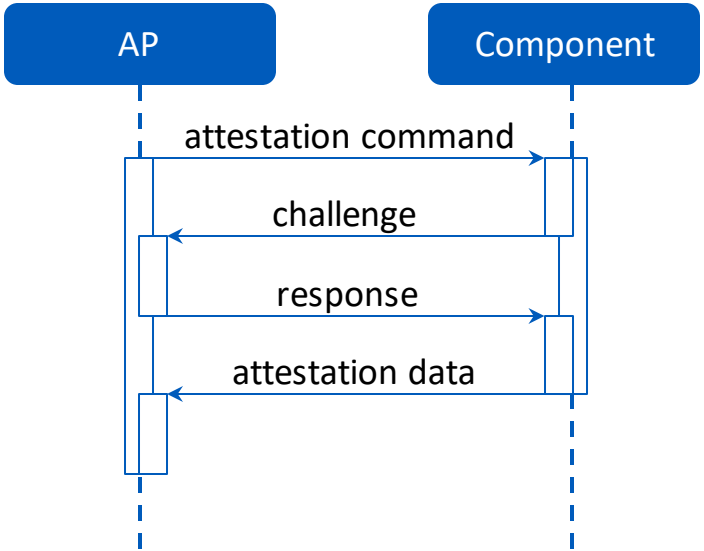
Gaoxiang Liu
Zheyuan Ma
Alex Eastman
Xi Tan
MD Armanuzzaman
Sagar Mohan
Afton Spiegel
Sai Bhargav Menta

Advised by: Prof. Ziming Zhao and Prof. Hongxin Hu

Design Overview

Design Overview

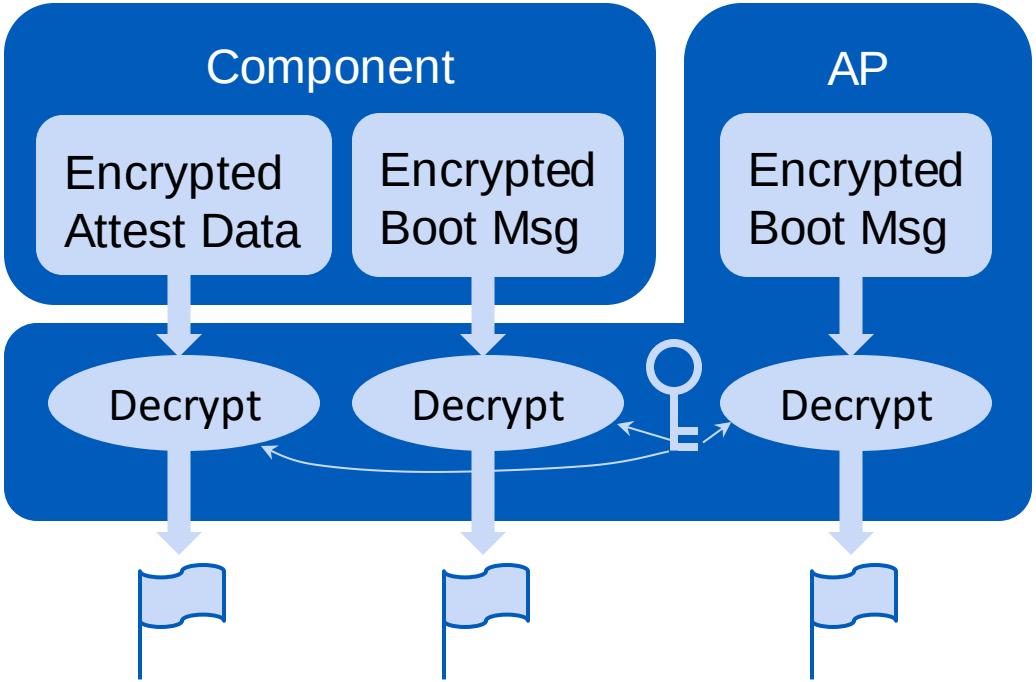
- Use the Monocypher library
- Use a challenge-response mechanism to authenticate
  - The challenge is a random number generated by the True Random Number Generator (TRNG)
  - It relies on a key shared between the AP and the component



Example: the component authenticates the AP during the attestation process

Defense Mechanisms

Sensitive Data (Flags) Encryption



Mitigating Brute-Force Attacks

- Use the Argon2 keyed-hash algorithm for the attestation PIN and replace token
  - Argon2 is for password hashing
  - Computing speed is deliberately slow
- Introduce delays in the PIN and token validation processes
- A longer delay is introduced after an unsuccessful attempt
  - The delay remains effective even after resetting the board

Mitigating Fault Injection Attacks

- Remove debugging messages and turn off the LED
  - They can be used as triggers for fault injection attacks
- Introduce random delays of several hundred CPU cycles
- Execute important conditional expressions twice
  - E.g., branching on PIN code validation

Additional Defenses

- Memory wiping
  - Zero out the memory area which contained sensitive data, such as keys, after each use
- Communication timeout
  - A timer is started after sending a message, and the response must be received before the timer expires
- Constant time comparator for the PIN code checking
  - Mitigates timing side-channel attacks

Attacks

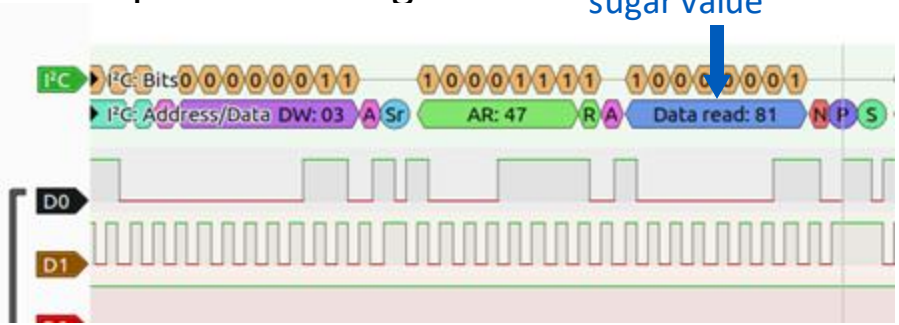
Brute-Force Attack

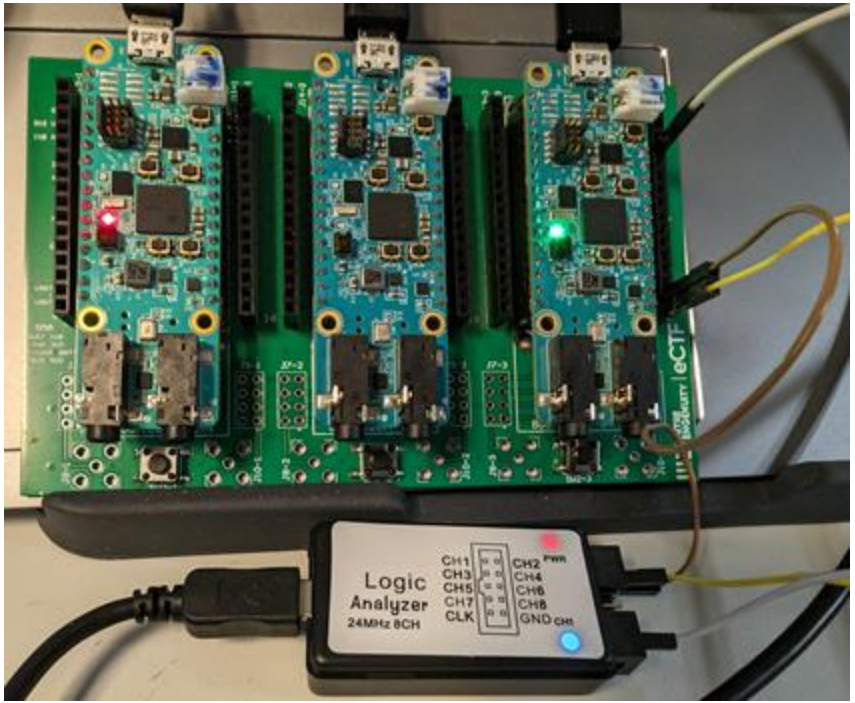
- Try all the possible PIN codes
- Use the Python UART library
- Utilize the three attack boards (3-threaded)
- Debug messages, such as “PIN Accepted!” tell when the correct PIN is found
- Finds the correct PIN within 15 hours for designs without delays

Replay Attack

- Use a logic analyzer to capture traffic on the I2C bus
- Replay specific captured messages
- The attack works if there is no message integrity check or if the checksum remains constant for a specific message

A valid high blood sugar value





Exploiting Other Design Flaws

- Same/no secrets for all deployments
  - Self-built firmware will be valid for any attack scenario
- Predictable keys
  - Global variables are 0
  - The keys, intended to be random by design, are not actually random.
- Weak/no validation
  - Sending a fixed value as the authentication token
    - The value can be captured and then replayed

Thoughts and Tips

Thoughts and Tips

- Don’t rush to submit
  - We had a buffer overflow bug last year
  - The defense points helped a lot this year
- Always encrypt sensitive data
- Use Elliptic Curve Cryptography (ECC) instead of RSA for asymmetric encryption
  - RSA will slow down the system
- Check the disassembly code from your firmware to make sure it works as expected
  - Use Ghidra or objdump
- Use multiple entropy sources for generating random numbers
- Utilize the hardware resources
  - E.g., the temperature sensor on the board last year and the TRNG on the board this year

Thank you!

Q & A