Lecture 17 (Remote Attestation)

SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust

1 Threat Model

1. System
   - immutable ROM
   - RAM erased at reset
2. Attacker
   - full control over software
   - no invasive hardware attacks
   - no side-channel attacks
3. Shared key between prover and verifier

2 Protocol

1. Verifier generates nonce n
2. Sends (n, a, b, x, x_{flag})
3. Prover sets $C = SMART(n, a, b, x, x_{flag}, -, -)$
4. If x_{flag} is true, then it runs $exec(x)$
5. It then sends C to verifier
6. If C is correct, then it accepts, else it rejects

3 Execution

1. User application starts SMART
2. Code attestation is performed using protected key (by SMART ROM code)
3. HMAC result is written to global memory at predefined location

4 Modifications to AVR and MSP430 Boards to Realize Implementation

this was discussed

5 Execution Time

1. 287ms for 1KB
2. 160ms for 512B
3. 48ms for 32B

This is very slow and inefficient

as engineers we should alleviate the issue and not manage it

5.1 Solution

1. Only analyze critical code
2. Construct a hardware accelerator for HMAC
3. Riju and Viresh working with students to find solutions

6 Time of Check Time of Use (TOCTOU)

1. When to call check?
2. How many times to call the check?
3. What if code is modified and reverted between this time period?

6.1 Solutions

1. Check randomly but have an upper bound
2. Check for smaller part of code more frequently - won't work since adversary will attack the other region