HKG18-503: My code is being hacked!

Joakim Bech and Ard Biesheuvel
Agenda

- Why?
- Terminology
- Encryption modes and IV’s
- RNG’s
- Side Channel Attacks
Why this talk?

- Eye opener for non-security developers
- Spread knowledge about a couple of (well known) pitfalls
- Side channel attacks have been a hot topic lately

Requires insight into software at a different level:

- Level 0: *Happy flow*
- Level 1: *Error/exception handling*
- Level 2: *Awareness of stack layout in memory, etc.*
- Level 3: *Awareness of ‘analog’ aspects like timing and glitching*
  
  → *Especially important for systems that may be vulnerable to HERO attacks*
Terminology

- **Authentication**
  - Verifies the identity (ex: showing ID card)

- **Integrity**
  - Ensure that something has not been modified (ex: software verification)

- **Confidentiality**
  - Keeping something secret (ex: encrypt sensitive data)

- **Non-repudiation**
  - Being accountable for actions (ex: a signature)
Block Cipher Modes - ECB

- Malleable - "easily influenced; pliable."
- Blocks are individually encrypted/decrypted
- Always same ciphertext at a given key (i.e., no randomness)
ECB bank transaction example

- Things an attacker could do:
  - Randomly change the amount
  - Replay a known amount
  - Change receiver
  - etc ...

Bank transaction

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joakim</td>
<td>Ard</td>
<td>$1</td>
</tr>
</tbody>
</table>

ECB block  ECB block  ECB block
Block Cipher Modes - CBC, CTR etc

- In almost all cases (data > blocksize), do **not** use ECB!
- Use another mode instead! Also, do **not** forget about integrity protection!
- CBC: Encryption not parallelizable, needs padding → “Padding oracle”
- CTR: Parallelizable, behaves as a stream cipher → “two time pad”
IV - Initialization Vector

- Without it the ciphertext would be the same at given key!
- IV can been seen as dummy block giving “randomness” to get the encryption going
- Does **not** have to be secret!
- Timestamp, random number even sequence number will do (in some cases)!
- Do **not** (re-)use the same IV for a given key/message!
Encrypt-then-MAC or MAC-then-encrypt?

MAC($K_m$, $E(K_e$, message))  ✔
Always provides Authenticated Encryption

$E(K_e$, MAC($K_m$, message))  ❌
Can be used, but not always secure to a chosen ciphertext attack (CCA)
RNG - Random Number Generators

- **TRNG** - True Random Number Generators
  - Impossible to reproduce (and predict)!
  - Example: Hardware, noise, clock jitter etc

- **PRNG** - Pseudo Random Number Generators
  - Generates a random sequence given by an initial seed
  - Example: `rand()` in ANSI C
  - Sometime also called **DRBG** - Deterministic Random Bit Generator

- **CSPRNG** - Cryptographically Secure Pseudorandom Number Generators
  - In short, a PRNG which is unpredictable

It is impossible to achieve reliable security without a good source for RNG!
Hello World Password check - What is wrong?

Password in binary

“OK” value != 0

Timing attack

Glitch attack

No input verification

```c
#include <stdio.h>
#include <string.h>

static char *password = "SuperSecret"

int verify_password(char *input, size_t len) {
    int login_granted = 1;
    if (memcmp(password, input, len)) {
        login_granted = 0;
    }
    return login_granted;
}

int main(int argc, char **argv) {
    char *input = argv[1];
    if (verify_password(input, strlen(input)) == 1) {
        printf("Unlocking the secret vault\n");
    } else {
        printf("Failed to authenticate\n");
    }
    return 0;
}
```
Timing attack

- Idea: Measure the running time for a certain operation
- Can totally compromise security on devices
  - Cache access (Spectre & Meltdown)
  - memcmp (comparing passwords)
  - Table based AES
  - Binary LtoR implementation (RSA - modular exponentiation)

- Strive for ...
  - Constant time comparison!
  - “Balance” operations like in the modular exponentiation

- Compilers can be an issue!

RSA private key exponentiation: bit “d”

- $d_i = 0$ (Square)
- $d_i = 1$ (Multiply)

Quicker → Square

Slower → Multiply
SPA / DPA - measure power consumption

- DPA - Differential Power Analysis
- SPA - Simple Power Analysis

- Many computations leaves distinct power consumption patterns, examples
  - Square + Multiply as mentioned on the timing attack side
  - Intermediate AES states
  - If / else ...

- These patterns can be used to mount timing attacks and other side channel attacks
Glitch attack

- Trick the CPU to make miscomputations!
  - Inject extra clock signals (needs external clock)
  - Momentarily cut the power (Vcc)

```c
int verify_password(char *input, size_t len) {
    int login_granted = 1;

    if (memcmp(password, input, len))
        login_granted = 0;

    return login_granted;
}
```

“0” means … not granted!
Tools and equipment

- x86 / Arm instructions: RDTSC, CNTxyz
- JTAG
- Oscilloscope
- Logic analyzer
- Special side channel attack tools, for example ChipWhisperer
Want to learn more?

- Coursera - Cryptography I (and II) - Dan Boneh
  - [https://www.coursera.org/learn/crypto](https://www.coursera.org/learn/crypto)

- Introduction to Cryptography - Christof Paar
  - [https://www.youtube.com/channel/UC1usFRN4LCMcflV7UjHNuQg](https://www.youtube.com/channel/UC1usFRN4LCMcflV7UjHNuQg)
Thank You

#HKG18
HKG18 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org
Time invariant AES encryption

- C implementations of AES are usually table based
  - 8 tables of 1 KB each for each direction
  - 4 tables for the inner rounds
  - 4 tables for the final round

- Tables are used to look up a 32-bit value for each byte of input

- The first round key is equal to the ‘master’ key and is xor’ed with the input before the first round → correlation

- Statistical analysis of the time it takes to individually encrypt thousands of packets can produce the key

- Solution: use slower algorithm that uses smaller tables, and preload them in the caches
  → crypto/aes_ti.c in the kernel tree is such a C implementation, individual architectures usually have SIMD or special instruction based ones