

Side-channel Attacks & Data remanence

Magnus Almgren, Erland Jonsson

Subset from the syssec course repository by:
Christian Platzer and Markus Kammerstetter
(inetsec@seclab.tuwien.ac.at)

Data remanence

- Often, the delete command does not really delete a file ...
 - See discussion in course book¹
 - Layers: OS – firmware in drive etc.
- Data remanence is
 - “Residual information remaining on storage media after clearing. “

¹Computer Security, Principles and Practice, 2nd edition, Stallings, Brown: p404—p407

A password check (0)

```
bool check_password(char *passwd)
{
for (int i=0; i<pass_len; i++)
{
    if (passwd[i] != stored_passwd[i])
        return false;
}

return true;
}
```

Side-channel attacks 1

- A **side channel attack** is any attack based on information gained from the *physical implementation* of an embedded system, for example a cryptosystem
- It normally requires **physical access** to the hardware
- A side-channel attack is an attack based on side-channel information, i.e. “extra” information that can be retrieved from the device.

Side-channel attacks 2

- Types of side-channel attacks:
 - **Timing attack** – attacks based on measuring how much time various computations take to perform.
 - **Power monitoring attack** - attacks based on observing the varying power consumption by the hardware during computation
 - **Electromagnetic attacks** – based on observing electromagnetic emanation, cp. TEMPEST (Sw. RÖjande Strålning = RÖS)
 - **Acoustic cryptanalysis** - attacks which exploit the sound produced during a computation
 - **Differential fault analysis** - in which secrets are discovered by introducing faults in a computation.

Timing Analysis

*Int. Secure Systems Lab
Technical University Vienna*

- We interact with the system and closely monitor the timing during operation
- For example, we might monitor the timing of a password check
 - Start timer when guessed password is sent to device
 - Stop timer when response (i.e. 'wrong password') is received
 - Compare time measurements for different guesses
 - See if we can draw a conclusion and determine the correct password

A bad password check (1)

```
bool check_password(char *passwd)
{
for (int i=0; i<pass_len; i++)
{
    if (passwd[i] != stored_passwd[i])
        return false;
}


return true;
}
```

A bad password check (2)

Terminates as soon a byte is wrong

```
bool check_password(char *password)
{
    for (int i=0; i<pass_len; i++)
    {
        if (passwd[i] != stored_passwd[i])
            return false;
    }

    return true;
}
```



➔ Based on timing information, we can easily guess the password

A better password check (1)

```
bool check_password(char *passwd)
{
    int err=0;
    for (int i=0; i<pass_len; i++)
    {
        err |= passwd[i] ^ stored_passwd[i];
    }

    if (err != 0)
        return false;
    return true;
}
```

A better password check (2)

```
bool check_password(char *passwd)
{
    int err=0;
    for (int i=0; i<pass_len; i++)
    {
        err |= passwd[i] ^ stored_passwd[i];
    }

    if (err != 0)
        return false;
    return true;
}
```



Constant time

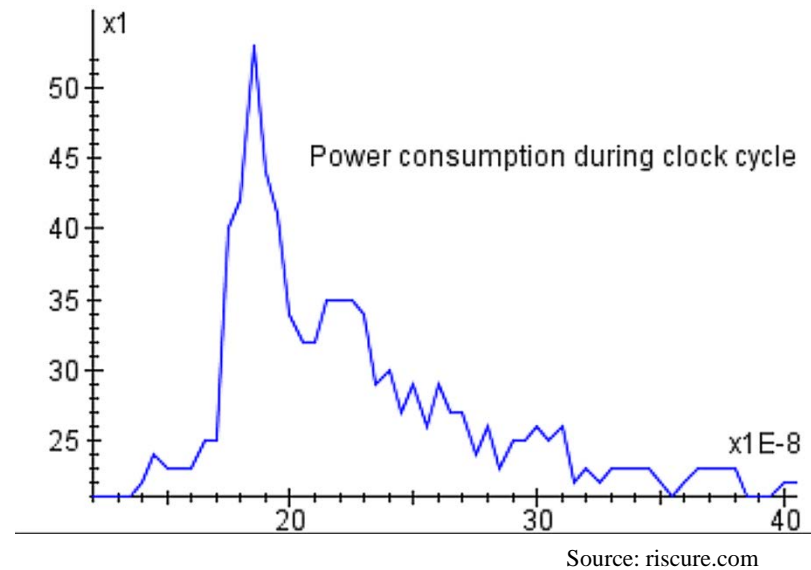
Simple Power Analysis (1)

*Int. Secure Systems Lab
Technical University Vienna*

- The power consumption of a processor depends on the instruction executed
- We closely monitor the power consumption during clock cycles (i.e. time domain)
- For a given instruction, the power consumption also depends on the data processed

Simple Power Analysis (2)

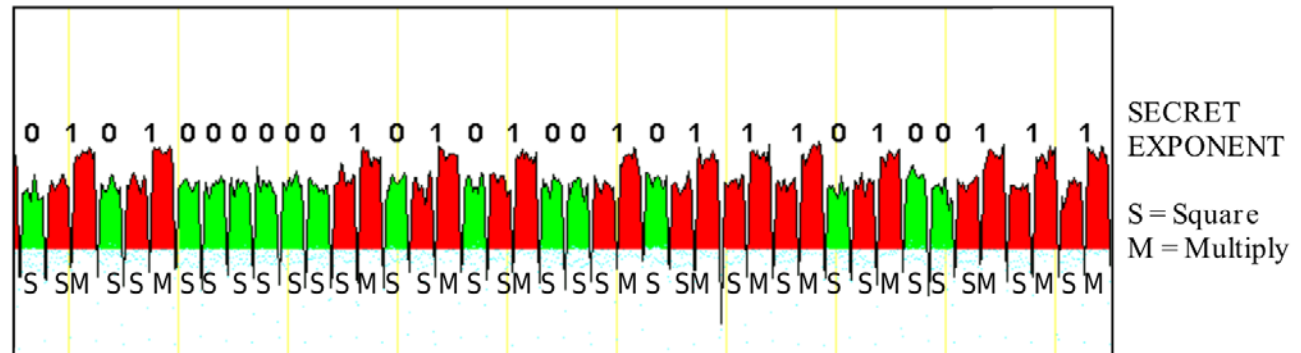
- Example:



- By analyzing the power consumption, it might be possible to determine the *instruction* and *data*

Vulnerable RSA exponentiation

- Example:



Source: 'Protecting FPGAs from Power Analysis', Cryptography Research Inc.

- Using SPA, we can completely recover the RSA secret key in this case !
- Drawback: We need a good S/N ratio for SPA to work, can be increased through averaging over multiple measurements