

A Quick Intro to Physical Attacks

Embedded Physical Attacks 101 @ HardwareIO

Lejla Batina

Kostas Papagiannopoulos

Radboud University, The Netherlands

NXP Semiconductors, Germany

How did we get here?

- Crypto in the 70s and 80s

- Crypto in the 90s and 00s

- Crypto in the 10s and 20s

What are Physical Attacks?

- Side-Channel Attacks

- Fault Injection Attacks

What do we cover in this course?

- Crypto in this course

- Side-Channels in this course

- Fault Injection in this course

How did we get here?



- ▶ The birth of modern cryptography (and rock music)

- ▶ The birth of modern cryptography (and rock music)
- ▶ DES (1975), RSA (1977) and many other early designs surfaced

- ▶ The birth of modern cryptography (and rock music)
- ▶ DES (1975), RSA (1977) and many other early designs surfaced
- ▶ Pure mathematical structures were proposed and analyzed

- ▶ The birth of modern cryptography (and rock music)
 - ▶ DES (1975), RSA (1977) and many other early designs surfaced
 - ▶ Pure mathematical structures were proposed and analyzed
 - ▶ We moved away from ‘closed’ approaches in favor of an open community of cryptanalysis
-

- ▶ The birth of modern cryptography (and rock music)
- ▶ DES (1975), RSA (1977) and many other early designs surfaced
- ▶ Pure mathematical structures were proposed and analyzed
- ▶ We moved away from ‘closed’ approaches in favor of an open community of cryptanalysis
- ▶ **Workflow:**

- ▶ The birth of modern cryptography (and rock music)
- ▶ DES (1975), RSA (1977) and many other early designs surfaced
- ▶ Pure mathematical structures were proposed and analyzed
- ▶ We moved away from ‘closed’ approaches in favor of an open community of cryptanalysis
- ▶ **Workflow:**
 1. Construct a cipher

- ▶ The birth of modern cryptography (and rock music)
 - ▶ DES (1975), RSA (1977) and many other early designs surfaced
 - ▶ Pure mathematical structures were proposed and analyzed
 - ▶ We moved away from ‘closed’ approaches in favor of an open community of cryptanalysis
 - ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
-

- ▶ The birth of modern cryptography (and rock music)
 - ▶ DES (1975), RSA (1977) and many other early designs surfaced
 - ▶ Pure mathematical structures were proposed and analyzed
 - ▶ We moved away from ‘closed’ approaches in favor of an open community of cryptanalysis
 - ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. Put it on the mainframe
-



- ▶ The advent of personal computing, the Internet (and hiphop music)

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used

- ▶ The advent of personal computing, the Internet (and hiphop music)
 - ▶ Everything is connected on the network
 - ▶ Every computer was vulnerable unless strong encryption was used
 - ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
-

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used
- ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
- ▶ AES should be mathematically secure *and* computationally efficient

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used
- ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
- ▶ AES should be mathematically secure *and* computationally efficient
- ▶ **Workflow:**

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used
- ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
- ▶ AES should be mathematically secure *and* computationally efficient
- ▶ **Workflow:**
 1. Construct a cipher

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used
- ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
- ▶ AES should be mathematically secure *and* computationally efficient
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used
- ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
- ▶ AES should be mathematically secure *and* computationally efficient
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. Make sure it is fast

- ▶ The advent of personal computing, the Internet (and hiphop music)
- ▶ Everything is connected on the network
- ▶ Every computer was vulnerable unless strong encryption was used
- ▶ Consensus was reached: every device should be well protected, so lets build AES (1998)
- ▶ AES should be mathematically secure *and* computationally efficient
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. Make sure it is fast
 4. Put it on your PC



- ▶ Smartphone revolution, the Internet of Things (and pop music)

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker
- ▶ **Workflow:**

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker
- ▶ **Workflow:**
 1. Construct a cipher

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. Make sure it is very efficient

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. Make sure it is very efficient
 4. Put it on every imaginable device

- ▶ Smartphone revolution, the Internet of Things (and pop music)
- ▶ An always-online army of embedded devices
- ▶ Smartphones, smartcards, USB dongles, RFID transport cards
- ▶ Every single one of them containing critical data
- ▶ Every single one of them is given to a potential attacker
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. Make sure it is very efficient
 4. Put it on every imaginable device
 5. Give such devices to everyone

- ▶ This was the breaking point

- ▶ This was the breaking point
- ▶ Hands-on device access was easy

- ▶ This was the breaking point
- ▶ Hands-on device access was easy
- ▶ Everyone could try hardware hacking!

- ▶ This was the breaking point
- ▶ Hands-on device access was easy
- ▶ Everyone could try hardware hacking!
- ▶ Many products were found insecure!

- ▶ This was the breaking point
- ▶ Hands-on device access was easy
- ▶ Everyone could try hardware hacking!
- ▶ Many products were found insecure!



What are Physical Attacks?

Input: 4-digit PIN code, Output: PIN verified or rejected

Process CheckPIN (pin[4])

```
int pin_ok=0;
if (pin[0]==5)
    if (pin[1]==9)
        if (pin[2]==0)
            if (pin[3]==2)
                pin_ok=1;
            end
        end
    end
end
return pin_ok;
```

Input: 4-digit PIN code, Output: PIN verified or rejected

Process CheckPIN (pin[4])

```
int pin_ok=0;
if (pin[0]==5)
    if (pin[1]==9)
        if (pin[2]==0)
            if (pin[3]==2)
                pin_ok=1;
            end
        end
    end
end
return pin_ok;
```

- ▶ What are the execution times of the process for PIN inputs [0,1,2,3], [5,3,0,2], [5,9,0,0]

Input: 4-digit PIN code, Output: PIN verified or rejected

Process CheckPIN (pin[4])

```
int pin_ok=0;
if (pin[0]==5)
  if (pin[1]==9)
    if (pin[2]==0)
      if (pin[3]==2)
        pin_ok=1;
      end
    end
  end
end
end
return pin_ok;
```

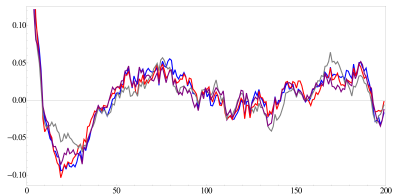
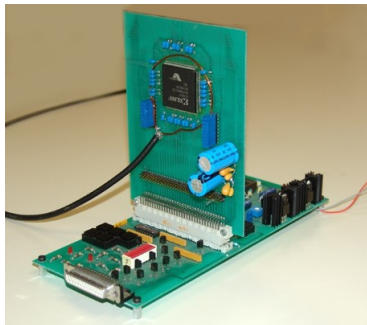
- ▶ What are the execution times of the process for PIN inputs [0,1,2,3], [5,3,0,2], [5,9,0,0]
- ▶ The execution time increases as we get closer to [5,9,0,2]

- ▶ A Side-Channel Attack *observes* data from the physical device, then uses it to bypass security

- ▶ A Side-Channel Attack *observes* data from the physical device, then uses it to bypass security
- ▶ **Workflow:**

- ▶ A Side-Channel Attack *observes* data from the physical device, then uses it to bypass security
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. **Make sure it is that is resists side-channel attacks**
 4. Give such devices to everyone

- ▶ A Side-Channel Attack *observes* data from the physical device, then uses it to bypass security
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. **Make sure it is that is resists side-channel attacks**
 4. Give such devices to everyone
- ▶ Several side-channels exist: timing, power consumption, electromagnetic emission of the device



Input: 128-bit plaintext, Output: 128-bit ciphertext

Process AES (plaintext)

```
for round = 1 until 10
  AddRoundKey;
  Sbox;
  ShiftRows;
  MixColumns;
end
return ciphertext;
```

Input: 128-bit plaintext, Output: 128-bit ciphertext

Process AES (plaintext)

```
for round = 1 until 10
  AddRoundKey;
  Sbox;
  ShiftRows;
  MixColumns;
end
return ciphertext;
```

- ▶ What if we glitch the voltage just enough to zero the 'round' variable?

Input: 128-bit plaintext, Output: 128-bit ciphertext

Process AES (plaintext)

```
for round = 1 until 10
  AddRoundKey;
  Sbox;
  ShiftRows;
  MixColumns;
end
return ciphertext;
```

- ▶ What if we glitch the voltage just enough to zero the 'round' variable?
- ▶ Then we essentially stop the AES encryption

- ▶ A Fault Injection Attack *alters* data in the physical device, then uses this to bypass security

- ▶ A Fault Injection Attack *alters* data in the physical device, then uses this to bypass security
- ▶ **Workflow:**

- ▶ A Fault Injection Attack *alters* data in the physical device, then uses this to bypass security
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. **Make sure it is that is resists side-channel and fault injection attacks**
 4. Give such devices to everyone

- ▶ A Fault Injection Attack *alters* data in the physical device, then uses this to bypass security
- ▶ **Workflow:**
 1. Construct a cipher
 2. Make sure it resists mathematical attacks
 3. **Make sure it is that is resists side-channel and fault injection attacks**
 4. Give such devices to everyone
- ▶ Several fault injection techniques exist: power glitching and electromagnetic glitching, laser injections



What do we cover in this course?

- ▶ **Crypto reminders:** we will discuss standard algorithms such as AES and RSA
- ▶ No need to implement them, we assume they are already in place, inside embedded devices
- ▶ We need to remember their basic structure: so try to freshen them up in Wikipedia ☺

- ▶ **Data analysis:** We will provide you with side-channel measurements for several devices, then we will teach you how to code standard side-channel attacks
- ▶ We will use the developed code to analyse measurements and extract the secret crypto key
- ▶ We will cover most standard side-channel techniques such as simple power analysis, differential power analysis and template attacks
- ▶ **Data capturing:** We also show how to perform side-channel measurements on the ChipWisperer device
- ▶ We recommend coding in Matlab or Octave: So do try to freshen your basic coding skills ☺

- ▶ Using the ChipWhisperer setup we will produce voltage-glitching effects
- ▶ We will use them to recover the secret key of RSA
- ▶ We recommend coding in Python: So do try to freshen your basic coding skills of standard Python and NumPy ☺



- ▶ Course Description: <https://hardware.io/berlin-2020/training/embedded-physical-attacks-101.php>
- ▶ Lejla Batina is currently a professor in the Digital Security group of the Computing Science Department at the Radboud University and active participant in the CHES and IACR communities.
- ▶ <https://www.cs.ru.nl/~lejla/>
- ▶ Kostas Papagiannopoulos is currently a senior vulnerability analyst in NXP semiconductors Hamburg and Ph.D. candidate in the Digital Security group of Radboud University in Nijmegen, the Netherlands.
- ▶ <https://kpcrypto.net/>
- ▶ Contact us: lejla@cs.ru.nl, kostaspap88@gmail.com