Cache Timing Attacks in Cryptography

Erik Zenner

Technical University Denmark (DTU)
Institute for Mathematics
e.zenner@mat.dtu.dk

DTU, Oct. 10, 2007

1. Introduction
2. Side-channel Attacks
3. Cache-timing Attacks
4. The AES Case
5. Final Remarks

Outline

1 Introduction

2 Side-channel Attacks

3 Cache-timing Attacks

4 The AES Case

5 Final Remarks

Personal Background

Name: Erik Zenner

- 1971: Born in Germany
- 1991-1993: Education as banker (Frankfurt)
- 1993-1999: Studied business and computer science (Mannheim, Edinburgh)
- 1999-2004: PhD in cryptography (Mannheim)
- 2004-2007: Chief cryptographer for Cryptico A/S (Copenhagen)
- Since March 2007: Assistant professor at MAT

Cryptography

What is cryptography?

Traditionally: The science of "secret codes", i.e. hiding the meaning of a message such that no unauthorised person can understand it.

Cryptography

What is cryptography?

Traditionally: The science of "secret codes", i.e. hiding the meaning of a message such that no unauthorised person can understand it.

More modern: "Cryptography is about communication in the presence of adversaries" (Ron Rivest, 1990).

Cryptography

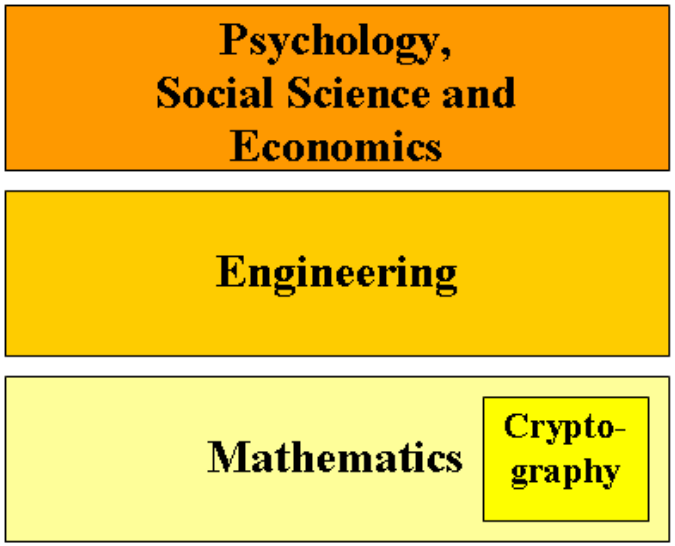
What is cryptography?

Traditionally: The science of "secret codes", i.e. hiding the meaning of a message such that no unauthorised person can understand it.

More modern: "Cryptography is about communication in the presence of adversaries" (Ron Rivest, 1990).

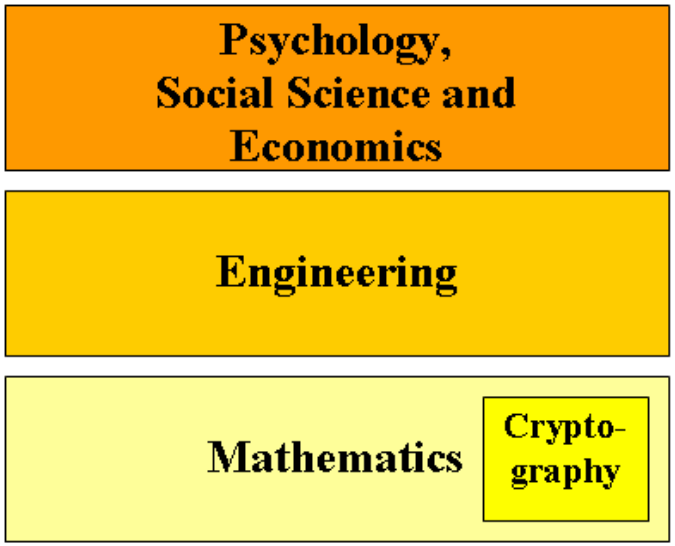
My understanding: Cryptography is the mathematical part of information security.

Cryptography in Context



Cryptography is just one of the sciences contributing the information security.

Cryptography in Context



Cryptography is just one of the sciences contributing the information security.

My interest:
Mediator between cryptography and neighbouring disciplines.

- Communicate (useful) theory to practitioners
- Develop practical solutions
- Feedback from "the real world" to theoreticians

In this talk

- Discuss *cache-timing attacks*
- Introductory level, no maths (sorry)!
- Show how encryption that seems unbreakable in theory can be breakable in practice

In this talk

- Discuss *cache-timing attacks*
- Introductory level, no maths (sorry)!
- Show how encryption that seems unbreakable in theory can be breakable in practice

Old saying:

In theory, there is not difference between theory and practice.
In practice, there is.

Outline

1 Introduction

2 Side-channel Attacks

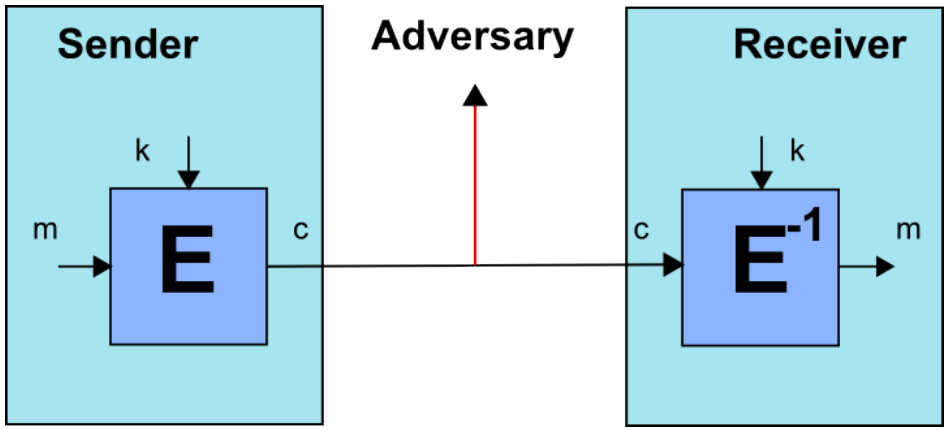
3 Cache-timing Attacks

4 The AES Case

5 Final Remarks

Shannon's communication model

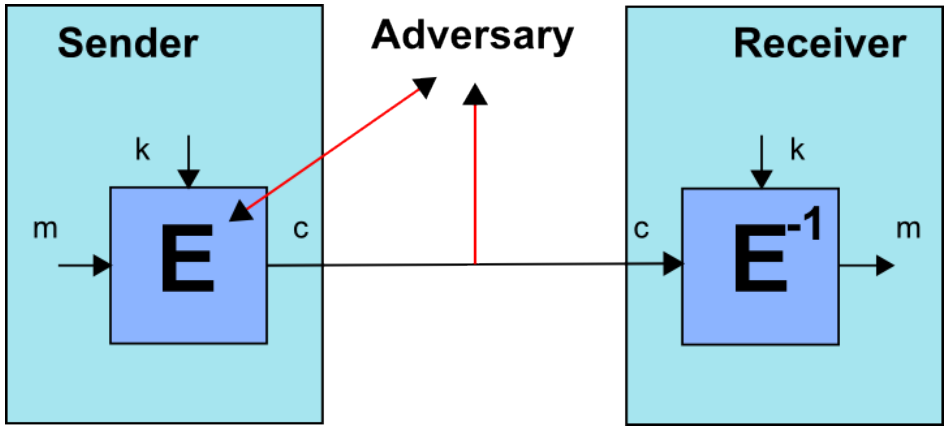
Traditional model (Shannon, 1949):



- Adversary knows the encryption algorithm E, but not the key k.
- Adversary observes ciphertext c.
- Adversary knows plaintext statistics, or maybe even part of plaintext m.

Side-channel attacks

Modified model:



In reality, additional information may leak to the attacker:

- Timing information
- Power consumption
- Fault induction

Relevance

Shannon model: Encryption happens in a secure box.

Side-channel model: Adversary has access to that box and can obtain certain real-world information.
Examples:

- Smart card
- Shared computer

Relevance: Not always, but sometimes!

Responsibility: Cryptographers or engineers?

Outline

1 Introduction

2 Side-channel Attacks

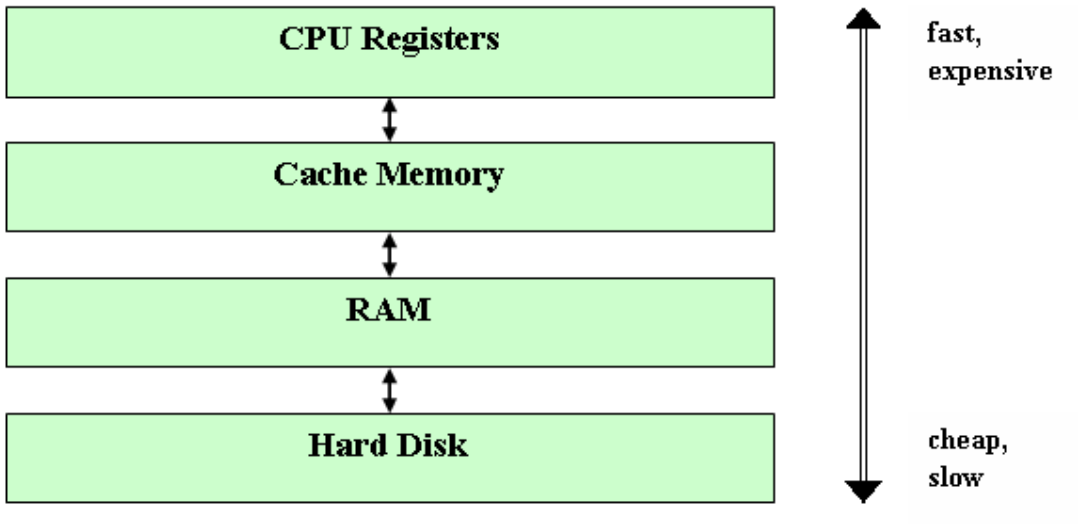
3 Cache-timing Attacks

4 The AES Case

5 Final Remarks

Memory Hierarchy (Simplified)

In a modern computer, different types of memory are used (simplified):



While CPU, RAM, and hard disk are protected against use by another user on the same machine, the cache is not.

Cache Workings (Simplified)

Motivation: Loading data from cache is much faster than loading data from RAM (by a factor of ≈ 10).

Working principle: Proceed as follows (simplified):

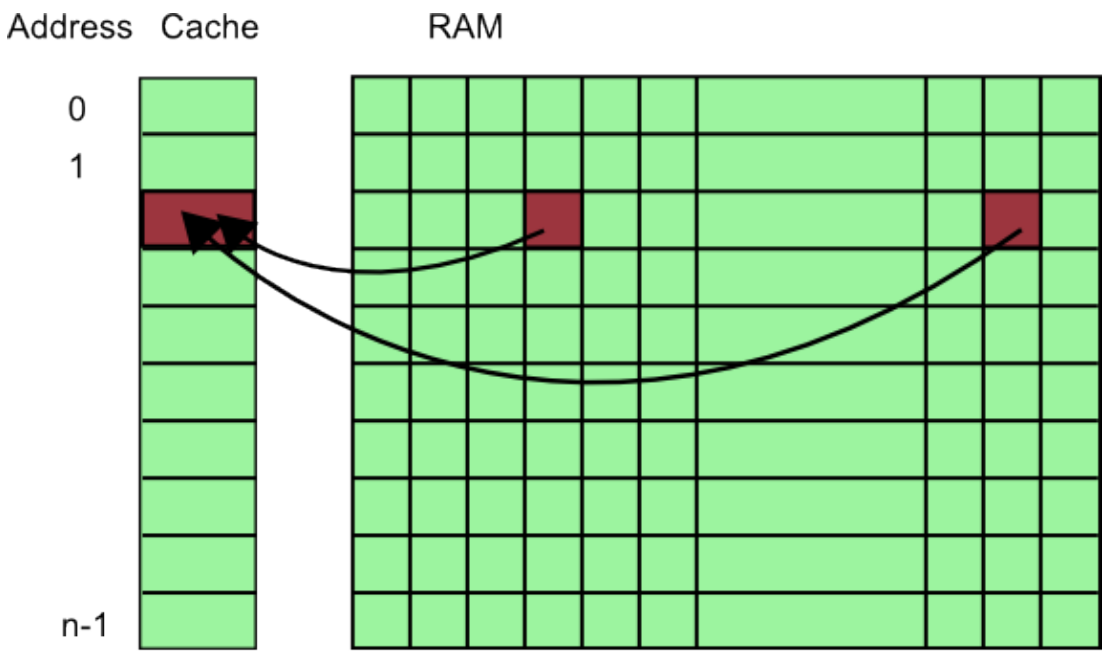
- If data from RAM address (a) is requested by the CPU, then check in the cache first.
- If present, use directly from cache.
- Otherwise, load data into cache address ($a \bmod n$), for future re-use.

Idea: Data that is used now will more likely be used again in the future.
$\Rightarrow$ Keeping copies in cache reduces the average loading time.

Cache Eviction (Simplified)

Problem: Cache is much smaller than RAM.

Consequence: Many RAM entries compete for the same place in cache.



Handling: New data overwrites old data (First in, first out).

Sample Attack Setting

Imagine two users A and B sharing a CPU. If user A knows that user B is about to encrypt, he can proceed as follows:

1. A fills all of the cache with his own data, then he stops working.
2. B does his encryption.
3. A measures loading times to find out which of his data have been pushed out of the cache.

This way, A learns which cache addresses have been used by B.
$\Rightarrow$ Internal (side-channel) information not available in the mathematical model!

Outline

1 Introduction

2 Side-channel Attacks

3 Cache-timing Attacks

4 The AES Case

5 Final Remarks

Advanced Encryption Standard (AES)

Officially: Encryption standard for (non-classified) U.S. government use.

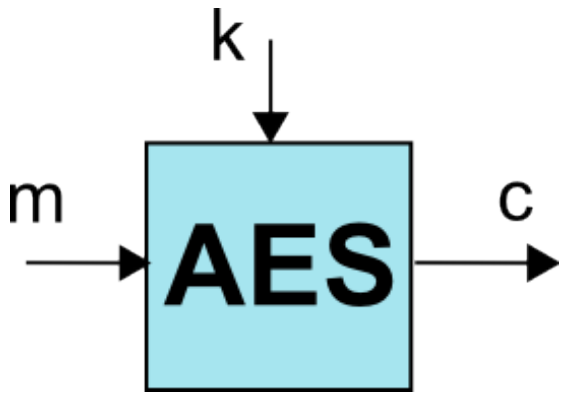
De facto: World standard for encryption.

- 1997: NIST calls for candidate algorithms
- 1998: 15 algorithms submitted (also by Lars Knudsen)
- 1998-2000: *Demolition Derby*
- 2000: Selection of the winner (Rijndael)
- 2001: Publication of the standard

$\Rightarrow$ Widely evaluated, considered to be secure.

What need to know about AES...

AES-128 transforms a 16-byte plaintext $m = (m_0, \ldots, m_{15})$ into a 16-byte ciphertext $c = (c_0, \ldots, c_{15})$, using a 16-byte key $k = (k_0, \ldots, k_{15})$.

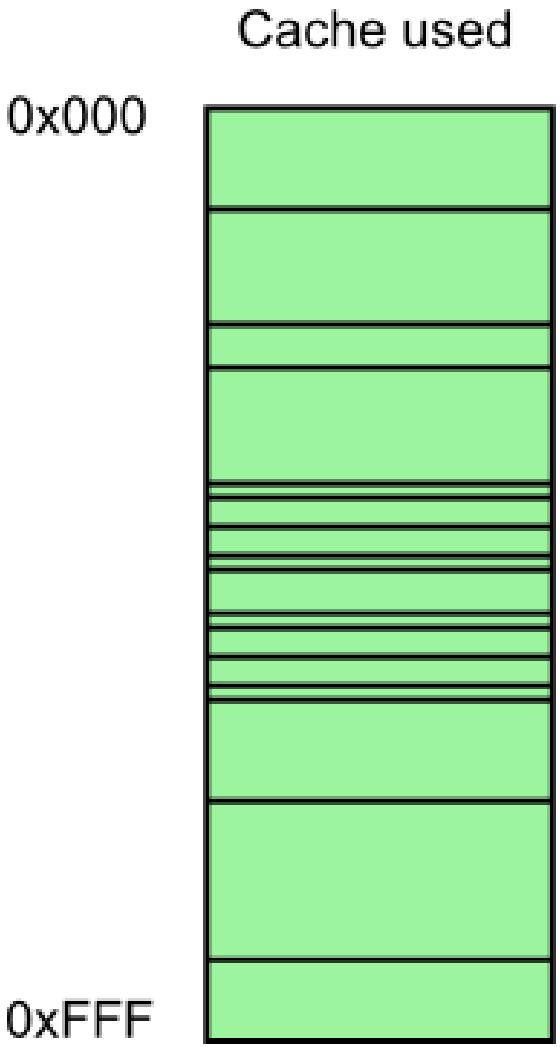


We give no full description of AES here.
All we need to know is step 1 of a typical AES-128 implementation:

- For all $i = 0, \ldots, 15$:
  Look up $F_i[m_i \oplus k_i]$ in a big table F_i.

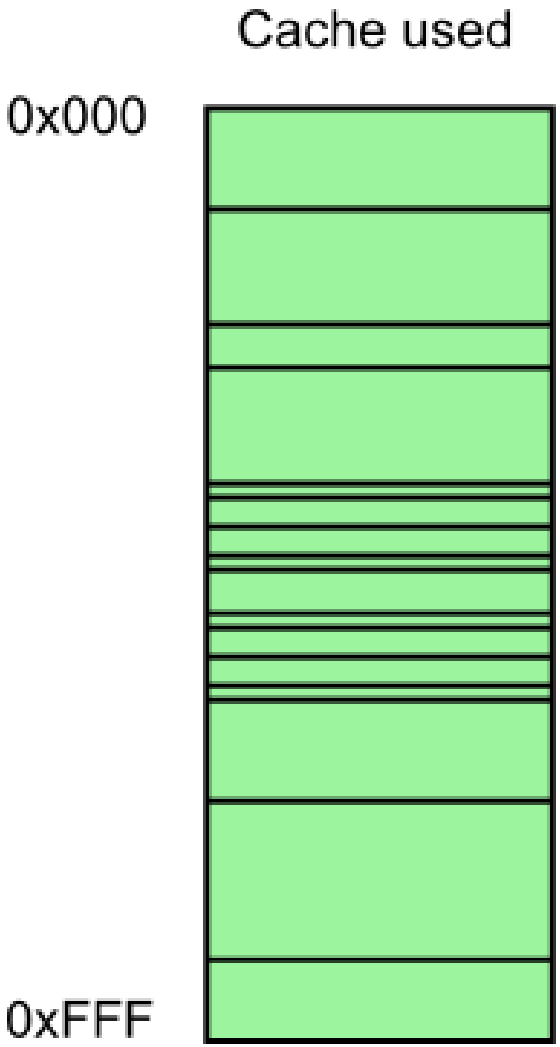
We ignore all the remaining steps here, we just point out that they also make use of table F_i.

Cache Timing Attack against AES (1)



1. Running a cache timing attack gives the adversary a table with this structure.

Cache Timing Attack against AES (1)



1. Running a cache timing attack gives the adversary a table with this structure.
2. We can clearly see where the table F_i lies in cache.
3. We can also see which entries in F_i have been accessed.

Cache Timing Attack against AES (2)

1. This gives us a list L of indices a for which $F_i[a]$ has been used.
2. In step 1, AES accessed the table for $F_i[m_i \oplus k_i]$.
   $\Rightarrow m_i \oplus k_i$ must be in L!
3. If we know plaintext m_i:
   1. Make list of candidates for $k_i = a \oplus m_i \ \ \forall a \in L$.
   2. Re-run attack and intersect the resulting lists.
4. Else:
   1. Use plaintext statistics.

Consequences

Shannon's model: The best known attack against AES-128 is brute-force key search:
Can be achieved by 10^{22} Pentium 4 processors within 100 years.

Side-channel model: One possible attack is a cache timing attack:
Can be achieved by 1 Pentium 4 processor within a few microseconds.

Consequences

Shannon's model: The best known attack against AES-128 is brute-force key search:
Can be achieved by 10^{22} Pentium 4 processors within 100 years.

Side-channel model: One possible attack is a cache timing attack:
Can be achieved by 1 Pentium 4 processor within a few microseconds.

Remember:

In theory, there is not difference between theory and practice.
In practice, there is.

Outline

1 Introduction

2 Side-channel Attacks

3 Cache-timing Attacks

4 The AES Case

5 Final Remarks

Practical Difficulties

For didactical reasons, we worked with a simplified cache model.

Real-world complexities include:

- Other processes (e.g. system processes) use the cache, too.
  $\Rightarrow$ We can not tell "encryption" cache accesses apart from others.
- Cache data is stored in larger blocks.
  $\Rightarrow$ We can only see which table areas were used, but not the precise table entry.

Nonetheless, as it turns out, these difficulties can be overcome in practice.

Developing Research Fields

Research questions on cache-timing attacks:

- Vulnerability of various cryptographic primitives:
  - Can we attack stream ciphers, MACs, digital signatures, etc.?
- New attacks based on cache timings:
  - In what ways can an attacker make use of a cache timing weakness?
- Cryptographic counter-measures:
  - How can we design a cipher such that no cache-timing attack is possible?
- Engineering counter-measures:
  - How can we implement a cipher such that no cache-timing attack is possible?

Wrapping Up

- We have learned about the difference between theoretical and practical cryptography.
- We have seen how a mathematically secure encryption function can break down completely due to implementation issues.
- We have (hopefully) learned that co-operation between theoreticians and practitioners is necessary to avoid such problems.

Wrapping Up

- We have learned about the difference between theoretical and practical cryptography.
- We have seen how a mathematically secure encryption function can break down completely due to implementation issues.
- We have (hopefully) learned that co-operation between theoreticians and practitioners is necessary to avoid such problems.

Questions?