NTNU | Norwegian University of Science and Technology

SIDE-CHANNEL ATTACKS 1: INTRO

TTM4205 – Lecture 9

Tjerand Silde

11.09.2026

Contents

Contents

Reference Material

These slides are based on:

- The referred papers in the slides
- JPA: parts of chapters 4 and 12
- DW: parts of chapter 13
- HHH: parts of chapters 1, 8, and 14

HHH is *The Hardware Hacking Handbook* by van Woudenberg and O'Flynn. Colin O'Flynn designed the ChipWhisperer boards we use in the lab.

Where This Fits

Side channels take up five lectures and the whole lab:

- **Today:** what leaks, how we classify attacks, and how we defend
- **Lecture 10:** setting up the ChipWhisperer hardware for the lab
- **Lecture 11:** power analysis against AES
- **Lecture 12:** timing and power attacks against RSA and ECC
- **Lecture 15:** side channels in post-quantum schemes

The lab is worth 30 points of your grade, with deadline December 4th at 23:59.

Contents

Black Box Crypto

We design the security of a cryptographic scheme to follow Kerckhoffs's principle: if everything about the scheme, except for the key, is known, then the scheme should be secure.

We then analyze the scheme mathematically as black-box algorithms that take some (public or secret) input and give some (public or secret) output, and prove that it is secure concerning the algorithm description and the public data with respect to the underlying hardness assumptions.

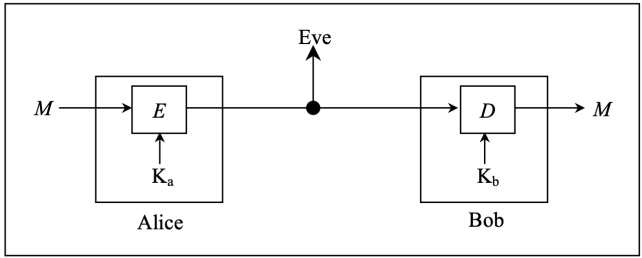


Figure 1: The traditional cryptographic model

Figure: https://csrc.nist.gov/csrc/media/events/physical-security-testing-workshop/documents/papers/physecpaper19.pdf

Black Box Crypto

However, security depends on your model. In practice, it matters how these algorithms are implemented and what kind of information the *physical* system leaks about the inner workings of the algorithm computing on secret data.

Q: What kind of information do you think might leak?

Leakage

- The time it takes to compute
- The power usage while computing
- The electromagnetic radiation...
- The temperature increase...
- The memory pattern accessed...
- The sounds your laptop makes...
- The clock frequency the processor picks...

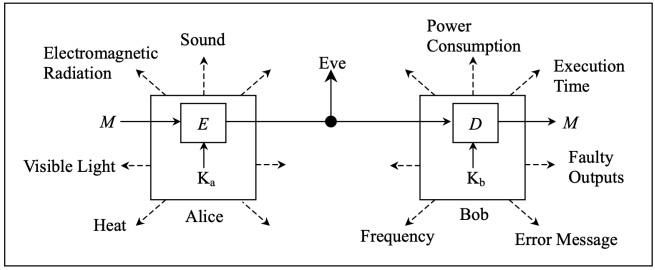


Figure 2: The cryptographic model including side-channel

Figure: https://csrc.nist.gov/csrc/media/events/physical-security-testing-workshop/documents/papers/physecpaper19.pdf

Figure: `https://eprint.iacr.org/2013/857.pdf`

Acoustic Cryptanalysis

Genkin, Shamir, and Tromer extract full 4096-bit RSA decryption keys from GnuPG on ordinary laptops, *within an hour*, from the sound the machine makes while decrypting chosen ciphertexts.

The measurement is done either with a plain mobile phone placed next to the computer, or with a sensitive microphone 4 meters away.

The acoustic channel carries under 20 kHz, six orders of magnitude below the clock rate. The leak is not the individual instruction, but the *aggregate* power draw of a long secret-dependent computation.

Examples

Q: What are real-world situations where side-channel information might leak?

Examples

- Credit cards connecting to ATMs
- Applications sharing resources
- Some publicly available crypto API
- Malware on your phone or laptop
- Crypto currency hardware wallet
- Cloud key management systems

Contents

Side-Channel Attacks

Side-channel attacks (SCA) are attacks that exploit *physical leakage* in an implemented scheme to break the underlying cryptography, that is, by extracting the secret keys.

We can categorize the attacks in several different ways.

Q: Can you, based on the list of leakage, imagine how?

Side-Channel Attacks

- Remote vs physical attacks
- Software vs hardware attacks
- Passive vs active attacks
- Invasive vs non-invasive attacks

The Attacker Model

Before an attack means anything we must say what the adversary can do:

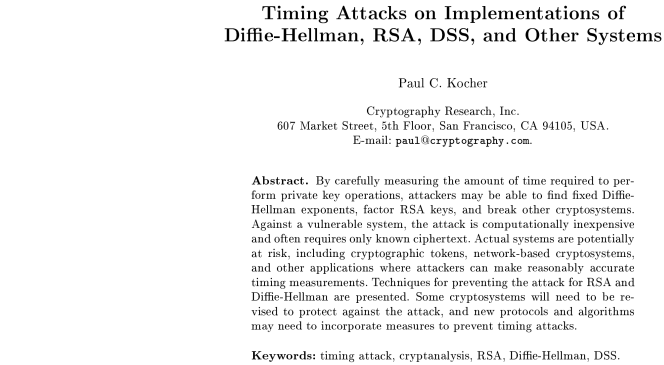
- **Access:** remote queries, co-located process, or the device in hand
- **Measurement:** what is observed, and how precisely
- **Control:** chosen inputs, repeated runs, resets to a known state
- **Budget:** how many measurements, and how much noise per trace

This is why “is this implementation secure?” has no answer on its own. Secure *against whom*, with *what* access, is the only question with content.

Remote vs Physical Attacks

Some side-channel attacks can be executed **remotely**, given information about how the algorithm is computed and access to timings, shared microarchitecture, or remotely shared sound.

For example decryption or signing queries online (remote server or WLAN), a process sharing your cache in a cloud instance, or sound through a feed such as a video call.



Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher

Cryptography Research, Inc.
607 Market Street, 5th Floor, San Francisco, CA 94105, USA.
E-mail: paul@cryptography.com.

Abstract. By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Against a vulnerable system, the attack is computationally inexpensive and often requires only known ciphertext. Actual systems are potentially at risk, including cryptographic tokens, network-based cryptosystems, and other applications where attackers can make reasonably accurate timing measurements. Techniques for preventing the attack for RSA and Diffie-Hellman are presented. Some cryptosystems will need to be revised to protect against the attack, and new protocols and algorithms may need to incorporate measures to prevent timing attacks.

Keywords: timing attack, cryptanalysis, RSA, Diffie-Hellman, DSS.

Figure: https://www.rambus.com/wp-content/uploads/2015/08/TimingAttacks.pdf

Kocher's Timing Attack (1996)

Modular exponentiation $c^d \bmod n$ by square-and-multiply does one squaring per bit of d, and an extra multiplication only where the bit is 1.

The running time is a sum of per-bit contributions. Guess the bits from the top: for a candidate value of bit i, predict the time over many ciphertexts, and keep the guess whose prediction correlates with the measurements.

This is the template: **a secret-dependent branch, repeated measurements, and a statistic that separates the two hypotheses.**

Remote Timing Attacks are Practical

David Brumley
Stanford University
dbrumley@cs.stanford.edu

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Figure: https://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf

Timing Attacks Are Practical (2003)

It was long assumed that network jitter buries timing differences, and that RSA is safe anyway. Brumley and Boneh broke both assumptions against OpenSSL.

Recovering a 1024-bit factor took 1,433,600 queries, about *two hours*, from a machine half a mile away across three routers.

It also works between two processes on one machine, and between two virtual machines on one host, which is the case that should worry a cloud tenant.

The fix is **blinding**: decrypt $r^e c$ instead of c for fresh random r, then divide the result by r. Cost 2–10%, enabled by default in OpenSSL from 0.9.7b.

Software vs Hardware Attacks

Some algorithms are computed in software and others in hardware, e.g., specialized circuits for computing AES or RSA.

This might impact memory allocation and SCA protection.

Passive vs Active Attacks

Some attacks are possible just by **listening** for information leakage, while other attacks require the adversary to take a more active role, e.g., by creating **(adaptive) queries** or by **injecting faults** into the computation.

Simple Power Analysis

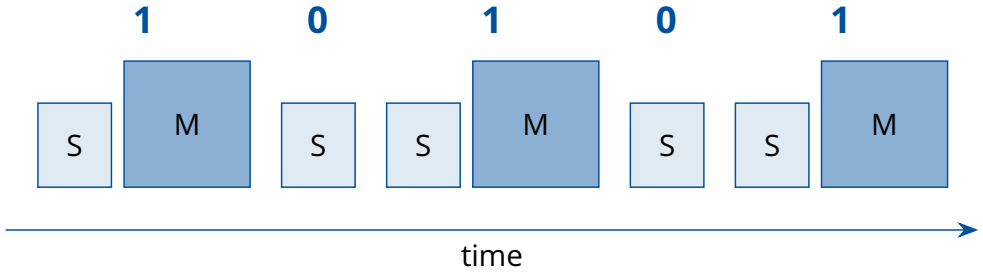
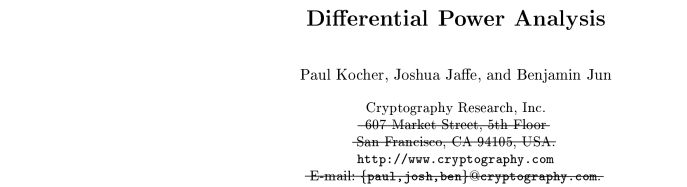


Figure: Square-and-multiply leaks the exponent: a squaring is followed by a multiplication exactly where the bit is 1. One trace can be enough.



Differential Power Analysis

Paul Kocher, Joshua Jaffe, and Benjamin Jun

Cryptography Research, Inc.
607 Market Street, 5th Floor
San Francisco, CA 94105, USA.
http://www.cryptography.com
E-mail: {paul,josh,ben}@cryptography.com

Abstract. Cryptosystem designers frequently assume that secrets will be manipulated in closed, reliable computing environments. Unfortunately, actual computers and microchips leak information about the operations they process. This paper examines specific methods for analyzing power consumption measurements to find secret keys from tamper resistant devices. We also discuss approaches for building cryptosystems that can operate securely in existing hardware that leaks information.

Keywords: differential power analysis, DPA, SPA, cryptanalysis, DES

Figure: https://paulkocher.com/doc/DifferentialPowerAnalysis.pdf

From Simple to Differential

Simple power analysis reads the secret off the *shape* of one trace. It works when the operation itself depends on the key, as in square-and-multiply, and it dies as soon as the implementation is regular.

Differential power analysis, Kocher, Jaffe, and Jun (1999), needs no visible structure. Collect many traces with known inputs, guess a small piece of the key, predict an intermediate value, and let the statistics decide.

The crucial point is that the guess is **local**: one byte of an AES round key at a time, so 16×256 hypotheses rather than 2^{128}.

Invasive vs Non-Invasive Attacks

An adversary that only measures time, power consumption, or electromagnetic radiation, leaving the device untouched, is **non-invasive**. Nothing shows afterwards that an attack took place.

A **semi-invasive** adversary opens the package to reach the surface of the chip, but makes no electrical contact with it: optical fault injection with a laser, localized EM probing, or heating.

An **invasive** adversary goes further and works on the silicon itself, using microprobes or a focused ion beam to read and modify internal signals. This is destructive and expensive, but nothing in the design is out of reach.

Vocabulary for the Next Lectures

- **Trace:** one recording of the side channel over one execution
- **Leakage model:** what we assume the trace measures, typically the Hamming weight of a value or the Hamming distance between two
- **Point of interest:** the sample in the trace where the target value is handled
- **Non-profiled:** attack the device directly, as in Differential Power Analysis
- **Profiled:** first characterize an identical device you control, then attack with far fewer traces (template attacks)

Traces are cheap and noisy, so almost every attack trades *more traces* against *less signal per trace*.

Contents

Checking Passwords

```
1  def isCorrectPassword(pw, user_input):
2      if len(pw) != len(user_input):
3          return False
4
5      for i in range(len(pw)):
6          if pw[i] != user_input[i]:
7              return False
8
9      return True
```

Q: What kind of information can you extract here?

Cracking Passwords

Possible vulnerabilities:

- The time depends on password length
- The time depends on correct guesses
- The attacker has unlimited trials

Protection: The time it takes to check must be independent of secrets, and we must rate-limit the number of trials.

Why the Timing Is So Damaging

The loop returns at the first wrong byte, so the running time counts how many leading bytes are correct. That turns the search from *guess the whole secret* into *guess one byte at a time*.

For a password of n bytes over an alphabet of size 256: blind search costs 256^n guesses, the timing attack costs $256 \cdot n$.

For $n = 16$ that is 2^{128} against 4096. The secret did not get smaller; we simply stopped being told about it all at once.

In reality one measurement is noisy, so each byte is tried a few hundred times and the median is used. Still nothing next to 2^{128}.

Doing It Right

Always touch every byte, and accumulate instead of branching:

```
def isCorrectPassword(pw, user_input): # pw is hashed
    user_input = hash(user_input)

    diff = 0
    for a, b in zip(pw, user_input): # as byte arrays
        diff |= a ^ b # no branch, no early exit

    return diff == 0
```

In practice, call `hmac.compare_digest` and compare *hashes* of fixed length, so the length itself stops leaking.

Contents

SCA Protection

Q: What are possible mitigation strategies to prevent side-channel leakage?

SCA Protection

- Constant time (secret independent) code
- Randomization of (secret) computation
- Splitting the secret into several parts
- Checking that outputs are correct/valid
- Hardware implementations of popular schemes
- Add noise to the computation (random delays)

What “Constant Time” Actually Means

Not “always takes the same number of seconds”. It means the *instruction and memory trace* must not depend on secrets:

- No branch whose condition depends on a secret
- No memory address that depends on a secret, including table indices
- No variable-latency instruction on secret data, such as modular inversion

The compiler is not on your side: it is free to turn your careful branchless code back into a branch. This is why the property must be *tested on the binary*, not argued from the source.

Contents

What to Remember

- A proof covers the algorithm; the attack targets the *implementation*
- Every attack needs a stated adversary: access, measurement, control, budget
- Secret-dependent branches, memory addresses, and instruction latencies are the leak
- Divide and conquer is what makes it cheap: guess one byte, not one key
- Countermeasures are constant time, blinding, masking, and output checks

Next: Into the Lab

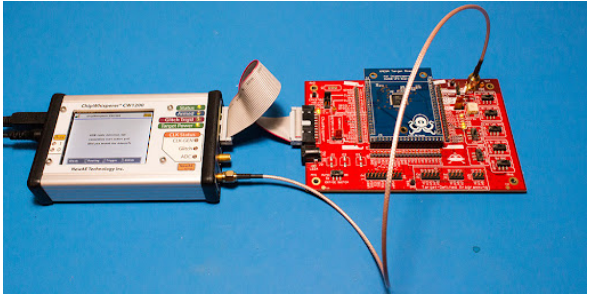


Figure: Next lecture we set up the ChipWhisperer, and then measure traces ourselves.

Questions?