Security Engineering

Elisabeth Oswald

Part 1 — What is Security/Crypto Engineering?

Outline

About this Unit

What is Crypto Engineering? And How does Crypto Theory differ from Crypto Practice?

Over to you

UNIT-HOWTO,1

Course Director and Lecturer: Elisabeth Oswald

- This is a 4ECTS = 2 UE course
- 2 hours of contact time: Tuesday 12-14, we either meet online (check the link in Moodle) in class (check Moodle for room). How we meet depends on regulations/convenience/preferences: I will always be available.
- All contact ‘slots’ are ‘student led’: i.e. I am there to answer questions and help, depending on your views, there will be joint slots with the "Advanced Topics in Cyber 2" people this year.
- All ‘taught material’ will be available online (videos,slides,handouts); You choose the pace at which you want to progress
- There is no written exam but only a project with a viva (or in other words, we’ll chat about your project).

UNIT-HOWTO, WHO AM I?

- I'm a researcher who looks into the challenges when deploying cryptography, mainly in the context of small/embedded devices
- My research is often "blue skies", and the results attract the interest of industry (aka people who need to design secure systems) and certifiers (aka people who need to decide if a produce satisfies certain security properties)
- My own background is a mix between maths and CS, and after starting out as an applied number theorist, I got dragged into applied statistics and statistical learning
- Besides my academic research, I do consulting for industry, and this course is inspired by what I have seen in practice.

I go on a first name basis with nearly everyone: please call me Elisabeth and use "du".

UNIT-HOWTO,2

This unit does not have any special prerequisites, but I do assume that you all have basic skills in mathematics (arithmetic) and computer science (ability to read pseudo code).

If you struggle with either of these then please be prepared to ask for help! I am happy to help out, but you must make yourselves known, best by approaching me via email or during class.

Ideally you will have done some crypto or security course prior to taking this one. If not, you may still be able to succeed and enjoy the experience, but you may need to ask more questions (see point I made before).

UNIT-HOWTO,3

How should you use your time?

This unit is worth 4 ECTS credits (aka approx. 100 hrs of work for an ‘average student’ to achieve an ‘average mark’).

- Approx. 15×2 hrs = 30 hrs you will spend each week watching lectures and reading notes/slides
- Each week there is also the opportunity to ask questions (w.r.t. the lectures or the coursework) and/or work on your coursework/project with us at hand; this gives another 30hrs
- The coursework will require a brief write-up and some preparation to present your findings to me. There are 40hrs reserved for this activity.

Thus if you engage during the term on a regular basis, you should be able to not just do the coursework but enjoy it, and you can in fact present your results whenever you are ready (thus you can complete this course within the allocated term).

THE CONTENT OF THIS UNIT

This course is about engineering cryptography: we will initially explore what this actually means, and how, in fact, ideas about engineering cryptography have changed over time.

There are many aspects to engineering cryptography, and to keep the course manageable whilst enabling you to develop some new skills, the coursework will be fairly focused on engineering a system that requires a secure symmetric encryption scheme at its' core.

We will approach things 'top down': based on engaging with ideas of what cryptographic engineering might be about, we will make a journey into security evaluation and testing, consider very concretely some ideas of how to create randomness and make things fast, and then we will take a longer look at dealing with physical side channels and fault attacks.

The coursework is tied to the 'lectures' in the sense that you will need the latter to make a security assessment of the former and suggest improvements.

Coursework

The coursework is the main vehicle for learning: it should drive your research and should get you thinking about the 'taught content'.

The coursework will consist of a 'specification' and a 'reference implementation': the specification will describe a core cryptographic component of a 'larger system'; the reference implementation will be the 'first attempt' of realising the specification. We will also provide a simulator that enables you to test for physical leaks and try out fault attacks.

Everybody will get their 'personalised' specification and corresponding reference implementation. Your task is then simple: find, describe, demonstrate, and fix as many problems that you can find in either the specification or the reference implementation.

The marking scheme will be roughly as follows: finding (1 mark), describing (up to 5 marks), demonstrating (up to 10 marks), and fixing (up to 10 marks). To achieve a pass mark you need to get 50 marks (out of 100), how the marks scale up probably needs adjusting based on what you find (I have never done this particular coursework before).

Introduction

The course is in several parts and we start off with the basic question of:

What is Security/Crypto Engineering?

- Theoretical cryptography vs. practical cryptography
- Changing challenges due to changing real life constraints and attacks
- Attacks only ever get better
- The need for guarantees in practice: evaluation and certification

Let's do a little warm up

Please take a moment (perhaps pause the video) and just think about what you believe that cryptography can do for you in practice. What does cryptography promise, and under which assumptions? If you are not sure where to start then take a simple example, like encrypting a file or signing an email (what crypto primitives do you recall are available for such tasks and what sort of security guarantees do they give, under what assumptions?).

Don't use the Internet for assistance. This exercise it's not about getting it right. It is about getting you from being passive to actively engaging with your learning.

So what about encrypting a file? If you attended the Cryptography course, then I'd hope that your answer would go beyond the "well I'd hope that I'm the only one being able to decrypt the file" ...

What assumptions did you make? E.g. about where the key lives? What information does an adversary get: Can they watch you encrypt the file, do they get access only afterwards, and how do these assumptions relate to notions like IND-CPA or IND-CCA?

LET'S ENCRYPT . . .

Clearly assumptions depend on the context:

encrypting/decrypting files on a USB stick: a USB stick can easily get stolen or lost, and it would be easy for an adversary to get physical access and record side channels)

encrypting/decrypting files on a server: unlikely that the server gets lost, or that physical access is possible, but possible that adversaries could get their code co-hosted

Irrespective of the context it is nearly always true that the assumptions made by classical theoretical cryptography are **not** met in practice: adversaries do get more information than what is provided in typical security games.

SOME NOTATION ...

Any message needs to be represented as a ‘number‘ via some suitable encoding. Thus any message can be thought of as an element from some space.

- m is the plaintext which is from the plaintext space: $m \leftarrow \mathcal{M}$
- c is the ciphertext which is from the ciphertext space: $c \leftarrow \mathcal{C}$
- sk is the secret key which is drawn at random from the key space: $sk \leftarrow \mathcal{K}$

Thus an encryption scheme consists of three functions:

- $Gen : \mathcal{K} \xrightarrow{\$} sk$ generates a secret key via randomly sampling an element from the keyspace
- $Enc : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$ maps an element from the plaintext space to the ciphertext space (using an element from the key space)
- $Dec : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$ recovers the plaintext from the ciphertext (using the secret key)

Correctness Property of Symmetric Encryption

Symmetric encryption

An encryption scheme in which the same key sk is used for both encryption and decryption is called **symmetric** encryption scheme.

Correctness

A symmetric encryption scheme is said to be correct if
$\forall m \in \mathcal{M}$ and $\forall sk \in \mathcal{K} : Dec_{sk}(Enc_{sk}(m)) = m$.

Clearly we will require (and need to check) that any concrete instantiation of an encryption scheme is at least correct.

LET'S REVISIT A GAME FOR SECURE ENCRYPTION

Let's consider the standard indistinguishability game for symmetric encryption. The adversary's goal is to distinguish between the encryption of m_0 and m_1.

Where are the assumptions about key storage?

Where are the assumptions about potential for extra information?

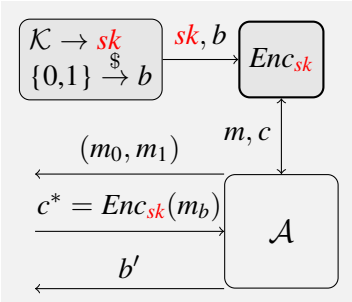

Figure: Exp ind-pas

LET'S REVISIT A GAME FOR SECURE ENCRYPTION — WITH LEAKAGE

Let's consider the standard indistinguishability game for symmetric encryption. The adversary's goal is to distinguish between the encryption of m_0 and m_1.

In practice, an adversary may be able to get extra information $\mathcal{L} = f(m, c, sk)$ every time a device accesses the secret key.

This is both information about message and/or the key.

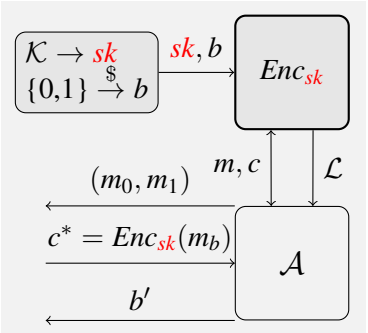

Figure: Exp ind-pas

Kerckhoffs' Principle

Let's also briefly revisit this famous principle. We accept it, and defend it, quite naturally in the context of theoretical crypto.

Kerckhoffs' Principle

"The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

But what about practical implementations? Should we extend Kerckhoffs' Principle to include implementation details?

Safeguarding against Security by Obscurity

Kerckhoffs' Principle

"The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

There is a lot of wisdom in the assumption that any secret cryptographic scheme will eventually be reverse engineered: because every system that has spread widely and wasn't fully open did eventually get reverse engineered.

Making a system open enables many experts to examine it. Thereby the chances increase to find potential problems and fix them before it gets deeply integrated in systems.

This is equally true for implementation details: but often companies see these implementations as valuable IP and therefore this principle is not adhered to.

Principle of sufficient key length?

Suppose that an adversary gets a large amount of ciphertexts. Perhaps they also get the corresponding plaintexts, but even if they don't, they can **guess and test** keys. Their goal is to determine the key.

Using **brute force key search** it may be possible to find the secret key via trying lots of key values. The only way to defend against it is by choosing a sufficiently large key space. Thus modern symmetric encryption schemes require keys to be drawn from space of at least 2^{128} elements (aka the key size of a modern symmetric encryption scheme is at least 128 bits).

Unfortunately, longer keys do not necessarily protect against attacks using extra information. Therefore we cannot simply safe practical security by increasing the key size.

So how does Crypto Theory differ from Crypto Engineering Practice?

Conventional security notions do not take into account the sometimes vast amount of extra information that adversaries have access to in real life.

Leakage resilient cryptography is still a fledgling area: leakage models are either not strong enough or too strong; i.e. resulting schemes either do not lead to secure implementations or they are very unwieldy.

There is even less theory available that can guide developers w.r.t efficient fault resilient implementations.

Therefore crypto engineers need to be able to correctly understand theoretical assumptions in order to be able to judge when the **do not** hold and engineer crypto with this in mind.

A "SIMPLE" EXAMPLE

Consider encrypting a website.

For IND-CPA security we assume that message m_i are drawn from the same input space and that they are of equal lenght, otherwise breaking IND-CPA is trivial.

Thus strictly speaking, to guarantee IND-CPA for encrypted websites, as engineer you need to ensure that all website (i.e. including the dynamic content that they might load such as ads) are of equal size.

Good luck ...

What does a Crypto Engineer need to be aware of?

In a nutshell: **any data dependent behaviour**.

E.g. data dependent behaviour might result in different running times, power consumption, EM emanation, sound, blinking lights (or other optical emanations), fan speeds, cache behaviour (visible as timing side channel),

Sometimes adversaries can “force” data dependent behaviour by injecting faults: either via special equipment, or via existing software interfaces, or by rapidly accessing memory.

Also programming mistakes can lead to issues: e.g. memory violations (e.g. C programming language is not memory safe)

Finally the improper use of randomness can have devastating effects on what crypto can deliver in practice.

This Course

This course is to explain how some of the most devastating attacks that are know today work, and to help you spot problems in software that enable such attacks.

Crypto engineering really is about developing skills:

- understanding the impact of optimisations on security
- relating crypto assumptions to reality
- paying attention to seemingly tiny details
- ability to utilise some basic testing regimes

The latter point indicates that structured testing and validation has a significant role to play here, but again, there is very little theory and tools to help.

Complementary Reading and Watching

This course consists not just of material that I produced, but I also want you to benefit from a range of existing other materials from some fantastic researchers out there.

- Crypto Engineering is part of a larger discipline called Security Engineering. Ross Anderson from Cambridge University has authored a comprehensive, and very readable book on this larger topic. Read chapter 1 of this book, available at `https://www.cl.cam.ac.uk/~rja14/Papers/SEv3-ch1-7sep.pdf`
- There is a corresponding Video available via YouTube from Ross, and I would encourage you to watch this segment here: `https://www.youtube.com/watch?v=gQbQPuwntqU`

The intended learning goals are that you better understand the significance of Kerckhoffs Principle and the size of the key space for the practical security of encryption schemes.