# Crypto Engineering

Elisabeth Oswald

Part 1 — What is Crypto Engineering?

# Outline

# Overview

# Crypto vs. Security Engineering

Crypto Engineering is a sub-area of Security Engineering.

The main difference is in the emphasis on systems thinking: it is stronger in Security Engineering than Crypto Engineering.

When engineers consider systems, they tend to focus on the technical components; but obviously systems also consist of humans; there are rules and regulations, incentives, etc.

Today we will make a detour into the less technical side of things, and consider the "human factor".

# USABILITY AS KEY REQUIREMENT

Many crypto courses start by very early on reviewing some principles put forward by Auguste Kerckhoffs.

Typically, and my courses are no exception, one of his key principles is **not** listed:

“Lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.” (taken verbatim from Wikipedia)

It should be obvious that systems that are complicated, annoying or inconvenient will not be adopted in practice: security features that aren’t used don’t really contribute to security though.

# Usability as Key Requirement

That systems must be useable is mostly obvious, but should the same principle also apply to crypto components?

I would argue that it does because any larger system is only as secure as it's weakest link: if the usability of a crypto component (be it a specification, piece of software, or bit of hardware) is poor, then it will compromise the security of the entire system eventually.

**Pause and reflect:** what crypto or security systems have you encountered in your life and how would you rate their usability?

# Usability as a Field of Research

Psychologists are the key experts to consult when it comes to usability; most research seems to come from their corner.

Within computer science, the field of “human computer interaction” (HCI) has championed the idea of systematically studying how humans and systems interact, and how to create useable systems.

Within the wider crypto and security community there has been only a very late acknowledgement of the the role that usability has to play in the adoption of crypto/security software/systems.

# WHY JOHNNY CAN'T ENCRYPT

This is the paper title of the now legendary paper published by Whitten and Tygar at Usenix 1999.

It reports on a usability experiment that they carried out: college students were asked to encrypt an email using PGP. They had 90 mins to complete the task.

They concluded that "Based on the results of our evaluation, we conclude that PGP 5.0's user interface does not come even reasonably close to achieving our usability standard – it does not make public key encryption of electronic mail manageable for average computer users." (quotation from that paper).

# What is Usability (according to Whitten and Tygar)

"Security software is usable if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don't make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it."

This sounds pretty reasonable, and is largely applicable to other situations too because by and large there will be some user who we'd like to

1. know what task they should be doing
2. know how achieve their task
3. don't make (dangerous) errors
4. will do the task again if needed

# How, what and why

Underpinning these usability conditions are three questions that any specification/description (for an interface, or software or hardware) must answer (for the user):

- What: am I meant to be doing with this.
- How: should I be using this.
- Why: does it work and why does it not work.

Addressing the “How” and “Why” in a suitable manner can help minimise errors based on misunderstanding or a lack of awareness of the risk that there is if things go wrong.

# Overview

# (Limitations to) Capability

We are most likely to make mistakes when given tasks that exceed our capabilities.

Humans can only focus their attention primarily on one task at any one time.

Alarm fatigue: Once alarms have been classified as unreliable, people stop paying attention to them.

Memory: A distinction is between Short Term Memory (STM) and Long Term Memory (LTM). When one tries to memorise an item, it needs to go round the STM loop a few times before it is transferred into the LTM.

Human behaviour is essentially goal-driven.

Actions that have to be carried out become “automated”.

# Human Error

James Reason showed that virtually all mistakes people make are predictable. They occur as a result of **latent failures** (organisation and local workplace conditions) and **active failures** (errors and violations by humans) in combination to allow the accident to occur.

Reason identifies four types of latent failures that are more likely to cause people to make errors.

- Individual factors include fatigue but also inexperience and a risk-taking attitude.
- Human Factors include the limitations of memory but also common habits and widely shared assumptions.
- Task factors include time pressure, high workload and multiple tasks, but monotony and boredom are equally error-inducing because people shift their attention to diversions. Uncertainty about roles, responsibilities and rules also lead to incorrect choices.
- Work environment factors include interruptions to tasks and poor equipment and information. People are also particularly prone to error when rules and procedures change.

# EXAMPLE

Consider the situation where cashiers have to type in their ID and PIN/password at the till before they operate it. If they are not just working at the till, but also helping customers on the shop floor, the additional login/logout process might quickly become a hassle.

It is easy to imagine the rule that you must login when working at the till degenerating into the rule that 'somebody' has to be logged in. Hence one person logs in and never logs out, thus allowing everybody (including possible adversaries) to operate the till as and when needed.

In this system the problem is one of incentives. If, for each worker, total sales attributed to their ID counted towards (e.g.) a bonus at the end of a working day, they'd be more willing to bear the hassle of logging in and out.

# Overview

# CLEAR AND CONCISE DESCRIPTIONS ARE DIFFICULT

# Clear and concise descriptions are difficult

# CLEAR AND CONCISE DESCRIPTIONS ARE DIFFICULT

# DES AND THE BYTE ORDER CONVENTION AS AN EXAMPLE

The DES specification explains that any plaintext block needs is understood as two halves $(L, R)$, and the byte ordering within each half is from little byte upwards, i.e. $L = L_1 L_2 ....$

How bytes are ordered (either in software, or in hardware, or in a protocol) is called "endian ness".

Big Endian: highest byte comes first (i.e. like you write things in a positional number system)

Little Endian: lowest bytes comes first (the other way around). Many systems use the "natural" big endian notation, but DES does not.

Endianness itself can be confusing, and there are entire threads about what DES does and how to survive when working on a big endian system ... e.g. `https://groups.google.com/forum/#!topic/sci.crypt/Zv-G9XuTu78`.

# OpenSSL API

Suppose you want to encrypt something . . . some people resort to using the OpenSSL library.

But what function to use? A “simple” encryption function that is provided in the current version still requires a host of input parameters: `https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption`.

This documentation is not easy to read and hence there are a number of tutorials available such as this: `https://tombuntu.com/index.php/2007/12/12/simple-file-encryption-with-openssl/`.

Can you spot the problem in the tutorial?

# OpenSSL API

But things were worse in the past: previously the encryption for CBC mode was the function call `alg_cbc_encrypt()`.

The problem was that this function used whatever was in another variable that held the IV and never updated the IV value.

As you know from Cryptography (and if not you should read up on CBC mode), CBC mode is very brittle and the IV really must be chosen from random, or the mode can be easily broken.

To use CBC mode with a fresh IV you had to use `alg_ncbc_encrypt()` instead.

# Overview

# WHY DO WE TAKE RISKS?

Risky behaviour could be seen as breaking some rules. Reber defined it as: “an action that jeopardises something of value”. Risky behaviour is something that is difficult to fully explain.

One potential contributing factor is that risk is an action which in society typically leads to (strong) negative feedback. But some people like this aversive feedback (i.e. the fright that comes with it provides a form of excitement).

The body releases, as a physiological response, a hormone called adrenalin. Adrenalin is also associated with euphoria, and this clearly feels good; the cycle of excitement (prior), thrill (during) and euphoria (after) associated with risk-taking can be sufficiently gratifying as to become potentially highly habit-forming.

# Risk as a personality trait vs Risk as a habit

Another way of understanding risk is as an inherent feature of someone's personality – that is, a part of one's psychological make-up, as opposed to a behaviour (that was learned and can be unlearned).

There is also the concept of 'risk homeostasis'. This relates to the observation that when we make a situation safer, then people tend to take higher risks, but when a situation is perceived as dangerous, then people tend to be more careful.

# Overview

# CRYPTOLOGGER EXPERIMENT

This year some researchers developed a new tool to scan crypto implementations for common vulnerabilities.

They found larger numbers of apps featuring these problems.

They also found that most app developers did not care.

`https://www.computer.org/csdl/proceedings-article/sp/2021/893400a160/1mbmHwIxTb2`

| ID | Rule Description | Ref. |
|---|---|---|
| *R-01* | Don't use broken hash functions (`SHA1`, `MD2`, `MD5`, ..) | [8] |
| *R-02* | Don't use broken encryption alg. (`RC2`, `DES`, `IDEA` ..) | [8] |
| *R-03* | Don't use the operation mode `ECB` with $>1$ data block | [5] |
| *R-04* † | Don't use the operation mode `CBC` (client/server scenarios) | [12] |
| *R-05* | Don't use a static (= constant) key for encryption | [5] |
| *R-06* † | Don't use a "badly-derived" key for encryption | [5] |
| *R-07* | Don't use a static (= constant) initialization vector (IV) | [5] |
| *R-08* † | Don't use a "badly-derived" initialization vector (IV) | [5] |
| *R-09* † | Don't reuse the initialization vector (IV) and key pairs | [46] |
| *R-10* | Don't use a static (= constant) salt for key derivation | [5] |
| *R-11* † | Don't use a short salt ($<64$ bits) for key derivation | [14] |
| *R-12* † | Don't use the same salt for different purposes | [46] |
| *R-13* | Don't use $<1000$ iterations for key derivation | [14] |

| ID | Rule Description | Ref. |
|---|---|---|
| *R-14* † | Don't use a weak password (score $< 3$) | [47] |
| *R-15* † | Don't use a NIST-black-listed password | [48] |
| *R-16* | Don't reuse a password multiple times | [48] |
| *R-17* | Don't use a static (= constant) seed for PRNG | [49] |
| *R-18* | Don't use an unsafe PRNG (java.util.Random) | [49] |
| *R-19* | Don't use a short key ($< 2048$ bits) for RSA | [13] |
| *R-20* † | Don't use the textbook (raw) algorithm for RSA | [50] |
| *R-21* † | Don't use the padding `PKCS1-v1.5` for RSA | [51] |
| *R-22* | Don't use HTTP URL connections (use HTTPS) | [16] |
| *R-23* | Don't use a static (= constant) password for store | [48] |
| *R-24* | Don't verify host names in SSL in trivial ways | [16] |
| *R-25* | Don't verify certificates in SSL in trivial ways | [16] |
| *R-26* | Don't manually change the hostname verifier | [16] |

# Overview

Crypto Engineering vs. Security Engineering

Types of errors

Errors based on misunderstandings

Errors based on Risk taking

Typical Mistakes Made when Using Crypto

Over to you

# 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.

- Psychology is an integral part of usable security, I suggest you read `https://www.cybok.org/media/downloads/Human_Factors_issue_1.0.pdf`
- There are also some videos from Ross Anderson, e.g. `https://www.youtube.com/watch?v=0-004dunKDU`

The intended learning goals are that you better understand the significance of the human factor when designing and implementing crypto.