

Applied Cryptology

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
(csdsp@bristol.ac.uk)

September 5, 2025

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
 - ▶ the associated notes page may be pre-populated with extra, written explanation of material covered in lecture(s), plus
 - ▶ anything with a “grey’ed out” header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- ▶ **Agenda:** a non-technical introduction to
 1. unit objectives,
 2. unit organisation, and
 3. some motivation (i.e., *why* the unit exists).

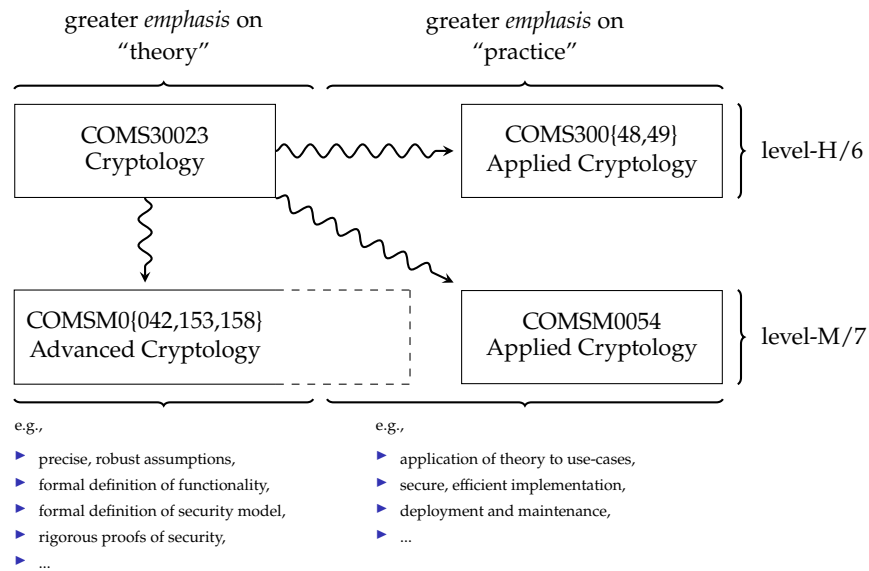
Notes:

Part 1: unit objectives, i.e., the “what” (1)

- ▶ **Question:** what *is* cryptography?
- ▶ (An) **answer:**
 - ▶ the field can be described as the sum of more specific sub-fields, namely
 - underlying Mathematics \simeq number theory, group theory, ...
 - cryptography \simeq design and analysis of (general) primitives and protocols
 - applied cryptography \simeq development of (specific) cryptographic solutions
 - cryptographic engineering \simeq implementing, deploying, and maintaining said solutions
 - ▶ keep in mind that
 1. cryptology \simeq cryptography + cryptanalysis
 2. cryptology \subset cybersecurity
 3. cryptology \supset Mathematics
 4. cryptology \supset encryption
 5. “crypto” = cryptography
 \neq block chain

Notes:

Part 1: unit objectives, i.e., the “what” (2)



Notes:

Part 1: unit objectives, i.e., the “what” (3)

Objectives

Put simply, after completing this unit you *should* be able to understand *and* apply concepts relating to

1. implementation techniques, e.g., multi-precision arithmetic
2. implementation attack and countermeasure techniques, e.g., timing attacks, constant-time implementation
3. cryptographic protocols and systems, e.g., TLS

set within the more general context of cryptology.

Notes:

Part 2: unit organisation, i.e., the “how” (1)

► Important:

1. The unit is delivered by the following members of (academic) staff

Daniel Page ⇒ Lecturer and Unit Director
François Dupressoir ⇒ Lecturer

supplemented by, e.g., a wider team who act in/as Teaching Support Roles (TSRs).

2. Even if/where we refer to a single unit, there are *really* multiple units:

COMS30048 ↦ *teaching* unit
COMS30049 ↦ *assessment* unit : level-H/6
COMSM0054 ↦ *assessment* unit : level-M/7

Notes:

Part 2: unit organisation, i.e., the “how” (1)

► Important:

3. At a high(er) level, the unit is delivered as a set of themes:

Theme #1 ⇒ “implementation challenges”
Theme #2 ⇒ “security challenges (i.e., attacks and countermeasures)”
Theme #3 ⇒ “use-cases, examples, and case-studies”

4. At a low(er) level, the unit involves the following activities:

lecture slot ⇒ synchronous, i.e., timetabled
 ⇒ in-person

lab. slot ⇒ synchronous, i.e., timetabled
 ⇒ in-person

Notes:

Part 2: unit organisation, i.e., the “how” (1)

► Important:

5. The *summative* assessment for this unit includes

summative coursework assignment \rightsquigarrow TB2, week 24
 \mapsto 100% weight = 20CP

Notes:

Part 2: unit organisation, i.e., the “how” (1)

► Important:

6. *Everything* related to the unit is accessible via *either*

- the *internal*-facing Blackboard-based unit web-site

<https://www.ole.bris.ac.uk>

or


- the *external*-facing GitHub-based unit web-site

<https://cs-uob.github.io/COMS30048>


or, more specifically,

| | | | |
|--|---------------|------------|---------------------------|
| unit-wide communication, e.g., announcements | \Rightarrow | Blackboard | } <i>internal</i> -facing |
| assessment submission, marks, and feedback | \Rightarrow | Blackboard | |
| discussion forum | \Rightarrow | Teams | |
| teaching material | \Rightarrow | GitHub | } <i>external</i> -facing |

Notes:

$$r \leftarrow g^x \implies$$


Notes:

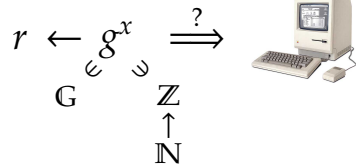
$$r \leftarrow_{G \in \mathcal{G}} g^x \implies$$


$\mathbb{Z} \uparrow \mathbb{N}$

Notes:

Part 3: unit motivation, i.e., the “why” (1)

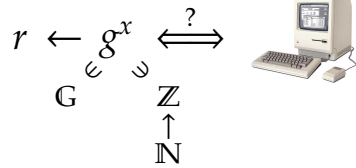
Implementation challenges



Notes:

Part 3: unit motivation, i.e., the “why” (1)

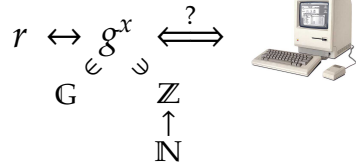
Implementation challenges



Notes:

Part 3: unit motivation, i.e., the “why” (1)

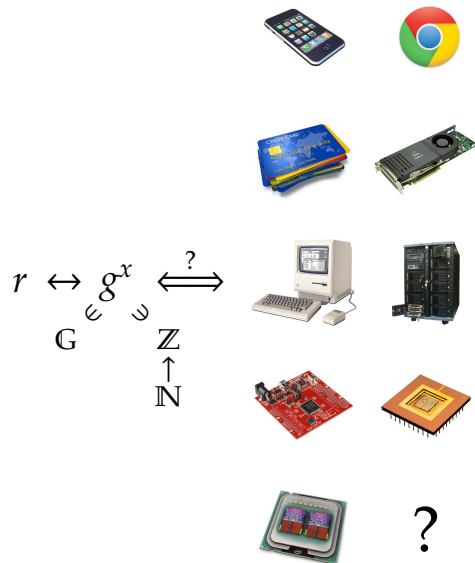
Implementation challenges



Notes:

Part 3: unit motivation, i.e., the “why” (1)

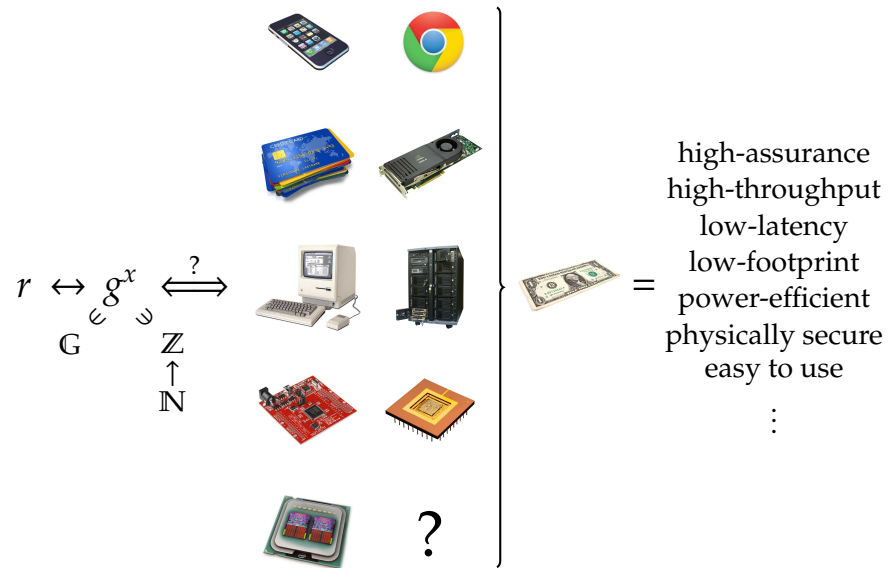
Implementation challenges



Notes:

Part 3: unit motivation, i.e., the “why” (1)

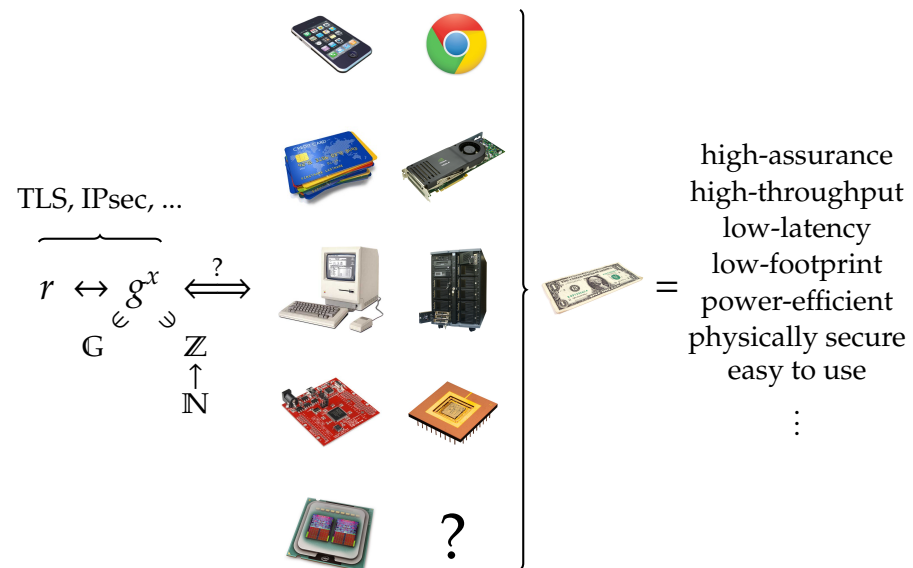
Implementation challenges



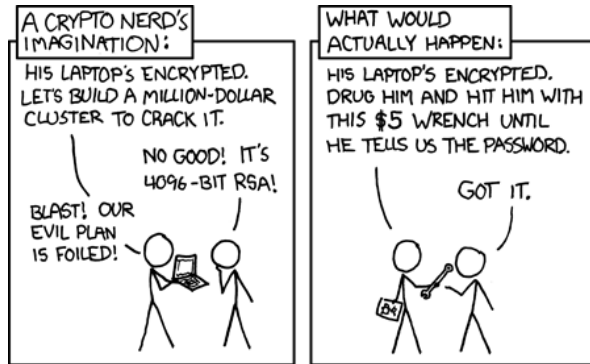
Notes:

Part 3: unit motivation, i.e., the “why” (1)

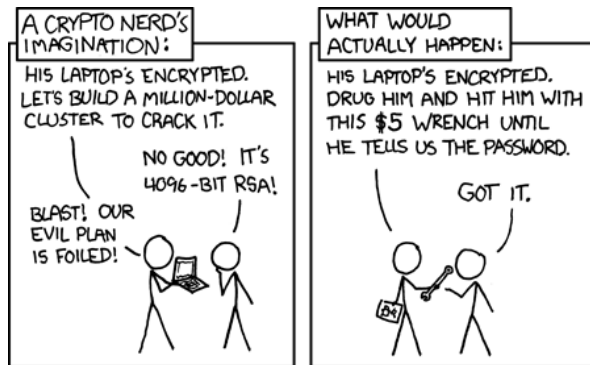
Implementation challenges



Notes:



<https://xkcd.com/538>



i.e.,

1. “black box” security model \leadsto cryptanalytic attack \approx focused on the *design*
 \approx attackers do what they *should*
2. “grey box” security model \leadsto implementation attack \approx focused on the *implementation*
 \approx attackers do what they *can*

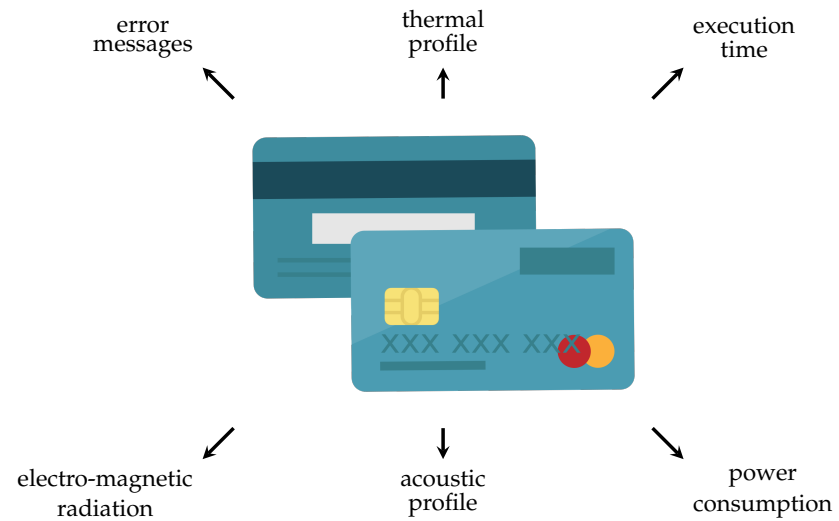
<https://xkcd.com/538>

Notes:

Notes:

Part 3: unit motivation, i.e., the “why” (3)

Security challenges



https://commons.wikimedia.org/wiki/File:Credit_or_Debit_Card_Flat_Icon_Vector.svg

© Daniel Page (CC BY-SA 4.0)
Applied Cryptology

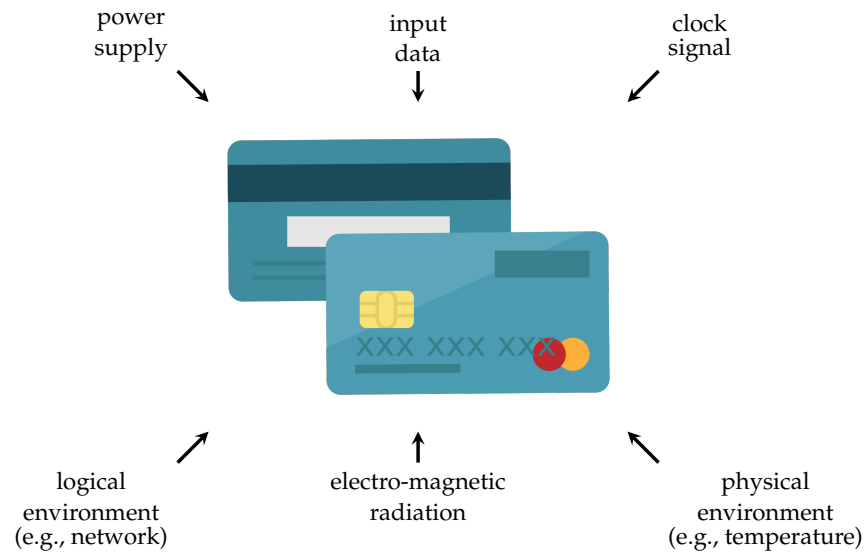
University of
BRISTOL

git # b282dbb9 @ 2025-09-03

Notes:

Part 3: unit motivation, i.e., the “why” (3)

Security challenges



https://commons.wikimedia.org/wiki/File:Credit_or_Debit_Card_Flat_Icon_Vector.svg

© Daniel Page (CC BY-SA 4.0)
Applied Cryptology

University of
BRISTOL

git # b282dbb9 @ 2025-09-03

Notes:

Conclusions (1)

RSA: Rivest, Shamir, and Adleman [3]

Each user must (privately) choose two large random numbers p and q to create his own encryption and decryption keys. These numbers must be large so that it is not computationally feasible for anyone to factor $n = p \cdot q$. (Remember that n , but not p or q , will be in the public file.) We recommend using 100-digit (decimal) prime numbers p and q , so that n has 200 digits.

To find a 100-digit “random” prime number, generate (odd) 100-digit random numbers until a prime number is found. By the prime number theorem [7], about $(\ln 10^{100})/2 = 115$ numbers will be tested before a prime is found.

► Challenges:

1. how can we generate random (enough) numbers,
2. how and where should we store key material once it's generated, and
3. is a 200-digit (or n -digit) key enough to prevent real attacks (even in m years time),
4. ...

<https://people.csail.mit.edu/rivest/Rsapaper.pdf>

© Daniel Page ([d.page@bristol.ac.uk](#))
Applied Cryptology

 University of
BRISTOL

git # b282dbb9 @ 2025-09-03

Conclusions (1)

RSA: Rivest, Shamir, and Adleman [3]

In the following sections we consider ways a cryptanalyst might try to determine the secret decryption key from the publicly revealed encryption key. We do not consider ways of protecting the decryption key from theft; the usual physical security methods should suffice. (For example, the encryption device could be a separate device which could also be used to generate the encryption and decryption keys, such that the decryption key is never printed out (even for its owner) but only used to decrypt messages. The device could erase the decryption key if it was tampered with.)

► Challenges:

1. what attacks exist *beyond* those a cryptanalyst might employ,
2. how can we generate and/or agree secure session keys between parties, and
3. what determines secure versus insecure erasure of data,
4. ...

<https://people.csail.mit.edu/rivest/Rsapaper.pdf>

© Daniel Page ([d.page@bristol.ac.uk](#))
Applied Cryptology

 University of
BRISTOL

git # b282dbb9 @ 2025-09-03

Notes:

Notes:

Conclusions (1)

RSA: Rivest, Shamir, and Adleman [3]

Computing $M^e \pmod n$ requires at most $2 \cdot \log_2(e)$ multiplications and $2 \cdot \log_2(e)$ divisions using the following procedure (decryption can be performed similarly using d instead of e):

- Step 1. Let $e_k e_{k-1} \dots e_1 e_0$ be the binary representation of e .
- Step 2. Set the variable C to 1.
- Step 3. Repeat steps 3a and 3b for $i = k, k-1, \dots, 0$:
 - Step 3a. Set C to the remainder of C^2 when divided by n .
 - Step 3b. If $e_i = 1$, then set C to the remainder of $C \cdot M$ when divided by n .
- Step 4. Halt. Now C is the encrypted form of M .

► Challenges:

1. how efficient and suitable is an implementation of this approach (versus alternatives) on a given platform,
2. how can we be sure an implementation doesn't leak information and isn't vulnerable to tampering, and
3. how should we use the resulting public-key encryption primitive within some application,
4. ...

<https://people.csail.mit.edu/rivest/Rsapaper.pdf>

© Daniel Page (dp@cs.bristol.ac.uk)
Applied Cryptology



git # b282dbb9 @ 2025-09-03

Conclusions (2)

Quote

In theory, there is no difference between theory and practice. But, in practice, there is.

– van de Snepscheut (https://en.wikiquote.org/wiki/Jan_L._A._van_de_Snepscheut)

► Take away points:

1. Practical realisation of theoretical cryptography is *hard*, but someone has to do it: since *you* are potentially them, you'll ideally do a good job!
2. Development and deployment of wider *systems* that utilise cryptography requires a deep, inter-disciplinary understanding of *both* dimensions ...
3. ... even then, various domain-specific challenges *must* be met somehow to avoid (epic) failure:
 - in many cases, failure to meet similar challenges is obvious, e.g., something just doesn't work,
 - in cryptography, the worst-case is that don't even *know* you don't understand until it's too late.

Notes:

Notes:

- A summary of (some) points that lead to the cited difficulty might read as follows:
 1. One cannot simply “assume the existence of X ” in the sense that for *every* component X needed, we also need a way to realise it in practice.
 2. The quality metrics for such a realisation are diverse, and often mutually exclusive (rather than graceful trade-offs); among tough decisions, security should *still* be first-class.
 3. Many primitives will represent an inherently expensive workload comprised of computationally-bound, highly numeric kernels; they often also exist in high-volume or high-throughput applications.
 4. The range of platforms one might need to realise a solution on is vast, and diverse; understanding and utilising the platform characteristics is vital.
 5. A practical security model (or attack surface) can be *much* wider than a theoretical analogue, and the implication of any *insecurity* are more tangible.
 6. Someone actually has to use the realisation, so human-error needs to be minimised: anecdotal evidence shows it is common for users to disable security features in an application if it improves performance or responsiveness.

Additional Reading

- ▶ *Wikipedia: Cryptography*. URL: <https://en.wikipedia.org/wiki/Cryptography>.
- ▶ *Wikipedia: Cryptographic engineering*. URL: https://en.wikipedia.org/wiki/Cryptographic_engineering.

Notes:

References

- [1] *Wikipedia: Cryptographic engineering*. URL: https://en.wikipedia.org/wiki/Cryptographic_engineering (see p. 53).
- [2] *Wikipedia: Cryptography*. URL: <https://en.wikipedia.org/wiki/Cryptography> (see p. 53).
- [3] R.L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Communications of the ACM (CACM)* 21.2 (1978), pp. 120–126 (see pp. 45, 47, 49).

Notes: