Crypto Engineering ’23

✦

Definitions

Pierre Karpman
pierre.karpman@univ-grenoble-alpes.fr
https://membres-ljk.imag.fr/Pierre.Karpman/tea.html

2023–09–26

The “symmetric” part of this course (with me)

- 5 CMs; 3 TDs; $2*(2*2) = 8$ TPs
- (Mostly) about symmetric encryption, authentication, (password) hashing
- Goal 1: understanding the models $\rightsquigarrow$ What can we/do we try to achieve?
- Goal 2: looking a bit at some design(s): the why and hows
- Goal 3: getting a few ideas of what can go terribly wrong :(

The practical part of the “asymmetric” part of this course (with me)

- 1*(2*2) $=$ 4 TPs
- Kangaroos for memory-efficient discrete logarithms computation

Crypto : what and what and what?

Crypto : “securing communication (and more) in the presence of adversaries”

- What kind of adversaries (what *attack model*)?
- Securing what (what kind of functionality)?

$\rightsquigarrow$ formal security definitions
Examples:

- Passive (blackbox) adversaries + authenticating stuff $\rightsquigarrow$ EUF-CMA security
- Active (blackbox) adversaries + hiding stuff $\rightsquigarrow$ IND-CCA2 security

Why do we care so much about definitions?

Formal definitions allow to:

- State clearly (...) and unambiguously what we want to achieve
- Define attacks (at a fine granularity: incl. cost, success rate)
- Express relations between different objects $\rightsquigarrow$ modularity of designs, (reduction) proofs of security

Everything is about definitions

Figure: From Calvin & Hobbes, Watterson

Designing a definition

Two challenges (among others):

- What does it mean to (say) keep something secret? How do you formalise (the absence of) learning?
- How do you formalise hardness? A "security level"?

Typical approaches:

- Try something (taking inspiration from an ideal world?), hope for the best?
- Use probabilities + complexity-based models (for a start)

Lack of confidentiality: an XKCD illustration

Figure: XKCD #257

$\rightsquigarrow$ Security is never absolute

Security always depends on the context $\rightsquigarrow$ make *assumptions* on the adversary's limitations, e.g.:

- doesn't know a certain value
- only has access to one encrypted message
- only has access to 3 out of 7 communication channels
- is limited to a 1GW power source

usually, everything trivially collapses if some assumptions don't hold $\rightsquigarrow$ important to always clearly state your assumptions

Bad definitions are useless; so are badly-used good ones

Knowing what definition to use depends on:

- the objective (more or less obvious)
- assumptions on the adversary, cf. above; below (somewhat less obvious)

↑ Typical crypto engineering

Again:

- a completely sound design may be completely broken for any practical usage
- *provable security* is relative

Typical panorama

Typical adversaries' capabilities (or not):

- passive ("eavesdropper") or active ("man in the middle")
- blackbox or with some physical access ($\rightsquigarrow$ side-channel attacks; fault attacks...)
  - Protecting coms is also about restricting physical access
- with limited time/memory ("computational") or not ("information theoretic")
- with limited data

Typical objectives:

- keeping a message secret ("encryption")
- proving an identity
- certifying a document ("authentication" or "signing")
- computing in a malicious environment / over encrypted data ("MPC"; "Homomorphic encryption")

Security definitions: typical high-level structure

- Adversary: algorithm with access to some oracle(s), trying to win some probabilistic game
- oracle: captures interaction with a system of interest, with some capability (nature of the oracle; allowed queries...); usually randomised, and *dependent on the system*
- algorithm's running time, #queries: how efficient is it (for a given result)?
- algorithm's winning probability, or *advantage* (over a dumb adversary) of winning the game (for a given cost): how good is it?

Security definition: typical games

Two main families:

- Decision games: try to distinguish two outcomes (is this oracle sampled from this or that distribution? is this the left or right choice?). Examples: IND-CCA2; PRP
  - "confidentiality" oriented
  - Measure of success: the *advantage* (over a random choice)
- Search games: try to find something that satisfies some property (passes some verification). Examples: EUF-CMA; second preimage
  - "authentication" oriented
  - Measure of success: success probability (...)

Advantage

Typical setting:

- $[\mathbb{O}]$ a list of oracles, sampled either from $\mathfrak{D}_0$ in *world* 0 or $\mathfrak{D}_1$ in world 1, each with probability 1/2; W an indicator variable for the two worlds
- $A^{[\mathbb{O}]}$ an algorithm with access to $[\mathbb{O}]$ that returns one bit and tries to decide if W is 0 or 1

Déf.: A *wins* if its return value equals the one of W $\rightsquigarrow$ $p_A^{\text{win}} := \Pr[A() = 1 : W = 1] + \Pr[A() = 0 : W = 0]$ $\rightsquigarrow$ extremely trivial to get a “large” (say, 1/2) p^{win} (how?)

Déf.: Distinguishing advantage: $\mathbf{Adv}_A^{\mathfrak{D}_0,\mathfrak{D}_1} := |\Pr[A() = 1 : W = 1] - \Pr[A() = 1 : W = 0]|$ $\rightsquigarrow$ not trivial anymore

Advantage of a problem; advantage function

One may define the advantage of:

- a specific algorithm (cf. above)
- a "full problem", as the max advantage of any algorithm
  - usually under some constraints, e.g. #queries (information-theoretical) or running time ((and memory)) (computational)
  - $\rightsquigarrow$ advantage function: $\mathbf{Adv}^{\mathfrak{D}_0,\mathfrak{D}_1}(q,t) = \max_{A_{q,t}} \mathbf{Adv}_A^{\mathfrak{D}_0,\mathfrak{D}_1}$ ($A_{q,t}$: set of all algorithms that make q queries to their oracle and run in time t)
  - (non-uniform/circuit approach)

Adversarial power: some orders of magnitude

For what kind of t's does it make sense to compute $\mathbf{Adv}(\cdot, t)$?

Say t counts how many times a cheap function is computed. Look at the time/energy/infrastructure to count up to 2^t for $t = \cdots$

- $\approx 40 \rightsquigarrow$ doable w/ a small Raspberry Pi cluster
- $\approx 60 \rightsquigarrow$ doable w/ a large CPU/GPU cluster
  - Already done (equivalently) several times in the academia:
  - Ex. RSA-768 (Kleinjung et al., 2010), 2000 core-years ($\equiv 2^{67}$ bit operations)
  - Ex. DL-768 (Kleinjung et al., 2016), 5300 core-years
  - Ex. SHA-1 collision (Stevens et al., and me!, 2017), 6500 core-years + 100 GPU-year ($\equiv 2^{63}$ hash computations)
- $\approx 80 \rightsquigarrow$ doable w/ an ASIC cluster (cf. Bitcoin mining)

Order of magnitude (cont.)

What about 128?

Objective: run a function 2^{128} times within 34 years ($\approx 2^{30}$ seconds), assuming:

- Hardware at 2^{50} iterations/s (that's pretty good)
- Trivially parallelizable
- 1000 W per device, no overhead (that's pretty good)

$\Rightarrow$

- $2^{128-50-30} \approx 2^{48}$ machines needed
- $\approx 280\,000\,000$ GW 'round the clock
  - $\approx 34\,000\,000$ EPR nuclear power plants (assuming 5 reactors/plant)

Looks hard enough...

¡It's not because you have a 128-bit parameter that one will need 2^{128} evaluations to break your system ‽

Advantage: interpretation guide

- Advantage is (by default) “terminal”: only evaluated when the adversary is done
- (But one may sometimes still *amplify* the advantage of an adversary by using it as a sub-routine)
- The “security level” associated with advantage $2^{-\kappa/2}$ may reasonably be defined as “κ bits” (cf. below)

Bit security

$\mathbf{Adv}^{\mathfrak{D}_0,\mathfrak{D}_1}$ is a function; not always easy to summarise the "security" it defines/assumes. Two natural options to define "'bit security":

- (The most common) As $\kappa := \log(t)$ for the minimal t s.t. $\mathbf{Adv}^{\mathfrak{D}_0,\mathfrak{D}_1}(\cdot, t) \geq c$ for a constant c (e.g. 2/3)
- As $\kappa' := -\log(\mathbf{Adv}^{\mathfrak{D}_0,\mathfrak{D}_1}(\cdot, 1))$

Usually do *not* match for (generic) decision problems: (conjecturally) $\kappa = 2\kappa'$ (cf. Watanabe & Yasunaga, 2021), but *do* match for (a suitable adaptation of the **Adv** def. to) search problems