Introduction to cryptology (GBIN8U16) Final Examination

2020-05-15

Instructions

— All documents are allowed.

— Communication regarding the exam is strictly forbidden.

— *All answers must be carefully justified to get maximum credit.*

— You may answer in English or French.

— Suggested digital format: raw text/markdown. Send at most *two* files (one main file plus one optional appendix) starting with *your student number_your name*.

— Indicative duration: 2 hours.

— Deadline: 2020-05-16T09:00+0200.

— Send your answers to `pierre.karpman@univ-grenoble-alpes.fr`.

Exercise 1: Crypto culture

Name one:

1. Block cipher.
2. MAC.
3. Block cipher mode of operation for *encryption* (at least).
4. Hash function.

Remark: You get double points for each sub-question if no one else provided the same answer, and triple points if the answer additionally cannot be found on wikipedia.

Exercise 2: Unknown babies and giants

Remark: For each algorithm that you need to specify, you may either use pseudocode or a clear textual description.

Let $\mathcal{S}$ be a set of $N \in \mathbb{N}$ arbitrary elements (*i.e.* elements that do not necessarily have a "natural" representation as an integer), where N is *a priori* unknown.

Q.1: You are given an oracle $\mathbb{O}$ that, whenever it is called, returns an element of $\mathcal{S}$ drawn uniformly at random *without replacement* (*i.e.* all elements are equally likely to be returned on the first call, but an element that was returned in a prior call cannot be returned anymore).

1. Specify an algorithm that uses $\mathbb{O}$ and returns N.
2. Analyse its time, query and memory complexity.

Q.2: The oracle $\mathbb{O}$ is now modified such that whenever it is called, it returns an element of $\mathcal{S}$ drawn uniformly at random *with replacement* (*i.e.* all elements are equally likely to be returned on *any* call).

1. Specify an algorithm that uses $\mathbb{O}$ and returns *an estimate* for N *and that has a better time complexity than the algorithm from **Q.1***.
2. Analyse its time, query and memory complexity. Be careful to justify the assumptions you may make on data structures to reach this complexity.
3. Assuming that your algorithm is not very precise in the estimate it returns, how could you make it more so without increasing its complexity?

Let now $\mathbb{G} = \langle g \rangle$ be a finite cyclic group of unknown order (or cardinality) N and g be one of its generators. Suppose that you know how to perform elementary operations in $\mathbb{G}$ (*i.e.* given a, $b \in \mathbb{G}$, you know how to compute $ab \in \mathbb{G}$; given a you know how to compute a^{-1}) in one time unit.

Q.3: Rephrase your algorithm of **Q.1** such that on input g it returns its order N (which is equal to the order of the group $\mathbb{G}$).

Q.4: Suppose now that you already know B and W such that $B \leq N \leq B + W$.

1. Specify an algorithm that takes g as input and returns N with time and memory complexity $\mathcal{O}(\sqrt{W})$.
2. Prove the stated complexity for your algorithm. Be careful to justify the assumptions you may make on data structures and elementary steps.

HINT: Notice that N can be written as $N = B + u\sqrt{W} + v$ where u, $v \in [\![0, \sqrt{W}]\!]$, and that this implies the equality $g^{B+u\sqrt{W}} = g^{-v}$ (since $g^{B+u\sqrt{W}+v} = g^N = g^0 = 1$).

Q.5: Suppose now that you do not know any upper and lower bound for N.

1. Adapt the algorithm of **Q.4** by starting from the (possibly incorrect) assumption that $N \in [\![0, 2]\!]$ and by increasing the size of the estimated interval at every step.
2. Analyse the time and memory complexity of your algorithm.

HINT: Recall that for $q \in \mathbb{R}$ one has that $\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q}$.

Q.6: Assume that $\mathbb{G}$ is a "candidate" group to be used in a Diffie-Hellman key exchange but that one first wants to check that its order is large enough to prevent generic attacks.

1. Which algorithms from **Q.3**$\sim$**5** are appropriate to perform this task?

Exercise 3: Cloaks and passwords, daggers and keys

Q.1: Let p be a passphrase of possibly more than 20 unicode characters, each stored on one byte or more.

1. What cryptographic primitive could you use as a *key derivation function* (KDF) to map p to a 128-bit string suitable for use as the key of a block cipher?

A ring of conspirators all share a common passphrase (such as: "Never believe it. I am more an antique Roman than a Dane. Here's yet some liquor left."). One conspirator is guarding a safe house behind a closed door, while a second wishes to prove his/her membership to the other.

Q.2:

1. Explain how to solve the above problem assuming that the safe house natively benefits from a "secure channel" (for instance one can slip a piece of paper under the door).

Q.2: We now assume that no such secure channel exists (for instance because of the use of a hermetic reinforced door), but that only passive adversaries (*i.e.* eavesdroppers) are a concern. Explain how to solve the problem in this case with a "challenge-response" protocol, and specify the most appropriate security definitions to formally express the requirements:

1. Using a KDF and a block cipher.
2. Using a KDF and a MAC.

Q.3: The conspirator guarding the door had the bad idea of deciding to draw the challenges at random, using a pseudo-random numbers generator with only N possible outputs, where N is "small".

1. Explain how a counterspy that was able to monitor enough (successful) runs of the protocol could gain access to the safe house without knowing the passphrase.
2. Assuming that the PRNG outputs are independent and uniform, how many runs should the counterspy monitor before trying to impersonate a conspirator without (much) risk?

Q.4: An alternative brutal strategy for a counterspy would be to try finding an appropriate secret by performing an exhaustive search.

1. What is the best target for this search in function of P (the number of candidate passphrases), P_R (the number of calls to the (known) conspirators' KDF one can make per second for a fixed unspecified monetary unit ¤), K (the number of possible keys of the used block cipher or MAC), K_R (the number of calls to the (known) conspirators' block cipher/MAC one can make per second for one ¤).

Q.5:

1. Describe a scenario where your protocol of **Q.2** is not sufficient to guarantee the security of the conspirators.

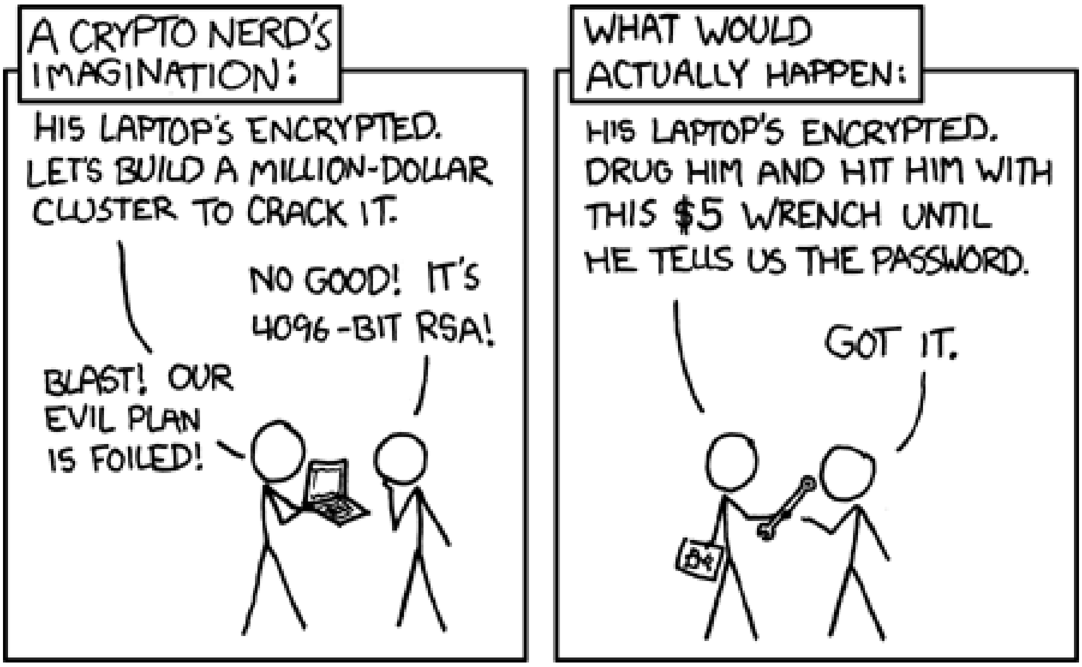

Figure 1: XKCD#538: Not an acceptable answer to **Q.5**.