Introduction to cryptology
(GBIN8U16)
Final Examination

2021-05-12

Instructions

— One two-sided A4 page of (handwritten or typed) notes allowed.

— *Except indicated otherwise, answers must be carefully justified to get maximum credit.*

— Not all questions are independent, but you may admit a result from a previous question by clearly stating it.

— You may answer in English or French.

— Duration: 3 hours.

Notation & definitions

We recall some notation and the following definitions, which are useful in Ex. 1 and 3.

— For any finite set $\mathcal{S}$, we write $X \twoheadleftarrow \mathcal{S}$ to mean that the random variable X is sampled uniformly from $\mathcal{S}$. Furthermore, in notation such as $X \twoheadleftarrow \mathcal{S}, Y \twoheadleftarrow \mathcal{S}$, the samplings of X and Y are independent (except specified otherwise).

— $\mathrm{Perms}(\{0,1\}^n)$ denotes the set of all permutations over $\{0,1\}^n$.

— $\mathrm{Funcs}(\mathcal{X})$ denotes the set of all functions $\mathcal{X} \to \{0,1\}^n$ (where n is either already defined from the context, or introduced by the definition).

— $\cdot||\cdot$ denotes string concatenation.

— $\log(.)$ is the (usual) logarithm function in base two.

Definition 1 (PRP advantage)**.** Let $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher, the *PRP advantage of* E is defined as: $\mathrm{Adv}_E^{\mathrm{PRP}}(q,t) =$

$$\max_{A_{q,t}} |\Pr[A_{q,t}^{\mathbb{O}}() = 1 : \mathbb{O} \twoheadleftarrow \mathrm{Perms}(\{0,1\}^n)] - \Pr[A_{q,t}^{\mathbb{O}}() = 1 : \mathbb{O} = E(k,\cdot), k \twoheadleftarrow \{0,1\}^\kappa]|$$

Where $A_{q,t}^{\mathbb{O}}$ denotes an algorithm that runs in time t and makes q queries to the oracle $\mathbb{O}$ it is given access to.

Definition 2 (PRF advantage)**.** Let $F : \{0,1\}^\kappa \times \mathcal{X} \to \{0,1\}^n$ be a family of functions, the *PRF advantage of* F is defined as: $\mathrm{Adv}_F^{\mathrm{PRF}}(q,t) =$

$$\max_{A_{q,t}} |\Pr[A_{q,t}^{\mathbb{O}}() = 1 : \mathbb{O} \twoheadleftarrow \mathrm{Funcs}(\mathcal{X})] - \Pr[A_{q,t}^{\mathbb{O}}() = 1 : \mathbb{O} = F(k,\cdot), k \twoheadleftarrow \{0,1\}^\kappa]|$$

Where $A_{q,t}^{\mathbb{O}}$ denotes an algorithm that runs in time t and makes q queries to the oracle $\mathbb{O}$ it is given access to.

Definition 3 (Existential forgeries)**.** Let $M : \{0,1\}^\kappa \times \mathcal{X} \to \{0,1\}^n$ be a MAC, then an *existential forgery* for M is an algorithm $A_{q,t}^{\mathbb{O}}$ that takes no input, with oracle access to $\mathbb{O} = M(k,\cdot), k \twoheadleftarrow \{0,1\}^\kappa$ to which it makes q queries, that runs in time t, and which outputs $(x,y) \in \mathcal{X} \times \{0,1\}^n$. The algorithm is said to *win the existential forgery game* if: 1) x was not queried by A to its oracle; 2) $\mathbb{O}(x) = y$. It *loses* otherwise.

Definition 4 (Universal forgeries)**.** Let $M : \{0,1\}^\kappa \times \mathcal{X} \to \{0,1\}^n$ be a MAC, then a *universal forgery* for M is an algorithm $A_{q,t}^{\mathbb{O}}$ that takes an input $x \in \mathcal{X}$, with oracle access to $\mathbb{O} = M(k,\cdot), k \twoheadleftarrow \{0,1\}^\kappa$ to which it makes q queries, that runs in time t, and which outputs $y \in \{0,1\}^n$. The algorithm is said to *win the universal forgery game* if: 1) x was not queried by A to its oracle; 2) $\mathbb{O}(x) = y$. It *loses* otherwise.

Exercise 1: PRP-PRF switching

We first consider an oracle $\mathbb{O} : \{0,1\}^n \to \{0,1\}^n$, which can be one of two things:

— In the *PRP world*, $\mathbb{O} \twoheadleftarrow \mathrm{Perms}(\{0,1\}^n)$. Said otherwise, it samples its outputs uniformly from $\{0,1\}^n$ *without* replacement.

— In the *PRF world*, $\mathbb{O} \twoheadleftarrow \mathrm{Funcs}(\{0,1\}^n)$. Said otherwise, it samples its outputs uniformly from $\{0,1\}^n$ *with* replacement.

Q.1: We consider an algorithm $A_q^{\mathbb{O}}$ which makes q (distinct) queries $x_1, \ldots, x_q$ to its oracle $\mathbb{O}$.

Give an estimate for the probability $\in [0,1]$ that there is a collision between two outputs of $\mathbb{O}$ in the PRP (resp. PRF) world, *i.e.* estimate the following:

1. $p_q^P := \Pr[\exists\, i, j \neq i, \mathbb{O}(x_i) = \mathbb{O}(x_j) : \mathbb{O} \twoheadleftarrow \mathrm{Perms}(\{0,1\}^n)]$;
2. $p_q^F := \Pr[\exists\, i, j \neq i, \mathbb{O}(x_i) = \mathbb{O}(x_j) : \mathbb{O} \twoheadleftarrow \mathrm{Funcs}(\{0,1\}^n)]$.

Only a brief justification of your answers is necessary.

Q.2: *Using your answers to the previous question*:

1. Specify a distinguisher $A^{\mathbb{O}}$ that returns 1 if $\mathbb{O}$ is believed to be in the PRP world, and 0 if it is believed to be in the PRF world.
2. Estimate its advantage $|\Pr[A_q^{\mathbb{O}}() = 1 : \mathbb{O} \twoheadleftarrow \mathrm{Perms}(\{0,1\}^n)] - \Pr[A_q^{\mathbb{O}}() = 1 : \mathbb{O} \twoheadleftarrow \mathrm{Funcs}(\{0,1\}^n)]|$ in function of the number of queries q made to the oracle only (*i.e.* where its running time may be arbitrary).[1]

[1] This is usually called an *information-theoretic* distinguisher, or a distinguisher in the *information theory* setting.

Q.3: We now consider a block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ s.t. $\mathrm{Adv}^{\mathrm{PRP}}_E(q,t) = t/2^\kappa$ when $q > \mathcal{O}(n/\kappa)$. We wish to analyse E in a "PRF setting".

1. Based on your distinguisher from **Q.2** and the definition of E, give a lower-bound for $\mathrm{Adv}^{\mathrm{PRF}}_E(q,t)$. You do not need to specify a matching distinguisher.

Q.4: We now consider a family of functions $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ s.t. $\mathrm{Adv}^{\mathrm{PRF}}_F(q,t) = t/2^\kappa$ when $q > \mathcal{O}(n/\kappa)$.

1. Is it possible to analyse F in a "PRP setting", *i.e.* to study $\mathrm{Adv}^{\mathrm{PRP}}_F(q,t)$?

Q.5:

1. Is it possible and meaningful to use a "good PRP" block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ in a context where a "good PRF" family of functions $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is expected? If yes, what would one "lose" by doing so?

2. Is it possible and meaningful to use a "good PRF" family of functions $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ in a context where a "good PRP" block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is expected? If yes, what would one "lose" by doing so?

Exercise 2: Ideal XOFs

Let $H : \{0,1\}^\star \to \{0,1\}^n$ be an ideal hash function, in that $\forall\, x \in \{0,1\}^\star, H(x) \twoheadleftarrow \{0,1\}^n$ (with the drawings for distinct inputs being independent). We wish to use H to build a hash function with a larger co-domain, while preserving the "idealness" of the resulting construction.

Q.1: We first consider $H' : \{0,1\}^\star \to \{0,1\}^{2n}$, $x \mapsto H(x) || H(x||1)$.

1. Show that not all outputs of H' are independent.

2. Could you call H' an ideal hash function?

Q.2: Let $H_0 : \{0,1\}^\star \to \{0,1\}^n$, $x \mapsto H(0||x)$, $H_1 : \{0,1\}^\star \to \{0,1\}^n$, $x \mapsto H(1||x)$.

1. Show that the *domain-separated* H_0 and H_1 are *independent* ideal hash functions (that is, show that their outputs are uniformly distributed and independent[2]).

Q.3:

1. Using H as a black box, specify (and justify) an ideal hash function construction $H'' : \{0,1\}^\star \to \{0,1\}^{\ell n}$, where $\ell > 1$ is a known integer.

Exercise 3: MAC definitions; RC4-MAC

We first consider a deterministic MAC $M : \{0,1\}^\kappa \times \mathcal{X} \to \{0,1\}^n$.

Q.1: Suppose that you know a universal forgery A for M that wins the universal forgery game with probability p^U and that runs in time t^U and makes q^U queries to its oracle.

1. Specify an existential forgery A' for M that uses A as a black box.

2. Analyse the cost t^E and q^E of A' and its success probability p^E.

[2] Quite obviously, their outputs are *not* independent from the ones of H, but this is irrelevant here.

Q.2: Suppose that you know an existential forgery A for M that wins the existential forgery game with probability p^E and that runs in time t^E and makes q^E queries to its oracle.

1. Specify a PRF distinguisher for M that runs in time $t^F \approx t^E$ and makes $q^F \approx q^E$ queries to its oracle.
2. Give a lower bound for $\mathrm{Adv}_M^{\mathrm{PRF}}(q^F, t^F)$ by analysing the advantage of your distinguisher.
3. Is the following (informally stated) scenario possible: "M is vulnerable to an existential forgery attack, but it is hard to distinguish from a random function"?
4. Show that the following (informally stated scenario) is possible: "There is no efficient existential forgery attack on M, but it is easy to distinguish it from a random function". Only a sketch of proof is required here.

Q.3: Recall that an assumption A_1 is said to be *stronger* than an assumption A_2 if breaking A_2 implies breaking A_1 with a similar cost, but breaking A_1 does not necessarily imply breaking A_2 with a similar cost. Consider the three following (informally stated) assumptions: A_1: M is hard to distinguish from a random function; A_2: there is no efficient universal forgery on M; A_3: there is no existential forgery on M.

1. Order the assumptions A_1, A_2, A_3 from weakest to strongest. *Be careful to justify your answer.*
2. Suppose that you need a MAC algorithm, and are magically given access to one that satisfies an assumption that you are free to choose; which of A_1, A_2 or A_3 would you pick (and why)?

RC4 is a stream cipher that can be used to (poorly) encrypt binary strings of arbitrary length in the following way:

1. Two communicating parties share a secret key k.
2. For each new plaintext p to be encrypted, one picks a unique initialisation vector v.
3. One runs a setup algorithm on the pair (k, v) that returns an initial state s (that depends on both k and v).
4. One runs the RC4 keystream generator on s, producing a keystream z of the same length as p.
5. The encryption of p is returned as $c := p \oplus z$, along with the initialisation vector v.

A designer suggests to use RC4 as the basis of a MAC algorithm. For simplicity, we assume that the input is at least 128-bit long, or that it has otherwise been padded up to that length (or longer) using an appropriate injective padding scheme. To authenticate a message one runs RC4 encryption on the input and returns the last 128 bits of the ciphertext as a tag. In more details:

1. Two communicating parties share a secret key k.
2. One runs a setup algorithm on the pair $(k, 0)$ that returns an initial state s.
3. For each new input x to be authenticated, one runs the RC4 keystream generator on s, producing a keystream z of the same length as x.
4. One encrypts x as $c := x \oplus z$; the last 128 bits of c are returned as the authentication tag of x.

Q.4:

1. Give (and analyse) a very efficient attack on RC4-MAC with respect to one of the three security notions studied in this exercise.

Exercise 4: Discrete logarithms with low weight

In this exercise, $\mathbb{G} = \langle g \rangle$ is a cyclic finite group of prime order p (*i.e.* with p elements $g^0, \ldots, g^{p-1}$). We write $n := \lceil \log(p) \rceil$, with the logarithm taken in base 2.

We recall that in such a group, the map $x \mapsto g^x$ is efficiently computable, but that its inverse $g^x \mapsto x$ is not in general; since inverting an *element* of the group is easy, the map $x \mapsto g^{-x}$ is also efficiently computable. We also recall that the baby-step/giant-step framework may be used to solve a discrete-logarithm problem in $\mathbb{G}$ with respect to the generator g, in the following way: first define ν as $\lceil \sqrt{p} \rceil$, then precompute the list $L_G := [(i, g^{i\nu});\ 0 \leq i \leq \nu]$, and on input g^a compute $L_B := [(i, g^a g^{-i});\ 0 \leq i \leq \nu]$. A collision between the two lists reveals the value $a \in [\![0, p-1]\!]$ of the desired discrete logarithm.

We now wish to design algorithms specialised to the case where the discrete logarithm of the input is known to have a small (binary) *weight*, where the weight $\mathrm{wt}(a)$ of an integer $a \in [\![0, p-1]\!]$ is defined as the integer w s.t. $a = \sum_{i=1}^{w} 2^{a_i}$, with the a_i's pairwise distinct integers in $[\![0, n-1]\!]$. In other words, w is the number of non-zero bits in the binary expansion of a. An instance of this problem "LWDLP" is specified as (g, g^a, w), with $\mathrm{wt}(a) = w$.

Q.1:

1. Specify a naïve exhaustive search algorithm for the LWDLP.
2. Do a time and memory cost analysis of this algorithm in the worst case.
3. Compare (roughly) this cost with the one of the baby-step/giant-step algorithm when $n = 256$ and $w = 10$.[3]

Q.2: Let now a be of even weight w and n be even.

1. Show that g^a can be written as $g^{a_B} g^{a_G}$ where $\mathrm{wt}(a_B) = \mathrm{wt}(a_G) = w/2$.

Let (g, g^a, w) be an LWDLP instance, and assume that you additionally know a partition of $[\![0, n-1]\!]$ into two sets $\mathcal{S}_B$ and $\mathcal{S}_G$ of size $n/2$ s.t. $\{a_1, \ldots, a_{w/2}\} \subseteq \mathcal{S}_B$ and $\{a_{w/2+1}, \ldots, a_w\} \subseteq \mathcal{S}_G$.

2. Specify a baby-step/giant-step algorithm for this variant of the LWDLP.
3. Do a time and memory cost analysis of this algorithm in the worst case.
4. Compare (roughly) this cost with the one of **Q.1** when $n = 256$ and $w = 10$.
5. In the general statement of the LWDLP, you do not know a partition $\mathcal{S}_B \cup \mathcal{S}_G$ of the above form; propose a (possibly randomised) strategy to accommodate this issue (no analysis of the resulting algorithm is required).

Q.3:

1. Given the current state of the art in computers performance, what could you say is a drawback of the baby-step/giant-step framework to solve a hard problem such as the LWDLP?

[3] You may use $\log(10!) \approx 21.8$.