Cryptography and Security — Final Exam
Solution

Serge Vaudenay

28.1.2021

- duration: 3h
- no documents allowed, except one 2-sided sheet of handwritten notes
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **not** answer any technical question during the exam
- readability and style of writing will be part of the grade

The exam grade follows a linear scale in which each question has the same weight.

1 Merkle's Puzzles

The following exercise is inspired from Secure Communications over Insecure Channels *by Merkle, published in the Communication of the ACM of April 1978 (pp. 294–299).*

We define a primitive which we call *puzzle* with the following algorithms:

- $\mathsf{Gen}(s, \lambda, i, k; r) \to p$ is probabilistic and generates a "puzzle" p from some input index i, a key k, a difficulty s, a parameter λ, and random coins r. We call a unit of time its average time complexity.
- $\mathsf{Solve}(s, \lambda, p) \to (i, k)$ is deterministic and recovers i and k from s, λ, and p. Its time complexity is upper bounded by 2^s units of time.

The correctness notion says

$$\forall s, \lambda, i, k \qquad \Pr[\mathsf{Solve}(s, \lambda, \mathsf{Gen}(s, \lambda, i, k)) \to (i, k)] \geq 1 - 2^{-\lambda}$$

The security notion intuitively says that any solving algorithm requires at least an average complexity of $\frac{1}{2}2^s$ units of time.

Let Enc be a symmetric encryption scheme over the space of (i, k, z) tuples (where z is a bitstring of length $s + \lambda$) with a key of length at least s. We assume that encryption and decryption take approximately the same amount of time. Given s random coins r, we define $\mathsf{Gen}(s, \lambda, i, k; r) = \mathsf{Enc}_{r\|0^*}(i, k, 0^{s+\lambda})$, where $r\|0^*$ is r padded with enough zero bits to be a key of correct length and $0^{s+\lambda}$ is a string of $s + \lambda$ zero bits.

Q.1 Prove that we can define a puzzle based on Gen (i.e. propose a Solve and verify all conditions of a puzzle).

We define $\mathsf{Solve}(s, \lambda, p)$ *as follows:*

1: ***for*** *all* r *of* s *bits* ***do***
2: $\quad \mathsf{Dec}_{r\|0^*}(p) \to x$
3: $\quad$ ***if*** x *parses to* $(i, k, 0^{s+\lambda})$ *for some* i *and* k ***then***
4: $\quad\quad$ ***return*** (i, k)
5: $\quad$ ***end if***
6: ***end for***

The number of iterations is upper bounded by 2^s*, which is also the complexity.*
Given s*,* λ*,* i*,* k*, the incorrect case corresponds to the existence of* r *and* r' *such that* $r \neq r'$ *and* $\mathsf{Dec}_{r'\|0^*}(\mathsf{Enc}_{r\|0^*}(i, k, 0^{s+\lambda}))$ *parses to* $(i', k', 0^{s+\lambda})$ *for some* i' *and* k'*. Given* r *and* r'*, this parse occurs with probability* $2^{-s-\lambda}$*, assuming* Dec *behaves like random. Since there are* $2^s - 1$ *possible* r'*, it exists with probability bounded by* $2^{-\lambda}$*. (Ok, this is a non-rigorous proof.)*
If the encryption is well done, solving the puzzle intruitively implies finding the right key r *which can only work by exhaustive search. (Ok, this is an even less rigorous proof.)*

Q.2 Let s, λ, and N be three public parameters. Alice generates $\mathsf{Gen}(s, \lambda, i, k_i; r_i) \to p_i$ for $i = 1, \ldots, N$ and randomly selected k_i and r_i. Alice picks a random permutation σ of $\{1, \ldots, N\}$, defines $p'_j = p_{\sigma(j)}$, $i = 1, \ldots, N$, and sends $p'_1, \ldots, p'_N$ to Bob. Bob picks a random j, runs $\mathsf{Solve}(s, \lambda, p'_j) \to (i, k_B)$ and sends i back to Alice. Alice sets $k_A = k_i$. The private outputs of Alice and Bob are k_A and k_B respectively.
Prove that we have a key agreement protocol (i.e. prove correctness). What is the complexity for Alice and Bob?

We have $(i_B, k_B) = \mathsf{Solve}(s, \lambda, \mathsf{Gen}(s, \lambda, \sigma(j), k_{\sigma(j)}; r_{\sigma(j)}))$*. Due to the correctness of the puzzle, with probability at least* $1 - 2^{-\lambda}$ *we have* $(\sigma(j), k_{\sigma(j)}) = (i_B, k_B)$ *so* $k_A = k_{i_B} = k_{\sigma(j)} = k_B$*.*
The complexity for Alice is roughly of N *units. The complexity for Bob is upper bounded by* 2^s *units.*

Q.3 In Q.2, what is the best passive attack Eve could do? (Study its complexity.) As s and N are based on the security parameter, propose a way to select them coherently.

If Eve solves a random subset of m *puzzles from Alice, it costs at least* $\frac{m}{2}2^s$ *and the probability that Bob solves one from these is* $\frac{m}{N}$*. Hence, the complexity/probability ratio is at least* $\frac{N}{2}2^s$*.*
It is reasonable to take $N = \frac{1}{2}2^s$ *to equate the complexity of Alice and Bob. Hence, the complexity of both Alice and Bob is* N *while the complexity/probability of Eve is* N^2*.*
This protocol is not so efficient but it was the very first proposal towards a secure key agreement protocol.

2 Crypto Choices

We want to design a secure communication system in which breaking it would be equivalent, in terms of complexity, to do an exhaustive search on a 128-bit secret key. In this exercise, we have to *propose* concrete cryptographic algorithms with some concrete parameters (such as the length of various objects). (*Proposing* can mean to name an existing standard algorithm instance which could be used with brief justifications.) To get the full grade, we should propose parameters which are neither too small not too large.

Q.1 Propose a symmetric encryption scheme for data of variable length. Specify the key length. Which security property is this algorithm supposed to guarantee?

We can suggest AES with 128-bit key. To accommodate variable length input, we should use it in CTR mode. This is made to protect the confidentiality of communication.

Q.2 For a message authentication code, what would be the appropriate key length and tag length? Propose an algorithm with those parameters. Which security property is this algorithm supposed to guarantee?

The appropriate key length is 128 bits. The appropriate tag length for unforgeability is also 128 bits. We could use HMAC-SHA256 with a tag truncate to 128 bits and a key of the same length but this would not be so efficient because SHA256 produce digests of 256 bits. A better choice would be to use CMAC which is also based on AES. We can actually combine with the previous question and use AES-CCM or AES-GCM. A MAC is made to authenticate data and to protect integrity.

Q.3 We want to hash a message to sign its digest. Propose an appropriate hash function. Which security property is important when used in hash-and-sign?

We can propose SHA256 which makes 256-bit digests. The important property is collision resistance to make sure that digests are unique.

Q.4 Propose a signature scheme which is appropriate with the proposed hash function.

We can use ECDSA on the P256 curve. It is a curve with close to 2^{256} *points working over the field of residues modulo a 256-bit prime number.*
Alternately, we can use RSA-PSS with a modulus of at least 2048 bits (more would be recommended, actually).

Q.5 Revisit all questions to offer the same security in a post-quantum era.

For encryption, we switch to AES-CTR with a 256-bit key.
For MAC, we can still use CMAC with 256-bit key. The tag-length (128 bits) is in theory too small but quantum forgery attacks exploiting a too small tag are harder than others.
We can use SHA512.
Signature cannot be ECDSA nor RSA. We need a post-quantum signature scheme (to come) for that.

3 ECDSA with Bad Randomness

We recall the ECDSA scheme. A point G in an elliptic curves generates a group of order a public prime n. The secret key is a residue d modulo n. The public key is a point $Q = dG$. To sign a message M, the signer picks a random $k \in \mathbf{Z}_n^*$ and computes $r = \mathsf{xcoord}(kG) \bmod n$ and $s = \frac{H(M)+dr}{k} \bmod n$, where H is a hash function. The signature is the pair (r, s). The signature is such that

$$r = \mathsf{xcoord}\left(\frac{H(M)}{s}G + \frac{r}{s}Q\right)$$

We consider a network of signers where the signer of index i has a key pair (d_i, Q_i). Signers use devices to sign but those devices have a poor random generator. We assume that there exists a subset of $\mathbf{Z}_n^*$ of cardinality v such that the random generator outputs k uniformly in this subset. We assume that n has 256 bits and that v has 100 bits.

Q.1 Describe a key recovery attack targetting one user. Analyze how many signatures are needed, the complexity, and the probability of success. Is this attack realistic?

After $\sqrt{v}$ signatures are made by the targetted signer, there is a constant probability of success that there exists two signatures having selected the same k, due to the birthday paradox. These two signatures are visible by having the same r. Let s_1 and s_2 be the two s values. We have $ks_1 = H(M_1) + dr$ and $ks_2 = H(M_2) + dr$ modulo n. This is a system of two linear equations in two unknowns k and d which can be easily solved to recover the secret key d.
We need $\sqrt{v} \sim 2^{50}$ signatures, a complexity of $\sqrt{v} \sim 2^{50}$, and a constant probability of success.
The attack is feasible. However, it is not realistic to assume that a single honest user will sign 2^{50} documents.

Q.2 In a network of N signers, we assume that each signer signs w documents at random. We assume $w \ll \sqrt{v}$. Given two different signers, what is (approximately) the probability p that they once used the same k value?

We count pairs of k values between the two users. We have w^2 pairs and each pair is matching, i.e. has twice the same k, with probability $\frac{1}{v}$. Hence, the expected number of matching pairs is $\frac{w^2}{v}$. If we neglect the probability to have several matching pairs, we obtain $p \approx \frac{w^2}{v}$.

Q.3 If each signer signs w documents, design an attack which recovers the key of some signers with good probability when $N \sim vw^{-2}$. Apply this to $w \sim 2^{35}$.
HINT: In a random graph of N vertices in which each possible edge is present with probability $\frac{1}{N}$, there exists a cycle with good probability.

We draw a graph in which each vertex represents a signer and each edge represent a value k in common between two signers. Each vertex has an unknown d and each edge has an unknown k. Due to our setting and the previous question, we have a random graph of N vertices in which each possible edge is present with probability $\frac{1}{N}$. Hence, we have a cycle with good probability. A cycle of t length t would contain t vertices and t edges and define $2t$ linear equations in $2t$ unknowns, which is solvable. This recovers the secret key of t signers.
With $w \sim 2^{35}$, $v \sim 2^{100}$, and $N \sim vw^{-2} \sim 2^{30}$,

4 Post-Quantum Bitcoins Wallet

In the bitcoin architecture, a transaction is an ECDSA signature by a holder of a key pair $(\mathsf{pk}, \mathsf{sk})$ on a document which contains

- pk;
- a list $p_1, \ldots, p_m$ of pointers to some transactions in which p_i paid x_i bitcoins to pk which have not been spent so far;
- a list $(y_1, \mathsf{pk}_1), \ldots, (y_n, \mathsf{pk}_n)$ of public keys to whom to pay the collected bitcoins and of positive numbers y_j such that $x_1 + \cdots + x_m = y_1 + \cdots + y_n$.

When a user wants to pay some other users for a total of $y_1 + \cdots + y_{n-1}$, he/she collects some of his bitcoins $x_1 + \cdots + x_m$ so that $x_1 + \cdots + x_m \geq y_1 + \cdots + y_{n-1}$, he/she computes y_n such that $x_1 + \cdots + x_m = y_1 + \cdots + y_n$ and if $y_n > 0$, he/she adds (y_n, pk) to pay the leftover to himself.

Q.1 In a quantum era, show that as soon as the transaction is made public, a powerful adversary can steal the $y_1 + \cdots + y_n$ bitcoins.

We consider an adversary who is powerful enough to possess a quantum computer. This adversary can compute the discrete logarithm of every pk_j, which appear in the transaction, by using the Shor algorithm, and obtain the secret keys sk_j. With this, the adversary can sign n transactions. The j-th transaction is signed with sk_j, the link to the posted transaction, and (y_j, pk') some some pk' of his choice. It will put the $y_1 + \cdots + y_n$ bitcoins on account pk'. Then, the adversary should cash those bitcoins before another adversary steals them.

Q.2 We change the structure of the transaction so that the (y_j, pk_j) pairs in the list of the recipients of the transaction are replaced by some $(y_j, H(\mathsf{pk}_j))$ pairs, by using a one-way hash function H. What users make public is now $H(\mathsf{pk})$, but users reveal their pk in the transaction. Using a notion of "one-time secret key" sk, show that we can mitigate the previous attack.

We now assume that pk is not public any more but that the hash of it is public. Until pk is revealed, its discrete logarithm cannot be computed. User could still receive payments on their $H(\mathsf{pk})$. The problem is that pk must be revealed when the user makes a transaction. Hence, one solution is that this user collects all his bitcoins $x_1 + \cdots + x_m$, makes his transaction, but pay the leftover to a freshly made pk'. He/she should not pay to the same pk. Hence, his/her account pk has no more bitcoins and he/she will use a new account pk'. Payments to his previous account should not be made any more. In such situation, sk is used only once.

Q.3 Describe a structure of a wallet in which each user would have a unique secret K.

We can use K as a key of a PRF. We define $\mathsf{sk}_t = \mathsf{PRF}(K, t)$ for $t = 0, 1, \ldots$ We define the exponentials pk_t and their hash. The wallet keeps a counter t. To receive a payment, the user gives $H(\mathsf{pk}_t)$ as a payment address. Every time the user makes a transaction, it uses sk_t then increment t.