Cryptography and Security — Final Exam
Solution

Serge Vaudenay

16.1.2023

- 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
- answers should not be written with a pencil

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

1 Symmetric-Key RSA

As people scare about quantum computers being able to factor RSA moduli, some people proposed to continue to use RSA by keeping the public key secret and end up with an encryption scheme with a symmetric key $K = (N, e, d)$. For simplicity, we consider plain RSA only. The purpose of the exercise is to make a quantum key recovery attack with chosen plaintext.

Q.1 Fully describe the symmetric-key RSA scheme between a sender Alice and a receiver Bob (key structure, message domains, key generation, encryption, decryption).

Key generation: Bob generates two different large prime numbers p and q and select an integer e which is coprime with $(p-1)(q-1)$. Compute $N = pq$ and d the multiplicative inverse of e modulo $(p-1)(q-1)$. Set the key to $K = (N, e, d)$. The key must be securely transmitted from Bob to Alice. We assume that N has ℓ bits, i.e. $2^{\ell-1} \leq N < 2^\ell$.
The message space is the set of bit strings. Each bitstring is encoded as an integer using the binary representation. Zero leading bits are ignored. A plaintext is limited to $\ell - 1$ bits. Ciphertext can have ℓ bits.
Encryption: the encryption of x with key $K = (N, e, d)$ is $y = x^e \bmod N$. Alice encrypts x this way then sends y to Bob.
Decryption: the decrption of y with key $K = (N, e, d)$ is $x = y^d \bmod N$. Bob decrypts y this way and retreives the plaintext x.

Q.2 Describe the game for key recovery with chosen plaintext with the symmetric scheme of the previous question.

```
1: run the key generation to select $K = (N, e, d)$
2: run $\mathcal{A}^{\mathsf{OEnc}} \to K'$
3: return $1_{K=K'}$
Oracle $\mathsf{OEnc}(x)$:
4: return $x^e \bmod N$
```

Q.3 In the case where e is small (e.g. $e = 65\,535$) and known by the adversary, propose an efficient (quantum) key-recovery attack with one known plaintext.

If the adversary gets an (x, y) pair, the adversary can compute $N' = x^e - y$ because e is small. This is a multiple of N. Then, the adversary can factor N' and isolate the two prime factors p and q of right length. Then, N and d can be derived.

Q.4 Propose an efficient (quantum) key-recovery attack with two chosen plaintexts, (e is not known any more and can be large as well).

The adversary can select a small x and get the encryption of x and x^2 which we denote by y and y' respectively. Then, the adversary computes $N' = y^2 - y'$ which is a multiplie of N. Then, the adversary can factor N' and isolate the two prime factors p and q of right length and deduce $N = pq$.
The adversary can then compute $\bar{x} = x \bmod p$ and $\bar{y} = y \bmod p$, then the discrete logarithm of $\bar{y}$ in basis $\bar{x}$ to deduce e. After getting e, the adversary can compute its inverse d modulo $(p-1)(q-1)$.

2 Hash-Based Signature

We consider a one-way hash function F from a set E to itself. We further consider a collision-resistant hash function H mapping an arbitrary message m to a digest $H(m)$ belonging to a given hash space. We analyze some digital signature schemes based on F and H.

Q.1 We recall the Lamport scheme with parameter n.

- Key generation: pick $2n$ random $\mathsf{sk}_{i,b} \in E$ for $i = 1, \ldots, n$ and $b \in \{0, 1\}$, compute $\mathsf{pk}_{i,b} = F(\mathsf{sk}_{i,b})$. We set $\mathsf{sk} = (\mathsf{sk}_{i,b})_{i=1,\ldots,n;b=0,1}$ and $\mathsf{pk} = (\mathsf{pk}_{i,b})_{i=1,\ldots,n;b=0,1}$.
- Hash space: $H(m) \in \{0, 1\}^n$.
- Signature: $\sigma = (\mathsf{sk}_{i,H(m)_i})_{i=1,\ldots,n}$.
- Verification: check $F(\sigma_i) = \mathsf{pk}_{i,H(m)_i}$ for $i = 1, \ldots, n$.

The scheme should be used to sign a single message but we investigate what happens if we sign several.

Q.1a Assume the adversary knows two signed messages (m_1, σ_1) and (m_2, σ_2) such that $H(m_1)$ and $H(m_2)$ differ on exactly d bit positions. Given a random m, what is the probability that the adversary can forge a signature for m?

The probability corresponds to $H(m)_i \in \{H(m_1)_i, H(m_2)_i\}$ for every i. For each i in which $H(m_1)$ and $H(m_2)$ differ, this always happens. For other i (there are $n - d$ of them), this happens with probability $\frac{1}{2}$. Hence, the probability to be able to forge is $\frac{1}{2^{n-d}}$.

Q.1b If m, m_1, m_2 are random, what are the expected value of d and the probability to forge?

The expected distance between $H(m_1)$ and $H(m_2)$ is $\frac{n}{2}$. The probability is roughly $2^{-\frac{n}{2}}$ if we directly plug the expected value of d in the formula. More precisely, the expected probability is

$$\sum_{d=0}^{n} \frac{1}{2^{n-d}} \binom{n}{d} 2^{-n} = \left(\frac{3}{4}\right)^n$$

Q.1c Propose a key-recovery chosen message attack using $\mathcal{O}(\log n)$ chosen messages, similar complexity, and success probability 1.

The idea is that we repeatedly pick random messages so that their signature would reveal half of the unknown secrets.

```
1: pick $m_1$ at random
2: set $I = \{1, \ldots, n\}$
3: for $j = 2$ to $\log_2 n$ do
4:     repeat
5:         pick $m_j$ at random
6:         set $d = \sum_{i \in I} 1_{H(m_j)_i \neq H(m_1)_i}$
7:     until $d \geq \frac{1}{2}\#I$
8:     set $I \leftarrow \{i \in I; H(m_j)_i = H(m_1)_i\}$
9: end for
10: return $m_1, m_2, \ldots$
```

We keep in I the indices i for which we are missing one $\mathsf{sk}_{i,b}$ value. At each iteration, we make sure that the selection of a new message reduces the size of I by at least half. Hence, the signature of all m_j reveals all secrets.
The repeat loop clearly repeats a constant number of times. Therefore, the complexity is $\mathcal{O}(\log n)$ and the success probability is 1.

Q.2 We recall the FORS scheme with parameters k and t.

- Key generation: pick kt random $\mathsf{sk}_{i,j} \in E$ for $i = 1, \ldots, k$ and $j = 1, \ldots, t$, compute $\mathsf{pk}_{i,j} = F(\mathsf{sk}_{i,j})$. We set $\mathsf{sk} = (\mathsf{sk}_{i,j})_{i=1,\ldots,k; j=1,\ldots,t}$ and $\mathsf{pk} = (\mathsf{pk}_{i,j})_{i=1,\ldots,k; j=1,\ldots,t}$.
- Hash space: $H(m) \in \{1, \ldots, t\}^k$.
- Signature: set $\sigma = (\mathsf{sk}_{i,H(m)_i})_{i=1,\ldots,k}$.
- Verification: check $F(\sigma_i) = \mathsf{pk}_{i,H(m)_i}$ for $i = 1, \ldots, k$.

This scheme is meant to be used to sign a few messages.

Q.2a After the signature of n random messages, what is, for each i, the expected number of indices j for which $\mathsf{sk}_{i,j}$ is revealed?

An $\mathsf{sk}_{i,j}$ remains secret after n signatures with probability $\left(1 - \frac{1}{t}\right)^n \approx e^{-\frac{n}{t}}$. So, the average number of revealed ones is $t(1 - e^{-\frac{n}{t}})$ for each i.

Q.2b Compute roughly the probability to be able to forge the signature of a random message after n random messages have been signed.

The probability that a required $\mathsf{sk}_{i,j}$ is known is $1 - e^{-\frac{n}{t}}$. Hence, the probability to be able to forge the signature is $(1 - e^{-\frac{n}{t}})^k$.

Q.2c Application: $k = 33$, $t = 2^6$. How many random messages can we sign without this probability becoming larger than $\frac{1}{2}$?

For $n = 247$, we have $(1 - e^{-\frac{n}{t}})^k < \frac{1}{2}$. For $n = 248$, we have $(1 - e^{-\frac{n}{t}})^k > \frac{1}{2}$. So, we can now sign 247 random messages. This is better than the $\log_2 n$ of the Lamport scheme.

3 DLP in GGM

We define the Discrete Logarithm Problem (DLP) in the Generic Group Model (GGM) as follows. Given a prime number q, we define the following game Γ:

```
Γ^A
1: pick $x \in \mathbf{Z}_q$ uniformly at random
2: set $\mathsf{Mem} \leftarrow (1, x)$
3: run $\mathcal{A}^{\mathsf{OAdd},\mathsf{OCmp}}(q) \rightarrow y$
4: return $1_{x=y}$

Oracle OAdd(i, j):
5: $S \leftarrow \mathsf{Mem}[i] + \mathsf{Mem}[j] \bmod q$
6: append $S$ to the list Mem
7: return

Oracle OCmp(i):
8: return $1_{\mathsf{Mem}[i]=0}$
```

We use a memory Mem which is defined as a list of write-only registers. If Mem is of length n, the registers are $\mathsf{Mem}[1], \ldots, \mathsf{Mem}[n]$. In Step 2, Mem is initialized with length $n = 2$ so that $\mathsf{Mem}[1] = 1$ and $\mathsf{Mem}[2] = x$. In Step 6, the length of Mem is incremented: Appending a value S to Mem means defining a new register $\mathsf{Mem}[n+1]$ set to S. Essentially, this model allows the adversary $\mathcal{A}$ to do group operations over $\mathbf{Z}_q$ in a blind manner through the OAdd oracle. The adversary does not see the content of the memory Mem but knows the group order q. Additionally, the adversary can test is a register contains 0 through the OCmp oracle. The DLP is then defined in the usual manner. The advantage of $\mathcal{A}$ is

$$\mathsf{Adv}_{\mathcal{A}} = \Pr[\Gamma^{\mathcal{A}} \rightarrow 1]$$

The goal of the exercise is to show that DDH is hard in GGM.

Q.1 Let $a, b \in \mathbf{Z}_q$ be fixed. Construct an (as efficient as possible) adversary $\mathcal{A}$ so that at the end of the game, the last memory register contains $a + bx \bmod q$. Analyze its complexity.

We use the double-and-add algorithm to have $a = a.\mathsf{Mem}[1]$ in a register, then $bx \equiv b.\mathsf{Mem}[2]$, then we finally add both results.
For instance, $a.\mathsf{Mem}[1]$ would first require to compute $2^i.\mathsf{Mem}[1]$ for $i = 1, 2, \ldots, \lfloor \log_2(a) \rfloor$. This is done by the sequence $\mathsf{OAdd}(1,1)$, $\mathsf{OAdd}(2,2)$,i $\mathsf{OAdd}(3,3)$, ... Then, we add together the results corresponding to a bit of a equal to 1. In the worst case, we have $2\lfloor \log_2(a) \rfloor$ oracle calls to OAdd.
The complexity, measures in the number of oracle calls to OAdd is bounded by $4\lfloor \log_2(q) \rfloor + 1$.

Q.2 By using only OAdd and OCmp and given two integers i and j, show how $\mathcal{A}$ can efficiently determine whether $\mathsf{Mem}[i] = \mathsf{Mem}[j]$ or not. Analyze its complexity.

One idea is first to compute $\mathsf{Mem}[i] + (q-1)\mathsf{Mem}[j]$, which would be the difference, then making a OCmp call to the result. Applying the previous question, the number of OAdd is bounded by $2\lfloor \log_2(q) \rfloor + 1$ and we need an extra OCmp.

Q.3 Propose an adversary $\mathcal{A}$ of advantage 1 of minimal complexity to solve DLP. Carefully discuss if it fits the GGM model.

We can use the baby-step giant-step algorithm which works in complexity $\mathcal{O}(\sqrt{q})$. One problem is that we use a hash table with group elements as a key, which is not offered in GGM. To emulate this data structure, we would need to make specific OCmp *comparisons which would increase the complexity to $\mathcal{O}(q)$.*
The Polard-Rho algorithm cannot be used as is because it uses the value of group elements which is not available here.

Q.4 If $\mathcal{A}$ never queries OCmp, prove that $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \frac{1}{q}$.

If no OCmp *is used, the adversary never gets any information about the content of the registers. Hence, what the adversary sees is independent from x. So, the output y is also independent from x. Since x is uniformly distributed, we have $x = y$ with probability $\frac{1}{q}$.*

Q.5 Prove by induction that a process which observes the queries made by $\mathcal{A}$ to the oracle can define 2-dimensional vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ such that for every i, we have $\mathsf{Mem}[i] = \boldsymbol{v}_i[1] + \boldsymbol{v}_i[2] \times x \bmod q$.

We can already see that it is true for $n = 1$ and $n = 2$ with $\boldsymbol{v}_1 = (1, 0)$ and $\boldsymbol{v}_2 = (0, 1)$. If this is true for the first $n-1$ register values, the n-th one is the result of an addition between two previous register values of some index i and j. Hence, we can set $\boldsymbol{v}_n = \boldsymbol{v}_i + \boldsymbol{v}_j \bmod q$ to obtain the result.

Q.6 Given $(a, b) \in \mathbf{Z}_q^2$ such that $(a, b) \neq (0, 0)$, prove that $\Pr[a + bx \bmod q = 0] \leq \frac{1}{q}$ over the random selection of $x \in \mathbf{Z}_q$.

If $b = 0$, we get that $a \neq 0$ so the probability is 0. If $b \neq 0$, we have $a + bx = 0$ equivalent to $x = -\frac{a}{b}$ which occurs with probability $\frac{1}{q}$. In all cases, $\Pr[a+bx \bmod q = 0] \leq \frac{1}{q}$.

Q.7 We define the alternate comparison procedure which is formalized as a subroutine of the adversary:

Subroutine $\mathsf{AltCmp}(i)$:

```
1: return $1_{v_i=(0,0)}$
```

where v_i is obtained from Q.5. We define $\mathcal{A}_t$, the adversary running exactly as $\mathcal{A}$ except that the t first queries to OCmp are made to AltCmp instead of Ocmp.
Prove that $\mathsf{Adv}_{\mathcal{A}_t} \leq \mathsf{Adv}_{\mathcal{A}_{t-1}} + \frac{1}{q}$.
HINT: define the event E that $\boldsymbol{v}_i \neq 0$ and $\mathsf{Mem}[i] = 0$, where i is the index which is queried to the t-th comparison oracle call.

We compare the executions of $\mathcal{A}_{t-1}$ and $\mathcal{A}_t$ step by step. They only differ when the t-th comparison is queried with some given input i. We let E be the event that $\boldsymbol{v}_i \neq 0$ and $\mathsf{Mem}[i] = 0$. If E does not occur, the two executions continue the same way. Hence, the advantage difference is bounded by $\Pr[E]$. Clearly, whatever $\mathcal{A}$ does before this time is independent from x, as no OCmp *call is made. Hence, $\boldsymbol{v}_i$ is independent from x. Using the previous questions, we have $\Pr[E] \leq \frac{1}{q}$.*

Q.8 Deduce $\mathsf{Adv}_{\mathcal{A}}(\lambda) \leq \frac{n+1}{q}$ when $\mathcal{A}$ is limited to n oracle calls to OCmp.

We apply $\mathsf{Adv}_{\mathcal{A}_t} \leq \mathsf{Adv}_{\mathcal{A}_{t-1}} + \frac{1}{q}$ *n times to obtain* $\mathsf{Adv}_{\mathcal{A}} \leq \mathsf{Adv}_{\mathcal{A}_n} + \frac{n}{q}$. *Then, we apply* $\mathsf{Adv}_{\mathcal{A}_n} = \frac{1}{q}$ *as it uses no* OCmp *comparison. It uses* AltCmp *comparisons which are done by the adversary without any oracle call.*