Cryptography and Security — Final Exam
Solution

Serge Vaudenay

24.1.2020

- 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 On Combining Two Hash Functions by Concatenation

The following exercise is inspired from Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions *by Joux, published in the proceedings of CRYPTO 2004 (2019 test-of-time award winner).*

In what follows, C is a compression function mapping a d-bit chaining value h and a ℓ-bit message block x to a d-bit value $C(h,x)$. Given the ℓ-bit blocks $x_1,\ldots,x_n$, we define

$$H(x_1,\ldots,x_n) = C(\ldots C(C(0^d,x_1),x_2)\ldots,x_n)$$

where 0^d is the bitstring of d bits with all bits set to 0.

Q.1 We assume that there is an algorithm $\mathcal{A}(h)\to(x,x')$ which, from h, produces a random pair of different ℓ-bit blocks x and x' such that $C(h,x)=C(h,x')$. We let T be the complexity of running the algorithm $\mathcal{A}$.
For $n\le d$, construct an algorithm, of complexity T multiplied by something small (i.e. less than $P(d)$ for some polynomial P), which returns $x_{i,j}$ ($i=1,\ldots,n$, $j=0,1$) such that for any $b_1,\ldots,b_n\in\{0,1\}$, we have $H(x_{1,b_1},\ldots,x_{n,b_n})=H(x_{1,0},\ldots,x_{n,0})$, and $x_{i,0}\neq x_{i,1}$ for $i=1,\ldots,n$.

We simply run

1: $h\leftarrow 0^d$
2: ***for*** $i=1$ *to* n ***do***
3: $\quad(x_{i,0},x_{i,1})\leftarrow\mathcal{A}(h)$
4: $\quad h\leftarrow C(h,x_{i,0})$
5: ***end for***

The complexity is essentially Tn plus the management of registers, which is small.

Q.2 Let H' be another hash function which hashes onto d' bits. We consider the combined hash function

$$\mathcal{H}(x) = H(x) \| H'(x)$$

(I.e. the concatenation of the two hash functions H and H'.) As an example, we may consider $d = d' = 128$. Prove that, with complexity $2^{\frac{d}{2}} + 2^{\frac{d'}{2}}$ multiplied by something small, we can find two different messages x and y of same length multiple of ℓ, such that $\mathcal{H}(x) = \mathcal{H}(y)$.

We set $n = \frac{d'}{2}$. We construct $\mathcal{A}$ to find a collision on $C(h, .)$ by using the birthday paradox with complexity $T \sim 2^{\frac{d}{2}}$.
We apply the previous question to obtain a way to construct 2^n messages of same hash by H. We hash all those $2^n = 2^{\frac{d'}{2}}$ messages with H'. Thanks to the birthday paradox, we are likely to obtain a collision on H'. This collision was already a collision for H. Hence, it is a collision for $\mathcal{H}$.
The complexity is essentially $n2^{\frac{d}{2}} + 2^{\frac{d'}{2}}$. With $d = d' = 128$, this is 2^{70} which is doable.

Q.3 Does concatenating hash functions significantly increase security, in terms of collision-resistance? (We expect a detailed answer. In particular, discuss the $d = d' = 128$ case.)

It does not increase security as expected when one of the two hash functions is of Merkle-Damgård type. We would intuitively expect that the complexity of collision search is $2^{\frac{d+d'}{2}} = 2^{\frac{d}{2}} \times 2^{\frac{d'}{2}}$, but it is $2^{\frac{d}{2}} + 2^{\frac{d'}{2}}$.
In the previous question with $d = d' = 128$, we would aim for the best attack to be of complexity 2^{128}, but we have an attack of complexity 2^{70}.

2 Discrete Logarithm in $\mathbf{Z}_{n^2}^*$

Let n be an arbitrary positive integer and $g = 1 + n$.

Q.1 In $\mathbf{Z}_{n^2}^*$, prove that g has order n.

Since $1 = (1+n)(1-n) + n^2$, $1-n$ *is the inverse of* $1+n$. *So,* g *belongs to* $\mathbf{Z}_{n^2}^*$.
We have

$$g^x = (1+n)^x = \sum_{i=0}^{x} \binom{x}{i} n^i \equiv 1 + xn \pmod{n^2}$$

Hence, $g^x \equiv 1$ *is equivalent to* $xn \bmod n^2 = 0$, *which is equivalent to* $x \bmod n = 0$.
Therefore, n *is the smallest positive power of* g *equal to* 1.

Q.2 Prove that the discrete logarithm problem is easy in $\langle g \rangle$.

We have already seen that $g^x \equiv 1 + xn$. *Hence, the discrete logarithm of* $y \equiv g^x$ *is*

$$x = \frac{(y \bmod n^2) - 1}{n}$$

which is easy to compute.

Q.3 Assume that n is prime. Given an algorithm $\mathcal{A}$ solving the discrete logarithm in $\mathbf{Z}_n^*$, construct an algorithm to solve the discrete logarithm in $\mathbf{Z}_{n^2}^*$.

Having an algorithm $\mathcal{A}$ *solving the discrete logarithm in* $\mathbf{Z}_n^*$ *means that there exists some* h *such that given input* $y \bmod n$, $\mathcal{A}(y \bmod n)$ *returns* x *such that* $y \equiv h^x \pmod{n}$ *when the output is correct. We prove below a more general result: given* $y \in \mathbf{Z}_{n^2}^*$ *and* $x \in \mathbf{Z}$, *if* $h = y^{\frac{1}{x} \bmod (n-1)} \bmod n$, *we can compute* x'' *such that* $y = h^{x''} \bmod n^2$ *if* $h^{n-1} \bmod n^2 > 1$, *or such that* $y = (gh)^{x''} \bmod n^2$ *otherwise.*
Given $y \in \mathbf{Z}_{n^2}^*$ *and* $x \in \mathbf{Z}$, *we let* $h = y^{\frac{1}{x} \bmod (n-1)} \bmod n$ *and consider* h *as an element of* $\mathbf{Z}_{n^2}^*$. *(Since* h *is coprime with* n, *it is coprime with* n^2 *too.) We have* $y \equiv h^x \pmod{n}$. *We have the discrete logarithm of* y *in basis* h *in* $\mathbf{Z}_n^*$. *We want to compute the discrete logarithm of* y *in basis* h *in* $\mathbf{Z}_{n^2}^*$, *when* h *is a generator.*
First of all, let us consider the order of h. *We have* $h^{n-1} \bmod n = 1$ *so,* $h^{n-1} \bmod n^2 = 1 + h'n$ *for some* $h' \in \mathbf{Z}_n$. *Hence, we have* $h^{n-1} \equiv g^{h'} \pmod{n^2}$. *If* $h' = 0$, *we are in the* $h^{n-1} \equiv 1 \pmod{n^2}$ *case. If* $h' > 0$, *we observe that* h' *is invertible modulo* n.
We let $z = yh^{-x} \bmod n^2$. *We have* $z \bmod n = 1$. *Hence,* $z = 1 + x'n$ *for some* $x' \in \mathbf{Z}_n$. *We have* $g^{x'} \equiv z \pmod{n^2}$. *We deduce* $y \equiv g^{x'} h^x \pmod{n^2}$.
If h' *is invertible modulo* n, *let* $h'' \in \mathbf{Z}_n$ *such that* $h'h'' \bmod n = 1$. *We have* $h^{x'h''(n-1)} \equiv g^{x'h'h''} \equiv g^{x'} \pmod{n^2}$. *Hence,* $y \equiv h^{x+x'h''(n-1)} \pmod{n^2}$ *which gives the discrete logarithm of* y *in basis* h.
If $h' = 0$, *we have* $h^{n-1} \equiv 1 \pmod{n^2}$. *We use the Chinese Remainder Theorem with the coprime* n *and* $n-1$ *to find* $x'' \in \mathbf{Z}_{n(n-1)}$ *such that* $x'' \bmod n = x'$ *and* $x'' \bmod (n-1) = x$. *We have* $g^{x''} \equiv g^{x'}$ *and* $h^{x''} \equiv h^x$. *Hence,* $(gh)^{x''} \equiv y$, *which gives the discrete logarithm of* y *in basis* gh.

3 A Post-Quantum Cryptosystem

We consider a ring R with a norm $\|\cdot\|$. For any $x \in R$, $\|x\|$ is a non-negative real number. It is such that $\|x\| = 0 \iff x = 0$, $\|x+y\| \leq \|x\| + \|y\|$, $\|x \times y\| \leq \|x\|.\|y\|$, and $\|-1\| = 1$. We further assume that there are values ℓ, τ, and a function encode from $\{0,1\}^\ell$ to R such that

$$\|\mathsf{encode}(\mathsf{pt}) - \mathsf{encode}(\mathsf{pt}')\| \leq \tau \implies \mathsf{pt} = \mathsf{pt}' \tag{1}$$

We assume that encode is easy to implement. We further assume that ring operations $+$ and $\times$ are easy to implement, as well as $\|\cdot\|$. We let $\varepsilon > 0$ be fixed. We define

- $\mathsf{Gen} \to (\mathsf{pk}, \mathsf{sk})$:
  Pick $A \in R$ at random. Pick $\mathsf{sk}, d \in R$ at random such that $\|\mathsf{sk}\| \leq \varepsilon$, $\|d\| \leq \varepsilon$. Set $B = A \times \mathsf{sk} + d$ and $\mathsf{pk} = (A, B)$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{pt}) \to \mathsf{ct}$:
  Parse $\mathsf{pk} = (A, B)$. Pick $t, e, f \in R$ at random such that $\|t\| \leq \varepsilon$, $\|e\| \leq \varepsilon$, $\|f\| \leq \varepsilon$. Set $U = t \times A + e$, $V = t \times B + f + \mathsf{encode}(\mathsf{pt})$, and $\mathsf{ct} = (U, V)$.

Q.1 Prove that for any $x \in R$, if there exists pt such that $\|x - \mathsf{encode}(\mathsf{pt})\| \leq \frac{\tau}{2}$, then pt is unique with this property.
In what follow, we define $\mathsf{decode}(x)$ as either pt such that $\|x - \mathsf{encode}(\mathsf{pt})\| \leq \frac{\tau}{2}$ if it exists, or $\perp$ otherwise. We further assume that decode is easy to implement.

Let pt *and* pt' *be such that* $\|x - \mathsf{encode}(\mathsf{pt})\| \leq \frac{\tau}{2}$ *and* $\|x - \mathsf{encode}(\mathsf{pt}')\| \leq \frac{\tau}{2}$. *We have*

$$\begin{aligned}\|\mathsf{encode}(\mathsf{pt}) - \mathsf{encode}(\mathsf{pt}')\| &\leq \|\mathsf{encode}(\mathsf{pt}) - x\| + \|x - \mathsf{encode}(\mathsf{pt}')\| \\ &\leq \|x - \mathsf{encode}(\mathsf{pt})\| + \|x - \mathsf{encode}(\mathsf{pt}')\| \\ &\leq \tau\end{aligned}$$

so $\mathsf{pt} = \mathsf{pt}'$. *We used* $\|x+y\| \leq \|x\| + \|y\|$, $\|-x\| \leq \|-1\|.\|x\| = \|x\|$, *and the property of* encode.

Q.2 Prove that if $\varepsilon \leq \frac{\tau/2}{1+\sqrt{\tau}}$, we can define an algorithm $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{pt}$ making a correct cryptosystem.

If ct *is the result of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{pt})$ *and* $(\mathsf{pk}, \mathsf{sk})$ *if the result of* Gen, *then*

$$V - U \times \mathsf{sk} = t \times d + f - e \times \mathsf{sk} + \mathsf{encode}(\mathsf{pt})$$

We have $\|t \times d + f - e \times \mathsf{sk}\| \leq 2\varepsilon^2 + \varepsilon$ *thanks to the properties of the norm. For* $\varepsilon \leq \frac{\tau/2}{1+\sqrt{\tau}}$, *we have* $2\varepsilon^2 + \varepsilon \leq \frac{\tau^2/2 + \tau/2(1+\sqrt{\tau})}{(1+\sqrt{\tau})^2}$. *This is less than* $\tau/2$ *if and only if* $\tau + (1+\sqrt{\tau}) \leq (1+\sqrt{\tau})^2$ *which is true. Hence, we have*

$$\|V - U \times \mathsf{sk} - \mathsf{encode}(\mathsf{pt})\| \leq \frac{\tau}{2}$$

Due to the previous question, we have

$$\mathsf{decode}(V - U \times \mathsf{sk}) = \mathsf{pt}$$

The algorithm $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{pt}$ *thus consists of computing* $\mathsf{decode}(V - U \times \mathsf{sk})$.

Q.3 We assume that there are $z_1, \ldots, z_n \in R$, with $n \geq \ell$, such that for any integers $\lambda_1, \ldots, \lambda_n$, we have $\|\lambda_1 z_1 + \cdots + \lambda_n z_n\| = \max_{1 \leq i \leq n} \|\lambda_i z_i\|$. We assume that there is a constant integer $K > \tau$ such that $\|K z_i\| = K$ for all i. Given $\mathsf{pt} = (\mathsf{pt}_1, \ldots, \mathsf{pt}_\ell)$ with $\mathsf{pt}_i \in \{0, 1\}$, $i = 1, \ldots, \ell$, we define $\mathsf{encode}(\mathsf{pt}) = \mathsf{pt}_1 K z_1 + \ldots + \mathsf{pt}_\ell K z_\ell$.
Prove that the hypothesis (1) on encode is satisfied.

We take pt *and* pt' *and we have*

$$\begin{aligned} \|\mathsf{encode}(\mathsf{pt}) - \mathsf{encode}(\mathsf{pt}')\| &= \left\|(\mathsf{pt}_1 - \mathsf{pt}'_1)K z_1 + \ldots + (\mathsf{pt}_\ell - \mathsf{pt}'_\ell)K z_\ell\right\| \\ &= \max_i \|(\mathsf{pt}_i - \mathsf{pt}'_i)K z_i\| \end{aligned}$$

Since $\| - 1\| = 1$, *we know that* $\| - x\| \leq \| - 1\|.\|x\| = \|x\|$ *and similarly,* $\|x\| = \|(-1) \times (-x)\| \leq \| - x\|$, *hence* $\| - x\| = \|x\|$ *for all* x. *We have* $\|K z_i\| = K$ *thus* $\| - K z_i\| = K$ *as well. Since* $\mathsf{pt}_i - \mathsf{pt}'_i \in \{-1, 0, 1\}$, *we have*

$$\|\mathsf{encode}(\mathsf{pt}) - \mathsf{encode}(\mathsf{pt}')\| = K \max_i |\mathsf{pt}_i - \mathsf{pt}'_i| = K.1_{\mathsf{pt} \neq \mathsf{pt}'}$$

Since $K > \tau$, $\|\mathsf{encode}(\mathsf{pt}) - \mathsf{encode}(\mathsf{pt}')\| \leq \tau$ *implies* $\mathsf{pt} = \mathsf{pt}'$.

4 Discrete Log -Based Signature with Domain Parameter

This exercise is about a software vulnerability in Windows 10 which was released last week. It was rated with *important* severity. It seems to apply to all Windows versions from the last 20 years.

> *The following exercise is inspired from* Digital Signature Schemes with Domain Parameters *by Vaudenay, published in the proceedings of ACISP 2004.*

We consider ECDSA, or any digital signature scheme based on the discrete logarithm problem which operate in a (multiplicatively denoted) group generated by some g element and such that $\mathsf{pk} = g^{\mathsf{sk}}$. We let Gen, Sign, and Verify be the components of the signature scheme. We assume they have the following form:

- $\mathsf{Gen}(g) \to (\mathsf{pk}, \mathsf{sk})$: pick a random sk then compute $\mathsf{pk} = g^{\mathsf{sk}}$.
- $\mathsf{Sign}(\mathsf{sk}, g, m) \to \sigma$: [for information only; the exercise can be solved without this algorithm] pick a random k, compute $r = f(g^k)$, $s = \frac{H(m)+r\cdot\mathsf{sk}}{k}$, $\sigma = (r, s)$.
- $\mathsf{Verify}(\mathsf{pk}, g, m, \sigma) \to 0/1$. [for information only; the exercise can be solved without this algorithm] make a few verifications plus $f\left(g^{\frac{H(m)}{s}}\mathsf{pk}^{\frac{r}{s}}\right) = r$.

[The rest of the specification is not useful for the exercise.] The correctness property says that for any generator g of the group and any sk and m, if $\mathsf{pk} = g^{\mathsf{sk}}$ and $\mathsf{Sign}(\mathsf{sk}, g, m) \to \sigma$, then $\mathsf{Verify}(\mathsf{pk}, g, m, \sigma) \to 1$.

In CryptoAPI (Crypt32.dll) in Windows 10, remote code validation needs a chain of certificates $\mathsf{chain}(C_1, \ldots, C_n)$ to validate a software s. We model a certificate C_i by $C_i = (m_i, \sigma_i)$, $i = 1, \ldots, n$. We say that chain is valid for s if we have the following properties:

- $m_1 = s$;
- for $i = 2, \ldots, n$, we parse $m_i = (\mathsf{info}_i, g_i, \mathsf{pk}_i)$ where g_i is a generator of the group $\langle g \rangle$;
- for $i = 1, \ldots, n-1$, σ_i is a valid signature of m_i when verified with g_{i+1} and pk_{i+1};
- pk_n is equal to the hard-coded root public key in CryptoAPI (it is the root public key).

Q.1 What is weird/unusual in the definition of chain?

> *The generator g_i seems to be changeable out from the common domain parameters. The last pk_n is checked to be the root pubic key, but there is no verifiation about g_n.*

Q.2 We consider an adversary who knows g and the root public key pk. Given an arbitrary software s, prove that the adversary can easily construct a valid chain with $n = 2$ for s.

> *We set $n = 2$, $g_2 = \mathsf{pk}_2 = \mathsf{pk}$, and we use $\mathsf{sk} = 2$ to compute $\sigma_1 \leftarrow \mathsf{Sign}(\mathsf{sk}_2, g_2, s)$ and. $\sigma_2 \leftarrow \mathsf{Sign}(\mathsf{sk}_2, g_2, m_2)$.*
> *We do have $\mathsf{pk}_2 = g_2^{\mathsf{sk}_2}$. Hence, due to the correctness property, σ_1 and σ_2 are valid signatures.*
> *Since we have $\mathsf{pk}_2 = \mathsf{pk}$, the chain is valid.*