Cryptography and Security — Final Exam

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

1 On Combining Two Hash Functions by Concatenation

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 $n \leq d$, 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 $i = 1,\ldots,n$.

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 $\mathcal{H}(x) = \mathcal{H}(y)$.

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.)

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.

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

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 $\mathbf{Z}^*_{n^2}$.

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 $\|x\| = 0 \Longleftrightarrow x = 0$, $\|x+y\| \leq \|x\| + \|y\|$, $\|x \times y\| \leq \|x\|.\|y\|$, and $\|-1\| = 1$. We further assume that there are values ℓ, τ, and a function encode from $\{0,1\}^\ell$ to R such that

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

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

- $\mathsf{Gen} \rightarrow (\mathsf{pk}, \mathsf{sk})$:
  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 $\mathsf{pk} = (A, B)$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{pt}) \rightarrow \mathsf{ct}$:
  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.

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

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.

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.

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:

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

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?

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.