Cryptology I (spring 2019)	Dominique Unruh

Exercise Sheet 5

Out: 2019-05-15	Due: 2019-05-29

Problem 1: One-way functions

Which of the following are one-way functions? For each function that is a one-way function, explain why (no formal proof required). For each function that is not a one-way function, write an attack in Python. (Code for all the functions, including test code is provided in `owf.py`. You only need to fill in the functions `adv`i for attacking function f_i.)

Hint: Out of the four functions, one is a OWF, the other three are not.

Note: Formally, of course, the question would have to be "is the function a (τ, ε)-OWF?" and τ and ε would have to be specified. I am omitting specific τ and ε, instead, you are to interpret "is an OWF" as "there is no attack in reasonable time and with resonable success probability".

Note: You may assume that the RSA assumption holds. And that E_{AES} is a PRF. (For reasonable τ, ε, again.)

Note: Remember that to break a one-way function, it is sufficient to find some preimage, not necessarily the "true" one that was fed into the one-way function.

(a) $f_1(x) := 0$ for all $x \in \{0,1\}^\eta$.

(b) $f(N, e, x) := (N, e, x^e \bmod N)$ where the domain of f is the set of all (N, e, x) where N is an RSA modulus, e is relatively prime to N, and $x \in \{0, \dots, N-1\}$.

(c) $f(N, e, x) := x^e \bmod N$ where the domain of f is the set of all (N, e, x) where N is an RSA modulus, e is relatively prime to N, and $x \in \{0, \dots, N-1\}$.

(d) $f(k, x) := E_{AES}(k, x)$.

Problem 2: Tree-based signatures

This problem refers to the tree-based construction of signature schemes from one-time signatures from Construction 4 in the lecture notes. You may assume that Lamport's signature scheme (Construction 2 in the lecture notes) is used as the underlying one-time signature scheme. (Where all messages are first hashed with a hash function H before signing with Lamport's scheme in order to fit in the message space.)

(a) As a warmup, let's attack Lamport's scheme. Assume Alice is sending random messages m, together with signatures $\sigma := \mathit{Sign}_{\mathit{Lamport}}(\mathit{sk}, m)$. Alice uses, though she should know better, the same sk for all messages m. (I am not specifying how many messages Alice sends and signs, you can assume that there are enough of them for your attack.)

The adversary gets all messages m and all corresponding signatures σ.

Describe how to efficiently compute sk from the received m, σ.

Note: Be explicit: describe all the actions and computations the adversary has to perform. (E.g., give the adversary in pseudocode.) It is not sufficient to say something like: "since two signatures are produced using the same key with a one-time signature scheme, the adversary can break the scheme". Remember that the underlying scheme is Lamport's one-time signature scheme.

(b) Assume someone has implemented the signature scheme incorrectly as follows: Instead of using randomness from the pseudorandom function F for the key-generation algorithm, it runs the key-generation normally (i.e., as probabilistic algorithms, with fresh randomness each time it is invoked).

Explain how to break the signature scheme. More precisely, show how to sign an arbitrary message m by performing only signature queries for messages $m' \neq m$.

Note: Be explicit: describe all the actions and computations the adversary has to perform. (E.g., give the adversary in pseudocode.) It is not sufficient to say something like: "since two signatures are produced using the same key with a one-time signature scheme, the adversary can break the scheme". Remember that the underlying scheme is Lamport's one-time signature scheme.

(c) **Bonus problem:** Lamport's signature scheme has public keys consisting of $2n$ n-bit blocks (assuming that the one-way function f has domain and range $\{0,1\}^n$). But it signs only messages consisting of a single n-bit block. In the tree-based construction, we need to sign two Lamport public keys, i.e., $4n$ n-bit blocks. Normally we solve this by converting Lamport's scheme into a one-time signature scheme for long messages by hashing the messages to be signed.

Here we explore a different possibility. Instead of hashing the $4n \times n$ bits, we XOR the blocks together. That is, from Lamport's scheme $(\mathit{KG}_{\mathit{Lamport}}, \mathit{Sign}_{\mathit{Lamport}}, \mathit{Verify}_{\mathit{Lamport}})$ we construct a one-time signature scheme $(\mathit{KG}_1, \mathit{Sign}_1, \mathit{Verify}_1)$ for $4n \times n$-bit messages as follows:

$\mathit{KG}_1 := \mathit{KG}_{\mathit{Lamport}}$. $\mathit{Sign}_1(\mathit{sk}, m_1 \| \dots \| m_{4n}) := \mathit{Sign}_{\mathit{Lamport}}(\mathit{sk}, \bigoplus_{i=1}^{4n} m_i)$ for $m_1, \dots, m_{4n} \in \{0,1\}^n$. $\mathit{Verify}_1(\mathit{pk}, m_1 \dots m_{4n}, \sigma) := \mathit{Verify}_{\mathit{Lamport}}(\mathit{pk}, \bigoplus_{i=1}^{4n} m_i, \sigma)$.

Now we can construct the tree-based signature scheme $(\mathit{KG}_{\mathit{tree}}, \mathit{Sign}_{\mathit{tree}}, \mathit{Verify}_{\mathit{tree}})$ from $(\mathit{KG}_1, \mathit{Sign}_1, \mathit{Verify}_1)$ without needing a hash function (as in Construction 4 in the lecture notes).

Your task: Break the resulting $(KG_{tree}, Sign_{tree}, Verify_{tree})$.

Note: It is not sufficient to just show that $(KG_1, Sign_1, Verify_1)$ is insecure. You have to break $(KG_{tree}, Sign_{tree}, Verify_{tree})$. All the other comments from the note of (b) also apply.