Cryptology I (spring 2018) Dominique Unruh

Exercise Sheet 10

Out: 2018-05-03 Due: 2018-05-11

Problem 1: Merkle-Damgård and the ROM

In the lecture, I explained the random oracle heuristic which suggests to model a hash function as a random oracle. It should be added that a (preferable) refinement of this heuristic is to model the compression function itself as a random oracle, and to model the hash function as some function constructed based on that compression function (using, e.g., Merkle-Damgård). The reason behind this is that constructions like Merkle-Damgård do not produce functions that behave like random functions (even if the underlying compression function is a random function).

Give an example why a hash function H constructed using the Merkle-Damgård construction should not be modeled as a random oracle. More precisely, find a cryptographic scheme which is secure when H is a random oracle (no security proof needed), but which is insecure when H is a Merkle-Damgård construction (even if the compression function is a random oracle).

Hint: Consider the construction of MACs from hash functions that is insecure when the hash function is constructed with Merkle-Damgård. This problem does not need a complicated construction!

Problem 2: Security proof in the ROM [Bonus problem]

This is a bonus problem.

Fix a hash function $H : \{0,1\}^* \to \{0,1\}^n$. We define the following block cipher with message and key space $\{0,1\}^n$:

- **Encryption** E**:** To encrypt $m \in \{0,1\}^n$ under key k, choose a random $r \in \{0,1\}^n$ and return the ciphertext $c := (r, m \oplus H(k\|r))$.
- **Decryption** D**:** To decrypt $c = (r, c')$ with key k, compute and return $m := H(k\|r) \oplus c'$.

Below is a proof that this encryption scheme is $(\tau, q_E, q_H, \varepsilon)$-IND-CPA[1] secure in the random oracle model. Fill in the gaps. (The length of the gaps is unrelated to the length of the text to be inserted.)

Proof. In the first game, we just restate the game from the IND-CPA security definition (in the random oracle model).

[1] q_E is the number of encryption oracle queries, and q_H the number of random oracle H queries performed by A.

Game 1. [1] ◇

To show that the encryption scheme is $(\tau, q_E, q_H, \varepsilon)$-IND-CPA secure, we need to show that

$$|\Pr[b = b' : Game\ 1] - \tfrac{1}{2}| \leq \varepsilon \tag{1}$$

As a first step, we replace the random oracle.

Game 2. Like Game 1, except that we define the random oracle H differently: [2] ◇

We have $\Pr[b = b' : Game\ 1] = \Pr[b = b' : Game\ 2]$.

One can see that the adversary cannot guess the key k (where k is the key used for encryption in Game 2), more precisely, the following happens with probability $\leq q_H 2^n$: "The adversary invokes $H(x)$ with $x = k\|r'$ for some r'." (We omit the proof of this fact.)

Let r_0 denote the value r that is chosen during the execution of $c \leftarrow E^H(k, m_b)$ in Game 2. Consider the following event: "Besides the query $H(k\|r_0)$ performed by $c \leftarrow E^H(k, m_b)$, there is another query $H(x)$ with $x = k\|r_0$ (performed by the adversary or by the oracle $E^H(k, \cdot)$." This event occurs with probability $q_H 2^{-n} + q_E 2^{-n}$. Namely, the adversary make such $H(x)$ queries with probability $\leq q_H 2^{-n}$ because [3], and each invocation of the oracle $E^H(k, m_b)$ makes such an $H(x)$ query with probability $\leq 2^{-n}$ because [4].

Thus, the response of the $H(k\|r_0)$-query performed by $c \leftarrow E^H(k, m_b)$ is a random value that is used nowhere else (except with probability $\leq (q_H + q_E)2^{-n}$). Thus, we can replace that value by some fresh random value.

Game 3. Like Game 2, except that we replace $c \leftarrow E^H(k, m_b)$ by $r_0 \xleftarrow{\$} \{0,1\}^n$, $h^* \xleftarrow{\$} \{0,1\}^n$, $c \leftarrow (r_0, m_b \oplus h^*)$. ◇

We have that

$$|\Pr[b = b' : Game\ 2] - \Pr[b = b' : Game\ 3]| \leq (q_H + q_E)2^{-n} = \varepsilon.$$

To get rid of m_b in Game 3, we use the fact that h^* is chosen uniformly at random and XORed on m_b. That is, we can replace $m_b \oplus h^*$ by [5].

Game 4. Like Game 3, except that we replace $c \leftarrow (r_0, m_b \oplus h^*)$ by [6]. ◇

We have that $\Pr[b = b' : Game\ 4] = \Pr[b = b' : Game\ 3]$. Notice that b is not used in Game 4, thus we have that $\Pr[b = b' : Game\ 4] =$ [7].

Combining the equations we have gathered, (1) follows. □

Problem 3: Yao's Garbled Circuits

(a) One application of secure function evaluation is the so-called "dating problem". Two parties A and B are wondering whether they should date, but none of them wishes to admit their interest unless they know that the other side is interested, too. The solution is to perform a two-party computation on their inputs a and b (where a and b are a bit corresponding to whether A or B wishes to date) that returns

$f(a, b) := a \wedge b$. (We ignore the fact that this is silly: by suggesting to run this SFE, one already expresses interest. But we could consider a case where some app is doing this automatically with all potential matches – a privacy preserving dating app.)

A and B want to use Yao's Garbled Circuits for this. (We ignore the fact that that protocol only has security against passive adversaries.) That is, A will have to pick some circuit C, and B some input x for that circuit. What should C and x be in this concrete case (i.e., how to convert a and b into C and x) so that B learns $f(a, b)$?

(b) Implement part of Yao's protocol. That is, implement a function `make_gate` that garbles a single gate. (Given four input keys, and four messages.) And a function `eval_gate` that recovers the message m_{ij} given the corresponding keys.

Use the template in `yao-gate.py`.