

Exercise Sheet #4

Advanced Cryptography 2022

Exercise 1 PRF Programming (Final 2013)

A function $\delta(s)$ is called negligible and we write $\delta(s) = \text{negl}(s)$ if for any $c > 0$, we have $|\delta(s)| = o(s^{-c})$ as s goes to $+\infty$.

Let s be a security parameter. For simplicity of notations, we do not write s as an input of games and algorithms but it *is* a systematic input.

A family $(f_k)_{k \in \{0,1\}^s}$ of functions f_k from $\{0,1\}^s$ to $\{0,1\}^s$ is called a PRF (Pseudo Random Function) if for any probabilistic polynomial-time oracle algorithm \mathcal{A} , we have that

$$|\Pr[\mathcal{A}^{f_K(\cdot)} = 1] - \Pr[\mathcal{A}^{f^*(\cdot)} = 1]| = \text{negl}(s)$$

where $K \in \{0,1\}^s$ is uniformly distributed, f^* is a uniformly distributed function from $\{0,1\}^s$ to $\{0,1\}^s$, $f_K(\cdot)$ denotes the oracle returning $f_K(x)$ upon query x , and $f^*(\cdot)$ denotes the oracle returning $f^*(x)$ upon query x .

Given a PRF $(f_k)_{k \in \{0,1\}^s}$, we construct a family $(g_k)_{k \in \{0,1\}^s}$ by $g_k(x) = f_k(x)$ if $x \neq k$ and $g_k(k) = k$. The goal of the exercise is to prove that $(g_k)_{k \in \{0,1\}^s}$ is a PRF.

We define the PRF game played by \mathcal{A} for g , f , and f^* by

Game Γ^g	Game Γ^f	Game Γ^*
1: pick $K \in \{0,1\}^s$	1: pick $K \in \{0,1\}^s$	1: pick $f^* : \{0,1\}^s \rightarrow \{0,1\}^s$
2: run $b = \mathcal{A}^{g_K(\cdot)}$	2: run $b = \mathcal{A}^{f_K(\cdot)}$	2: run $b = \mathcal{A}^{f^*(\cdot)}$
3: give b as output	3: give b as output	3: give b as output

For each integer i , we define an algorithm \mathcal{A}_i (called a *hybrid*) which mostly simulates \mathcal{A} until it makes the i th query. More concretely, \mathcal{A}_i simulates every step and queries of \mathcal{A} while counting the number of queries. When the counter reaches the value i , \mathcal{A}_i does not make this query k but it stops and the queried value k is returned as the output of \mathcal{A}_i . If \mathcal{A} stops before making i queries, \mathcal{A}_i stops as well, with a special output \perp . We define the following games:

Game Γ_i^f	Game Γ_i^*
1: pick $K \in \{0,1\}^s$	1: pick $f^* : \{0,1\}^s \rightarrow \{0,1\}^s$
2: run $k = \mathcal{A}_i^{f_K(\cdot)}$	2: run $k = \mathcal{A}_i^{f^*(\cdot)}$
3: if $k = \perp$, stop and output 0	3: if $k = \perp$, stop and output 0
4: pick $x \in \{0,1\}^s$	4: pick $x \in \{0,1\}^s$
5: if $f_k(x) = f_K(x)$, stop and output 1	5: if $f_k(x) = f^*(x)$, stop and output 1
6: output 0	6: output 0

Let $F(\Gamma)$ be the event that any of the queries by \mathcal{A} in game Γ equals K . We assume that the number of queries by \mathcal{A} is bounded by some polynomial $P(s)$.

1. Show that $|\Pr[\Gamma^f \rightarrow 1] - \Pr[\Gamma^* \rightarrow 1]| = \text{negl}(s)$.
2. Show that $\Pr[\Gamma^g \rightarrow 1 | \neg F(\Gamma^g)] = \Pr[\Gamma^f \rightarrow 1 | \neg F(\Gamma^f)]$ and $\Pr[\neg F(\Gamma^g)] = \Pr[\neg F(\Gamma^f)]$.
3. Deduce $|\Pr[\Gamma^g \rightarrow 1] - \Pr[\Gamma^f \rightarrow 1]| \leq \Pr[F(\Gamma^f)]$.
4. Show that $\Pr[F(\Gamma^f)] \leq \sum_{i=1}^{P(s)} \Pr[\Gamma_i^f \rightarrow 1]$.
5. Show that $|\Pr[\Gamma_i^f \rightarrow 1] - \Pr[\Gamma_i^* \rightarrow 1]| = \text{negl}(s)$ for all $i \leq P(s)$.
6. Show that $\Pr[\Gamma_i^* \rightarrow 1] = \text{negl}(s)$ for all $i \leq P(s)$.
7. Deduce $|\Pr[\Gamma^g \rightarrow 1] - \Pr[\Gamma^* \rightarrow 1]| = \text{negl}(s)$.

Exercise 2 A Weird Signcryption (Midterm 2019)

We consider the plain RSA cryptosystem (RSA.Gen, RSA.Enc, RSA.Dec) and a digital signature scheme (DS.Gen, DS.Sign, DS.Ver). We construct a signcryption scheme as follows:

SC.Gen

- 1: $\text{RSA.Gen} \rightarrow (\text{ek}, \text{dk})$
- 2: $\text{DS.Gen} \rightarrow (\text{sk}, \text{vk})$
- 3: $\text{pubk} \leftarrow (\text{ek}, \text{vk})$
- 4: $\text{privk} \leftarrow (\text{dk}, \text{sk})$
- 5: **return** $(\text{pubk}, \text{privk})$

SC.Send($\text{pubk}_B, \text{privk}_A, \text{pt}$) //user A sends a message to B

- 1: $\text{parse}(\text{ek}_B, \text{vk}_B) \leftarrow \text{pubk}_B$
- 2: $\text{parse}(\text{dk}_A, \text{sk}_A) \leftarrow \text{privk}_A$
- 3: $\text{ct} \leftarrow \text{RSA.Enc}(\text{ek}_B, \text{pt})$
- 4: $\sigma \leftarrow \text{DS.Sign}(\text{sk}_A, \text{ct})$
- 5: **return** (ct, σ)

so that A can send (ct, σ) to B . Once B obtains pt , he can show $\text{proof} = (\text{vk}_A, \text{ek}_B, \text{ct}, \sigma, \text{pt})$ as a proof that A sent pt . We call this property *non-repudiation*.

1. Describe the algorithm using $(\text{pubk}_A, \text{privk}_B)$ to receive (ct, σ) and compute pt , as well as the algorithm to verify the proof.
2. Given $(\text{vk}_A, \text{ct}, \sigma)$ such that $\text{DS.Ver}(\text{vk}_A, \text{ct}, \sigma)$ is true and given an arbitrary pt , prove that we can easily find ek such that $(\text{vk}_A, \text{ek}, \text{ct}, \sigma, \text{pt})$ is a valid proof.
3. Propose a fix to this problem so that we have non-repudiation.