Homework 6
COMP543 Fall 2020 - Modern Cryptography

Erhan Tezcan 0070881

11.11.2020

---

1. Quesitons

Q1: Where can/should we use hybrid proofs? How many hybrids should there be, and why?

A1: We use hybrid proofs when a certain primitive is applied iteratively. There are 3 rules regarding hybrids:

(1) There will be a first hybrid H^{first} and last hybrid H^{last}.
(2) Given two neighbors H^i and H^{i+1}, if we can distinguish them then the system will be broken, e.g. H^{first} and H^{last} will be distinguishable.
(3) There can only be polynomially many hybrids.

Q2: Define negligible, polynomial, noticeable, non-negligible, exponential functions and probabilities fully formally. Use several equivalent definitions. Try to employ Big-Oh notation as well. Moreover, answer the following:

a. $neg(k) \times poly(k) = ?$
b. $poly(k)/exp(k) = ?$
c. $poly(k)/superpoly(k) = ?$
d. $noticeable(k)/poly(k) = ?$

A2: Let us include the definition of Big-O here, $f(n) = \mathcal{O}(g(n))$ means that there exist positive integers c and n' such that $\forall n > n'$ it holds that $f(n) \leq c \times g(n)$. $poly(k)$ is just a polynomial.

- `poly`: $poly(k)$ is a polynomial that is in the form: $poly(k) = a_k x^k + a_{k-1}x^{k-1} + \ldots + a_1 x + a_0$ where $k \in \mathbb{Z}^+$ and $\forall i \in \{0, \ldots, k\}, a_i \in \mathbb{R}$. (shown as $\mathcal{O}(x^k)$).
- `exp`: A function $f(n)$ is an exponential function such that $f(n) = ab^n$ where $b \in \mathbb{R}$ and $b > 1$. $exp(n)$ is a special case of this, as $exp(n) = e^n$.
- `negl`: $negl(k)$ is when **for all** polynomials p there is an N such that for all integers $n > N$ it holds that $f(n) < 1/p(n)$. Equivalently, $f(n)$ is negligible if and only if $f(n) \in \mathcal{O}(1/p(n))$ for all positive polynomials p. Another equivalent definition is that $f(n)$ is negligible if and only if $p(n)f(n)$ converges to 0 for all positive polynomials p.

- `noticable`: *noticable*(k) is when **there exists** a polynomial p and an N such that for all integers $n > N$ it holds that $f(n) \geq 1/p(n)$. Equivalently, $f(n)$ is noticable if and only if $f(n) \in -(1/p(n))$ for some positive polynomial p. Another equivalent definition is that $f(n)$ is noticable if and only if $p(n)f(n)$ goes to ∞ for some positive polynomial p. $1/p(n)$ is bounded by $\mathcal{O}(f(n))$
- `non-negl`: Non-negligible is when if **there exists** a polynomial p and **no** N **exists** such that for all integers $n > N$, $f(n) < 1/p(n)$. Any function that is not negligible is non-negligible. Note that `noticable` and `non-negligible` are not necessarily the same!

A superpolynomial is a function that is not upperbounded by any polynomial.

a. $neg(k) \times poly(k) = neg(k)$
b. $poly(k)/exp(k) = neg(k)$ as the reciprocal of $exp(k)$ is negligible.
c. $poly(k)/superpoly(k) = neg(k)$ because *superpoly* grows larger than *poly*.
d. $noticeable(k)/poly(k) = noticeable(k)$.

Q3: Formally define computational indistinguishability, and discuss where it can be applied. Especially, considering what you have seen until this point, where have you already applied computational indistinguishability definition (possibly without knowing it)?

A3: Two probability ensembles $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$ are *computationally indistinguishable* if for every PPT distinguisher $\mathcal{D}$ there exists a negligible function `negl` s.t.

$$\left| \underset{x\leftarrow X_n}{Pr}[\mathcal{D}(1^n, x) = 1] - \underset{y\leftarrow Y_n}{Pr}[\mathcal{D}(1^n, y) = 1] \right| \leq \texttt{negl}(n)$$

For example, pseudorandomness is just a special case of this, where one distribution ensemble is the output of a PRG and the other is actually a random distribution. Most of the definitions and assumptions we made so far is a special case or varient of computational indistinguishability.

Q4: Why do we talk about "families"[1] of functions?

A4: It is because we are using just a “member” of that family during our experiments. We need many functions that show the properties

[1]KL Book 2nd ed. p. 244

defined by the family, such that we can randomly choose a function in there and use it.

Think of the first parameter as an index which chooses a function from that family. This is required so that an outsider does not know which function we are using.

Q5: Formally define a one way function.

A5: A function f is a one way function if:

(1) **(Easy to Compute)**: $\forall x$, there exists a polynomial time algorithm M_f computing f, i.e. $M_f(x) = f(x)$

(2) **(Hard to Invert)**: $\forall$`PPT` algorithms $\mathcal{A}$, it should be infeasible to calculate the inverse of f. Let $y = f(x)$. This is also shown as:

$$\underset{x \leftarrow \{0,1\}^n; y=f(x)}{Pr}[\mathcal{A}(1^n, y) \in f^{-1}(y)] \leq \texttt{negl}(n)$$

Q6: Formally define a one way permutation.

A6: A function f is a one way permutation if f is a one way length-preserving bijective function.

Q7: Prove that if f is a one-way function, then the function g defined by $g(x_1, x_2) = (f(x_1), x_2)$ where $|x_1| = |x_2|$ is also a one-way function. Observe that g reveals half of its input, but is nevertheless one-way.

A7: We had two conditions for a one-way function, **Easy-to-Compute** and **Hard-to-Invert**, immediately we can see that g is **Easy-to-Compute**. So what we have to do is show for all PPT algorithms $\mathcal{A}$ there exists a negligible function `negl` s.t.

$$Pr[\texttt{Invert}_{\mathcal{A},g}(n) = 1] \leq \texttt{negl}(n)$$

Equivalently,

$$\underset{x_1 \leftarrow \{0,1\}^n, x_2 \leftarrow \{0,1\}^n}{Pr}[\mathcal{A}(1^n, g(x_1, x_2)) \in g^{-1}(g(x_1, x_2))] \leq \texttt{negl}(n)$$

We can think of g as a function that gives half of the input to f, and acts like an identity for the other half. Inverting identity function is easy, however inverting f is infeasible, therefore inverting g should be infeasible, thus g is a one-way function.

Q8: Prove that if G is a pseudorandom generator with expansion factor $l(n) = n+1$ then G' with the construction given below is a PRG with the expansion factor of $l'(n) = p(n)$ where $p(n)$ is polynomial in

n. On input $s \in \{0,1\}^n$, algorithm G' does the following (see figure 6.1 in ED1 or figure 7.1 in ED2 of the textbook):

Algorithm 1 Algorithm of G'

```
1: Set $t_0 := s$
2: for $i = 1, \dots, p(n)$ do
3:     Let $s_{i-1}$ be the first $n$ bits of $t_{i-1}$, and let $\sigma_{i-1}$ denote the remaining $i-1$ bits. (When $i = 1$, $s_0 = t_0$ and $\sigma_0$ is the empty string.)
4:     Set $t_i := G(s_{i-1})||\sigma_{i-1}$
5: end for
6: Output $t_{p(n)}$
```

A8: The proof will start by assuming that if there exists a PPT algorithm $\mathcal{A}$ that breaks G', then we can construct an algorithm $\mathcal{B}$ that breaks G. We will define a game now.

Algorithm 2 `chal` at the beginning.

```
1: Chal chooses a bit $b \leftarrow \{R, PR\}$
2: if b = R then
3:     $r \leftarrow \{0,1\}^{n+1}$
4: else
5:     $x \leftarrow \{0,1\}^n$; $r \leftarrow G(x)$
6: end if
7: Chal gives $1^n$ and $r$ to $\mathcal{B}$
```

Define some hybrids as follows:

- $H^i : s \leftarrow \{0,1\}^{n+i}$ and then continue from level i.
- $H^{first} = H^0 : s \leftarrow \{0,1\}^n; r \leftarrow G'(s); r \in \{0,1\}^{n+p(n)}$, this is like a PR experiment.
- $H^{last} = H^{p(n)} : s \leftarrow \{0,1\}^{p(n)}, r = s$, this is like a R experiment.

If $\mathcal{A}$ wins with $\epsilon(n)$ probability then $\mathcal{B}$ wins with at least $\epsilon(n)/p(n)$ probability, which means at least one neighbor of hybrids were distinguishable. However, we know that all neighbors are indistinguishable as the theorem states, therefore $\mathcal{B}$ wins only with negligible probability.

Algorithm 3 Code of $\mathcal{B}$

```
1: $\mathcal{B}$ takes $1^n$ and $r$ from Chal
2: $i \leftarrow \{1, \ldots, p(n)\}$
3: $\sigma_{j-1} \leftarrow \{0,1\}^{j-1}$
4: $t_{j-1} := r||\sigma_{j-1}$
5: for $i = j, \ldots, p(n)$ do
6:     Let $s_{i-1}$ be the first $n$ bits of $t_{i-1}$, and let $\sigma_{i-1}$ denote the remaining $i-1$ bits. (When $i = 1$, $s_0 = t_0$ and $\sigma_0$ is the empty string.)
7:     Set $t_i := G(s_{i-1})||\sigma_{i-1}$
8: end for
9: $w = t_{p(n)}$
10: $\mathcal{B}$ gives $1^n$ and $t_{p(n)}$ to $\mathcal{A}$
11: $\mathcal{A}$ returns a response $b' \in \{R, PR\}$, which we then forward to Chal
```

Algorithm 4 chal at the end.

```
1: if b = b' then
2:     output 1
3: else
4:     output 0
5: end if
```