CS6111: Foundations of Cryptography

Assignment 4
Saurab Sirpurkar
CS16B039

Instructions

- Nov 15.
- Hints added on Nov 14.

References

- Introduction to Cryptography - Delfs and Knebl
- A Graduate Course in Applied Cryptography - Boneh and Shoup (link)
- Introduction to Modern Cryptography - Katz and Lindell

1 PRFs and PRGs

1. (6 points) Let F_s be a (length-preserving) pseudorandom permutation family.
   (a) (2 points) Show that $f(x,y) = F_x(y)$ is not a one-way function.
   (b) (2 points) Show that $f(y) = F_{0^n}(y)$ is not one-way. (where $n = |y|$)
   (c) (2 points) Prove that $f(x) = F_x(0^n)$ is one-way. (where $n = |x|$)

   Hint: You can give proofs by counter-example. Just like OWFs with cases, PRFs are allowed to be "easy" for some keys. Proving it is a one-way function must be done by reduction to the PRF security game.

Solution:

(a) We say that a keyed permutation is efficient if there is a polynomial time algorithm computing $F_k(x)$ given k and x, **as well as a polynomial-time algorithm computing $F_k^{-1}(x)$ given k and x**. (Katz Lindell, pg.94). Also, every PR permutation is a keyed permutation satisfying more discriminating conditions. From here, we can infer that PR permutations are poly-time invertible. (Even more simply put, the very fact that F is a **permutation** means it has a poly-computer as well as a poly-inverter because the definiton demands F be one-one for all keys)
From the above specification, we see why $f(x,y) : \{0,1\}^n * \{0,1\}^n \to \{0,1\}^n$ is not a OWF. For an output

'y_o', simply choose a random key 'k' in $\{0,1\}^n$ and compute $F_k^{-1}(y_o)$ - which can be done by a deterministic poly-time algorithm from the condition mentioned above. Therefore, since $\forall k, f(k, F_k^{-1}(y_o)) = y_o$, f is not a OWF.

(b) Same as above. Instead of using a random key, the key k = 0^n can be used.

(c) In case $f(x) = F_x(0^n)$ is not a OWF, we will form a distinguisher for the PR permutation family F_s. This reduction would mean that f must be a OWF.

Reduction:

$\mathbb{A}$: A PPT non-negl inverter of f. (assume that it exists)

$\mathbb{D}$: A distinguisher of F_s family (to be constructed)

Algorithm for $\mathbb{D}$:

```
Query the oracle for the ecnryption of $0^n (= y_o)$
Call the subroutine $A(y_o)(= x_o)$
Compute $F_{x_o}(0^n)(= z)$
Return 1 if $z = y_o$
Return 0 otherwise
```

If the oracle was using a PRF, then the probability that $\mathbb{D}$ returns 1 is : nonnegl Adv of $\mathbb{A}$

If the oracle was using a random funtion instead, the probability that it returned 1 is : negl

(Because $\mathbb{A}$ cannot find a pattern in the completely random response x_o from the oracle) Therefore, we have

$$|Pr[\mathbb{D}^{F_k(.)} = 1] - Pr[\mathbb{D}^{f(.)} = 1]| > nonnegl - negl > \text{negl(n)}$$

Hence, a contradiction to the fact that F_s is a PR permutation family. Therefore, $\mathbb{A}$ does not exist and hence, $f(x) = F_x(0^n)$ is a OWF.

2. (3 points) Let f be a length-preserving one-way function, and let hc be a hardcore predicate for f. Define G as $G(x) = f(x)\|hc(x)$. Is G necessarily a pseudorandom generator?

Solution: No, it is true only when f is a one-way 'permutation'. If f is not a OWP, but is a OWF, we can show a counterexample where $G(x)$ need not be a pseudorandom generator.

Consider $h(x_1x_2x_3.....x_n) = 0\|f(x_2x_3......x_n)$. From Assignment-2 Q2.6, we know that such a construction is also a OWF (except that lower order bits are instead of higher order and zero is padded). Now, let us check whether G, when constructed using such a 'h' is still a PRG. Consider:

$$G(x) = h(x)\|hc_h(x)$$

A distinguisher $\mathbb{D}$ that returns 1 if first bit of sampled string is zero and 0 otherwise follows:

$$|Pr[\mathbb{D}(G) = 1] - Pr[\mathbb{D}(U_n) = 1]| = 1 - \frac{1}{2} = \frac{1}{2}$$

This is clearly non-negl and therefore G is not a PRG.

3. (3 points) Given a pseudorandom function $F_s : \{0,1\}^{k+\log k} \to \{0,1\}$, construct a new pseudorandom function $F'_s : \{0,1\}^k \to \{0,1\}^k$ and prove that it is secure.

Solution: We have a PRF $f : \{0,1\}^{(n+logn)} \to 0,1$. From this, we have to construct $g : \{0,1\}^n \to \{0,1\}^n$.
The main idea is to use the function 'f', 'n' number of times, to generate random bits of g's output. To handle the mismatch in number of bits in the input, we use the form $y_i = F_s(x \circ [i]_2)$ where $[]_2$ denotes the binary representation of a number with padding to make it '$logn$' long.

Consider the function:

$$F'_s(x) = f(x\|[1]_2)\|f(x\|[2]_2)\|....f(x\|[n]_2)$$

Claim: The so constructed F'_s is a PRF.
Proof:
If there is a non-negl distinguisher (say D) for F'_s we will show that there will be one for F_s as well (call it B).
B can use D as a subroutine and work in this way:
B acts as a mediator between D and the oracle of F'_s (which we will see is something that B emulates using the oracle of F_s).
If D asks for a query $\mathbb{O}(x)$, B can instead give

$$\mathbb{O}(x) = \mathbb{O}^{Fs}(x\|[1]_2)\|\mathbb{O}^{Fs}(x\|[2]_2)\|.......\mathbb{O}^{Fs}(x\|[n]_2)$$

to D. In the cases that B faces a random oracle, then $\mathbb{O}(x)$ is obviously random. In case B faces a PRF oracle then D too would face an replicated version of PRF oracle (of F'_s).
If B just returns whatever D returns (0 or 1):

$$Pr(\frac{\text{B returns 1}}{\text{Oracle is Random}}) = Pr(\frac{\text{D returns 1}}{\text{Oracle is Random}})$$

$$Pr(\frac{\text{B returns 1}}{\text{Oracle is that of a PRF}}) = Pr(\frac{\text{D returns 1}}{\text{Oracle is that of a PRF}})$$

But since D is non-negl, we have:

$$Pr(\frac{\text{D returns 1}}{\text{Oracle is random}}) - Pr(\frac{\text{D returns 1}}{\text{Oracle is that of a PRF}}) > \epsilon(n)$$

From the equations above, we see that

$$Pr(\frac{\text{B returns 1}}{\text{Oracle is random}}) - Pr(\frac{\text{B returns 1}}{\text{Oracle is that of a PRF}}) > \epsilon(n)(i.e.non-negl)$$

(As we have shown the Oracle randomness equivalence above)A contradiction: F_s is inverted with non-negl probability. Therefore F'_s is a PRF.

4. (4 points) Let $\tilde{s}$ denote the bitwise negation of $s \in \{0,1\}^l$.

   (a) (2 points) Is $G_a(s) = G(s\|\tilde{s})$ a PRG? (where $G : \{0,1\}^{2l} \to \{0,1\}^{2l+1}$ is a PRG)

(b) (2 points) Is $G_b(s) = G(s)||G(\bar{s})$ a PRG? (where $G : \{0,1\}^l \to \{0,1\}^{l+1}$ is a PRG)

Hint: To prove it is not a PRG you should be able to predict at least one bit of the PRF (violating the NBU property). The bit could be a repeat of previous bits or even a constant value. To prove that it is a PRG you should reduce it to the PRG security game.

Solution: As per the hint given, we need to use the fact that next-bits are unpredictable in PRGs.

We know that $f(x \circ y) = x \circ g(y)$ is a OWP whenever g is. As G has an expansion factor of n+1, let us consider the example of simplest Blum-Micali generator which is G(x) = f(x)||hc(x), where f is some OW permutation as constructed above.

(a)

$$G_a(s) = G(s||\bar{s}) = f(s||\bar{s})||hc(s||\bar{s}) = s||g(\bar{s})||hc_f(s||\bar{s})$$

This clearly cannot be a PRG because a predictor $\mathbb{D}$ who has access to the blackbox of g can simply invert the first 'n' bits of the sampled string and observe that the next-n bits are $g(\bar{s})$. This violates the next-bit unpredictability as 'l' bits are predicted with full accuracy (As OWFs are deterministic).

(b) Let us assume $|s|$ is even. With a slight modification, if we define G(x) = f(x)||hc(x) where $f(x_1 \circ x_2) = x_1 \circ g(x_2)$ where g is a OW permutation over $\{0,1\}^{\frac{l}{2}}$, then we can see that:

$$G_b(s_1 \circ s_2) = G(s_1 \circ s_2)||G(\overline{s_1} \circ \overline{s_2}) = s_1 \circ g(s_2) \circ hc_f(s) \circ \overline{s_1} \circ g(\overline{s_2}) \circ hc_f(\bar{s})$$

A predictor $\mathbb{D}$ can always succeed in predicting the bits $[l+2, l+3, ..., \frac{3l}{2}]$ by making them the inverted versions of $[1, 2, ..., \frac{l}{2}]$. Therefore, this violates the next-bit unpredictability principle of PRG.
Note: If l is not even, just split the bits in 'f' unevenly and the rest of the proof holds true.

Feistel Networks

5. (3 points) Prove that a two-round Feistel network using pseudorandom round functions is not pseudorandom. *Reference for Feistel networks: Katz-Lindell V2 Section 7.6, and Wikipedia (Feistel cipher)*

Solution: If k_1 and k_2 are two successive keys, then a message m = $m_1||m_2$ after first round will take the form

$$m_2 \circ m_1 \oplus F_{k_1}(m_2)$$

After the second application, it will be:

$$m_1 \circ F_{k_1}(m_2), m_2 \oplus F_{k_1}(m_2)$$

Now, observe the left half of it. The xor value will purely depend on l and r independently. i.e., R2(x,y) = (m,n) implies R2($x \oplus a$,y) = ($m \oplus a$, n). If a distinguisher checks this exact condition, using x and $x \oplus a$, in two queries, it will always be correct when it faces PRF oracle. But on a random oracle, the probability that this happens is of course $\frac{1}{2^{\frac{n}{2}}}$ because every bit must coincide with that of a. Therefore, there is a poly non-negl D and hence two round Feistel networks are not PRFs.

6. (3 points) Prove that a three-round Feistel network using pseudorandom round functions is not strongly pseudorandom. *Hint: Use a distinguisher that makes two queries to the permutation and one query to its inverse.*

Solution: This is one of the proofs in Luby-Rackoff's works. Although the three round variant is pseudorandom, it is not strongly pseudorandom. Use the following strategy:

```
Query (a,b) ← Enc(l,r)
Query (c,d) ← Enc(l'($\neq l$),r)
Query (e,f) ← Dec(c,d+l-l')
Return 1 if f == c + r - a
Return 0 otherwise
```

On a Fiestal cipher, this test always passes. This can be seen by writing the three round expressions explicitly. However, on a truly random permutation, it passes with a negl probability. (Because, 'f' should be equal to a fixed value of c + r -a and this happens only with probability $\frac{1}{2^{\frac{n}{2}}}$). Therefore, the above distinguisher is non-negl and therefore it's not strongly pseudorandom.
(Note that the decryption query was given because we are testing for strong pseudorandomness and it is allowed by defintion)

7. (3 points) Consider the keyed permutation F^* defined as $F^*_k(x) := \mathsf{Feistel}_{F_k,F_k,F_k}(x)$. (Note that the same key is used in each round.) Show that F^* is not pseudorandom.

Solution: This is similar to Q5. If the same key 'k' is used through three rounds, we have

$$F^*_k(l,r) = (r \oplus F_k(l \oplus F_k(r)), l \oplus F_k(r) \oplus F_k(r \oplus F_k(l \oplus F_k(r))$$

Now, if we swap the first half and last half of this output, after three rounds, we get (r,l) as the output. A distinguisher can leverage this property by outputting '1' whenever

$$O(swap(O(l,r))) = swap(l,r) = (r,l)$$

and '0' otherwise. With a PRF oracle, he will always be correct and with a random oracle, he will be right only $\frac{1}{2^n}$ (l' should be same as r and r' should be same as l) which is negl. Therefore, F^* is not pseudorandom.

2 Fun with PRFs

Let $\{F_s : \{0,1\}^k \to \{0,1\}^k \mid s \in \{0,1\}^k\}$ be a family of pseudorandom functions with k-bit seed s. Prove whether the following new families are also pseudorandom functions or not: ($\circ$ denotes concatenation)

1. (2 points) $F_s^a(x) = F_{0^k}(x) \circ F_s(x)$

Solution: No, not a PRF. A distinguisher $\mathbb{D}$ can simply compute $F_{0^k}(x_o)$ for some x_o and can simply check with the oracle if the first-k bits match. When PRF: Probability = 1. When Random: Probability = $\frac{1}{2^k}$. The difference is non-negl.

2. (2 points) $F_s^b(x) = G(s) \oplus x$ where $G : \{0,1\}^k \to \{0,1\}^{2k}$ is a PRG.

Solution: No, not a PRF. G(s) is deterministic. Therefore, in a single security game, it can be removed by xor-ing the Oracle outputs of two inputs. Hence, the algo:

```
Get $y_1 = \mathbb{O}(x_1)$ and $y_2 = \mathbb{O}(x_2)$
Return 1 if ($x_1 \oplus x_2 == y_1 \oplus y_2$)
Return 0 otherwise
```

Will always succeed over Oracle using a PRF but will succeed exactly half times over a random Oracle. 1 - $\frac{1}{2} = \frac{1}{2}$ is clearly non-negl.

3. (2 points) $F_s^c(x) = F_{s_1}(x) \oplus s_2$ where $s = s_1 \circ s_2$ and $|s_1| = |s_2| = k$

Solution: We have, $F_s^c(x) = F_{s_1}(x)$. As $s_1 \circ s_2$ is uniformly randomly sampled, we can infer that s_1 and s_2 are as random as strings uniformly sampled from $\{0,1\}^n$.
Now, if a distinguisher accesses the oracle for say x_o, then

$$F_{s1}(x_o)s_2$$

is received. Suppose it has a non-negl distinguisher D. Then a new distinguisher (A) for F_s can be constructed. A simply takes the output of oracle of F_s and xors a random string s' to it (Let us assume it can generate random strings) and gives it to D. Let A return whatever B does.
We know that D is successful with a non-negl probability, where probability is over s1. On inspection, we find that A succeeds whenever D does. This is because, when A faces a random oracle, B will face something that is random too. However, if A faces a PRF oracle, B will face a emulated PRF oracle that follows our construction. Therefore, as B can distinguish between PRF and random oracle with non-negl and A,B return same output always, we can conclude that A is a non-negl distinguisher. But we know F_s is a PRF. Hence a contradiction and F^c is a PRF.

4. (2 points)

$$F^d(x) = \begin{cases} F_s(x) \text{ when } x \neq 0^k \\ a \circ b \text{ when } x = 0^k \end{cases}$$

where a is the first $\lfloor k/2 \rfloor$ bits of s and b is the last $\lceil k/2 \rceil$ bits of $F_s(x)$

Solution: Credits for the idea: `http://www.cs.brown.edu/courses/cs151/doc/sol07.pdf`.
The idea here is to use a function such that it uses only the first half of it's key for encryption. Let us define:

$$F_{s_1\|s_2}(x) = F^*_{G(s_1)}(x)$$

where F is our original PRF and F* is some PRF family and G is a PRG with 2n expansion factor. Since $s_1\|s_2$ is sampled uniformly randomly, this is a PRF because for all poly-time purposes, G(s_1) acts uniformly random.

We immediately see why F^d is not a PRF. This is because leaking first half of the key is as bad as leaking the whole of it (As G can be ran by adversary to find key for F^*). Therefore, a trivial distinguisher that can check if F(0^n) $= a \circ b = F^*_a(0^n)$ which is all computable can distinguish F with a probability 1 on PRF oracle. On a random oracle, the probability that this holds true is only $\frac{1}{2^{\frac{l}{2}}}$. Hence F^d need not be a PRF.

5. (3 points) $F^e_s(x) = F_s(0 \circ x) \circ F_s(1 \circ x)$

Solution: $F^e : \{0,1\}^n \rightarrow \{0,1\}^{2n}$. We will show that it is a PRF. If not, there is a non-negl Distinguisher D. Using that, we will construct an adversary for F_s called A.
Whenever D queries over x, A will emulate the oracle by sending $\mathbb{O}(0 \circ x) \circ \mathbb{O}(1 \circ x)$ to D. A outputs exactly what B does (either 0 or 1). From here, the argument is similar to Q3. Also note that non-negl(2n) = non-negl(n).

6. (3 points) $F^f_s(x) = F_s(G(x))$ where G is a pseudorandom generator from $\{0,1\}^{k/2} \rightarrow \{0,1\}^k$.

Solution: F^f is a PRF. We can give a similar reduction based proof here as well. Suppose F^f is not a PRF, then we know that it has a non-negl distinguisher. Call it D. Then We will construct a non-negl distinguisher for F_s family (say A).

Suppose A knows how to compute G. This can be done in poly-time. A can emulate the oracle of D by using it's own oracle. If D asks for O(x), A can instead send $O^A(G(x))$ (where O is oracle of D and O^A is A's oracle.)

A outputs whatever D does. Similar to above proofs, we can see that D faces random oracle whenever A does and a PRF (F^f) oracle whenever A faces PRF (F_s) oracle. Therefore, their outputs would be identical for either oracle. D can predict the which oracle it interacts with, nonnegligibly, thereby giving this power to A, but F_s is PRF so there should be no such A. Hence a contradiction

7. (3 points) $F^g_s(x) = G(F_s(x))$ where G is a pseudorandom generator from $\{0,1\}^k \rightarrow \{0,1\}^{2k}$.

Solution: This forms a PRF family. If not, there is a non-negl distinguisher D. Let us construct A that can distinguish F_s.
'A' emulates D's oracle by giving it G($O^A(x)$) whenever D asks for image of x. (Here O^A is A's oracle). After training, A can return whatever D does.
We know that G is a PR generator given a random seed. We can take it's output as perfectly random for all poly-time purposes as long as it's seed is. Therefore, G(O^A(x)) is purely random when the oracle is random. When the oracle A uses is that of a PRF, then we can see that D will face an emulation of PRF oracle over which we have assumed it has a non-negl advantage. Therefore, the oracle emulation is perfect (i.e., PRF to PRF and random to random). As A outputs whatever D does, and because D finds out the nature of oracle with non-negl probability, A also does so on F_s. This is a contradiction and hence F^g is PRF family.
(Though G is a PRG, we treat is just like a wrapper from k to 2k, as it is deterministic. This is because we have shown that our oracle to oracle emulation is perfect irregardless of G's PR property.)

3 PRFs as MACs

A message authentication code (MAC) is a method for ensuring that the data received came from the right person. Suppose Alice and Bob share a secret s and Bob wants to send m to Alice. Then a MAC $M_s(m)$ is an efficiently computable function family, using seed s, such that even if an active attacker Eve queries $M_s(.)$ on a set of messages of Eve's choice, Eve still cannot authenticate any message not explicitly queried.

A more formal definition is as follows: If A is a algorithm that is allowed to query a function O (the oracle), then let $Q \leftarrow A^O(i)$ denote the set of A's queries and answers upon termination with oracle O and input i. Now, a function family $\{M_s(.)\}_s$ is a MAC family if for all ppt A:

$$\Pr_{s \leftarrow \{0,1\}^k}[((m,x),Q) \leftarrow A^{M_s}(1^k) : x = M_s(m) \text{ and } (m,x) \notin Q] = negl(k)$$

With more English: Adversary A is allowed to query the value of $M_s(.)$ for a random seed s (A knows the MAC family $\{M_s\}$ but the seed s being used is secret). For each query m, A receives output $x = M_s(m)$. Let the set of query-output pairs (m,x) be Q. Now, A "wins" by now producing a **new** pair (m,x) such that $(m,x) \notin Q$. M_s is secure if any ppt A can only win with negligible probability. Note that a ppt algorithm can only make a polynomial number of queries. As usual, the role of input 1^k is to allow A to run in $poly(k)$ time (as opposed to $poly(\log k)$).

1. (3 points) Let $\{F_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}$ be a PRF family. Show (by reduction) that it is a MAC family.

Solution: We will show a reduction and form a contradiction to PRF-ness of F_s. Suppose E (Eve) can break the MAC authentication over F_s. Let us construct a distinguisher D for F_s from this.
Since D has the access to oracle of F_s, it can help answer the queries in the training phase of E. That is, if E asks for encryption of x, i.e., $M_s(x)$, just provide it with $O(x)$.

When E's training ends, it will output a (m,x) pair which may or may not be correct as per MAC authentication. However, as we have assume E breaks MAC authentication, (m,x) is correct with a non-negl probability: considered over random sampling of secret keys.

A uses E in this way but also makes an extra query. It checks if x==O(m) and returns 1 only if this is satisfied. Now, let us see how often A will be correct:

$$Pr(\text{A is correct}) = \frac{1}{2}[Pr(\frac{\text{A returns 1}}{\text{A faced a PRF oracle}}) + Pr(\frac{\text{A returns 0}}{\text{A faced a random oracle}})]$$

From the behaviour of A, we have:

$$Pr(\text{A is correct}) = \frac{1}{2}[Pr(\frac{x == O(m)}{\text{A faced a PRF oracle}}) + Pr(\frac{x! = O(m)}{\text{A faced a random oracle}})]$$

$$Pr(\text{A is correct}) = \frac{1}{2}[Pr(\frac{x == O(m)}{\text{A faced a PRF oracle}}) + 1 - Pr(\frac{x == O(m)}{\text{A faced a random oracle}})]$$

The first term is same as the non-negl probability with which E breaks MAC authentication. The third term is the probability that (m,x) is a vaild guess- in the wild. Therefore, we have the advantage

$$= \frac{1}{2}(\mathsf{p}(n) - \frac{1}{2^k})$$

This is clearly non-negl as p(n) is a non-negl. A contradiction that F_s is PRF. Therefore PRF families are also MAC.

2. (2 points) Is it the case that a MAC family is also a PRF family? Prove.

Solution: In a MAC family, it is hard to guess a (m,x) pair where as in a PRF family it is hard to guess if such pairs come from a random mapping or a poly-time pseudo random mapping. The notions sound similar as they capture the difficulty of guessing the behaviour of a family of mappings. However, their mathematical definitions make them different.

As a counter example, consider M_1 to be a MAC family. Then construct another MAC family that follows:

$$s \leftarrow R\{0,1\}^n; M_{2s}(x) = 0 \circ M_{1s}(x)$$

Clearly this is not a PRF as a trivial first-bit-based distinguisher can distinguish this from a random family with non-negl probability of 0.75.

3. (4 points) Is it true that if MACs exist then PRFs exist? Briefly defend your answer.

Hint: To show existence, you can just construct a OWF using a MAC. Explain why. The reason this is easier than building a PRF directly is because the security of an OWF is based on inverting being hard, while the security of a PRF is based on distinguishing being hard (which maybe be easy as you saw in part 2).

Construct the OWF such that inverting the output will give you the seed of the PRF. It may be the case that many seeds lead to the same output on a given message, so take many messages. Since the question asks for a brief defense (you don't have to lawyer up), just assume the following statement: the number of seeds that all have the same output on polynomially-many randomly-chosen messages is small (say, at most polynomial in n). This simplifies your argument when building a OWF.

Solution: Brief idea from http://cs.brown.edu/courses/cs151/doc/sol07.pdf:
As OWF's existence implies that of PRF's, it will be enough to show that MAC's existence implies that of OWF's. Therefore, using a MAC family, giving a construction of a OWF will suffice.

Using the function $f(x) = f(s, m_1, m_2, ..., m_{2k}) = m_1 \circ m_2.... \circ m_{2k} \circ M_s(m_1) \circ M_s(m_2).... \circ M_s(m_{2k})$, we can infer that whenever it is inverted, 's' will be spilled. However, this 's' might still not be the secret key Alice and Bob initially shared but might coincidentally be producing all these M_s's correctly.
Obviously, the next step is to show that such coincidences occur pretty rarely, thereby translating a f's inversion into MAC forgery almost always (except for that minute probability).