6.875: Cryptography and Cryptanalysis | September 22, 2021

Last updated: November 22, 2021

Problem Set 2

Instructor: Vinod Vaikuntanathan

TAs: Lali Devadas and Sacha Servan-Schreiber

Instructions.

- **When:** This problem set is due on **October 6, 2021** before **11pm ET**.
- **How:** You should use LaTeX to type up your solutions (you can use our LaTeX template from the course webpage). Solutions should be uploaded on Gradescope as a single pdf file.
- **Acknowledge your collaborators:** Collaboration is permitted and encouraged in small groups of at most three. You must write up your solutions *entirely on your own* and *acknowledge your collaborators*.
- **Reference your sources:** If you use material from outside the lectures, you must reference your sources (papers, websites, wikipedia, ...).
- **When in doubt, ask questions:** Use Piazza or the TA office hours for questions about the problem set. See the course webpage for the timings.

Problem 1. Let's Encrypt *and* Authenticate!

Let $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure encryption scheme and $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$ be an EUF-CMA secure scheme (defined below). Suppose Alice and Bob meet up in their secret hideout before Alice leaves to study abroad on Mars, and generate two secret keys k_1 and k_2, for encryption and authentication, respectively. That is, we define their algorithm $\mathsf{Gen}'(1^\lambda)$ to return $k_1 \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda)$ and $k_2 \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$.

While Alice is on Mars, they can only send each other messages via a public Earth-Mars broadcast (which is monitored by their nemesis E.V.E.), but they still want to communicate in a private and authenticated way. For Alice and Bob, this just means IND-CPA and EUF-CMA security (note that in the real world, we want much stronger guarantees.

Definition 1 (IND-CPA-security)
Let $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme with message space* $\mathcal{M}$ *and key space* $\mathcal{K}$ *with security parameter* λ*. Sample secret key* $k \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda)$ *and define encryption oracle* $\mathsf{Enc}(k, \cdot)$*, which on query* m*, outputs* $\mathsf{Enc}(k, m)$*. This scheme is* ***IND-CPA-secure*** *(a.k.a. computationally indistinguishable against chosen plaintext attacks) if for all PPT algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* negl *such that for all* λ

$$\Pr\left[\begin{array}{l} k \leftarrow \mathsf{Gen}(1^\lambda); \\ (m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Enc}(k,\cdot)}(1^\lambda); \\ b \xleftarrow{R} \{0,1\};\ c \leftarrow \mathsf{Enc}(k, m_b); \\ b' \leftarrow \mathcal{A}_2^{\mathsf{Enc}(k,\cdot)}(1^\lambda, c, \mathsf{state}) : \\ b' = b \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda).$$

Definition 2 (EUF-CMA-security)
Let $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$ *be a message authentication scheme with message space* $\mathcal{M}$ *and key space* $\mathcal{K}$ *with security parameter* λ*. Let* $k \leftarrow \mathsf{Gen}(1^\lambda)$ *and define MAC oracle* $\mathsf{MAC}(k, \cdot)$*, which on query* m*, outputs* $\mathsf{MAC}(k, m)$*. This scheme is* ***EUF-CMA-secure*** *(a.k.a. existentially unforgeable against chosen message attacks) if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that for all* λ

$$\Pr\left[\begin{array}{l} k \leftarrow \mathsf{Gen}(1^\lambda); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{MAC}(k,\cdot)}(1^\lambda) : \\ m^* \notin Q \text{ and } \mathsf{Verify}(k, m^*, \sigma^*) = 1 \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where Q *is the set of messages that* $\mathcal{A}$ *queried to the oracle.*

For each of the following:

- **Construct algorithms** $\mathsf{Dec}', \mathsf{Verify}'$ **such that** $\mathcal{E}_1 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Dec}')$ **is a correct encryption scheme and** $\mathcal{E}_2 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Verify}')$ **is a correct message authentication scheme (both schemes will use both keys).**
- **Either prove** $\mathcal{E}_1$ **is IND-CPA secure and** $\mathcal{E}_2$ **is EUF-CMA secure via reductions, or provide a attack on at least one.**

(a) $\mathsf{Transmit}(k_1, k_2, m) = \mathsf{Enc}(k_1, (m, \mathsf{MAC}(k_2, m)))$.

Solution

Define algorithms $\mathsf{Dec}'(k_1, k_2, \cdot)$ and $\mathsf{Verify}'(k_1, k_2, \cdot)$ as follows:

$\mathsf{Dec}'(k_1, k_2, c)$	$\mathsf{Verify}'(k_1, k_2, c)$
1 : $(m, t) \leftarrow \mathsf{Dec}(k_1, c)$	1 : $(m, t) \leftarrow \mathsf{Dec}(k_1, c)$
2 : **return** m	2 : **return** $\mathsf{Verify}(k_2, m, t)$

First we show that the scheme $\mathcal{E}_1 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Dec}')$ is perfectly correct. We have that for all messages $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l} (k_1, k_2) \leftarrow \mathsf{Gen}'(1^\lambda); \\ c \leftarrow \mathsf{Transmit}(k_1, k_2, m) : \\ \mathsf{Dec}'(k_1, k_2, c) = m \end{array}\right] = \Pr\left[\begin{array}{l} k_1 \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda) : \\ \mathsf{Dec}(k_1, \mathsf{Enc}(k_1, m)) = m \end{array}\right] = 1,$$

where the first equality follows from the definition of Dec' and the second equality follows from perfect correctness of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$.
Now we show that the scheme $\mathcal{E}_2 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Verify}')$ is perfectly correct. We again have that for all messages $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l} (k_1, k_2) \leftarrow \mathsf{Gen}'(1^\lambda); \\ c \leftarrow \mathsf{Transmit}(k_1, k_2, m) : \\ \mathsf{Verify}'(k_1, k_2, c) = m \end{array}\right] = \Pr\left[\begin{array}{l} k_2 \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda) : \\ \mathsf{Verify}(k_2, (m, \mathsf{MAC}(k_2, m))) = 1 \end{array}\right] = 1,$$

where the first equality follows from the definition of Verify' and the perfect correctness of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ and the second equality follows from perfect correctness of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$.

Solution (continued...)

This scheme **is** secure. We will show reductions to the computational indistinguishability of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ and the existential unforgeability of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$.

Computational indistinguishability

Suppose that $\mathcal{B}$ breaks the IND-CPA security of $\mathcal{E}_1$. Then $\mathcal{A}$ defined as follows breaks the IND-CPA security of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$:

Algorithm $\mathcal{A}$

$k_2 \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$
while in query phase :
  receive m from $\mathcal{B}$
  $t \leftarrow \mathsf{MAC}(k_2, m)$
  send (m, t) to challenger as query
  receive $c = \mathsf{Enc}(k_1, (m, t))$
  send c to $\mathcal{B}$
receive challenge messages m_0, m_1 from $\mathcal{B}$
$t_0 \leftarrow \mathsf{MAC}(k_2, m_0), \ t_1 \leftarrow \mathsf{MAC}(k_2, m_1)$
send challenge messages $(m_0, t_0), (m_1, t_1)$ to challenger
receive challenge ciphertext $c^* = \mathsf{Enc}(k_1, (m_b, t_b))$ for $b \xleftarrow{R} \{0, 1\}$
send c^* to $\mathcal{B}$ as challenge ciphertext
receive guess b' from $\mathcal{B}$
send b' to challenger as guess

Existential unforgeability

Suppose that $\mathcal{B}$ breaks the EUF-CMA security of $\mathcal{E}_2$. Then $\mathcal{A}$ defined as follows breaks the EUF-CMA security of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$:

Algorithm $\mathcal{A}$

$k_1 \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda)$
while in query phase :
  receive m from $\mathcal{B}$
  send m to challenger as query
  receive $t = \mathsf{MAC}(k_2, m)$
  $c \leftarrow \mathsf{Enc}(k_1, (m, t))$
  send c to $\mathcal{B}$
receive forgery (m', c') from $\mathcal{B}$
$(m', t') \leftarrow \mathsf{Dec}(k_1, c')$
send (m', t') to challenger as forgery

(b) $\mathsf{Transmit}(k_1, k_2, m) = (\mathsf{Enc}(k_1, m), \mathsf{MAC}(k_2, m))$.

Solution

Define algorithms $\mathsf{Dec}'(k_1, k_2, \cdot)$ and $\mathsf{Verify}'(k_1, k_2, \cdot)$ as follows:

$\mathsf{Dec}'(k_1, k_2, c)$	$\mathsf{Verify}'(k_1, k_2, c)$
1 : parse $c = (c_1, c_2)$	1 : parse $c = (c_1, c_2)$
2 : $m \leftarrow \mathsf{Dec}(k_1, c_1)$	2 : $m \leftarrow \mathsf{Dec}(k_1, c_1)$
3 : **return** m	3 : **return** $\mathsf{Verify}(k_2, m, c_2)$

First we show that the scheme $\mathcal{E}_1 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Dec}')$ is perfectly correct. We have that for all messages $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}(k_1, k_2) \leftarrow \mathsf{Gen}'(1^\lambda);\\ c \leftarrow \mathsf{Transmit}(k_1, k_2, m) :\\ \mathsf{Dec}'(k_1, k_2, c) = m\end{array}\right] = \Pr\left[\begin{array}{l}k_1 \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda) :\\ \mathsf{Dec}(k_1, \mathsf{Enc}(k_1, m)) = m\end{array}\right] = 1,$$

where the first equality follows from the definition of Dec' and the second equality follows from perfect correctness of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$.

Now we show that the scheme $\mathcal{E}_2 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Verify}')$ is perfectly correct. We again have that for all messages $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}(k_1, k_2) \leftarrow \mathsf{Gen}'(1^\lambda);\\ c \leftarrow \mathsf{Transmit}(k_1, k_2, m) :\\ \mathsf{Verify}'(k_1, k_2, c) = m\end{array}\right] = \Pr\left[\begin{array}{l}k_2 \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda) :\\ \mathsf{Verify}(k_2, (m, \mathsf{MAC}(k_2, m))) = 1\end{array}\right] = 1,$$

where the first equality follows from the definition of Verify' and the perfect correctness of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ and the second equality follows from perfect correctness of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$.

Solution (continued...)

This scheme is **not** secure. Let $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$ be a EUF-CMA-secure message authentication scheme. Define $\mathsf{MAC}'_k(m) = (m, \mathsf{MAC}_k(m))$. Suppose that $\mathcal{B}$ breaks the EUF-CMA security of $\left(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}', \mathsf{Verify}\right)$. Then $\mathcal{B}'$ defined as follows breaks the EUF-CMA security of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$:

Algorithm $\mathcal{B}$'

while in query phase :
  receive m from $\mathcal{B}$
  send m to challenger as query
  receive $t = \mathsf{MAC}(k_2, m)$
  send (m, t) to $\mathcal{B}$
receive forgery $(m', (m', t'))$ from $\mathcal{B}$
send (m', t') to challenger as forgery

Thus $\left(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}', \mathsf{Verify}\right)$ is also EUF-CMA-secure. However, if we use this as our message authentication scheme, then the scheme $\mathcal{E}_1 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Dec}')$ is obviously not semantically secure, because the ciphertext includes the message in the clear! To formalize this, define $\mathcal{A}$ which wins the IND-CPA game with probability $1 \geq \mathsf{nonnegl}(\lambda)$ against $\mathcal{E}_1$ as follows:

Algorithm $\mathcal{A}$

send challenge messages m_0, m_1 s.t. $m_0 \neq m_1$
receive challenge ciphertext $c = (\mathsf{Enc}(k_1, m_b), (m_b, \mathsf{MAC}(k_2, m_b)))$ for $b \xleftarrow{R} \{0,1\}$
parse $c = (c_1, (c_2, c_3))$
if $c_2 = m_0$: **return** 0
else : **return** 1

(c) $\mathsf{Transmit}(k_1, k_2, m) = (\mathsf{Enc}(k_1, m), \mathsf{MAC}(k_2, \mathsf{Enc}(k_1, m)))$.

Solution

Define algorithms $\mathsf{Dec}'(k_1, k_2, \cdot)$ and $\mathsf{Verify}'(k_1, k_2, \cdot)$ as follows:

$\mathsf{Dec}'(k_1, k_2, c)$	$\mathsf{Verify}'(k_1, k_2, c)$
1 : parse $c = (c_1, c_2)$	1 : parse $c = (c_1, c_2)$
2 : $m \leftarrow \mathsf{Dec}(k_1, c_1)$	2 : **return** $\mathsf{Verify}(k_2, c_1, c_2)$
3 : **return** m	

First we show that the scheme $\mathcal{E}_1 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Dec}')$ is perfectly correct. We have that for all messages $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}(k_1, k_2) \leftarrow \mathsf{Gen}'(1^\lambda);\\ c \leftarrow \mathsf{Transmit}(k_1, k_2, m):\\ \mathsf{Dec}'(k_1, k_2, c) = m\end{array}\right] = \Pr\left[\begin{array}{l}k_1 \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda):\\ \mathsf{Dec}(k_1, \mathsf{Enc}(k_1, m)) = m\end{array}\right] = 1,$$

where the first equality follows from the definition of Dec' and the second equality follows from perfect correctness of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$.

Now we show that the scheme $\mathcal{E}_2 = (\mathsf{Gen}', \mathsf{Transmit}, \mathsf{Verify}')$ is perfectly correct. We again have that for all messages $m \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}(k_1, k_2) \leftarrow \mathsf{Gen}'(1^\lambda);\\ c \leftarrow \mathsf{Transmit}(k_1, k_2, m):\\ \mathsf{Verify}'(k_1, k_2, c) = m\end{array}\right] = \Pr\left[\begin{array}{l}k_2 \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda);\\ c' \leftarrow \mathsf{Enc}(k_1, m):\\ \mathsf{Verify}(k_2, (c', \mathsf{MAC}(k_2, c'))) = 1\end{array}\right] = 1,$$

where the first equality follows from the definition of Verify' and the second equality follows from perfect correctness of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$.

Solution (continued...)

This scheme **is** secure. We will show reductions to the computational indistinguishability of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ and the existential unforgeability of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC})$.

Computational indistinguishability

Suppose that $\mathcal{B}$ breaks the IND-CPA security of $\mathcal{E}_1$. Then $\mathcal{A}$ defined as follows breaks the IND-CPA security of $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$:

Algorithm $\mathcal{A}$

$k_2 \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$
while in query phase :
  receive m from $\mathcal{B}$
  send m to challenger as query
  receive $c = \mathsf{Enc}(k_1, m)$
  $t \leftarrow \mathsf{MAC}(k_2, c)$
  send (c, t) to $\mathcal{B}$
receive challenge messages m_0, m_1 from $\mathcal{B}$
send challenge messages m_0, m_1 to challenger
receive challenge ciphertext $c^* = \mathsf{Enc}(k_1, m_b)$ for $b \xleftarrow{R} \{0, 1\}$
$t^* \leftarrow \mathsf{MAC}(k_2, c^*)$
send (c^*, t^*) to $\mathcal{B}$ as challenge ciphertext
receive guess b' from $\mathcal{B}$
send b' to challenger as guess

Existential unforgeability

Suppose that $\mathcal{B}$ breaks the EUF-CMA security of $\mathcal{E}_2$. Then $\mathcal{A}$ defined as follows breaks the EUF-CMA security of $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify})$:

Algorithm $\mathcal{A}$

$k_1 \leftarrow \mathsf{Gen}_{\mathsf{Enc}}(1^\lambda)$
while in query phase :
  receive m from $\mathcal{B}$
  $c \leftarrow \mathsf{Enc}(k_1, m)$
  send c to challenger as query
  receive $t = \mathsf{MAC}(k_2, c)$
  send (c, t) to $\mathcal{B}$
receive forgery $(m', (c', t'))$ from $\mathcal{B}$
send (c', t') to challenger as forgery

Problem 2. Building one-way functions

Suppose f is a length-preserving[1] one-way function. In this problem, we write

- $\oplus$ to denote bitwise XOR,
- $||$ to denote concatenation of bit-strings,
- $\bar{x}$ to denote the bitwise complement of x.

For each of the following functions f', either prove that f' is always a OWF (by a reduction to the one-wayness of f), or provide a counter example showing that f' is not always a OWF for some OWF f.

(a) $f'(x, y) = f(x)||f(x \oplus y)$, where $|x| = |y|$.

Solution

f' **is** a one-way function. Let $|x| = |y| = \lambda$. Suppose towards contradiction that f' is not one-way. Then, there exists a PPT algorithm $\mathcal{A}$ which takes as input $(1^\lambda, z := f'(x, y))$ and finds a pre-image for z with non-negligible probability. Formally, we have that

$$\Pr\left[\begin{array}{l}(x, y) \xleftarrow{R} \{0,1\}^\lambda \times \{0,1\}^\lambda; \\ (x', y') \leftarrow \mathcal{A}(1^\lambda, z := f'(x, y)) : f'(x', y') = z\end{array}\right] \geq \delta(\lambda)$$

for some non-negligible function δ and where the probability is taken over the randomness of (x, y) and the randomness of $\mathcal{A}$.

We show how to construct a PPT inverter $\mathcal{B}$ for f as follows.

$\mathcal{B}$

---

receive input $(1^\lambda, z := f(x))$

$r \xleftarrow{R} \{0,1\}^\lambda$

$z' \leftarrow z \mid\mid f(r)$

$(x', y') \leftarrow \mathcal{A}(1^\lambda, z')$

return x'

Analysis: The distribution of $z' := z \mid\mid f(r)$ provided to $\mathcal{A}$ is identical to the distribution $f'(x, y)$ for random x, y because it holds that for all uniformly random $a, b \in \{0,1\}^\lambda$, $(a, a \oplus b) \approx (a, b)$ (i.e. $a \oplus b$ looks uniformly random because b is uniformly random). In this case, $f(r)$ is distributed identically to $f(x \oplus y)$ and independently from $z = f(x)$ given that r and y are both uniformly random and independent strings.

Let $\delta(\lambda)$ be the probability that $\mathcal{A}$ successfully inverts f'. Given that $f'(x', y') = z' = z \mid\mid f(r)$, we have that $f(x') = z$, which implies that x' is a pre-image for $z = f(x)$. Therefore, $\mathcal{B}$ succeeds in inverting f whenever $\mathcal{A}$ succeeds in inverting f', that is, with probability at least $\delta(\lambda)$. As such, we get that if f' is **not** one-way, then f is not one-way either. But this is a contradiction; thus f' is one way.

[1] For every $x \in \{0,1\}^*$, it holds that $|f(x)| = |x|$.

(b) $f'(x) = f(\bar{x})$

Solution

f' **is** a one-way function. Suppose towards contradiction that there exists a PPT algorithm $\mathcal{A}$ which takes as input $(1^\lambda, z := f'(x))$ and finds a pre-image for z with non-negligible probability. Formally, we have that

$$\Pr\left[\begin{array}{l} x \xleftarrow{R} \{0,1\}^\lambda; \\ x' \leftarrow \mathcal{A}(1^\lambda, z := f'(x)) : f'(x') = z \end{array}\right] \geq \delta(\lambda)$$

for some non-negligible function δ and where the probability is taken over the randomness of x and the randomness of $\mathcal{A}$.

We show how to construct a PPT inverter $\mathcal{B}$ for f as follows.

$\mathcal{B}$

receive input $(1^\lambda, z := f(x))$

$y \leftarrow \mathcal{A}(1^\lambda, z)$

return $\bar{y}$

Analysis: The distribution of the input provided to $\mathcal{A}$ matches the input distribution expected by $\mathcal{A}$: $\mathcal{A}$ expects $f'(r) = f(\overline{r})$ for uniformly random r (which we have if we set $r = \overline{x}$). $\mathcal{A}$ outputs a pre-image y such that $f'(y) = f(\bar{y}) = z$ with probability $\delta(\lambda)$. As such, $f(\bar{y}) = z$ and therefore $\bar{y}$ is the pre-image of f. $\mathcal{B}$ therefore succeeds in inverting f with probability $\delta(\lambda)$ (the same as $\mathcal{A}$) in polynomial time, so f is not one-way. But this is a contradiction; thus f' is one-way.

(c) $f'(x) = f(x)_{[1:|x|-1]}$

Solution

f' is **not** a one-way function. Let g be an arbitrary length-preserving one-way function.

Define $f(x) = \begin{cases} x & \text{if } x_{[\lambda/2+1:\lambda]} = 0^{\lambda/2} \\ g(x_{[1:\lambda/2]}) \mid\mid 0^{\lambda/2-1} \mid\mid 1 & \text{otherwise.} \end{cases}$

Suppose towards contradiction that f is not one-way. Then there exists some PPT algorithm $\mathcal{A}$ such that $\Pr\Big[x \xleftarrow{R} \{0,1\}^\lambda; x' \leftarrow \mathcal{A}(1^\lambda, f(x)) : f(x') = f(x)\Big] \geq \delta(\lambda)$ for some non-negligible function δ and where the probability is over the randomness of x and $\mathcal{A}$. We show how to construct a PPT inverter $\mathcal{B}$ for g as follows:

$\mathcal{B}$

receive input $(1^{\lambda/2}, g(z))$

$x' \leftarrow \mathcal{A}(1^\lambda, g(z) \mid\mid 0^{\lambda/2-1} \mid\mid 1)$

return $z' = x'_{[1:\lambda/2]}$

Analysis: The distribution of $g(z) \mid\mid 0^{\lambda/2-1} \mid\mid 1$ does not exactly match the distribution expected by $\mathcal{A}$, which is $f(r)$ for uniformly random r. It does not match when $r_{[\lambda/2+1:\lambda]} = 0^{\lambda/2}$. However, this only occurs with negligible probability, so the probability that $\mathcal{B}$ succeeds in inverting h is equal to $\delta(\lambda) - \mathsf{negl}(\lambda)$, which is still non-negligible. We will now formally bound the success probability of $\mathcal{B}$. For convenience, we will write S to denote the set $\{0,1\}^{\lambda/2} \setminus \{0^{\lambda/2}\}$.

$$\begin{aligned}
\delta(\lambda) &\leq \Pr\Big[x \xleftarrow{R} \{0,1\}^\lambda; x' \leftarrow \mathcal{A}(1^\lambda, f(x)) : f(x') = f(x)\Big] \\
&= \sum_{x \in \{0,1\}^\lambda} \frac{1}{2^\lambda} \Pr\big[x' \leftarrow \mathcal{A}(1^\lambda, f(x)) : f(x') = f(x)\big] \\
&= \sum_{z \in \{0,1\}^{\lambda/2}} \frac{1}{2^\lambda} \Pr\big[x' \leftarrow \mathcal{A}(1^\lambda, f(z \mid\mid z_2)) : f(x') = f(z \mid\mid z_2)\big] \\
&\qquad\qquad + \sum_{z_2 \in S} \sum_{z \in \{0,1\}^{\lambda/2}} \frac{1}{2^\lambda} \Pr\big[x' \leftarrow \mathcal{A}(1^\lambda, f(z \mid\mid z_2)) : f(x') = f(z \mid\mid z_2)\big] \\
&\leq \frac{2^{\lambda/2}}{2^\lambda} + \sum_{z_2 \in S} \sum_{z \in \{0,1\}^{\lambda/2}} \frac{1}{2^\lambda} \Pr[x' \leftarrow \mathcal{A}(1^\lambda, g(z) \mid\mid 0^{\lambda/2-1} \mid\mid 1) : \\
&\qquad\qquad\qquad\qquad g(x'_{[1:\lambda/2]}) \mid\mid 0^{\lambda/2-1} \mid\mid 1 = g(z) \mid\mid 0^{\lambda/2-1} \mid\mid 1] \\
&= \frac{1}{2^{\lambda/2}} + (2^{\lambda/2} - 1) \cdot \sum_{z \in \{0,1\}^{\lambda/2}} \frac{1}{2^\lambda} \Pr\Big[x' \leftarrow \mathcal{A}(1^\lambda, g(z) \mid\mid 0^{\lambda/2-1} \mid\mid 1) : g(x'_{[1:\lambda/2]}) = g(z)\Big] \\
&\leq \frac{1}{2^{\lambda/2}} + \sum_{z \in \{0,1\}^{\lambda/2}} \frac{1}{2^{\lambda/2}} \Pr\Big[x' \leftarrow \mathcal{A}(1^\lambda, g(z) \mid\mid 0^{\lambda/2-1} \mid\mid 1) : g(x'_{[1:\lambda/2]}) = g(z)\Big] \\
&= \frac{1}{2^{\lambda/2}} + \Pr\Big[z \xleftarrow{R} \{0,1\}^{\lambda/2}; z' \leftarrow \mathcal{B}(1^{\lambda/2}, g(z)) : g(z') = g(z)\Big] \\
\implies &\Pr\Big[z \xleftarrow{R} \{0,1\}^{\lambda/2}; z' \leftarrow \mathcal{B}(1^{\lambda/2}, g(z)) : g(z') = g(z)\Big] \geq \delta(n) - \frac{1}{2^{\lambda/2}}
\end{aligned}$$

$\delta(\lambda) - \frac{1}{2^{\lambda/2}}$ is non-negligible, but this is a contradiction. Thus f is one-way.

Solution (continued...)

Define $\mathcal{A}$' as follows:

$\mathcal{A}$'

1 : receive input $(1^\lambda, f'(x))$

2 : **return** $f'(x) \mid\mid 0$

Note that $\forall\ x \in \{0,1\}^\lambda, f'(x) = f(x)_{[1:\lambda-1]} = c \mid\mid 0^{\lambda/2-1}$ for some $c \in \{0,1\}^{\lambda/2}$
$\implies f(f(x) \mid\mid 0) = f(c \mid\mid 0^{\lambda/2}) = f'(c \mid\mid 0^{\lambda/2})_{[1:\lambda-1]} = c \mid\mid 0^{n/2-1} = f'(x)$.
Then $\Pr\Big[x \xleftarrow{R} \{0,1\}^\lambda; x' \leftarrow \mathcal{A}'(1^\lambda, f'(x)) : f'(x') = f(x)\Big] = 1$, which is non-negligible. Thus f' is not one-way.

Note. Here is a different approach suggested by students that doesn't quite work: Suppose for contradiction that f' is a OWF for all choices of OWFs f. Then, consider the following sequence of functions,

$$\begin{aligned} f_0(x) &:= f(x), \\ f_i(x) &:= 0||f_{i-1}(x)_{[0:n-1]}, && \text{for } 1 \leq i \leq n. \end{aligned}$$

By our assumption, if f_i is a OWF, then f_{i+1} is also a OWF, since concatenating a 0 cannot make a OWF a non-OWF. So by induction, if f_0 is a OWF, it must be true that $f_n = 0^n$ is also a OWF, reaching a contradiction.
The main issue with this solution is that at each step there could be some loss (in either security or time) which could end up quite large after n steps. The assumption "if f_i is a OWF then f_{i+1} is a OWF", which is equivalent to "if there is an adversary $\mathcal{A}_{i+1}$ for f_{i+1} then there is also an adversary $\mathcal{A}_i$ for f_i", needs to be made more quantitative. Suppose that your adversary $\mathcal{A}_{i+1}$ for f_{i+1} runs in time T_{i+1}, and the adversary $\mathcal{A}_i$ for f_i runs in time $T_i = 2T_{i+1}$. Note that $\mathcal{A}_i$ still runs in polynomial time. But when you repeat this for n steps, you end up with an adversary $\mathcal{A}_0$ for f_0 that runs in time $T_0 = 2^n T_n$, which is exponentially big. But this does not contradict the one-wayness of the original function $f = f_0$.

(d) $f'(x) = f(f(x))$

Solution

f' is **not** a one-way function. Let $|x| = \lambda = \lambda_\ell + \lambda_r$ for integers $\lambda_\ell = \lceil \frac{\lambda}{2} \rceil$ and $\lambda_r = \lfloor \frac{\lambda}{2} \rfloor$. Let $h(\cdot)$ be any length-preserving one-way function.

Define $f(x) = \begin{cases} 0^\lambda & \text{if } x_1 \cdots x_{\lambda_\ell} = 0^{\lambda_\ell}, \\ 0^{\lambda_\ell} \mid\mid h(x_{\lambda_\ell+1}, \ldots, x_{\lambda_r}) & \text{otherwise.} \end{cases}$

Clearly f is efficiently computable. Moreover, using f as defined above, it is easy to see that $f'(x) = f(f(x)) = 0^\lambda$ for all inputs x, making f' trivially invertible. Hence, all that remains is to show that f (as constructed above) is a one-way function.

Suppose, towards contradiction, that f is not one-way. Then, there exists a PPT algorithm $\mathcal{A}$ that inverts f with non-negligible probability. Formally, we have that

$$\Pr\left[\begin{array}{l} x \xleftarrow{R} \{0,1\}^\lambda; \\ x' \leftarrow \mathcal{A}(1^\lambda, y := f(x)) : f(x') = y \end{array} \right] \geq \delta(\lambda)$$

for some non-negligible function δ and where the probability is taken over the randomness of x and the randomness of $\mathcal{A}$. We show how to construct a PPT inverter $\mathcal{B}$ for h as follows:

$\mathcal{B}$

receive input $(1^\lambda, y = h(x))$

$y' \leftarrow 0^{\lambda_\ell} \mid\mid y$

$x' \leftarrow \mathcal{A}(1^\lambda, y')$

return $x'_{[\lambda_\ell+1:r]}$

Analysis: The distribution of y' matches the distribution expected by $\mathcal{A}$ *conditioned* on $x_{[1:\lambda_\ell]} \neq 0^{\lambda_\ell}$. Therefore, $\mathcal{B}$'s success hinges on $x_{[1:\lambda_\ell]} \neq 0^{\lambda_\ell}$. The probability that for random $x \in \{0,1\}^\lambda$, $x_{[1:\lambda_\ell]} = 0^{\lambda_\ell}$ is $\frac{1}{2^{\lambda_\ell}}$ and thus negligible. As such, the probability that $\mathcal{B}$ succeeds in inverting h is equal to $\delta(\lambda) - \mathsf{negl}(\lambda)$, which is still non-negligible (see a formal probability analysis of this style in the solution to (c) above), so h is not one-way. But this is a contradiction; thus f is one way.
However, as we show above, $f'(x) = f(f(x))$ is **not** one-way given that *any* input $x \in \{0,1\}^\lambda$ results in 0^λ as output, so an inverter can return anything and succeed with probability 1.

Problem 3. More fun with one-way functions!

Alice comes across a function $f(x, y) = (g_1(x), g_2(y))$ based on two one-way functions g_1, g_2. She wants to try and invert this function. Let $x, y \in \{0,1\}^\lambda$. Her friend Bob has a access to a special black-box algorithm $\mathcal{B}$ which, on input $(g_1(x), g_2(y))$, computes the inner product of x and y mod 2, denoted $\langle x, y\rangle$ mod 2. Specifically, $\mathcal{B}(1^\lambda, g_1(x), g_2(y))$ outputs

$$\langle x, y\rangle \bmod 2 = \sum_{i=1}^{\lambda} x_i y_i \bmod 2.$$

He's willing to help Alice by giving her access to $\mathcal{B}$.

(a) **Suppose that $\mathcal{B}$ outputs the correct inner product with near-perfect probability $1 - \mathsf{negl}(\lambda)$ on random x and y. Prove that with Bob's help, Alice can use $\mathcal{B}$ to invert f with non-negligible probability.**

(b) **Now, suppose $\mathcal{B}$ outputs the correct inner product with probability $\frac{1}{2} + \epsilon$ for some constant $\frac{1}{4} < \epsilon < \frac{1}{2}$, again for random x and y. Prove that Alice can still use $\mathcal{B}$ to invert f with non-negligible probability. (The runtime of Alice's inverter can depend on ϵ.)**

Solution

Part (a) (near-perfect $\mathcal{B}$): Observe that the inner product is a linear function. As such, for any random $r \in \{0,1\}^\lambda$, it holds that

$$\begin{aligned}
&\langle a, b \oplus r\rangle \oplus \langle a, r\rangle \\
&\qquad = \left(\sum_{i=1}^{\lambda} a_i(b + r_i) \bmod 2 - \sum_{i=1}^{\lambda} a_i r_i \bmod 2\right) \bmod 2 \\
&\qquad = \sum_{i=1}^{\lambda} a_i(b + r_i - r_i) \bmod 2 \\
&\qquad = \langle a, b\rangle.
\end{aligned}$$

Recall that $\oplus$ is commutative and that $a + b \bmod 2 = a \oplus b = a - b \bmod 2$.
Moreover, if r is random, then so is $b \oplus r$. We can then construct the following PPT algorithms $\mathcal{A}_{g_1}$ and $\mathcal{A}_{g_2}$ which compute $\langle x, z\rangle$ and $\langle y, z\rangle$, respectively, for **any** z (not necessarily random) by invoking $\mathcal{B}$. Later, we use $(\mathcal{A}_{g_1}, \mathcal{A}_{g_2})$ to fully invert f.

Algorithm $\mathcal{A}_{g_1}(1^\lambda, g_1(x), z)$

$r \xleftarrow{R} \{0,1\}^\lambda$

$b_0 \leftarrow \mathcal{B}(1^\lambda, g_1(x), g_2(r))$

$b_1 \leftarrow \mathcal{B}(1^\lambda, g_1(x), g_2(z \oplus r))$

return $b_0 \oplus b_1$

Algorithm $\mathcal{A}_{g_2}(1^\lambda, g_2(y), z)$

$r \xleftarrow{R} \{0,1\}^\lambda$

$b_0 \leftarrow \mathcal{B}(1^\lambda, g_1(r), g_2(y))$

$b_1 \leftarrow \mathcal{B}(1^\lambda, g_1(z \oplus r), g_2(y))$

return $b_0 \oplus b_1$

First, we analyze correctness of $(\mathcal{A}_{g_1}, \mathcal{A}_{g_2})$. Consider $\mathcal{A}_{g_1}$ first (the same argument applies to $\mathcal{A}_{g_2}$). $b_0 = \langle x, r\rangle$ and $b_1 = \langle x, z \oplus r\rangle$. Moreover, r and $z \oplus r$ are **uniformly random** because r is uniformly random. Therefore, we know that $\mathcal{B}$ fails in computing each inner product with only negligible probability, and an easy union bound tells us that it computes both correctly with probability $1 - \mathsf{negl}(\lambda)$. Finally, we have that $b_0 \oplus b_1 = (\langle x, r\rangle \bmod 2) \oplus (\langle x, z \oplus r\rangle \bmod 2) = \langle x, z\rangle \bmod 2$ as required. Thus, we have that $\mathcal{A}_{g_1}$ (and $\mathcal{A}_{g_2}$ by symmetry) successfully compute the inner product for **any** z with probability $1 - \mathsf{negl}(\lambda)$ using $\mathcal{B}$ as a subroutine.

Solution (continued...)

How do we use $(\mathcal{A}_{g_1}, \mathcal{A}_{g_2})$ to invert f? Because $(\mathcal{A}_{g_1}, \mathcal{A}_{g_2})$ work for arbitrary z, we can build an inverter for f as follows.

Algorithm $\mathcal{A}_f(1^\lambda, f(x, y))$

> **parse** $f(x, y) = (g_1(x), g_2(y))$
> **for** $i = 1 \dots \lambda$ **do**
> $e_i \leftarrow$ ith row of the identity matrix. // $e_i = (0, \dots, 0 \underbrace{1}_{\texttt{ith index}}, 0, \dots, 0)$
> $x_i \leftarrow \mathcal{A}_{g_1}(1^\lambda, g_1(x), e_i)$
> $y_i \leftarrow \mathcal{A}_{g_2}(1^\lambda, g_2(y), e_i)$
> **endfor**
> $x' := x_1 \dots x_\lambda$
> $y' := y_1 \dots y_\lambda$
> **return** (x', y')

Analysis: Because $(\mathcal{A}_{g_1}, \mathcal{A}_{g_2})$ work with near-perfect probability (we're in the case where $\mathcal{B}$ succeeds with probability $1 - \mathsf{negl}(\lambda)$ over random inputs), we know that each inner product $\langle x, z \rangle$ computed by $\mathcal{A}_{g_1}$ is correct with probability $1 - \mathsf{negl}(\lambda)$ by the analysis above. Likewise for the inner product $\langle y, z \rangle$ computed by $\mathcal{A}_{g_2}$.

$\mathcal{A}_f$ recovers each bit of x and y by running $\mathcal{A}_{g_1}$ and $\mathcal{A}_{g_2}$, respectively. To see this, notice that $\langle x, e_i \rangle \bmod 2 = x_i$ (the ith bit of x). The same holds for $\langle y, e_i \rangle$. By the union bound, the probability that $\mathcal{A}_f$ fails to recover the correct x is bounded by

$$\lambda \Pr[\mathcal{A}_{g_1} \text{ fails or } \mathcal{A}_{g_2} \text{ fails}] \leq 2\lambda \Pr[\mathcal{A}_{g_1} \text{ fails }] \leq 2\lambda \mathsf{negl}(\lambda) = \mathsf{negl}'(\lambda),$$

for some negligible function negl'.
Therefore, using Bob's $\mathcal{B}$, Alice successfully inverts f, with probability at least $1 - \mathsf{negl}'(\lambda)$, which is non-negligible.

Part (b) (less-perfect $\mathcal{B}$): We now consider the case where $\mathcal{B}$ succeeds with probability $\frac{1}{2} + \epsilon$ where $\frac{1}{4} < \epsilon < \frac{1}{2}$ on random inputs. Intuitively, we will have to run $\mathcal{B}$ repeatedly and take the majority output. We will determine the number of repeated trials t required using a Chernoff bound. Define $\mathcal{A}_{g_1}$ and $\mathcal{A}_{g_2}$ as in part 1.
To aid our analysis, we will define four sets:

$$\mathsf{Good}_{g_1} = \{s \in \{0,1\}^\lambda : \mathcal{B}(g_1(x), g_2(s)) = \langle x, s \rangle \bmod 2\}$$
$$\mathsf{Good}_{g_2} = \{s \in \{0,1\}^\lambda : \mathcal{B}(g_1(s), g_2(y)) = \langle y, s \rangle \bmod 2\}$$

$$\mathsf{Bad}_{g_1} = \{s \in \{0,1\}^\lambda : \mathcal{B}(g_1(x), g_2(s)) \neq \langle x, s \rangle \bmod 2\}$$
$$\mathsf{Bad}_{g_2} = \{s \in \{0,1\}^\lambda : \mathcal{B}(g_1(s), g_2(y)) \neq \langle y, s \rangle \bmod 2\}.$$

We will focus our analysis on $\mathcal{A}_{g_1}$ (which involves sets Good_{g_1} and Bad_{g_1}), with the understanding that the case for $\mathcal{A}_{g_2}$ (with sets Good_{g_2} and Bad_{g_2}) follows by symmetry.

Solution (continued...)

Let's compute the probability that $\mathcal{A}_{g_1}$ fails. First, observes that $|\mathsf{Good}_{g_1}| = (\frac{1}{2}+\epsilon)2^\lambda$ and $|\mathsf{Bad}_{g_1}| = (\frac{1}{2}-\epsilon)2^\lambda$. Next, we will upper bound the probability of failure so as to be able to compute a lower bound on the success probability.

$$\begin{aligned}
&\Pr[\mathcal{A}_{g_1}(1^\lambda, g_1(x), z) \neq \langle x, z\rangle \bmod 2] \\
&\qquad = \Pr[b_0 \oplus b_1 \neq \langle x, z\rangle \bmod 2] \\
&\qquad = \Pr[\mathcal{B}(g_1(x), g_2(z\oplus r)) \neq \langle x, z\oplus r\rangle \bmod 2 \text{ OR } \mathcal{B}(g_1(x), g_2(r)) \neq \langle x, r\rangle \bmod 2] \\
&\qquad = \Pr[z\oplus r \in \mathsf{Bad}_{g_1} \text{ OR } r \in \mathsf{Bad}_{g_1}] \\
&\qquad \leq \Pr[z\oplus r \in \mathsf{Bad}_{g_1}] + \Pr[r \in \mathsf{Bad}_{g_1}] \\
&\qquad = \left(\frac{1}{2}-\epsilon\right) + \left(\frac{1}{2}-\epsilon\right) = 1-2\epsilon \\
&\qquad = \frac{1}{2} - \delta \quad \text{ for } \delta = 2\epsilon - \frac{1}{2}.
\end{aligned}$$

(Note that we have $0 < \delta < \frac{1}{2}$ by assumption that $\frac{1}{4} < \epsilon < \frac{1}{2}$.) We have that $\mathcal{A}_{g_1}$ (and $\mathcal{A}_{g_2}$ by symmetry) succeed with probability $\frac{1}{2}+\delta$. We now compute the number of trials t (in terms of δ) necessary to ensure that $\mathcal{A}_f$ succeeds with probability $p = 1-\mathsf{negl}(\lambda)$ by applying the Chernoff and union bounds.
First, observe that the probability that $\mathcal{A}_f$ succeeds (denoted p) is

$$p \geq 1 - \lambda \Pr[\mathcal{A}_{g_1} \text{ fails OR } \mathcal{A}_{g_2} \text{ fails}] = 1 - 2\lambda \cdot \Pr[\mathcal{A}_{g_1} \text{ fails}]$$

by the union bound and independence (and symmetry) of $\mathcal{A}_{g_1}$ and $\mathcal{A}_{g_2}$. To see why, observe that $\mathcal{A}_f$ must successfully recover each bit of x (resp. y) and is successful in doing so when the majority bit over t trials output by $\mathcal{A}_{g_1}$ (resp. $\mathcal{A}_{g_2}$) is correct. We now compute the value of t in terms of δ needed to make $p = 1-\mathsf{negl}(\lambda)$. Let X_i be the indicator random variable such that $X_i = 1$ with probability $\frac{1}{2}+\delta$ and $X_i = 0$ with probability $\frac{1}{2}-\delta$. By the Chernoff bound,

$$\Pr[\mathsf{majority}(X_1,\ldots,X_t) = 1] \geq 1 - e^{-\frac{1}{1+2\delta}t\delta^2}$$

If we set $t = \frac{c\lambda \ln 2}{\delta^2}$ with a small constant $c = (1+2\delta)$ then we get that

$$\Pr[\mathsf{majority}(X_1,\ldots,X_t) = 1] \geq 1 - e^{-\frac{1}{1+2\delta}\frac{\lambda(1+2\delta)\ln 2}{\delta^2}\delta^2} = 1 - \frac{1}{2^\lambda}.$$

Recall that this is the success probability for recovering **one** bit from $\mathcal{A}_{g_1}$ (resp. $\mathcal{A}_{g_2}$). Moreover, note that t is polynomial in λ, which ensures that $\mathcal{A}_f$'s runtime is still polynomial. Plugging this value of t in the our union bound above, we get that with $t = \frac{c\lambda \ln 2}{\delta^2}$ trials, the probability that $\mathcal{A}_f$ succeeds is lower bounded by:

$$p \geq 1 - 2\lambda\left(\frac{1}{2^\lambda}\right) = 1 - \frac{2\lambda}{2^\lambda} = 1 - \mathsf{negl}(\lambda).$$

As such, we've shown that Alice can invert f with overwhelming probability even in the case where $\mathcal{B}$ provided by Bob succeeds in computing the inner product with some small advantage $\frac{1}{2}+\epsilon$ (where $\frac{1}{4} < \epsilon < \frac{1}{2}$). As a side note: it is possible (although quite challenging) to invert f when $\mathcal{B}'s$ advantage is any non-negligible $\epsilon > 0$. This is the core result of Goldreich-Levin, which was presented in lecture.

Problem 4. Random self-reducibility

(a) Recall the Computational Diffie–Hellman (CDH) assumption from lecture.

CDH assumption: Given $(\mathbb{G}, g, g^a, g^b)$ where p is prime, $\mathbb{G}$ is a cyclic group of order $p-1$, and a, b are *random* in $\mathbb{Z}_{p-1}$, it is computationally intractable to compute g^{ab}.

Suppose that Bob has an instance of a Diffie-Hellman tuple $(\mathbb{G}, g, g^x, g^y)$ for some **worst-case** $x, y \in \mathbb{Z}_{p-1}$. Bob has no idea how to compute g^{xy} for these values of x and y, but Alice does have an algorithm $\mathcal{A}$ which on input $(\mathbb{G}, g, g^\alpha, g^\beta)$ for **random** $\alpha, \beta \xleftarrow{R} \mathbb{Z}_{p-1}$, can output $g^{\alpha\beta}$ with non-negligible probability over the sampling of α, β.

Prove that CDH is random self-reducible in $\mathbb{Z}_p^*$. I.e., Bob can use Alice's $\mathcal{A}$ to solve his worst-case instance.

Solution

Note 1: full credit given only for additive rerandomization solution, partial credit for multiplicative solution (which isn't correct in the case where $\mathsf{GCD}(a, p-1) \neq 1$ *or* $\mathsf{GCD}(b, p-1) \neq 1$*)*

Let $\mathcal{A}$ be Alice's algorithm for random instances. Bob constructs $\mathcal{B}$ which uses $\mathcal{A}$ to solve his worst-case instance as follows.

Algorithm $\mathcal{B}(\mathbb{G}, g, g^x, g^y)$

receive as input $(\mathbb{G}, g, g^x, g^y)$

$a, b \xleftarrow{R} \mathbb{Z}_{p-1}$

$g^\alpha \leftarrow g^x g^a, \quad g^\beta \leftarrow g^y g^b$

$g^\gamma \leftarrow \mathcal{A}((\mathbb{G}, g, g^\alpha, g^\beta))$

$g^z \leftarrow g^\gamma g^{-xb} g^{-ya} g^{-ab}$

return g^z

Analysis: First, observe that $\mathcal{B}$ runs in polynomial time over the runtime of $\mathcal{A}$ given that it only calls $\mathcal{A}$ once (we do not have any requirements on the runtime of $\mathcal{A}$; just that the *reduction* is polynomial time). Next, observe that g^α and g^β as computed by $\mathcal{B}$ are uniformly random elements of $\mathbb{G}$ (given that a and b are uniformly random) and hence match the distribution expected by $\mathcal{A}$. Next, it holds that if $\mathcal{A}$ succeeds, then so does $\mathcal{B}$. Specifically,

$$\begin{aligned} g^\gamma &= g^{(x+a)(y+b)} g^{-xb} g^{-ya} g^{-ab} \\ &= g^{xy+xb+ya+ab} g^{-xb-ya-ab} \\ &= g^{xy} \end{aligned}$$

which is exactly the output required. Finally, it is important to note that $\mathcal{B}$ can easily compute $g^{xb} = (g^x)^b$ and $g^{ya} = (g^y)^a$ without knowing x, y. As such, $\mathcal{B}$ succeeds with the same probability as $\mathcal{A}$ when given a uniformly random input. This proves that CDH is random self-reducible.

(b) Consider the following variant of the CDH assumption (sometimes referred to as the n-CDH assumption).

n**-CDH assumption:** Given n-CDH tuple $(\mathbb{G}, g, g^a, g^{a^2}, \ldots, g^{a^{n-1}})$ where p is prime, $\mathbb{G}$ is a cyclic group of order p, and a is *random* in $\mathbb{Z}_{p-1}$, it is computationally intractable to compute g^{a^n}.

Alice claims that n-CDH is not random self-reducible. Bob, however, believes that it is.

Who is correct? Is the n-CDH assumption random self-reducible? Prove your answer.

Solution

n-CDH is random self-reducible.

First consider $n = 3$. Let $\mathcal{A}$ be an algorithm solving the 3-CDH problem on **random** instances. We will construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve a worst-case instance of 3-CDH.

Algorithm $\mathcal{B}(\mathbb{G}, g, g^a, g^{a^2})$

$x \xleftarrow{R} \mathbb{Z}_{p-1}$

$g^b \leftarrow g^a g^x, \quad g^{b^2} \leftarrow g^{a^2} g^{2ax} g^{x^2}$

$g^\beta \leftarrow \mathcal{A}((\mathbb{G}, g, g^b, g^{b^2}))$

$g^\alpha \leftarrow g^\beta g^{-3xa^2} g^{-3ax^2} g^{-x^3}$

return g^α

Analysis: $\mathcal{B}$ clearly runs in polynomial time over the runtime of $\mathcal{A}$ as it only invokes $\mathcal{A}$ once (again, we do not have any requirements on the runtime of $\mathcal{A}$). Moreover, the input given to $\mathcal{A}$ is distributed identically to the expected input given that: $g^b = g^{a+x}$ is uniformly random when x is random and it holds that

$$g^{b^2} = g^{(a+x)^2} = g^{a^2+2ax+x^2} = g^{a^2} g^{2ax} g^{x^2},$$

which is also randomly distributed due to x. Hence, $\mathcal{A}$ gets as input a uniformly random instance of n-CDH.

If $\mathcal{A}$ succeeds then $g^\beta = g^{(a+x)^3}$, with some algebra, we see that

$$\begin{aligned} g^\alpha &= g^\beta g^{-3xa^2} g^{-3ax^2} g^{-x^3} \\ &= g^{(a+x)^3} g^{-3xa^2} g^{-3ax^2} g^{-x^3} \\ &= g^{a^3+3a^2x+3ax^2+x^3} g^{-3xa^2} g^{-3ax^2} g^{-x^3} \\ &= g^{a^3} \cancel{g^{3a^2x} g^{-3a^2x}} \cancel{g^{3ax^2} g^{-3ax^2}} \cancel{g^{x^3} g^{-x^3}} \\ &= g^{a^3}, \end{aligned}$$

which is exactly the output required by $\mathcal{B}$. Moreover, observe that $\mathcal{B}$ can efficiently compute $g^{-3a^2x} = (g^{a^2})^{-3x}$ and $g^{-3ax^2} = (g^a)^{-3x^2}$ without knowing a. Therefore, $\mathcal{B}$ succeeds whenever $\mathcal{A}$ succeeds on a uniformly random input. It then follows that 3-CDH is random self-reducible.

Arbitrary $n > 3$. Intuitively, the same reduction idea holds for $n > 3$. That is, $\mathcal{B}$ can always compute a random instances of n-CDH and remove the randomness from the resulting degree-n polynomial output produced by $\mathcal{A}$.

To show this formally, consider: $g^a, g^{a^2}, \ldots, g^{a^{n-1}}$. $\mathcal{B}$'s first task is to compute $g^b, g^{b^2}, \ldots, g^{b^{n-1}}$ such that b is **random**. We will now show that any $g^{(a+x)^i}$ is efficiently computable given only $g^{a^i}, g^{a^{i-1}}, \ldots, g^a$ and then substituting $b = (a + x)$ will result in a uniformly random n-CDH instance given that x (and hence b) is random.

Formally, we need to show two things.

1. Given g^{a^i} for all $0 \le i < n$ we can compute $g^{(a+x)^i}$ for all $0 \le i < n$. This will allow us to compute a random instance of the n-CDH problem by choosing a random x.
2. Given $g^{(a+x)^n}$ and g^{a^i} for all $0 \le i < n$ for any n we can compute g^{a^n}. This will allow us to solve the original, worst-case, instance of n-CDH.

Solution (continued...)

We first show that given g^{a^i} for all $0 \le i < n$ we can compute $g^{(a+x)^i}$ for all $0 \le i < n$. We prove this by induction. For the base case $i = 0$ this claim follows trivially. For $i > 0$, we observe that

$$g^{(a+x)^i} = g^{\sum_{k=0}^{i} \binom{i}{k} a^k x^{i-k}},$$

by the binomial theorem. We know x and hence can easily compute $x^2, \dots, x^i$. Moreover, each term in the sum is efficiently computable given only g^{a^j} for all $0 \le j \le i$. To see why, observe that $\binom{i}{k}$ is just a constant and

$$g^{\binom{i}{k} a^k x^{i-k}} = (g^{a^k})^{\binom{i}{k} x^{i-k}},$$

doesn't require the discrete log of g^{a^k} in order to compute. The final sum is then easily computed as

$$\prod_{k=0}^{i} (g^{a^k})^{\binom{i}{k} x^{i-k}} = g^{\sum_{k=0}^{i} \binom{i}{k} a^k x^{i-k}} = g^{(a+x)^i},$$

as required. Using the above fact, we conclude that $\mathcal{B}$ can always compute a fresh random instance of n-CDH given only $g^a, \dots, g^{a^{n-1}}$ by evaluating a linear function in the exponent over the a^i's.
Now, we show that given $g^{(a+x)^n}$ it is possible to recover g^{a^n}. Observe that:

$$g^{(a+x)^n} = g^{\sum_{k=0}^{n} \binom{n}{k} a^k x^{n-k}}.$$

Using $g^a, \dots, g^{a^{n-1}}$ we can recover g^{a^n} by computing:

$$\begin{aligned} g^{(a+x)^n} \Big/ \prod_{k=1}^{n-1} (g^{a^k})^{\binom{n}{k} x^{n-k}} &= g^{\sum_{k=0}^{n} \binom{n}{k} a^k x^{n-k}} \Big/ g^{\sum_{k=0}^{n-1} \binom{n}{k} a^k x^{n-k}} \\ &= g^{\sum_{k=0}^{n} \binom{n}{k} a^k x^{n-k} - \sum_{k=0}^{n-1} \binom{n}{k} a^k x^{n-k}} \\ &= g^{\binom{n}{n} a^n} \\ &= g^{a^n}. \end{aligned}$$

Therefore, $\mathcal{B}$ can always "correct" the output (by removing all the extra terms) and recover the answer g^{a^n} for its original instance. We thus conclude that n-CDH is random self-reducible.

Problem 5. Designatable PRFs

Recall the definition of pseudorandom functions presented in lecture. We have also seen some constructions of PRFs, from other primitives like PRGs. In this problem we will consider a variant of PRFs.

We define a **designatable PRF** to be a PRF family $\mathcal{F} = \{f_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$ for $n = n(\lambda)$ equipped with two special PPT algorithms $\mathsf{Designate} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ and

$\mathsf{Execute} : \{0,1\}^n \times \{0,1\}^{n-m} \to \{0,1\}^n$ such that for any $k \in \{0,1\}^n, y \in \{0,1\}^m, z \in \{0,1\}^{n-m}$,

$$\mathsf{Execute}\Big(\mathsf{Designate}(k, y), z\Big) = f_k(y||z).$$

In other words, Designate takes in the key k and some "prefix" y, and outputs a designated key k_y. Given the designated key k_y, Execute can compute $f_k(x)$ for any $x \in \{0,1\}^n$ that has the prefix y (i.e. $x = y||z$ for some $z \in \{0,1\}^{n-m}$).

Additionally, we require that any PPT algorithm $\mathcal{A}$ given a designated key k_y for prefix y, can *only* compute $f_k(x)$ with non-negligible probability for x with the prefix y. That is, for any $y, y' \in \{0,1\}^m, z \in \{0,1\}^{n-m}$ with $y \neq y'$, we have

$$\Pr[k_y \leftarrow \mathsf{Designate}(k, y) : \mathcal{A}(k_y, y'||z) = f_k(y'||z)] \leq \mathsf{negl}(\lambda).$$

Prove that if PRFs exist, so do designatable PRFs.

Solution

First, we show that the existence of PRFs implies that PRGs exist, and then we construct a designatable PRF given a PRG.

Let $\mathcal{F}_0 = \{f_s : \{0,1\}^n \to \{0,1\}^n\}_{s \in \{0,1\}^n}$ be a PRF family for $n = n(\lambda)$. Define $G : \{0,1\}^n \to \{0,1\}^{2n}$ such that $G(s) = f_s(0^n)||f_s(1^n)$. Then we claim that G is a PRG which doubles the input length. For contradiction, suppose that G is not a PRG, so there is a polynomial time adversary $\mathcal{A}_0$ that distinguishes between G on a random s and a truly random string of length $2n$, with non-negligible advantage. Then we can construct a distinguisher $\mathcal{D}$ which breaks the security of the PRF family $\mathcal{F}_0$. Let $\mathcal{O}$ be an oracle to either the a truly random function or a function $f_s \in \mathcal{F}_0$ for a random s.

Algorithm $\mathcal{D}^{\mathcal{O}}$

$y_0 \leftarrow \mathcal{O}(0^n)$

$y_1 \leftarrow \mathcal{O}(1^n)$

return $\mathcal{A}_0(y_0||y_1)$

Note that for a random oracle $\mathcal{O}$, $y_0||y_1$ is truly random, but if $\mathcal{O}$ is an oracle to the PRF $f_s \in \mathcal{F}_0$ then $y_0||y_1 = G(s)$. Then it is clear that $\mathcal{D}$ also has the same non-negligible advantage as $\mathcal{A}_0$, in distinguishing between the two cases: $\mathcal{O}$ is a random oracle or it is an oracle to f_s.

Given such a PRG, we now construct a designatable PRF family. Let G be a PRG with expansion factor $\ell(n) = 2n$. Let F be the GGM PRF as defined in class. We already know that F is a PRF. Define

$$\mathsf{Designate}(k, y) = G_{y_m}(G_{y_{m-1}}(\cdots(G_{y_1}(k))))$$

and

$$\mathsf{Execute}(k_y, z) = G_{z_{n-m}}(\cdots(G_{z_1}(k_y))).$$

By construction $\mathsf{Designate}$ and $\mathsf{Execute}$ clearly satisfy the requirement that

$$\mathsf{Execute}(\mathsf{Designate}(k, y), z) = F_k(y||z).$$

Suppose towards contradiction that there exists some PPT algorithm $\mathcal{A}$ such that for some $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ with $x \neq y||z \ \forall \ z \in \{0,1\}^{n-m}$, we have

$$\Pr_{k \xleftarrow{R} \{0,1\}^n}[k_y \leftarrow \mathsf{Designate}(k, y); w \leftarrow \mathcal{A}(k_y, x) : w = F_k(x)] \geq \mathsf{nonnegl}(\lambda).$$

We will construct a distinguisher $\mathcal{B}$ which breaks the security of $\mathsf{Designate}$ which, like F, is a PRF by the argument from class, as follows. Let $\mathcal{O}$ be the PRF challenger; where $\mathcal{O}$ is either an oracle to a truly random function or the output of $\mathsf{Designate}$.

Algorithm $\mathcal{B}^{\mathcal{O}}$

$k_y \leftarrow \mathcal{O}(y)$

$w \leftarrow \mathcal{A}(k_y, x)$

$k_x \leftarrow \mathcal{O}(x_{[1:m]})$

if $w = \mathsf{Execute}(k_x, x_{[m+1:n]})$: **return** 1

else return 0

Then if we use $\mathbb{F}_n$ to denote the set of all functions $\{0,1\}^n \to \{0,1\}^n$, we have

$$\Pr_{k \xleftarrow{R} \{0,1\}^n}[\mathcal{B}^{\mathsf{Designate}(k,\cdot)}(1^\lambda) = 1] = \Pr_{k \xleftarrow{R} \{0,1\}^n}[k_y \leftarrow \mathsf{Designate}(k, y); w \leftarrow \mathcal{A}(k_y, x) : w = F_k(x)] \geq \mathsf{nonnegl}(\lambda)$$

and

$$\Pr_{R \xleftarrow{R} \mathbb{F}_n}[\mathcal{B}^R(1^\lambda) = 1] = \Pr_{k_y, k_x \xleftarrow{R} \{0,1\}^n}[w \leftarrow \mathcal{A}(k_y, x) : w = \mathsf{Execute}(k_x, x_{[m+1:n]})] \leq \mathsf{negl}(\lambda).$$

Solution (continued...)

This implies that

$$\left| \Pr_{k \xleftarrow{R} \{0,1\}^n} [\mathcal{B}^{\mathsf{Designate}(k,\cdot)}(1^\lambda) = 1] - \Pr_{R \xleftarrow{R} \mathbb{F}_n} [\mathcal{B}^{R}(1^\lambda) = 1] \right| \geq \mathsf{nonnegl}(\lambda)$$

which is a contradiction, since $\mathsf{Designate}(k, \cdot)$ is a PRF. Thus such an $\mathcal{A}$ cannot exist.