

Fundamentals of Cryptography: Problem Set 6

Due Wednesday Oct 19, 3PM

Collaboration is permitted (and encouraged); however, you must write up your own solutions and acknowledge your collaborators.

If a problem has **0pt**, it will not be graded.

Problem 0 Read Section 9 (Number Theory and Cryptographic Hardness Assumption) and 12 (Public-Key Encryption) of “Introduction to Modern Cryptography” by Katz & Lindell or Section 10, 11 of “A Graduate Course in Applied Cryptography” by Boneh & Shoup.

Problem 1 (5pt) Show that 5-round Feistel is not an indifferentiable construction of random permutation based on random oracles. The definition of *indifferentiability* can be found in Section 8.10.3 Random oracles: safe modes of operation of “A Graduate Course in Applied Cryptography”.

Given t random oracles $\mathcal{O}_1, \dots, \mathcal{O}_t : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$, the t -round Feistel network is a permutation on $\{0, 1\}^{2\lambda}$, defined as:

Feistel $^{\mathcal{O}_1, \dots, \mathcal{O}_t}(x_0, x_1)$, takes (x_0, x_1) as the input.

For each $i = 1, 2, \dots, t$:

Set $x_{i+1} \leftarrow x_{i-1} \oplus \mathcal{O}_i(x_i)$.

Output (x_t, x_{t+1}) as the output.

In the class, we discussed how 4-round Feistel is not an indifferentiable construction of random permutation. Here, you are going to extend the negative statement to 5-round. That is, you need to construct an efficient adversary $\mathcal{A}$ such that for any efficient simulator $\mathcal{S}$,

$$\Pr_{\mathcal{O}_1, \dots, \mathcal{O}_5 : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda} [\mathcal{A}^{\mathcal{O}_1, \dots, \mathcal{O}_5, \text{Feistel}^{\mathcal{O}_1, \dots, \mathcal{O}_5} \text{ and its inverse}}(1^\lambda) \rightarrow 1] \\ - \Pr_{\text{permutation } P : \{0, 1\}^{2\lambda} \rightarrow \{0, 1\}^{2\lambda}} [\mathcal{A}^{\mathcal{S}^{P, P^{-1}}, P, P^{-1}}(1^\lambda) \rightarrow 1]$$

is non-negligible.

Notation: If the round number in the subscript looks annoying to you, you can use A, B, C, D, E, F, G instead of $x_0, x_1, \dots, x_6$.

Problem 2 (6pt) Unsafe Groups As we mentioned in the class, the hardness of Diffie-Hellman depends on the group generation algorithm. Let **Gen** be a group generation algorithm. $\text{Gen}(1^\lambda)$ samples a cyclic group G together with a generator g . Technically, the generation algorithm outputs the description of G – including efficient algorithms for multiplication, inversion. For efficiency, the size of G is at most $2^{\text{poly}(\lambda)}$ (otherwise a group element can not be represented by $\text{poly}(\lambda)$ bits).

Part A. Show that, if all prime factors of $|G|$ are small, then computational Diffie-Hellman is easy. More precisely, show that computational Diffie-Hellman problem for group generated by **Gen** is easy if, for any (G, g) sampled by $\mathbf{Gen}(1^\lambda)$, $|G|$ can be efficiently computed and all prime factors of $|G|$ are no more than $\text{poly}(\lambda)$.

Part B. Show that, if some prime factors of $|G|$ is small, then decisional Diffie-Hellman is easy. More precisely, show that decisional Diffie-Hellman problem for group generated by **Gen** is easy if, for any (G, g) sampled by $\mathbf{Gen}(1^\lambda)$, $|G|$ can be efficiently computed and at least one prime factors of $|G|$ is no more than $\text{poly}(\lambda)$.

Problem 3 (4pt) Matrix Diffie-Hellman Let **Gen** be a cyclic group generation algorithm. The DDH assumption (parameterized by **Gen**) says that, if $(G, g) \leftarrow \mathbf{Gen}(1^\lambda)$

$$(G, g, g^a, g^b, g^{ab}) \approx_c (G, g, g^a, g^b, g^c)$$

where a, b, c are independently and uniformly sampled from $\mathbb{Z}_{|G|} = \{0, 1, \dots, |G| - 1\}$. Show that DDH assumption implies the *matrix DDH assumption*, that is, for $h, w = \text{poly}(\lambda)$,

$$(G, g, g^{a_1}, \dots, g^{a_h}, g^{b_1}, \dots, g^{b_w}, g^{a_1 b_1}, \dots, g^{a_h b_w}) \approx_c (G, g, g^{a_1}, \dots, g^{a_h}, g^{b_1}, \dots, g^{b_w}, g^{c_{1,1}}, \dots, g^{c_{h,w}})$$

where $a_1, \dots, a_h, b_1, \dots, b_w, c_{1,1}, \dots, c_{h,w}$ are independently and uniformly sampled from $\{1, \dots, |G|\}$. This assumption is called matrix DDH, because it can be written as the following matrix form

$$\left(G, g, g \begin{bmatrix} a_1 \\ \vdots \\ a_h \end{bmatrix}, g \begin{bmatrix} b_1 \\ \vdots \\ b_w \end{bmatrix}, g \begin{bmatrix} a_1 b_1 & \cdots & a_1 b_w \\ \vdots & \ddots & \vdots \\ a_h b_1 & \cdots & a_h b_w \end{bmatrix} \right) \approx_c \left(G, g, g \begin{bmatrix} a_1 \\ \vdots \\ a_h \end{bmatrix}, g \begin{bmatrix} b_1 \\ \vdots \\ b_w \end{bmatrix}, g \begin{bmatrix} c_{1,1} & \cdots & c_{1,w} \\ \vdots & \ddots & \vdots \\ c_{h,1} & \cdots & c_{h,w} \end{bmatrix} \right).$$

Here we use the conventional notation

$$g \begin{bmatrix} c_{1,1} & \cdots & c_{1,w} \\ \vdots & \ddots & \vdots \\ c_{h,1} & \cdots & c_{h,w} \end{bmatrix} := \begin{bmatrix} g^{c_{1,1}} & \cdots & g^{c_{1,w}} \\ \vdots & \ddots & \vdots \\ g^{c_{h,1}} & \cdots & g^{c_{h,w}} \end{bmatrix}.$$

Problem 4 (8pt) Play with Diffie-Hellman Let **Gen** be a cyclic group generation algorithm. For any (G, g) sampled by $\mathbf{Gen}(1^\lambda)$, $|G|$ is a prime and can be efficiently computed.

Part A. Worst-case to average case reduction. Assume there is a p.p.t. adversary $\mathcal{A}$ such that for any (G, g) sampled by $\mathbf{Gen}(1^\lambda)$,

$$\Pr_{x, y \leftarrow \mathbb{Z}_{|G|}} \left[\mathcal{A}(G, g, g^x, g^y) = g^{xy} \right] \geq \frac{1}{\text{poly}(\lambda)}.$$

That is, $\mathcal{A}$ solves CDH in the average case. Construct another p.p.t. adversary $\mathcal{B}$ such that for any (G, g) sampled by $\mathbf{Gen}(1^\lambda)$, for any $x, y \in \mathbb{Z}_{|G|}$,

$$\Pr \left[\mathcal{B}(G, g, g^x, g^y) = g^{xy} \right] \geq 99\%.$$

That is, $\mathcal{B}$ should solve CDH with good probability in the worst case.

Part B. Show the equivalence between the following variants of CDH assumption. In all the variants, (G, g) are sampled by $\text{Gen}(1^\lambda)$, and x, y are random in $\mathbb{Z}_{|G|}$.

CDH assumption	given G, g, g^x, g^y	hard to find g^{xy}
“square” CDH assumption	given G, g, g^x	hard to find g^{x^2}
“inverse” CDH assumption	given G, g, g^x	hard to find $g^{x^{-1}}$
“division” CDH assumption	given G, g, g^x, g^y	hard to find $g^{x/y}$

To show problem A is at least as hard as problem B, the proof should be a reduction that assumes an adversary $\mathcal{A}$ can solve problem A with non-negligible probability, and constructs another adversary $\mathcal{B}$ solving problem B with non-negligible probability.

For some of the reductions, you may additionally assume that $\text{Gen}(1^\lambda)$ always samples g as a random generator in G .

Problem 5 (8pt) The strong RSA assumption says that, for any p.p.t. adversary $\mathcal{A}$,

$$\Pr\left[x^e = y \text{ and } e \geq 3 \text{ is an odd integer} : \mathcal{A}(N, y) \rightarrow (x, e)\right] \leq \text{negl}(\lambda),$$

where $N = pq$ and p, q are two random λ -bit safe primes, and y is sampled from $\mathbb{Z}_N^*$.

Consider the following keyed function:

- $\text{Gen}(1^\lambda)$ generates the key as follows. Sample two random λ -bit safe primes p, q , set $N = pq$, sample $s \leftarrow \mathbb{Z}_N^*$, and output key $k = (N, p, q, s)$.
- Function $f(k, i)$ takes as inputs a key k , and $i \in \{1, \dots, m\}$. The output of $f(k, i)$ is $m = \text{poly}(\lambda)$ bits long, and is defined as

$$f(k, i) = s^{1/e_i} \pmod{N},$$

where e_i is the i -th odd prime. In other words, $f(k, i) = x$ such that $x^{e_i} = s$.

Part A. Show that $f(k, i)$ can be computed by a poly-time algorithm.

Part B. Given a key $k = (N, p, q, s)$, for any set $S \subseteq \{1, \dots, m\}$, define k_S as

$$k_S = (N, s^{1/\prod_{i \in S} e_i} \pmod{N}).$$

Show that, there is poly-time algorithm Eval , such that for any $S \subseteq \{1, \dots, m\}$, for any $i \in S$

$$\text{Eval}(k_S, S, i) = f(k, i).$$

Part C. Show that, under the strong RSA assumption, any p.p.t. adversary wins the following game with negligible probability

- Generate $k \leftarrow \text{Gen}(1^\lambda)$.
- The adversary chooses sets $S_1, S_2, \dots \subseteq \{1, \dots, m\}$ and receives $k_{S_1}, k_{S_2}, \dots$
- Eventually, the adversary outputs i, y .
- The adversary wins if and only if $f(k, i) = y$ and $i \notin \bigcup_j S_j$.

Remark: By the pigeonhole principle, if $m \gg \lambda$, for any key k , there must exist distinct S, S' such that $k_S = k_{S'}$. Say $i \in S \setminus S'$. In some sense, $f(k, i)$ is revealed by k_S , but “is not revealed by $k_{S'}$ ”.

Remark: We are constructing a “constrained PRF” whose input domain is $\{1, \dots, m\}$, such that it is possible to generate a constrained key for any constraints.

Technically speaking, the keyed function f is not a PRF, because its output is not pseudorandom. (RSA assumption does not imply indistinguishability from uniform.) This gap can be closed by using the Goldreich-Levin hard-core predicate.