Problem 1.

Denote by A, B, C, D, E, F, G the intermediate values.

The distinguisher arbitrarily picks C, D and $\Delta \neq 0$, computes E, Δ' from

$$E = C \oplus \mathcal{O}_3(D), \qquad E \oplus \Delta' = C \oplus \mathcal{O}_3(D \oplus \Delta).$$

As the consequence,

$$E \oplus \Delta' = (C \oplus \Delta') \oplus \mathcal{O}_3(D), \qquad E = (C \oplus \Delta') \oplus \mathcal{O}_3(D \oplus \Delta).$$

Therefore, we can consider 4 evaluations of the Feistel network, where the three intermediate values are $C \oplus \alpha\Delta', D \oplus (\alpha \oplus \beta)\Delta, E \oplus \beta\Delta'$ respectively

$$\begin{array}{lllllll}
A_{0,0} & B_{0,0} & C & D & E & F_{0,0} & G_{0,0} \\
A_{0,1} & B_{0,1} & C & D \oplus \Delta & E \oplus \Delta' & F_{0,1} & G_{0,1} \\
A_{1,0} & B_{1,0} & C \oplus \Delta' & D \oplus \Delta & E & F_{1,0} & G_{1,0} \\
A_{1,1} & B_{1,1} & C \oplus \Delta' & D & E \oplus \Delta' & F_{1,1} & G_{1,1}
\end{array}$$

Let $(A_{\alpha,\beta}, B_{\alpha,\beta})$ and $(F_{\alpha,\beta}, G_{\alpha,\beta})$ denote the corresponding input and output for each $\alpha, \beta \in \{0, 1\}$. It always holds that

$$\bigoplus_{\alpha,\beta\in\{0,1\}} B_{\alpha,\beta} = 0, \qquad \bigoplus_{\alpha,\beta\in\{0,1\}} F_{\alpha,\beta} = 0.$$

But in the random permutation (RP) model, it is hard to find such four input/output pairs. Thus it is impossible to construct an efficient simulator in the ideal model.

Problem 2.

Part A. We construct an algorithm to compute g^{xy} given $\{g^x, g^y\}$ and the discription of G. First, compute $|G|$ and give its standard factorization, i.e. $|G| = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$. Becasuse g is the generator of G, thus it suffices to compute $xy \pmod{|G|}$. Further, by CRT, it suffices for us to compute $xy\ (\bmod\ p_i^{\alpha_i})$ for every $i \in \{1, \dots k\}$ (by $|G| \leq 2^{\mathrm{poly}(\lambda)}$ we know $k \leq \mathrm{poly}(\lambda)$). Suppose that

$$
\begin{aligned}
x \pmod{p_i^{\alpha_i}} &= x_0 + y_1 p_i + \cdots + x_{\alpha_i - 1} p_i^{\alpha_i - 1}, \\
y \pmod{p_i^{\alpha_i}} &= y_0 + y_1 p_i + \cdots + y_{\alpha_i - 1} p_i^{\alpha_i - 1},
\end{aligned}
$$

where

$$
x_0, x_1, \dots, x_{\alpha_i - 1}, y_0, y_1, \dots, y_{\alpha_i - 1} \in \mathbb{Z}_{p_i}.
$$

Now consider the group generated by $g^{p_1^{\alpha_1 - 1} \cdots p_k^{\alpha_k}}$. It is a subgroup of G of order p_1, and $(g^x)^{p_1^{\alpha_1 - 1} \cdots p_k^{\alpha_k}} = (g^{p_1^{\alpha_1 - 1} \cdots p_k^{\alpha_k}})^{x_0}$ lies in this subgroup. Given that $p_i \leq \mathrm{poly}(\lambda)$, we can simply enumertate $0, 1, \dots, p_i - 1$ to find x_0. Now we can get $g^{x - x_0}$, and $(g^{x - x_0})^{p_1^{\alpha_1 - 2} \cdots p_k^{\alpha_k}} = (g^{p_1^{\alpha_1 - 2} \cdots p_k^{\alpha_k}})^{x_1}$ lies in a subgroup of order p_i. Similarly by enumerating we get x_1. Keep doing this we can get all $x_0, \dots, x_{\alpha_i - 1}$ and thus get $x \pmod{p_i^{\alpha_i}}$. Similarly we can get $y \pmod{p_i^{\alpha_i}}$. By simply multiplying them we get $xy \pmod{p_i^{\alpha_i}}$. In this way we break the computational DH assumption.

Part B. Assume that $|G| = p^\alpha q$, where $p \leq \mathrm{poly}(\lambda)$. We have shown in part A that given g^x and g^y, $xy \pmod{p^\alpha}$ can be computed in polynomial time. Note that for a random $z \xleftarrow{\$} \mathbb{Z}_{|G|}$, $z \pmod{p^\alpha}$ is also uniformly random in $\mathbb{Z}_{p^\alpha}$.

Now, for an adversary equiped with the description of group G and $\{g^x, g^y, g^z\}$, where z is either uniformly random in $\mathbb{Z}_{|G|}$ or $z = xy$. It can simply compute $z \pmod{p^\alpha}$ and $xy \pmod{p^\alpha}$. If they are identical then output 0 (means $z \xleftarrow{\$} \mathbb{Z}_{|G|}$), else ouput 1 (means $z = xy$). This brings out an advantage of $1 - \frac{1}{p^\alpha}$.

Problem 3.

Start with the vector DDH assumption.

$$\left(g, g^a, g^{b_1}, \ldots, g^{b_w}, g^{ab_1}, \ldots, g^{ab_w}\right) \approx_c \left(g, g^a, g^{b_1}, \ldots, g^{b_w}, g^{c_1}, \ldots, g^{c_w}\right).$$

Intuitively, the vector DDH assumption is implied by the DDH assumption because

$$\begin{aligned}
&\left(g, g^a, g^{b_1}, \ldots, g^{b_w}, g^{ab_1}, \ldots, g^{ab_w}\right) \\
\approx_c &\left(g, g^a, g^{b_1}, \ldots, g^{b_w}, g^{c_1}, g^{ab_2}, \ldots, g^{ab_w}\right) \\
&\vdots \\
\approx_c &\left(g, g^a, g^{b_1}, \ldots, g^{b_w}, g^{c_1}, \cdots, g^{c_j}, g^{ab_{j+1}}, \ldots, g^{ab_w}\right) \\
&\vdots \\
\approx_c &\left(g, g^a, g^{b_1}, \ldots, g^{b_w}, g^{c_1}, \ldots, g^{c_w}\right).
\end{aligned}$$

And the vector DDH assumption implies the matrix DDH assumption because

$$\left(g, g^{\begin{bmatrix} a_1 \\ \vdots \\ a_h \end{bmatrix}}, g^{\begin{bmatrix} b_1 \\ \vdots \\ b_w \end{bmatrix}}, g^{\begin{bmatrix} a_1b_1 & \cdots & a_1b_w \\ \vdots & \ddots & \vdots \\ a_hb_1 & \ldots & a_hb_w \end{bmatrix}}\right) \approx_c \left(g, g^{\begin{bmatrix} a_1 \\ \vdots \\ a_h \end{bmatrix}}, g^{\begin{bmatrix} b_1 \\ \vdots \\ b_w \end{bmatrix}}, g^{\begin{bmatrix} c_{1,1} & \cdots & c_{1,w} \\ a_2b_1 & \cdots & a_2b_w \\ \vdots & \ddots & \vdots \\ a_hb_1 & \ldots & a_hb_w \end{bmatrix}}\right) \approx_c \cdots$$

$$\approx_c \left(g, g^{\begin{bmatrix} a_1 \\ \vdots \\ a_h \end{bmatrix}}, g^{\begin{bmatrix} b_1 \\ \vdots \\ b_w \end{bmatrix}}, g^{\begin{bmatrix} c_{1,1} & \cdots & c_{1,w} \\ \vdots & \ddots & \vdots \\ c_{i,1} & \cdots & c_{i,w} \\ a_{i+1}b_1 & \cdots & a_{i+1}b_w \\ \vdots & \ddots & \vdots \\ a_hb_1 & \ldots & a_hb_w \end{bmatrix}}\right) \approx_c \left(g, g^{\begin{bmatrix} a_1 \\ \vdots \\ a_h \end{bmatrix}}, g^{\begin{bmatrix} b_1 \\ \vdots \\ b_w \end{bmatrix}}, g^{\begin{bmatrix} c_{1,1} & \cdots & c_{1,w} \\ \vdots & \ddots & \vdots \\ c_{h,1} & \cdots & c_{h,w} \end{bmatrix}}\right).$$

This intuition can be formalized as follows.

Assume $\mathcal{D}_V$ is a distinguisher for the vector DDH problem. Construct a distinguisher $\mathcal{D}$ for the DDH problem.

> $\mathcal{D}(g, x, y, z)$
>
> Sample random $j^* \in \{1, \ldots, w\}$.
>
> For each $j \in \{1, \ldots, w\} \setminus \{j^*\}$, sample $b_j \leftarrow \mathbb{Z}_q$
>
> For each $j \in \{1, \ldots, j^* - 1\}$, sample $c_j \leftarrow \mathbb{Z}_q$
>
> Run $\mathcal{D}_V(g, x, g^{b_1}, \ldots, g^{b_{j^*-1}}, y, g^{b_{j^*+1}}, \ldots, g^{b_w}, g^{c_1}, \ldots, g^{c_{j^*-1}}, z, x^{b_{j^*+1}}, \ldots, x^{b_w})$ and output what $\mathcal{D}_V$ outputs.

The distinguisher $\mathcal{D}$ formalizes the hybrid argument. In particular, for random a, b, c, the execution of $\mathcal{D}(g, g^a, g^b, g^{ab})$ conditioning on $j^* = j$ is exactly the same as the execu-

Problem 3 continued on next page...

tion of $\mathcal{D}(g, g^a, g^b, g^c)$ conditioning on $j^* = j - 1$. Thus

$$\begin{aligned}
&\Pr[\mathcal{D}(g, g^a, g^b, g^{ab})] - \Pr[\mathcal{D}(g, g^a, g^b, g^c)] \\
&= \frac{1}{w}\sum_{j=1}^{w} \Pr\Big[\mathcal{D}(g, g^a, g^b, g^{ab}) \,\Big|\, j^* = j\Big] - \frac{1}{w}\sum_{j=1}^{w} \Pr\Big[\mathcal{D}(g, g^a, g^b, g^c) \,\Big|\, j^* = j\Big] \\
&= \frac{1}{w}\bigg(\Pr\Big[\mathcal{D}(g, g^a, g^b, g^{ab}) \,\Big|\, j^* = 1\Big] - \Pr\Big[\mathcal{D}(g, g^a, g^b, g^c) \,\Big|\, j^* = w\Big]\bigg) \\
&= \frac{1}{w}\bigg(\Pr\big[\mathcal{D}_V(g^{b_1}, \ldots, g^{b_w}, g^{ab_1}, \ldots, g^{ab_w})\big] - \Pr\big[\mathcal{D}_V(g^{b_1}, \ldots, g^{b_w}, g^{c_1}, \ldots, g^{c_w})\big]\bigg).
\end{aligned}$$

So the DDH assumption implies the vector DDH assumption.

Similarly, assume $\mathcal{D}_M$ is a distinguisher for the matrix DDH problem. We can construct a distinguisher $\mathcal{D}_V$ for the vector DDH problem as

$\mathcal{D}_V(g, x, y_1, \ldots, y_w, z_1, \ldots, z_w)$

Sample random $i^* \in \{1, \ldots, h\}$.

For each $i \in \{1, \ldots, h\} \setminus \{i^*\}$, sample $a_i \leftarrow \mathbb{Z}_q$

For each $i \in \{1, \ldots, i^* - 1\}$, sample $c_{i,1}, \ldots, c_{i,w} \leftarrow \mathbb{Z}_q$

Run $\mathcal{D}_M\left(g, \begin{bmatrix} g^{a_1} \\ \vdots \\ g^{a_{i^*-1}} \\ x \\ g^{a_{i^*+1}} \\ \vdots \\ g^{a_h} \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_w \end{bmatrix}, \begin{bmatrix} g^{c_{1,1}} & \cdots & g^{c_{1,w}} \\ \vdots & \ddots & \vdots \\ g^{c_{i^*-1,1}} & \cdots & g^{c_{i^*-1,w}} \\ z_1 & \cdots & z_w \\ y_1^{a_{i^*+1}} & \cdots & y_w^{a_{i^*+1}} \\ \vdots & \ddots & \vdots \\ y_1^{a_h} & \cdots & y_w^{a_h} \end{bmatrix}\right)$ and output what $\mathcal{D}_M$ outputs.

The same hybrid argument shows that the advantage of $\mathcal{D}_V$ solving the vector DDH problem (comparing to the random guess) is $\frac{1}{h}$ of the advantage of $\mathcal{D}_M$ solving the matrix DDH problem. So the vector DDH assumption implies the matrix DDH assumption.

Problem 4.

Part A. Construct $\tilde{\mathcal{B}}$ as follows:

- On input (G, g, g^x, g^y), sample $a, c \xleftarrow{\$} \mathbb{Z}^*_{|G|}, b, d \xleftarrow{\$} \mathbb{Z}_{|G|}$ uniformly.
- Call $\mathcal{A}(G, g, (g^x)^a \cdot g^b, (g^y)^c \cdot g^d)$ and obtain $\mathcal{A}$'s output h.
- Return $\left(\frac{h}{g^{xad} \cdot g^{ycb} \cdot g^{bd}}\right)^{1/ac}$.

Since $ax + b, cy + d$ are uniform in $\mathbb{Z}_{|G|} \times \mathbb{Z}_{|G|}$, $h = g^{(ax+b)(cy+d)}$ holds and $\tilde{\mathcal{B}}$ returns g^{xy} with probability at least $1/\operatorname{poly}(\lambda)$.

To amplify the success probability to 99%, notice that when $h \neq g^{(ax+b)(cy+d)}$, the output of $\tilde{\mathcal{B}}$ is uniformly random in G. Hence $\mathcal{B}$ need only run $\operatorname{poly}(\lambda)$ many $\tilde{\mathcal{B}}$'s independently then take the majority.

Part B.

CDH $\Rightarrow$ Square CDH Assume adversary $\mathcal{A}$ breaks the CDH assumption, square CDH (G, g, g^x) can be solved by

1. Sample $r \xleftarrow{\$} \mathbb{Z}_{|G|}$.
2. Query $h \leftarrow \mathcal{A}(G, g, g^x, g^x \cdot g^r)$.
3. Return $h/(g^x)^r$.

Square CDH $\Leftrightarrow$ Inverse CDH For adversary $\mathcal{A}$ against square CDH, an adversary taking as input (G, g, g^x) need only output $\mathcal{A}(G, g^x, g)$ to break inverse CDH. Similarly if $\mathcal{A}$ breaks inverse CDH, $\mathcal{A}(G, g^x, g)$ will output result to square CDH (G, g, g^x) with non-negligible probability.

Square + Inverse CDH $\Rightarrow$ Division CDH If $\mathcal{A}_1, \mathcal{A}_2$ breaks square CDH and inverse CDH respectively, one can construct adversary breaking division CDH on input (G, g, g^x, g^y) by

1. $h_0 \leftarrow \mathcal{A}_2(G, g, g^y)$.
2. $h_1 \leftarrow \mathcal{A}_1(G, g, h_0 \cdot g^x), h_2 \leftarrow \mathcal{A}_1(G, g, h_0), h_3 \leftarrow \mathcal{A}_1(G, g, g^x)$.
3. Return $\left(\frac{h_1}{h_2 h_3}\right)^{(|G|+1)/2}$.

With high probability, $h_0 = g^{y^{-1}}, h_1 = g^{x^2 + 2x/y + y^{-2}}$.

Division CDH $\Rightarrow$ CDH On input (G, g, g^x, g^y), call division CDH adversary $\mathcal{A}$ twice to obtain $h_0 \leftarrow \mathcal{A}(G, g, g^x, g^y)$ and $h_1 \leftarrow \mathcal{A}(G, g^x, (g^x)^r, h_0)$ for $r \xleftarrow{\$} \mathbb{Z}^*_{|G|}$. Output $h_1^{1/r}$ finally.

Problem 5.

Part A. Because $p = 2p' + 1, q = 2q' + 1$ are safe primes, $\varphi(N) = 4p'q'$, for all $i \in [m]$, $\gcd(\varphi(N), e_i) = 1$ holds. Using extended Euclidean algorithm, $a \in \mathbb{Z}_{\varphi(N)}$ can be found in polynomial time such that

$$ae_i = 1 \pmod{\varphi(N)}.$$

Hence we know that if $x^{e_i} = s \pmod N$, then

$$x = x^{ae_i} = s^a \pmod N.$$

This directly yields a polynomial time algorithm. On input (N, p, q, s, i), calculate $f(k, i)$ as follows:

1. Calculate $\varphi(N) = (p-1)(q-1)$.
2. Find $a \leftarrow e_i^{-1} \pmod{\varphi(N)}$ using extended Euclidean algorithm.
3. Calculate $x \leftarrow s^a \pmod N$.
4. Output x.

Part B. Construct Eval as follows. On receiving input $((N, x_S), s, i)$ where $x_S^{\prod_{j\in S} e_j} = s$, Eval includes the following steps:

1. Calculate $b \leftarrow \prod_{j \in S, j \neq i} e_j$.
2. Calculate $x \leftarrow x_S^b \pmod N$.

It's obvious that Eval runs in polynomial time and outputs the correct $f(k, i)$.

Part C. Proof by contradiction. If there exists a PPT adversary $\mathcal{A}$, which wins the given experiment with non-negligible probability $p(n)$, we'll construct a PPT adversary $\mathcal{A}'$, who wins the strong RSA experiment with non-negligible probability, thus contradicts the RSA assumption.

In the construction, $\mathcal{A}'$ calls $\mathcal{A}$ and emulates the security experiment for $\mathcal{A}$. Concretely, given input $(N, y), \mathcal{A}$ involves the following steps:

1. Calculate $e_1, \cdots e_m$.
2. Call $\mathcal{A}$, and receive queries $S_1, \cdots, S_\ell$ from $\mathcal{A}$.
3. Compute $S = \bigcup_{j=1}^{\ell} S_j$.
4. For each query S_j, calculate $r_j = \prod_{t \in (S - S_j)} e_t$, and give y^{r_j} to $\mathcal{A}$ as response.
5. Receive the output (i, x) from $\mathcal{A}$.
6. Find (u, v) such that $ue_i + v \cdot \prod_{t \in S} e_t = 1$ using extended Euclidean algorithm.
7. Calculate $z \leftarrow y^u x^v \pmod N$.

Problem 5 continued on next page...

8. Output (z, e_i).

Because $m = \text{poly}(n), \mathcal{A}'$ runs in polynomial time. Next we show that with non-negligible probability, $\mathcal{A}'$ finds (z, e_i) such that $z^{e_i} = y$.

We first analyze the emulation that $\mathcal{A}'$ gives to $\mathcal{A}$. From the standpoint of $\mathcal{A}$, it is running exactly the same as when the k is $(N, p, q, y\Pi_{t\in S}e_t)$, where y is chosen uniformly in $\mathbb{Z}_N^*$. Therefore, with non-negligible probability $p(n), \mathcal{A}$ outputs (i, x) such that

$$x^{e_i} = y^{\prod_{t\in S} e_t} \pmod N. \tag{1}$$

Because $i \notin S, \gcd\left(e_i, \prod_{t\in S} e_t\right) = 1$, therefore, with Euclidean algorithm, $\mathcal{A}'$ succeeds in finding u, v. Remember that (u, v) is such pair that

$$ue_i + v \cdot \prod_{t\in S} e_t = 1. \tag{2}$$

Combining equation (1) and (2)

$$y = y^{ue_i + v} \prod_{t\in S} e_t = y^{ue_i} x^{ve_i} = \left(y^u x^v\right)^{e_i}. \tag{3}$$

Equation (3) shows that $z^{e_i} = y$. This holds as long as x is correctly given by $\mathcal{A}$. Because $\mathcal{A}$ succeeds with non-negligible probability, $\mathcal{A}'$ also succeeds with non-negligible probability. This finishes our proof.

Remark: One should note that, if the queres in step 2 of the game in part C are queried by the adversary one-by-one, as opposed to all-at-once, the above reduction would not work (because adversary $\mathcal{A}'$ cannot calculate S in step 3 and 4).

To fix this, it only needs to notice that $m = \text{poly}(\lambda)$. Thus adversary $\mathcal{A}'$ can simply guess the index $\mathcal{A}$ will output in step 5 is i_0, and set S to be $\{1, \ldots, m\} - \{i_0\}$. If its guess is right, i.e. $i = i_0$, then $\mathcal{A}'$ can run other step correctly and output z. In this way

$$\Pr[\mathcal{A}' \text{ breaks the strong RSA}] \geq \frac{1}{\text{poly}(\lambda)} \Pr[\mathcal{A} \text{ wins the game in part C}].$$