A Round and Communication Efficient Secure Ranking Protocol

Abstract. In this work, we consider realizing a ranking functionality $(m_1, \cdots, m_n) \mapsto (r_1, \cdots, r_n)$ in the non-adaptive malicious model, where $r_i = 1 + \sharp\{m_j : m_j < m_i\}$. Generically, it has been solved by a general multi-party computation construction (via a circuit formulation). However, such a solution is inefficient in either round complexity or communication complexity. In this work, we propose an efficient construction without a circuit. Our protocol is constant round and efficient in communication complexity. Furthermore, we show it is directly secure in the non-adaptive malicious model (i.e., without a compiler, as is used in many general constructions).

1 Introduction

A general multi-party computation paradigm was initially studied by Yao [13]. In this paradigm, any cryptographic functionality can be solved as a special case. Such a functionality can be first formalized as a circuit that non-cryptographically solves it. Then, a semi-honestly secure construction is proposed to realize the circuit. In the semi-honest model, all the parties (including corrupted parties) strictly follow the protocol specification. This, of course, does not suffice for real applications. To obtain a realistically secure protocol, a compiler is proposed which, given a semi-honestly secure protocol, outputs a maliciously secure protocol. This approach is generally very powerful, as one need not take care of the problem's specific structure (beyond the circuit). However, since a generic solution does not exploit the detailed structure, it is in general not efficient. Therefore, in the practical point of view, it is very interesting and valuable to investigate a specific problem without following the generic approach. With this motivation in mind, we study a multi-party millionaire problem below. There are n parties, $P_1, \cdots, P_n$. Each P_i has m_i dollars for $1 \le m_i \le N$. We would like P_i to obtain the rank of m_i but nothing beyond this. Formally, we are interested in realizing the *ranking* functionality $(m_1, \cdots, m_n) \mapsto (r_1, \cdots, r_n)$, where $r_i = 1 + \sharp\{m_j : m_j < m_i\}, i = 1, \cdots, n$.

1.1 Related Work

Following Yao [13], many authors [6, 9, 8, 5, 10] worked in the circuit paradigm. However, prior to the work by Cramer *et al.* [3] and by Jakobsson *et al.* [11], none of them have achieved both the non-adaptive malicious security and the communication complexity $O(kn|C|)$, where communication complexity is the total number of bits sent by all parties, $|C|$ is the circuit size, n is the the number of parties, and k is the security parameter. Applying to a ranking functionality, one currently can only achieve communication complexity $O(kn^3)$ in the non-adaptive malicious model. On the other hand, [3] has a round complexity $O(d)$ and [11] has a round complexity $O(d + n)$, where d is the depth of circuit size. Therefore, for the ranking problem, no construction is known such that it is constant round with communication complexity $O(kn^3)$ and that it is secure in the non-adaptive malicious model.

1.2 Contribution

In this work, we construct an efficient n-party ranking protocol. Our protocol has a constant round complexity. Assume k is the security parameter and each party P_i has an input $m_i \in \{1, \cdots, N\}$.

Then our protocol have a communication complexity $O(kn^2(n+N))$. Therefore, if $N = O(n)$ which is the typical case, the communication complexity is $O(kn^3)$. Furthermore, we prove that our construction is secure in the non-adaptive malicious model.

This work is organized as follows. Section 2 introduces the security model. Section 3 introduces a building block: a zero sharing protocol. Section 4 introduces our ranking protocol with ideal functionalities and its security analysis. Section 5 considers how to realize the ideal functionalities introduced in Section 4 and obtain a full functional ranking protocol.

2 Security Model

In this section, we introduce the security model for non-adaptive malicious adversary. In this model, we first have an ideal process in which the computation is essentially achieved via a trusted third party and the adversary capability is rather weak. Then, we turn to the realistic real process which capture the non-adaptive malicious attack in the real world. Finally, a protocol is said to be secure if the real process and ideal process has essentially identical performance.

Ideal Process. We consider the ideal process w.r.t. a non-adaptive malicious adversary $\mathcal{S}$. Let f be an ideal functionality and $\mathcal{F}$ be a trusted third party. Let $P_1, \cdots, P_n$ be n parties involved in the execution. $\mathcal{S}$ has an arbitrary auxiliary input z and P_i has an input $x_i \in D$. Before the protocol starts, $\mathcal{S}$ can select a set of parties $\Phi \subset \{P_1, \cdots, P_n\}$ for corruption. As a result, all the inputs of these parties are provided to $\mathcal{S}$. In addition, their future actions are fully taken by $\mathcal{S}$. After the protocol starts, the execution is described as follows.

- Upon input x_i, an uncorrupted P_i forwards it to $\mathcal{F}$. Upon receiving an output from $\mathcal{F}$, P_i outputs it directly.
- $\mathcal{S}$ can change the input x_i of a corrupted party P_i to $x'_i \in D \cup \{\perp\}$, and sends x'_i to $\mathcal{F}$.
- Upon receiving $x'_1, x'_2, \cdots, x'_n$ from all parties ($x'_i = x_i$ for an uncorrupted P_i), $\mathcal{F}$ may ask $\mathcal{S}$ for a message. Then, he follows f to compute output $(o_1, o_2, \cdots, o_n)$ from $(x'_1, \cdots, x'_n)$ and the response from $\mathcal{S}$ (if any). By default, if some $x'_i = \perp$, then $o_1 = \cdots = o_n = \perp$. Finally, $\mathcal{F}$ asks $\mathcal{S}$ to deliver o_i to P_i.
- $\mathcal{S}$ can deliver o_i or $\perp$ for an uncorrupted P_i. In any case, the secret part of o_i will be kept invisible from $\mathcal{S}$. Finally, $\mathcal{S}$ can output whatever he wishes.

Let r_0 and r_S be the random input of $\mathcal{F}$ and $\mathcal{S}$, respectively. The joint execution of a protocol in the ideal process, denoted by $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(z)}(\boldsymbol{x}; r_0, r_S)$, is a concatenation of the outputs for uncorrupted parties as well as the adversary $\mathcal{S}$. We use random variable $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(z)}(\boldsymbol{x})$ to denote $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(z)}(\boldsymbol{x}; r_0, r_S)$ with uniform random input r_0, r_S.

Real Process. Let Γ be a protocol to realize a functionality f. Let $P_1, \cdots, P_n$ be n parties involved in the execution. Let $\mathcal{A}$ is a PPT adversary with an auxiliary input z. Before the protocol starts, $\mathcal{A}$ specify a set of parties $\Phi \subset \{P_1, \cdots, P_n\}$ for corruption. As in the ideal process, once a party is corrupted, his secret input is provided to $\mathcal{A}$. In addition, his future action is fully taken by $\mathcal{A}$. After the protocol starts, the real process is described as follows.

- Upon input x_i, an corrupted P_i exactly follows the protocol Γ to answer the incoming message and generates its output o_i.
- The action for each corrupted P_i is fully taken by $\mathcal{A}$.

In addition, we assume the channel is authenticated with a guaranteed delivery. That is, for any message M from an uncorrupted P_i, $\mathcal{A}$ must deliver M to the specified receiver without any change. Let r_A, r_i, $(P_i \notin \Phi)$ are the random input for $\mathcal{A}$ and P_i, respectively. o_A, o_i, $(P_i \notin \Phi)$ are the outputs for $\mathcal{A}$, P_i, respectively. Similar to the ideal process, we can define the joint execution, denoted by $\mathrm{REAL}_{\Gamma,\mathcal{A}(z)}(\boldsymbol{x}; r_A, \{r_i : P_i \notin \Phi\})$, to be a concatenation of outputs for uncorrupted parties and the output of $\mathcal{A}$. We use $\mathrm{REAL}_{\Gamma,\mathcal{A}(z)}(\boldsymbol{x})$ to denote the variable of the joint execution with uniform random input.

Definition 1. *Let Γ be an n-party protocol to implement a functionality f. Γ is said to be secure in a malicious but non-adaptive model if for any probabilistic polynomial-time adversary $\mathcal{A}$ in this model there exists an expected polynomial-time ideal adversary $\mathcal{S}$ such that*

$$\left\{ \mathrm{IDEAL}_{f,\mathcal{S}(z)}(\boldsymbol{x}) \right\}_{z,\boldsymbol{x}} \stackrel{c}{\equiv} \left\{ \mathrm{REAL}_{\Gamma,\mathcal{A}(z)}(\boldsymbol{x}) \right\}_{z,\boldsymbol{x}},$$

where $\stackrel{c}{\equiv}$ means computational indistinguishability, z is auxiliary input and $\boldsymbol{x} = (x_1, \cdots, x_n)$ is the input vector for parties.

3 A Zero-Sharing Protocol

Consider functionality $\mathcal{F}_0 : (1^\kappa, \cdots, 1^\kappa) \mapsto \langle (f_1(x), \cdots, f_n(x) \rangle$, where $f_i(x)$ is uniformly at random from $F_q[x]$ of degree at most $n-1$ such that $\sum_{i=1}^n f_i(x) = 0$. In the ideal process, the adversary is allowed to choose $f_i(x)$ for corrupted parties. In this section, we construct a zero-sharing protocol (see Table 1) to realize $\mathcal{F}_0$. Let $R_{2dl} = \{(g^{y_1}h^{y_2}, (y_1, y_2)) : y_1, y_2 \in Z_q\}$. In our protocol, we employ a multi-party function $\mathcal{F}_{m2dl}$, which is defined as follows.

Definition 2. (Multi-party Functionality: $\mathcal{F}_{m2dl}$)
$\mathcal{F}_{m2dl}$ does the following, running $P_1, \cdots, P_n$ and $\mathcal{S}$, and parameterized by relation R_{2dl}.

- *Upon receiving $(F_{jt}, (f_{jt}, f'_{jt}))_{t=1}^n$ from each P_j, $\mathcal{F}_{m2dl}$ verifies if $(F_{jt}, (f_{jt}, f'_{jt})) \in R_{2dl}$. It holds for all i and t, $\mathcal{F}_{m2dl}$ sends message* **ok** *to $P_1, \cdots, P_n$ and $\mathcal{S}$, and terminates; otherwise, it does nothing.*

After the protocol execution, P_i can compute $u_{ij} = f_i(j)$, for each $j = 1, \cdots, n$. All parties can compute $U_{ij} = g^{u_{ij}}h^{u'_{ij}}$, where $u'_{ij} = f'_i(j)$. Note the sharing scheme has a property that for each j, $\sum_i u_{ij} = 0$. That is why we call it a zero-sharing protocol. This property is important for our ranking protocol.

Lemma 1. *Our protocol in Table 1 securely realizes $\mathcal{F}_0$. Specifically, if Φ is the set of corrupted parties with $|\Phi| \leq n-1$, then $f_i(x)$ for uncorrupted P_i is uniformly at random in $F[x]$ with degree at most $n-1$ such that the only constraint is $\sum_{P_i \notin \Phi} f_i(x) = -\sum_{P_i \in \Phi} f_i(x)$. In addition, $\{u_{ij}\}_{i \notin \Phi, 1 \leq j \leq n}$ is uniform at random in $Z_q^{n \times (n-|\Phi|)}$ with only constraint $\sum_{i \in \Phi} u_{ij} = -\sum_{i \notin \Phi} u_{ij}$, for each $j = 1, \cdots, n$.*

Proof. We now show that for any PPT real process adversary $\mathcal{A}$, there exists an expected polynomial-time ideal process adversary $\mathcal{S}$ such that the two executions are indistinguishable. $\mathcal{S}$ first prepares G_q, p, g, computes $h = g^a$, for $a \leftarrow Z_q$. Then he runs $\mathcal{A}$ with G_q, p, g, h. In turn, he will receive a set Φ for corruption. Then he corrupts Φ in the ideal process. For simplicity, we use P_i to represent a party in the internal simulation and $\tilde{P}_i$ the party in the external ideal process.

Params:	G_q, p, g, h;
Output:	$f_i(x) \in Z_q[x]$ for each $P_i, i = 1, \cdots, n$.

1. For each $j = 1, \cdots, n$, P_i takes $f_{ij}(x) = \sum_{t=0}^{n-1} f_{ijt}x^t, f'_{ij}(x) = \sum_{t=0}^{n-1} f'_{ijt}x^t \in Z_q[x]$ randomly such that $\sum_j f_{ij}(x) = 0$ and $\sum_j f'_{ij}(x) = 0$. Then he computes $F_{ijt} = g^{f_{ijt}}h^{f'_{ijt}}$. He *privately* sends $(f_{ij}(x), f'_{ij}(x))$ to P_j, and makes F_{ijt} public.
2. P_j verifies if $\prod_{j=1}^{n} F_{ijt} = 1$ and if $F_{ijt} = g^{f_{ijt}}h^{f'_{ijt}}$ for all i, t. If the verification fails, P_j broadcasts a complaint toward unsuccessful P_i. P_i tries to resolve the complaint by making $(f_{ij}(x), f'_{ij}(x))$ public. If the complaint still remains unresolved, $(f_{ij}(x), f'_{ij}(x)) = (0, 0)$ by default for all j. P_j computes $f_j(x) = \sum_{i=1}^{n} f_{ij}(x)$, $f'_j(x) = \sum_{i=1}^{n} f'_{ij}(x)$.
3. $\mathcal{F}_{m2dl}$ is invoked such that P_j proves $(F_{jt}, (f_{jt}, f'_{jt})) \in R_{2dl}$ for all t, where $F_{jt} = \prod_{i=1}^{n} F_{ijt}$, $f_j(x) := \sum_{t=0}^{n-1} f_{it}x^t$, $f'_j(x) := \sum_{t=0}^{n-1} f'_{it}x^t$. If **ok** is received, P_i accepts and outputs $(f_i(x), f'_i(x))$; otherwise, he outputs $\perp$.

Table 1. An Efficient Zero Sharing Protocol **Zero-Sharing**(n)

$\mathcal{S}$ follows the real protocol execution to interact with $\mathcal{A}$ for Step one and Step two. Assume that he concludes f_j, for each uncorrupted P_j. In Step three, upon receiving $(F_{jt}, (f_{jt}, f'_{jt}))$ for each corrupted P_j from $\mathcal{A}$, $\mathcal{S}$ verifies if $(F_{jt}, (f_{jt}, f'_{jt})) \in R_{2dl}$. If yes, $\mathcal{S}$ sends message **ok** back to $\mathcal{A}$, and externally sends f_{jt} for corrupted $\tilde{P}_j$ to $\mathcal{F}_0$. When $\mathcal{F}_0$ asks him to deliver the messages, he does it faithfully. On the other hand, if the verification fails, $\mathcal{S}$ sends $\perp$ for corrupted party $\tilde{P}_j$ to $\mathcal{F}_0$ (note if no party is corrupted, **ok** must occur). Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ does. The difference between this simulated execution and real execution is that the outputs for uncorrupted parties might be different. In order to be consistent with the ideal process, the output for an uncorrupted P_i in the internal simulation is supposed to output $\tilde{f}_j(x)$, where $\tilde{f}_j(x)$ is the polynomial sent to $\tilde{P}_j$ from the ideal functionality (but invisible to $\mathcal{S}$). However, in the simulated execution, the output $f_j(x)$ of P_j might be different from $\tilde{f}_j(x)$. However, this is not a problem as given the adversary view in the simulated system, we can *consistently* reformulate the output of P_j to $\tilde{f}_j(x)$. Indeed, take a fixed party $P_r \notin \Phi$, reformulate $f_{rj}(x)$ to $\tilde{f}_{rj}(x) = f_{rj}(x) + \tilde{f}_j(x) - f_j(x)$ and $f'_{rj}(x)$ to $\tilde{f}'_{rj}(x) = f'_{rj}(x) - a^{-1}(\tilde{f}_j(x) - f_j(x))$, for each $P_j \notin \Phi$. After the reformulation, adversary view does not change. However, the distribution of the reformulated simulation is exactly according to the real protocol since for each $j = 1, \cdots, n$

$$\begin{aligned} \tilde{f}_{rj}(x) + \textstyle\sum_{i \neq r} f_{ij}(x) &= \tilde{f}_j(x) - f_j(x) + \textstyle\sum_{i=1}^{n} f_{ij}(x) = \tilde{f}_j(x), \\ \tilde{f}'_{rj}(x) + \textstyle\sum_{i \neq r} f'_{ij}(x) &= \tilde{f}'_j(x) - f'_j(x) + \textstyle\sum_{i=1}^{n} f'_{ij}(x) = \tilde{f}'_j(x). \end{aligned}$$

Thus, our protocol realizes the functionality $\mathcal{F}_0$. The second statement is immediate from the first statement. □

4 Our Hybrid Ranking Protocol

In this section, we introduce our cryptographic ranking protocol in the hybrid model, see Table 2. First of all, we assume that all parties have jointly run a zero sharing protocol. Thus, each P_i has the secret output $f_i(x), f'_i(x)$ and public output F_{jt} for all j and t. Let $U_{ij} = \prod_{t=0}^{T} F_{it}^{j^t}$ (publicly computable), $u_{ij} = f_i(j), u'_{ij} = f'_i(j)$ (known to P_i). Our ranking protocol is divided into two phases: input commitment and rank computation. In the input commitment phase, each P_i commits to m_i: $B_{i1} = h^{x_{i1}}, \cdots, B_{i(m_i)} = h^{x_{i(m_i)}}, B_{i(m_i+1)} = \sigma h^{x_{i(m_i+1)}}, \cdots, B_{iN} = \sigma h^{x_{iN}}$. Let

$R_{dl} = \{(h^x, x) : x \in Z_q\}$ and $R_{or} = R_{dl} \cup \{(\sigma h^x, x) : x \in Z_q\}$. An ideal functionality $\mathcal{F}_{mor^+}$ is invoked in which each P_i proves $(B'_{it}, x'_{it}) \in R_{or}$ for all t and $(B_{i1}, x_{i1}) \in R_{dl}$.

Definition 3. (Multi-party Functionality: $\mathcal{F}_{mor^+}$)
$\mathcal{F}_{mor^+}$ does the following, running $P_1, \cdots, P_n$ and $\mathcal{S}$, and parameterized by relations R_{or} and R_{dl}.

- *Upon receiving (B_{i1}, x_{i1}) and $\{(B'_{it}, x'_{it})\}_{t=1}^N$ from each P_i, $\mathcal{F}_{mor^+}$ verifies if $(B'_{it}, x'_{it}) \in R_{or}$ and $(B_{i1}, x_{i1}) \in R_{dl}$. It holds for all i and t, $\mathcal{F}_{mor^+}$ sends message* **ok** *to $P_1, \cdots, P_n$ and $\mathcal{S}$, and terminates; otherwise, it does nothing.*

If $\mathcal{F}_{mor^+}$ does not send **ok**, each P_i aborts; otherwise, he is ready for phase two. Note the successful verification by $\mathcal{F}_{mor^+}$ implies that B_{i1} encode bit 0 and B'_{it} encodes bit 0 or 1. Since B_{i1} encodes bit 0 and $B_{i(N+1)} = \sigma$ encodes bit 1, it follows $\{(B_{it}, x_{it})\}_1^N$ must be an appropriate encoding for an unknown m_i.

In the rank computation stage, each P_i essentially uses an oblivious transfer to send a blinded comparison bit between m_i and m_j to P_j. To do this, he does a fresh commitment to m_i using $\{B_{ijt}\}_{t=1}^N$, and then computes $D_{ijt} = (B_j\sigma^{-t})^{x_{ijt}} z_1^{u_{ij}}$. Let

$$R_{rc} = \Big\{ \Big(\{A_t || \tilde{A}_t || D_t\}_1^N || U || \Delta,\ \{x_t || \tilde{x}_t\}_1^N || u || u' \Big) :\quad \tilde{A}_t A_t^{-1} = h^{\tilde{x}_t - x_t}, (A_t, x_t) \in R_{or}$$
$$D_t = (\Delta \cdot \sigma^{-t})^{\tilde{x}_t} z_1^u, U = g^u h^{u'}, \Delta \in G_q, x_t, \tilde{x}_t, u, u' \in Z_q \}.$$

Note that normally $\Big(\{B_{it} || B_{ijt} || D_{ijt}\}_1^N || U_{ij} || B_j, \{x_t || x_{ijt}\}_1^N || u_{ij} || u'_{ij} \Big) \in R_{rc}$. In the protocol, functionality $\mathcal{F}_{mrc}$ below is invoked in which each P_i proves that $\{x_{it} || x_{ijt}\}_1^N || u_{ij} || u'_{ij}$ is a witness of $\{B_{it} || B_{ijt} || D_{ijt}\}_1^N || U_{ij} || B_j$ w.r.t R_{rc}, parameterized by σ, z_1.

Definition 4. (Multi-party Functionality: $\mathcal{F}_{mrc}$)
$\mathcal{F}_{mrc}$ does the following, running $P_1, \cdots, P_n$ and $\mathcal{S}$, and parameterized by R_{rc}, σ and z_1.

- *Upon receiving $\Big(\{B_{it} || B_{ijt} || D_{ijt}\}_1^N || U_{ij} || B_j,\quad \{x_{it} || x_{ijt}\}_1^N || u_{ij} | u'_{ij} \Big)$ from P_i, $\mathcal{F}_{mrc}$ checks it is consistent w.r.t. R_{rc}. If it holds, then $\mathcal{F}_{mrc}$ sends message* **ok**(P_i, P_j) *to P_j and $\mathcal{S}$; otherwise, it ignores the verification for (P_i, P_j).*

Since $D_{ijt} = h^{y_j x_{ijt}} z_1^{u_{ij}} \times \sigma^{(m_j - t)x_{ijt}}$, P_j can compute $\xi_j = B_{j(m_j)} \cdot D_{ij(m_j)}^{-1/y_j} = \sigma^{\delta_{ij}} z_1^{u_{ij}/y_j}$, where the bit $\delta_{ij} = 0$ if $m_j \leq m_i$; $\delta_{ij} = 1$ otherwise. Note $\sum_i \delta_{ij} = r_j - 1$ and $\sum_i u_{ij} = 0$. Completeness of our protocol follows.

Now we formally prove the security of our ranking protocol.

Theorem 1. *Our hybrid ranking protocol is secure in the non-adaptive malicious model.*

Proof. We show that for any PPT real process adversary $\mathcal{A}$, there exists an expected polynomial-time ideal process adversary $\mathcal{S}$ such that the joint executions in the two processes are indistinguishable. $\mathcal{S}$ internally runs the simulated real process with $\mathcal{A}$, playing the uncorrupted parties. At the same time, he externally involves the ideal process execution, on behalf of corrupted party $\tilde{P}_i \in \Phi$. The code of $\mathcal{S}$ is described as below.

- $\mathcal{S}$ first gets the input m_i for each $\tilde{P}_i \in \Phi$, and provides to $\mathcal{A}$ as the input for P_i. He prepares $p, g, h, \sigma = h^a$ for $a \leftarrow Z_q$ and runs $\mathcal{A}$ with it.

Params: p, g, σ, h, z_1; **Input:** m_i for $P_i, i = 1, \cdots, n$.

PHASE ONE: Input Commitment.

1. Each P_i takes $x_{it} \leftarrow Z_q$, computes $B_{it} = \begin{cases} h^{x_{it}} & \text{if } 1 \leq t \leq m_i, \\ \sigma h^{x_{it}} & \text{if } m_i < t \leq N \end{cases}$. Then P_i broadcasts $\langle B_{i1}, \cdots, B_{iN} \rangle$.
2. Let $B'_{it} = B_{i(t+1)} B_{it}^{-1}$ for $1 \leq t \leq N$, where by default $B_{i(N+1)} = 1$. Let $x'_{it} = x_{i(t+1)} - x_{it}$ and $x_{i(N+1)} = 0$. $\mathcal{F}_{mor^+}$ is invoked in which each P_i proves $(B_{i1}, x_{i1}) \in R_{dl}$ and $(B'_{it}, x'_{it}) \in R_{or}$ for all t. If a message **ok** is not received from $\mathcal{F}_{mor^+}$, P_i aborts; otherwise, he is ready for Phase Two.
   Let $B_i = \sigma^N \cdot \prod_{t=1}^{N} B_{it}^{-1} = \sigma^{m_i} h^{y_i}$, for $y_i = -\sum_{t=1}^{N} x_{it}$. P_i keeps y_i secret.

PHASE TWO: Rank Computation.

1. P_i prepares the outgoing message to P_j as follows.
   - P_i takes $x_{ijt} \leftarrow Z_q$, set $B_{ijt} = \begin{cases} h^{x_{ijt}} & \text{if } 1 \leq t \leq m_i, \\ \sigma h^{x_{ijt}} & \text{if } m_i < t \leq N \end{cases}$. In other words, P_i commits to m_i again.
   - P_i computes $D_{ijt} = (B_j \cdot \sigma^{-t})^{x_{ijt}} \cdot z_1^{u_{ij}}$, sends D_{ijt} to P_j for all t.
2. $\mathcal{F}_{mrc}$ is invoked in which each P_i proves $\left(\{B_{it}||B_{ijt}||D_{ijt}\}_{t=1}^{N}\, ||U_{ij}||B_j, \{x_{it}||x_{ijt}\}_{t=1}^{N}\, ||u_{ij}||u'_{ij}\right) \in R_{rc}$ to P_j.
3. P_j computes $\xi_i = B_{ij(m_j)} \cdot D_{ij(m_j)}^{-1/y_j}$. Finally, he derives $\xi = \prod_{i=1}^{N} \xi_i = \sigma^{r_j - 1}$ and obtains r_j by trial.

Table 2. Our Hybrid Ranking Protocol

- $\mathcal{S}$ plays the role of uncorrupted P_i to compute $B_{it} = h^{x^*_{it}}$, where $x^*_{it} \leftarrow Z_q$, $1 \leq t \leq N$. Note that $\mathcal{S}$ does not know the real input m_i. Thus, he is unable to do a real commitment to m_i. Then, $\mathcal{S}$ simulates the ideal functionality $\mathcal{F}_{mor^+}$. If all the inputs from corrupted parties (controlled by $\mathcal{A}$) have been successfully verified, $\mathcal{S}$ (simulating $\mathcal{F}_{mor^+}$) sends **ok** to $\mathcal{A}$; otherwise, he does nothing.
- $\mathcal{S}$ calculates $\tilde{m}_i$ from the input to $\mathcal{F}_{mor^+}$ for corrupted P_i. He then externally sends $\tilde{m}_i$ for $\tilde{P}_i$ to $\mathcal{F}_0$. And in turn, he obtains the corresponding r_i. He then chooses arbitrary $\tilde{m}_i$ for uncorrupted P_i such that each $\tilde{m}_j$ for corrupted P_j has a rank r_i among $(\tilde{m}_1, \cdots, \tilde{m}_n)$. Then for uncorrupted P_i, he defines $x_{it} = x^*_{it}$ if $t \leq \tilde{m}_i$; $x_{it} = x^*_{it} - a$ if $\tilde{m}_i < t \leq N$. Under this formulation, $\{B_{it}\}_1^N$ is an encoding of $\tilde{m}_i$ as computed in Step 1 of Phase One.
- In Phase Two, $\mathcal{S}$ faithfully interacts with $\mathcal{A}$ by simulating $\mathcal{F}_{mrc}$ and all uncorrupted P_i with $\{x_{it}\}_1^N$. In the external execution (ideal process), $\mathcal{S}$ delivers r_j to uncorrupted $\tilde{P}_j$ if and only if $\mathbf{ok}(P_i, P_j)$ for all i has been computed in the internal execution for all corrupted P_i. Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ does.

The view of $\mathcal{A}$ is different from the real process: for uncorrupted P_i, (1) Input commitment is dummy instead a commitment of m_i; (2) $\tilde{m}_i$ is not equal to m_i for uncorrupted P_i. In the remaining part, we show this modification does change the distribution of $\mathcal{A}$'s view.

Let the simulated game be Γ. Consider the mental game Γ_1 of Γ, where the only difference is that for uncorrupted P_i, the input commitment is for $\tilde{m}_i$ instead of first being dummy and reformulating later. The view of $\mathcal{A}$ under this change is identically distributed in Γ, as x^*_{it} in Γ is uniformly at random (thus x_{it} obtained in the reformation is random) in Z_q.

Now we will show that the adversary view in Γ_1 and the variant of Γ_1, where $\tilde{m}_i$ for uncorrupted P_i is replaced by m_i, is indistinguishable. We do this using a sequence of game techniques.

First we modify Γ_1 to Γ_2 such that in Phase Two, for each uncorrupted P_i, $D_{ijt} = h^{y_j x_{ijt}} \cdot \beta_{ijt}^{\tilde{m}_j - t} \cdot z_1^{u_{ij}}$, where $\beta_{ijt} \leftarrow G_q$. Note if $\beta_{ijt} = \sigma^{x_{ijt}}$, then Γ_2 becomes Γ_1. We show that the execution in these two games are indistinguishable. If this were not true, we construct an adversary $\mathcal{B}_2$ to break DDH assumption. Given $(h, \sigma, \alpha, \beta)$ (either DH tuple or random tuple), $\mathcal{B}_2$ takes $(\alpha_{ljt}, \beta_{ljt}) \leftarrow Rud_1(h, \sigma, \alpha, \beta)$, $1 \leq t \leq N, 1 \leq j \leq n$, for each uncorrupted P_l (algorithm Rud is presented in Appendix A). He then follows Γ_1 (and Γ_2) for input commitment stage. In the second phase, he follows the simulator $\mathcal{S}$ in Γ_2, except that for uncorrupted P_i, $h^{x_{ijt}}$ in defining B_{ijt} is taken as α_{ijt} and that $D_{ijt} = \alpha_{ijt}^{y_j} \cdot \beta_{ijt}^{\tilde{m}_j - t} \cdot z_1^{u_{ij}}$. Finally, $\mathcal{B}_2$ feeds the execution output to the distinguisher and outputs whatever he does. Note if $(h, \sigma, \alpha, \beta)$ is DH tuple, then the simulated game by $\mathcal{B}_2$ is distributed as in Γ_1; otherwise, it is distributed according to Γ_2. Thus, the distinguishability between Γ_1 and Γ_2 implies the non-negligible advantage of $\mathcal{B}_2$, a contradiction to DDH assumption.

We modify Γ_2 to Γ_3 such that in Phase Two, for any uncorrupted pair P_i and P_j, $D_{ijt} = \gamma_{ijt} \cdot \beta_{ijt}^{\tilde{m}_j - t} \cdot z_1^{u_{ij}}$ for $\gamma_{ijt} \leftarrow G_q$ (instead of $\gamma_{ijt} = h^{y_j x_{ijt}}$). We show the execution in Γ_2 and Γ_3 is indistinguishable; otherwise, consider a DDH breaker $\mathcal{B}_3$. Upon receiving input (h, μ, ν, γ), $\mathcal{B}_3$ first takes $(h, \mu_j, \nu_j, \gamma_j) \leftarrow Rud_1(h, \mu, \nu, \gamma)$ (See Appendix A for details) for each uncorrupted P_j and then further takes $(h, \mu_j, \nu_{ijt}, \gamma_{ijt}) \leftarrow Rud_0(h, \mu_j, \nu_j, \gamma_j)$ for $1 \leq t \leq N$ and each uncorrupted P_i. He follows the simulation in Γ_2 for input commitment except that $h^{x_{jN}}$ is computed as $\mu_j \cdot \prod_{t=1}^{N-1} h^{-x_{jt}}$. Note this simulation is distributed identically as in Γ_2 (and Γ_3) as μ_j is uniform in G_q. In Phase Two, $\mathcal{B}_3$ follows the simulation in Γ_3, except that for any uncorrupted pair P_i, P_j, (1) $h^{x_{ijt}}$ in computing B_{ijt} is defined to be ν_{ijt}; (2) $D_{ijt} = \gamma_{ijt} \cdot \beta_{ijt}^{\tilde{m}_j - t} \cdot z_1^{u_{ij}}$, where $\beta_{ijt} \leftarrow G_q$ as in Γ_2 and γ_{ijt} is the value just derived from Rud. By the property of Rud algorithm, if (h, μ, ν, γ) is a DH tuple, then the simulated game is identically distributed as Γ_2; otherwise, it is according to Γ_3. Thus, the distinguishablity between Γ_2 and Γ_3 implies the distinguishability of DDH, contradiction.

Consider the mental game Γ_4, a variant of Γ_3 while $\tilde{m}_i$ is replaced by m_i (registered by $\tilde{P}_i$ to the ideal functionality). We show that the views of $\mathcal{A}$ between Γ_3 and Γ_4 are identically distributed. Otherwise, commitments for $\{\tilde{m}_i\}_{P_i \notin \Phi}$ and $\{m_i\}_{P_i \notin \Phi}$ using the method in Step 1 can be distinguished. However, this is impossible since the two set of commitments have identical distribution. Here is details. Upon input $\{B_{it}\}_1^N$ for each uncorrupted P_i (either commit of m_i or commitment of $\tilde{m}_i$), a distinguisher $\mathcal{B}_4$ simulates Step 2 normally (as in Γ_3). Phase Two is simulated as follows.

- For each uncorrupted P_i, $B_{ijt} = B_{it} \cdot h^{\Delta_{ijt}}$ for each j, t, where $\Delta_{ijt} \leftarrow Z_q$.
- For each uncorrupted P_i and corrupted P_j, take $D_{ijt} \leftarrow G_q$ for $t \neq \tilde{m}_j$. Fix $P_{i_0} \notin \Phi$. Take $D_{i_0 j(\tilde{m}_j)} = B_{i_0 j(\tilde{m}_j)}^{y_j} \cdot \sigma^{-R_j y_j} \cdot z_1^{u_{ij}}$, where $R_j = \sharp\{i : P_i \notin \Phi, m_i \geq \tilde{m}_j\}$. For $i \neq i_0$, take $D_{ij(\tilde{m}_j)} = B_{ij(\tilde{m}_j)}^{y_j} \cdot z_1^{u_{ij}}$. Note $y_j = -\sum_{t=1}^N x_{jt}$ and x_{jt} is obtained from the input to functionality $\mathcal{F}_{mor^+}$ $\mathcal{A}$ in Phase One.
- For uncorrupted P_i and P_j, take $D_{ijt} \leftarrow G_q$.

Now we claim that no matter the input to the simulator is commitment of $\{\tilde{m}_i\}$ or $\{m_i\}$, the above simulation is consistent with Γ_4. It suffices to show that $D_{ij(\tilde{m}_j)}$ for uncorrupted P_i and corrupted P_j is according to Γ_4. Notice for corrupted P_j, $\tilde{m}_j$ is ranked r_j for both cases $\{m_i\}_{P_i \notin \Phi}$ and $\{\tilde{m}_i\}_{P_i \notin \Phi}$, it follows that $\sharp\{i : P_i \notin \Phi, m_i \geq \tilde{m}_j\} = \sharp\{i : P_i \notin \Phi, \tilde{m}_i \geq \tilde{m}_j\}$. Thus, we only need to consider the case, where for uncorrupted P_i $\{B_{it}\}_1^N$ is the commitment of $\tilde{m}_i$. Our key point is that for any j, $\{u_{ij} : P_i \notin \Phi\}$ are uniform in Z_q with only constraint $\sum_{P_i \notin \Phi} u_{ij} = -\sum_{P_i \in \Phi} u_{ij}$ (by Lemma 1). Note that $B_{ijt} = \sigma^{\delta_{it}} h^{x_{ijt}}$, where the bit $\delta_{it} = 0$ if and only if $t \leq \tilde{m}_i$. Thus, in the simulation, $D_{i_0 j \tilde{m}_j} = h^{y_j x_{i_0 j \tilde{m}_j}} \times \sigma^{y_j \delta_{i_0 \tilde{m}_j}} \cdot \sigma^{-y_j R_j} \cdot z_1^{u_{i_0 j}}$; for other uncorrupted P_i, $D_{ij\tilde{m}_j} = h^{y_j x_{ij\tilde{m}_j}} \times \sigma^{y_j \delta_{i\tilde{m}_j}} \cdot z_1^{u_{ij}}$. Note that $\sigma^{-R_j} \prod_{P_i \notin \Phi} \sigma^{\delta_{i\tilde{m}_j}} \cdot z_1^{u_{ij}} = \prod_{P_i \notin \Phi} z_1^{u_{ij}}$. Let $\tilde{u}_{i_0 j} = u_{i_0 j} - (R_j - \delta_{i_0 \tilde{m}_j}) y_j \log_{z_1} \sigma$, and $\tilde{u}_{ij} = u_{ij} + \delta_{i\tilde{m}_j} \log_{z_1} \sigma$ for other uncorrupted P_i. Note that $\sum_{P_i \notin \Phi} \tilde{u}_{ij} = \sum_{P_i \notin \Phi} u_{ij}$, we have that $\{\tilde{u}_{ij}\}_{P_i \notin \Phi}$ can be regarded as another feasible assignment for $\{u_{ij}\}_{P_i \notin \Phi}$. Furthermore, $\{\tilde{u}_{ij}\}_{P_i \notin \Phi}$ is according to the real distribution since $\{u_{ij}\}_{P_i \notin \Phi}$ is uniform at random with the only constraint on their additive sum. Therefore, the simulation is actually distributed as in Γ_4. Our claim follows.

Furthermore, notice adversary view in $\Gamma_1, \cdots, \Gamma_4$ when $\tilde{m}_i = m_i$ for all uncorrupted P_i is still indistinguishable. On the other hand adversary view in Γ_1 with $\tilde{m}_i = m_i$ for all uncorrupted P_i is according to the real execution. Thus, the execution of ideal process by $\mathcal{S}$ is indistinguishable from the execution of the real process by $\mathcal{A}$. ■

5 Full Ranking Protocol

Cramer *et al.* [3] demonstrates a transformation which, given any $\sum$-protocol, outputs a secure 3-round multi-party (parallel) $\sum$-protocol. In this section, we realize our building functionalities $\mathcal{F}_{m2dl}, \mathcal{F}_{mor^+}, \mathcal{F}_{mrc}$ using their transformation. Then a full ranking protocol can be obtained by (sequentially) composing these realizations with the hybrid protocol in the last section. We start with the notion of $\sum$-protocol.

5.1 $\sum$-Protocol

Let R be a binary relation consisting of pair (x, w), where x is a public string and w is a witness of polynomial length. Consider a 3-round proof of knowledge protocol for $(x, w) \in R$, where x is the common input and w is the private input for the prover. The prover starts with a message a. The verifier responds with a challenge e. Finally, the prover responds with a finishing message z. Then the verifier accepts if and only if $ver(a, e, z, x) = 1$ for a public algorithm ver. Such a protocol is said to be a $\sum$-protocol if it satisfies the following.

- **Completeness.** If the prover is given a private input w such that $(x, w) \in R$, then the verifier always accepts.
- **Special Honest Verifier Zero-knowledge.** For any e, one can efficiently compute (a, e, z) such that (a, e, z) is according to the real distribution with a fixed e.
- **Witness Extraction.** For a fixed x, one can efficiently extract witness w from any two accepting transcripts (a, e, z) and (a, e', z') with $e' \neq e$.

Useful Examples. The protocols presented in Appendix B are $\sum$-protocols $\pi_{dl}, \pi_{2dl}, \pi_{or}$, and π_{rc} for relations R_{dl}, R_{2dl}, R_{or} and R_{rc} respectively. These examples will soon be applied to realize our building functionalities.

5.2 Secure Multi-Party $\sum$-Protocol

Let $R_1, \cdots, R_v$ be v binary relations. Assume there are n parties $P_1, \cdots, P_n$. Let $\boldsymbol{\Lambda} = (\boldsymbol{\Lambda}_1, \cdots, \boldsymbol{\Lambda}_n)$, where $\boldsymbol{\Lambda}_i$ is a collection of numbers taken from $\{1, \cdots, v\}$ with replacement (i.e., taking two identical numbers is allowed). Let $x_{i,j}$, for $j \in \boldsymbol{\Lambda}_i$ and $1 \leq i \leq n$, be the common input for all parties. Suppose $\mathcal{F}_{\boldsymbol{\Lambda}}$ be an n-party functionality in which each P_i proves the knowledge of witness w_{ij} *s.t.* $(x_{ij}, w_{ij}) \in R_j$, for each $j \in \boldsymbol{\Lambda}_i$. The following has been established by Cramer *et al.* [3] (Section 6.3 in their paper).

Lemma 2. *Let $R_1, \cdots, R_v$ be v binary relations. Let π_i be a $\sum$-protocol for relation R_i for $1 \leq i \leq v$. Then there exists a 3-round multi-party protocol $\pi_{\boldsymbol{\Lambda}}$ realizing $\mathcal{F}_{\boldsymbol{\Lambda}}$. In addition, if π_i $(1 \leq i \leq v)$ has a communication complexity upper bounded by $O(K)$, then the communication complexity of $\pi_{\boldsymbol{\Lambda}}$ is upper bounded by $O(K \sum_{i=1}^{n} |\boldsymbol{\Lambda}_i|)$.*

Now we are ready to realize $\mathcal{F}_{m2dl}, \mathcal{F}_{mor^+}, \mathcal{F}_{mrc}$. From Lemma 2, we we can easily conclude the following result.

Corollary 1. *There exists an 3-round multi-party protocol π_{m2dl} (resp. π_{mor^+}, π_{mrc}) realizing the ideal functionality $\mathcal{F}_{m2dl}$ (resp. $\mathcal{F}_{mor^+}, \mathcal{F}_{mrc}$). Furthermore, the communication complexity of π_{m2dl} (resp. π_{mor^+}, π_{mrc}) is $O(n^2k)$ (resp. $O(nNk)$, $O(nNk)$), where k is the security parameter (i.e., the length of p).*

Proof. In $\mathcal{F}_{m2dl}$, each P_i proves the knowledge of witness for N instances w.r.t R_{2dl}. In $\mathcal{F}_{mor^+}$, each P_i proves the knowledge of witness of N instances w.r.t. R_{or} and one instance w.r.t. R_{dl}. In $\mathcal{F}_{mrc}$, each P_i proves the knowledge of witness of one instance w.r.t. R_{rc}. On the other hand, π_{dl}, π_{2dl}, π_{or} and π_{rc} are $\sum$-protocols for R_{dl}, R_{2dl}, R_{or} and R_{rc} respectively. Thus, the first part follows from Lemma 2. The second part follows since π_{dl}, π_{2dl}, π_{or} and π_{rc} have communication complexity $O(k), O(k), O(k), O(Nk)$, respectively. ■

5.3 Full Ranking Protocol

Now we are ready to state our full ranking protocol. Let *FulRank* be the ranking protocol in the last section but functionalities $\mathcal{F}_{m2dl}$, $\mathcal{F}_{mor^+}, \mathcal{F}_{mrc}$ are replaced by $\pi_{m2dl}, \pi_{mor^+}, \pi_{mrc}$ respectively.

Theorem 2. *FulRank is a ranking protocol realizing the ranking functionality in the non-adaptive malicious model. In addition, FulRank is a constant round complexity and has a communication complexity $O(n^2k(N+n))$.*

Proof. Since *FulRank* is obtained from the hybrid protocol via sequential compositions, the security follows. It has a constant round complexity since the hybrid protocol, and π_{m2dl}, π_{mor^+}, π_{mrc} all are constant round. Since the zero-sharing protocol has the communication complexity $O(n^3k)$ and the main ranking part has $O(n^2Nk)$, it follows that the whole protocol has $O(n^2k(N+n))$. ■

References

1. M. Bellare, A. Boldyreva, S. Micali, Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements, *EUROCRYPT'00*, pp. 259-274, 2000.
2. M. Bellare and S. Goldwasser, Verifiable Partial Key Escrow, *ACM CCS'97*, pp. 78-91, 1997.
3. R. Cramer, I. Damgrd, J. Nielsen, Multiparty Computation from Threshold Homomorphic Encryption, *EUROCRYPT'01*, pp. 280-299, 2001.
4. R. Cramer, I. Damgard, and B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, *CRYPTO'94*, LNCS 839, Y. Desmedt (ed.), SpringerVerlag, 1994.
5. R. Cramer, I. Damgard, and U. Maurer, Gemeral secure multi-party computation from any linear secret sharing scheme, *Advances in Cryptology-EUROCRYPT'00*, B. Preneel (Ed.), LNCS 1807, Springer-Verlag, pp 316-334, 2000.
6. M. Franklin, Comlexity and Security of Distributed Protocols, *Ph. D thesis*, Columbia University, 1993.
7. M. Franklin and Haber, Joint encryption and message-efficient computation, *Journal of Cryptology*, 9(4): 217-234, 1996.
8. Z. Galil, S. Haber, and M. Yung. Cryptographic computation: secure fault-tolerant protocol and the public-key model. In *Advances in Cryptology-CRYPTO'87*, C. Pomerance (Ed.), LNCS 293, Spriner-Verlag, New York, 1988.
9. O. Goldreich, S. Micali, and A. Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, *STOC'87*, pp. 218-229, New York City, 25-27 May, 1987.
10. Y. Ishai and E. Kushilevitz, Randomizing polynomials: a new representation with application to random efficient secure computation, *FOCS '00*, pp. 294-304, 2000.
11. M. Jakobsson and A. Juels, Mix and match: secure function evaluation via ciphertexts, *Advances in Cryptology-ASIACRYPT'00*, T. Okamoto (Ed.), LNCS 1976, Springer-Verlag, pp. 162-177, 2000.
12. V. Shoup, On Formal Models for Secure Key Exchange, Available at http://philby.ucsd.edu/cryptolib/1999.html.

13. A. C. Yao, Protocols for secure computations (extended abstract), *FOCS'82*, pp. 160-164, 1982.

Appendix A Diffie-Hellman Self-reduction.

Now we introduce a self-reduction technique [1, 12]. Let p, q be two large primes with $p = 2q+1$; G_q be the subgroup of order q in Z_p^*. And $g \in G_q \setminus \{1\}$. Thus, $\langle g \rangle = G_q$. Given a bit x and a triple (g^a, g^b, g^c), a self-reduction algorithm Rud_x in $[1, 12]$ can efficiently compute a new triple $(g^{a'}, g^{b'}, g^{c'})$ with the properties in Table 3.

Table 3. Properties of Output from Self-reduction Rud_x

	$x = 0$	$x = 1$
$c = ab$	$a' = a$ & b' random in Z_q & $c' = a'b'$	a', b' random in Z_q & $c' = a'b'$
$c \neq ab$	$a' = a$ &b', c' random in Z_q	a', b', c' random in Z_q

For example, if the input is $x = 0$ and a triple (g^a, g^b, g^{ab}), then the output will be $(g^a, g^{b'}, g^{ab'})$, where b' is uniformly random in Z_q. For simplicity, we use $(g^{a'}, g^{b'}, g^{c'}) \leftarrow Rud_x(g^a, g^b, g^c)$ to denote a random output of Rud with input x and (g^a, g^b, g^c).

Appendix B

In our protocol, $\sum$-protocols for R_{dl}, R_{2dl}, R_{or} and R_{rc} are presented in Tables 4, 5, 6, 7. All these protocols are not new. For example, Tables 6 [2] and Table 7 both are examples of the general OR protocol in [4].

Common Input:	$G_q, p, h, X = h^x$.
Auxiliary Input:	x for Prover.

1. Prover takes $x' \leftarrow Z_q$, computes $X' = h^{x'}$. Then he sends X' to Verifier.
2. Verifier takes $e \leftarrow Z_q$ and sends to Prover.
3. Prover computes $r = ex + x'$ and sends r to Verifier.
4. Verifier checks if $h^r = X^e X'$. He accepts if the check is successful.

Table 4. The $\sum$-Protocol π_{dl} for relation R_{dl}

Common Input:	G_q, p, g, h, and $S = g^x h^y$.
Auxiliary Input:	(x, y) for Prover.

1. Prover takes $x', y' \leftarrow Z_q$, computes $S' = g^{x'} h^{y'}$ and sends it to Verifier.
2. Verifier takes $e \leftarrow Z_q$ and sends back to Prover.
3. Prover computes $r_1 = ex + x', r_2 = ey + y'$ and sends (r_1, r_2) to Verifier.
4. Verifier checks if $S^e S' = g^{r_1} h^{r_2}$. The proof is accepted if the verification is successful.

Table 5. The $\sum$-Protocol π_{2dl} for relation R_{2dl}

Common Input:	G_q, p, h, σ, $X = h^x$ or σh^x.
Auxiliary Input:	x for Prover.

1. If $X = h^x$, Prover takes $w_1, r_2, c_2 \leftarrow Z_q$, computes $R_1 = h^{w_1}, R_2 = h^{r_2}(X/\sigma)^{-c_2}$.
   If $X = \sigma h^x$, Prover takes $w_2, r_1, c_1 \leftarrow Z_q$, computes $R_1 = h^{r_1} X^{-c_1}, R_2 = h^{w_2}$.
   Then he sends (R_1, R_2) to Verifier.
2. Verifier takes $c \leftarrow Z_q$ and sends back to Prover.
3. If $X = h^x$, Prover computes $c_1 = c - c_2, r_1 = w_1 + c_1 x$.
   If $X = \sigma h^x$, Prover computes $c_2 = c - c_1, r_2 = w_2 + c_2 x$.
   Then he sends r_1, r_2, c_1, c_2 to Verifier.
4. Verifier checks if $c_1 + c_2 = c$, $h^{r_1} = R_1 \cdot X^{c_1}$, $h^{r_2} = R_2 \cdot (X/\sigma)^{c_2}$. He accepts if the verification is successful.

Table 6. The $\sum$-Protocol π_{or} for relation R_{or}

Common Input: $p, g, h, \sigma, z_1, \{B_t || \tilde{B}_t || D_t\}_{t=1}^N || U || B$
Auxiliary Input: $\{x_t || \tilde{x}_t\}_{t=1}^N || u || u'$ for Prover.

1. If $B_t = h^{x_t}$, Prover takes $x^*_{t1}, u^*_{t1}, u'^*_{t1}, r_{t2}, \tilde{r}_{t2}, s_{t2}, s'_{t2}, e_{t2} \leftarrow Z_q$, computes

$$B^*_{t1} = h^{x^*_{t1}}, B^*_{t2} = h^{r_{t2}}(B_t/\sigma)^{-e_{t2}}, \qquad \tilde{B}^*_{t1} = h^{\tilde{x}^*_{t1}}, \tilde{B}^*_{t2} = h^{\tilde{r}_{t2}}(\tilde{B}_t/\sigma)^{-e_{t2}},$$
$$D^*_{t1} = (B\sigma^{-t})^{\tilde{x}^*_{t1}} z_1^{u^*_{t1}}, D^*_{t2} = (B\sigma^{-t})^{\tilde{r}_{t2}} z_1^{s_{t2}} D_t^{-e_{t2}}, \qquad U^*_{t1} = g^{u^*_{t1}} h^{u'^*_{t1}}, U^*_{t2} = g^{s_{t2}} h^{s'_{t2}} U^{-e_{t2}}.$$

   If $B_t = \sigma h^{x_t}$, Prover takes $x^*_{t2}, u^*_{t2}, u'^*_{t2}, r_{t1}, \tilde{r}_{t1}, s_{t1}, s'_{t1}, e_{t1} \leftarrow Z_q$, computes

$$B^*_{t1} = h^{r_{t1}} B_t^{-e_{t1}}, B^*_{t2} = h^{x^*_{t2}}, \qquad \tilde{B}^*_{t1} = h^{\tilde{r}_{t1}} \tilde{B}_t^{-e_{t1}}, \tilde{B}^*_{t2} = h^{\tilde{x}^*_{t2}},$$
$$D^*_{t1} = (B\sigma^{-t})^{\tilde{r}_{t1}} z_1^{s_{t1}} D_t^{-e_{t1}}, D^*_{t2} = (B\sigma^{-t})^{\tilde{x}^*_{t2}} z_1^{u^*_{t2}}, \qquad U^*_{t1} = g^{s_{t1}} h^{s'_{t1}} U^{-e_{t1}}, U^*_{t2} = g^{u^*_{t2}} h^{u'^*_{t2}}.$$

   Then he sends $\{B^*_{t1} || B^*_{t2},\ \tilde{B}^*_{t1} || \tilde{B}^*_{t2},\ D^*_{t1} || D^*_{t2},\ U^*_{t1} || U^*_{t2}\}_1^N$ to Verifier.
2. Verifier takes $e \leftarrow Z_q$ and sends back to Prover.
3. If $X = h^x$, Prover computes $e_{t1} = e - e_{t2}, r_{t1} = x^*_{t1} + e_1 x_t, \tilde{r}_{t1} = \tilde{x}^*_{t1} + e_1 \tilde{x}_t, s_{t1} = u^*_{t1} + e_1 u, s'_{t1} = u'^*_{t1} + e_1 u'$.
   If $X = \sigma h^x$, Prover computes $e_{t2} = e - e_{t1}, r_{t2} = x^*_{t2} + e_2 x_t, \tilde{r}_{t2} = \tilde{x}^*_{t2} + e_2 \tilde{x}_t, s_{t2} = u^*_{t2} + e_2 u, s'_{t2} = u'^*_{t2} + e_2 u'$.
   Then he sends $\{r_{t1} || r_{t2}, \tilde{r}_{t1} || \tilde{r}_{t2}, s_{t1} || s_{t2}, s'_{t1} || s'_{t2}, e_{t1}, e_{t2}\}_1^N$ to Verifier.
4. Verifier checks if $e_{t1} + e_{t2} = e$, and if the following for all t.

$$h^{r_{t1}} = B^*_{t1} B_t^{e_{t1}}, h^{r_{t2}} = B^*_{t2}(B_t/\sigma)^{e_{t2}}, \qquad h^{\tilde{r}_{t1}} = \tilde{B}^*_{t1} \tilde{B}_t^{e_{t1}}, h^{\tilde{r}_{t2}} = \tilde{B}^*_{t2}(\tilde{B}_t/\sigma)^{e_{t2}},$$
$$(B\sigma^{-t})^{\tilde{r}_{t1}} z_1^{s_{t1}} = D^*_{t1} D_t^{-e_{t1}}, (B\sigma^{-t})^{\tilde{r}_{t2}} z_1^{s_{t2}} = D^*_{t2} D_t^{-e_{t2}}, \qquad g^{s_{t1}} h^{s'_{t1}} = U^*_{t1} U^{-e_{t1}}, g^{s_{t2}} h^{s'_{t2}} = U^*_{t2} U^{-e_{t2}}.$$

   He accepts if the verification is successful.

Notation: $B_t = h^{x_t}$ and $\tilde{B}_t = h^{\tilde{x}_t}$, or, $B_t = \sigma h^{x_t}$ and $\tilde{B}_t = \sigma h^{\tilde{x}_t}$; $U = g^u h^{u'}$; $D_t = (B\sigma^{-t})^{\tilde{x}_t} z_1^u$.

Table 7. The $\sum$-protocol π_{rc} for relation R_{rc}