An Efficient Protocol for the Commit-Prove-Fair-Open functionality

Ou Ruan\(^a\), Cai Fu\(^*\), Guohua Cui\(^a\)

\(^a\) College of Compute Science & Technology, Huazhong University of Science & Technology, Wuhan, China, 430074

\(^b\) College of Compute Science & Technology, Hubei University of Technology, Wuhan, China, 430068

ruanou@21cn.com, 12695133@qq.com, cgh3986@126.com

*: Corresponding author: Cai Fu, 12695133@qq.com, stand_fucai@126.com

Research supported in part by the National Natural Science Funds of China(60903175)

Abstract. In TCC 2006, Garay et al. introduced the notion of "commit-prove-fair-open" functionality in order to achieve what they called "resource fairness" of secure multi-party computation(MPC) with corrupted majority. The protocol realizing this notion of fairness follows the gradual release approach and, further, it can be proven secure in the simulation paradigm and enjoys composition properties.

In this paper, we show a more efficient resource-fair protocol of FCPO based on a new variant of Garay et al. time-lines and simplified Camenisch-Shoup(sCS) commitment, whose communication and computation complexity are less than 1/5 of Garay et al. construction. In addition, our new protocol allows commitment to value 0, which is not possible in the plain Garay et al. construction.

Keywords: commit-prove-fair-open functionality, resource fairness, time-lines, secure multi-party computation

1 Introduction

In TCC 2006, Garay et al.[1] introduced the notion of resource-fair secure multi-party computation(MPC) [2] with corrupted majority, which essentially means that if one party learns the output of the protocol, then so can all other parties, as long as they expend roughly the same amount of resources. Their approach follows the gradual release approach[3,4,5] and, further, it can be proven secure in the simulation paradigm and enjoys composition properties. Turning to constructions of resource-fair secure MPC protocol, they defined the “commit-prove-fair-open”(FCPO) functionality and designed a resource-fair protocol GradRel that securely realizes it using a new variant of Garay and Jakobsson’s “time-lines” [4]. With this functionality, they constructed resource-fair secure MPC protocols based on the Canetti et al.[6] and Cramer et al.[7] MPC protocols, and 2008, Kiraz and Schoenmakers[8] showed a resource-fair secure two-party computation(2PC) protocol.

However, the plain GradRel construction is very expensive and doesn’t allow commitment to value 0. First, in GradRel construction, when commit to a value x, it need make five timed commitments to x\(_1\), x\(_2\), x\(_3\), x\(_4\) and y, where\{x\(_1\), x\(_2\), x\(_3\), x\(_4\)\} is a random permutation of \{ x,-x,xV,-xV \} (one of the four elements is a quadratic residue. V is an arbitrary element in \(Z^*_N\) with Jacobi symbol -1) and \(y \in \{1,4,9,16\}\) indicates which \(x_i\) is the \(x_i\) (\(y = i^2\) means that \(x_i = x\)), and it also need prove these commitments are consistent.
Second, in GradRel construction the committed value couldn’t be equal to 0, because the commitment \( z = (u(x))^y \cdot x = 0 \) if \( x = 0 \).

In this paper, we show a more efficient protocol securely realizing \( F_{\text{CPFO}} \) functionality based on a new variant of Garay et al. time-lines and simplified Camenisch-Shoup(sCS) commitment[9].

**Our Contribution.** First, in our construction, when commit to a value \( x \), we only need make one commitment to \( x \), instead of making five commitments. So the communication and computation complexity of our protocol are less than 1/5 of Garay et al. construction. Second, our construction allows commitment to value 0.

**Related works on fairness of secure MPC.** Fairness is a very desirable property for secure MPC protocols. Informally, a protocol is fair if either all the parties learn their (private) outputs, or none of them learns anything. However, 1986, Cleve[10] has proved that there doesn’t exist fair MPC protocols with corrupted majority. Then, many researchers turn to achieve some form of fairness. The first approach is the optimistic model which adds to the model a third party, where if fairness is breached the third party may be contacted to restore fairness. 2000, Cachin and Camenisch [11] designed an efficient optimistic fair secure 2PC protocol. The second approach uses a mechanism known as “gradual release”. In this approach, the output is released gradually with the property that if an abort occurs, then the adversary has not learned much more about the output than the honest parties. 2003, Pinkas[5] first presented a fair secure 2PC protocol based on a variant of Boneh and Naor timed commitment[3] that could be gradually opening. However, it couldn’t be proven secure according to the simulation paradigm. In order to resolve this problem of gradual release approach, 2006, Garay et al.[1] introduced the notion of resource-fair secure MPC protocol. The third approach is 1/p-secure computation[12], which requires only 1/p-indistinguishability rather than indistinguishability. 2009, Moran et al.[13] constructed an 1/p-secure two-party coin-flipping protocol. 2010, Beimel et al.[14] extended Moran et al. results to multi-party model when less than 2/3 of the parties are malicious. The same year, Gordon and J. Katz[15] constructed an 1/p-secure 2PC protocol.

2 Cryptographic Tools

2.1 Simplified Camenisch-Shoup (sCS) Commitment Scheme

sCS encryption scheme is a homomorphic variant of Camenisch-Shoup(CS) cryptosystem[16], which uses shorter keys and can be used as a commitment scheme(sCS commitment).

**Setup:** A trusted third party generates a safe RSA modulus \( N = pq \), where \( p \) and \( q \) are safe primes such that \( p = q = 3 \mod 4 \), \( p = 2p' + 1, q = 2q' + 1, \mid p \mid = \mid q \mid, p \neq q \), and \( p, q, p', q' \) are all primes, picks a random element \( g^* \in \mathbb{Z}_N^* \) and an element \( g = (g^*)^{2N} \). The common reference string is \((N, g)\), which also implicitly defines the element \( \alpha = 1 + N \). In the following of this paper, we treat all multiplications and exponentiations as operations in \( \mathbb{Z}_N^* \), unless stated otherwise.

**Commit phase.** Consider message \( m \) as an integer in \([0, N]\). Committer picks a random value
\( r \in [0, N/4], \) and computes \( c = sCSCom(m) = \alpha^m g^r \), then sends \( c \) to verifier.

**Open phase:** Committer sends verifier \((m, r)\). Verifier accepts if \( c = \alpha^m g^r \).

### 2.2 Time-lines

In Crypto 2000, Boneh and Naor [3] first constructed a timed commitments scheme under *generalized Blum-Blum-Shub (GBBS)* assumption. 2002, Garay and Jakobsson [4] introduced the notion of *reusable time-lines* which allow an amortization of the generation costs, and constructed time-lines by reusing the Boneh-Naor’s commitments. 2006, Garay *et al.* [1] gave a more efficient time-lines based on Garay and Jakobsson’s time-lines construction. Our time-lines follows Garay *et al.* construction, but has different modulus \((\text{mod } N^2)\).

**Setup:** Let \( \kappa \) be a positive integer representing a certain security parameter. For 80-bit security, one can take \( \kappa = 80 \). A trusted third party chooses \((N, g)\) like sCS commitment and does:

1. Compute the master time-line
   
   (1) Compute \( u_i = g^{2^{x_i}} \mod N^2 = \text{ReqSq}_{g^2, g} (2^\kappa - 2^\kappa - i) \) for \( 0 \leq i \leq \kappa \). This computation can be performed by computing \( a = 2^{x_i} - 2^{x_i} \mod \varphi(N^2) \) and then \( u_i = g^a \).

   (2) Generate a proof that \( u_i (0 \leq i \leq \kappa) \) are well-formedness, which we assume is non-interactive. For details of the proof, we refer to [5].

   (3) Output \( L = (N, g, \tilde{u}) \) (the master time-line), along with the above proof.

2. Compute the derived time-line
   
   (1) Pick a random value \( \alpha \in [0, N/4] \).

   (2) Set \( h = g^\alpha \), and compute a derived time-line \( L' = (N, h, \vec{v}) \), where \( v[i] = (u[i])^\alpha \) for \( 0 \leq i \leq \kappa \).

**Lemma 1 (Strong Pseudorandomness)[1]**: Let \( L = (N, g, \tilde{u}), L' = (N, h, \vec{v}) \) and \( \kappa \) be as the above. Let \( \delta \) be as in the GBBS assumption. Let \( \vec{w} \) be the vector containing the last \((l + 1)\) elements in \( \vec{v} \), i.e., \( \vec{w} = (v[\kappa - l], v[\kappa - l + 1], \ldots, v[\kappa]) \). Let \( A \) be a algorithm whose running time is bounded by \( \delta \cdot 2^l \) and \( R \) be a random element in \( Z_{N^2}^* \). Then, assuming the *composite decisional Diffie-Hellman assumption (CDDH)*[17] and GBBS hold, there exists a negligible function \( \varepsilon(\cdot) \) such that, for any \( A \),

\[
\Pr\left[A(N^2, g, \tilde{u}, h, \vec{w}, v[\kappa - l - 1]) = 1\right] - \Pr\left[A(N^2, g, \tilde{u}, h, \vec{w}, R^2) = 1\right] \leq \varepsilon(\kappa).
\]
3 A new protocol for \( F_{\text{CPPFO}} \) functionality

3.1 The Commit-prove-fair-open Functionality

The formal definition of \( F_{\text{CPPFO}} \) functionality is given in Figure 1, which combines the commit phase and prove phase of the \( F_{\text{CPPFO}} \) functionality in [1].

<table>
<thead>
<tr>
<th>Functionality ( F_{\text{CPPFO}}^R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_{\text{CPPFO}}^R ) is parameterized by a polynomial-time computable binary relation ( R ). It proceeds as follows, running with parties ( P_1, P_2, ..., P_n ) and an adversary ( S ).</td>
</tr>
</tbody>
</table>

**Commit and prove phase:** Receive message \((\text{commit}, sid, x_i, \overrightarrow{w_i})\) from every party \( P_i \), if \( R(\overrightarrow{w_i}, x_i) = 1 \), broadcast \((\text{RECEIPT \& PROOF}, sid, P_i, \overrightarrow{w_i})\) to all parties and \( S \).

**Open phase:** Wait to receive message \((\text{open}, sid)\) from party \( P_i \), \( 1 \leq i \leq n \), and a message \((\text{deliverat}, sid, s)\) from \( S \). As soon as all \( n \) open messages and the deliverat message are received, output \((\text{fairdeliver}, sid, 0, \{(DATA, (x_1, x_2, ..., x_n)), P_i\}_{1 \leq i \leq n} \cup \{(DATA, (x_1, x_2, ..., x_n)), S\}), s)\).

Figure 1: The commit-prove-fair-open functionality \( F_{\text{CPPFO}} \) with relation \( R \).

3.2 Realizing \( W(F_{\text{CPPFO}}) \): Protocol \( \text{NewGradRel} \)

Our new protocol, \( \text{NewGradRel} \), securely realizes wrapped functionality \( W(F_{\text{CPPFO}}) \) in the \((F_{\text{CRS}}, F_{\text{ZK}})\)-hybrid model, which is a new variant of Garay et al. construction[1] based on sCS commitment and the previous section time-lines. The general wrapper functionality \( W() \) [1] provides an interface to any functionality and \( W(F) \) is the resource fairness model of functionality \( F \). For the protocol construction, we need the \( F_{\text{ZK}} \) functionality[1] for the following relations.

**Diffie-hellman quadruple:** \( \text{DH}=\{(N^2,g,h,x,y), \gamma \} | h = g^\gamma \mod N^2 \land y = x^\gamma \mod N^2 \} \)

**sCS commitment equal:**

\[ \text{ComEq}=\{(N^2,g,h,u,c), (\gamma,m) \} | h = g^\gamma \mod N^2 \land \alpha^m = c / u^\gamma \mod N^2 \} \]

The detailed protocol is as follows.

**Setup:** Compute the common reference string \( L=(N,g, \overline{u} = \{u_i\}_{i=1}^r) \) as the previous section time-lines.

**Round 1 (Commit and Prove phase).** For each party \( P_i \), \( 1 \leq i \leq n \), upon receiving input \((\text{commit}, sid, x_i)\) from the environment, performs the following steps:

1. Pick a random value \( \gamma_i \in [0, N/4] \).
(2) Set \( h_i = g^{v_i} \), and compute a derived time-line \( L_i = (N, h_i, \overline{v_i}) \), where \( v[j] = u[j]^{\gamma_j}, 1 \leq j \leq \kappa \).

(3) Commit to \( x_i \in [0, N] \), \( c_i = sCSCom(x_i) = \alpha^x \cdot v_i[\kappa] = \alpha^x \cdot (u[\kappa])^{\gamma_i} \). Broadcast message (\text{COMMIT,} \text{sid,} \text{P}_i, h_i, c_i)\).

(4) Send message (\text{zk-prove,} \text{sid,} 0, \text{P}_i, (N^2, g, h_u, u[\kappa], c_i, (\gamma_i, x_i)) \) to the \( F_{ZK}^{ComEq} \) functionality.

(5) After receiving messages (\text{ZK-PROVE,} \text{sid,} 0, \text{P}_i, (N^2, g, h_u, u[\kappa], c_i)) from \( F_{ZK}^{ComEq} \), broadcast (\text{RECEIPT&PROOF,} \text{sid,} \text{P}_i, h_i, c_i)\).

**Round** \( r = 2, \ldots, \kappa + 1 \) (Open phase). Let \( l = r - 1 \). For each party \( \text{P}_i, 1 \leq i \leq n \), does:

(1) Broadcast (\text{RELEASE,} \text{sid,} \text{v}_i[l]) and send message (\text{zk-prove,} \text{sid,} r, (N^2, g, h_u, u[l],

\text{v}_i[l]), \text{\gamma}_i) \) to the functionality \( F_{ZK}^{DH} \).

(2) After receiving all \( n \) \text{RELEASE} and \text{ZK-PROOF} messages, proceed to the next round. Otherwise, if any of the broadcast messages is missing, go to panic mode.

At the end of round \((\kappa + 1)\), compute \( x_j = m_j = (c_j \cdot (v_j[\kappa])^{-1})^2 \), \( m_j = (m_j - 1) / N \) (over the integers), \( x_j = m_j / 2 \mod N \), for \( 1 \leq j \leq n \). Output(\text{DATA,} \text{sid,} x_1, \ldots, x_n) and terminate.

**Panic mode**: For each party \( \text{P}_i, 1 \leq i \leq n \), does:

- Send (\text{dealoffer,} \text{sid,} \phi, n \cdot \delta \cdot 2^{\kappa-1}) to the environment.

- If the environment responds with (\text{dealaccept,} \text{sid,} \phi), use \text{v}_i[l - 1] from the previous round to compute \( v_i[\kappa] \) as \( v_i[\kappa] = \text{RepSq}_{N, v_i[l-1]}(2^{\kappa-1} - 1) \), then compute \( x_j \) as the above method, for \( 1 \leq j \leq n \). Output(\text{DATA,} \text{sid,} x_1, \ldots, x_n) in round \((\kappa + 1)\) and terminate.

- Otherwise, output \( \bot \) in round \((\kappa + 1)\) and terminate.

**Theorem 2.** Under the decision composite residuosity(DCR)[18], strong RSA and GBBS assumption, the above protocol \text{NewGradRel} securely realizes the ideal functionality \( W(F_{CPF}) \) in the \( (F_{CRS}, F_{ZK}^{DH}, F_{ZK}^{ComEq}) \)–hybrid model, assuming static corruptions.

**Proof.**

Let \( A \) be a \( t \)-bounded adversary that operates against protocol \text{NewGradRel}. We construct an ideal adversary \( S \) with access to \( W(F_{CPF}) \), which simulates a real execution of protocol \text{NewGradRel} with \( A \) such that for all \( t \)-bounded environment \( Z \) and adversary \( A \), we have

\[
HYB_{NewGradRel,A,Z}(F_{CRS}, F_{ZK}^{DH}, F_{ZK}^{ComEq}) \approx IDEAL_{W(F_{CPF}), S^{4}(t), Z}.
\]
First, we assume that \( W(\text{CPFO}) \) uses the same relation \( \text{ComEq} \) of \text{NewGradRel} as its parameterized relation \( \text{R} \), and it has access to \( \text{F}_{\text{CRS}} \).

Recall that \( S \) interacts with the ideal functionality \( W(\text{CPFO}) \) and with the environment \( Z \). The ideal adversary \( S \) starts by invoking a copy of \( A \) and running a simulated interaction of \( A \) with the environment \( Z \) and parties running the protocol. Messages received from \( Z \) are forwarded to the simulated \( A \), and messages sent by the simulated \( A \) to its environment are forwarded to \( Z \). Furthermore, \( S \) also plays the roles of the ideal \( F_{\text{DK}} \) and \( F_{\text{ZK}} \) functionalities. \( S \) proceeds as follows:

**Initialization step:** \( S \) simulates \( F_{\text{CRS}} \), i.e., \( S \) chooses the common reference string \( L=(N, g, \tilde{u} = \{u_i\}^\kappa) \) as in the real protocol. Then \( S \) sets \( \Lambda = \varphi(N^2) \) (since \( S \) generates \( N \), it knows the factorization of \( N \)).

**Commitment and Proof by an uncorrupted party:** When \( S \) sees a broadcast message \((\text{RECEIPT} \& \text{PROOF}, sid, P_i,h_i,c_i)\) From \( W(\text{CPFO}) \), it means that an uncorrupted party \( P_i \) has committed to a value. \( S \) then simulates the two broadcast messages in the real world:

\[ \text{(COMMIT}, sid, P_i,h_i,c_i) \text{ from } P_i \text{ and } (\text{ZK-PROVE}, sid, 0, P_i, (N^2, g, h_i, u[l], \kappa, l, c_i)) \text{ From } F_{\text{ZK}} \text{ComEq}. \]

**Commitment and Proof by a corrupted party:** When \( S \) sees a broadcast message \((\text{COMMIT} }, sid, P_i,h_i,c_i) \) from a corrupted party \( P_i \) (controlled by \( A \)), it means that \( P_i \) is committing to a value. Then, \( S \) acts as the \( F_{\text{ZK}} \text{ComEq} \) functionality and expects the message \((\text{zk-prove}, sid, 0, (N^2, g, h_i, u[l], \kappa, l, c_i), (\gamma, r, x)) \). If it is received and verified, \( S \) sends message \((\text{commit, sid, } x_i, h_i, c_i) \) to \( W(\text{CPFO}) \) on behalf of \( P_i \).

**Simulating the open phase:** In the open phase, \( S \) first sends message \((\text{deliverat, sid, } (\kappa + 1)) \) to \( W(\text{CPFO}) \). Then, \( S \) simulates the gradual opening of the uncorrupted parties. Let

\[ m = \kappa - \left\lfloor \log_2 \left( \frac{t}{\delta} \right) \right\rfloor - 1 \]

then \( S \) behaves differently in the first \( m-1 \) rounds of the open phase from the last \( \kappa - m + 1 \) rounds; the difference lies in the release value used in simulating the uncorrupted parties (i.e., the value \( x \) in the message \((\text{RELEASE, sid, } P_i, x) \) sent by the uncorrupted parties).

- In the first \( m-1 \) rounds of the open phase, \( S \) simply uses a random value each time for the value being released. For each uncorrupted party \( P_i \), \( S \) randomly generates \( v_i[l] \in \mathcal{Q}_{N^2} \) (\( \mathcal{Q}_{N^2} \) denotes the quadratic residues modulo \( N^2 \)) and fakes two broadcast messages: \((\text{RELEASE, sid, } P_i, v_i[l]) \) from \( P_i \) and \((\text{ZK-PROVE, sid, } P_i, (N^2, g, h_i, u[l], v_i[l]) \)) from ideal functionality \( F_{\text{DK}} \). Then, \( S \) waits to receive the release messages from all the corrupted parties, as well as their zk-prove messages. \( S \) proceeds to the next round if all the anticipated messages are received and verified. If any of the messages is missing, or any of the proofs is incorrect, \( S \) sends \((\text{noinvest, sid, } 0) \) to \( W(\text{CPFO}) \) and goes to panic mode.
At round $m$ of the open phase, $S$ switches its strategy. $S$ needs to produce a correct commitment because from this round on the adversary may be able to force open the commitment. So $S$ should find the openings in this round. First, $S$ sends the messages (open, sid) to $W(F_{CPFO})$ on behalf of every corrupted party $P_i$, and then sends (invest, sid, 0, $n \cdot 2^{x-m+1}$) to $W(F_{CPFO})$. It then immediately receives the opening of all the committed values in the message (DATA, sid, $x_1, x_2, \ldots, x_n$) from $W(F_{CPFO})$. Once $S$ knows the committed value $x_i$ from uncorrupted $P_i$, $S$ can now produce a “real” derived time-line for $P_i$ backward, which is consistent with $x_i$. We know that the end point of the time-line must be $c_i/\alpha^x$ which is a quadratic residue (the proof is shown in the Appendix A), and thus the other points should be the roots of it. So, for each uncorrupted party $P_i$, $S$ can compute $w_i = \left( c_i/\alpha^x \right) \left( 2^{x-m} \right)^{\mod \Lambda}$, which is the $2^{x-m}$ th root of $c_i/\alpha^x$. Then $S$ fakes broadcast messages (RELEASE, sid, $P_i,w_i$) from $P_i$ and (ZK-PROVE, sid,m, $P_i(\ N^2,g,h_i, u[l], w_i))$ from $F_{ZK}^{DH}$. Then, $S$ waits to receive the release messages from all the corrupted parties, as well as their zk-prove messages. As in the previous rounds, it proceeds to the next round if all the messages are received and verified. Otherwise $S$ goes to the panic round.

From round $m$ on, $S$ simulates the gradual opening using the time-line generated in round $m$, i.e., $S$ sends the message (RELEASE, sid, $P_i$, ReqSq$_{N^2,m_i}(2^{x-m} - 2^{x-i})$) from $P_i$ and simulates the corresponding messages from $F_{ZK}^{DH}$ in the $l$th round. Then, $S$ acts as in the previous rounds.

Panic mode: Here $S$ simulates each uncorrupted party asking for a deal. For an uncorrupted party $P_i$, $S$ sends (dealoffer, sid, $P_i$, 0, $n \cdot 2^{x-m+1} - 1$) to $W(F_{CPFO})$. After this, $S$ simply forwards messages appropriately between $W(F_{CPFO})$, $P_i$, and the environment.

Finally, $S$ outputs what the simulated $A$ outputs.

We now prove that $Z$ cannot distinguish an interaction of Protocol NewGradRel with $A$ from an interaction in the ideal process with $W(F_{CPFO})$ and $S$. In order to show this, we examine several hybrid experiments:

(I) Real interaction: This is the interaction of $Z$ with $A$ and Protocol NewGradRel.

(II) Real interaction with the simulated ZK functionalities: This is the interaction of $Z$ with $S$, as described above, except that: (1) it does not simulate the CRS functionality, but instead has access to $F_{CRS}$; (2) it also emulates the $W(F_{CPFO})$ functionality; (3) Finally, it behaves differently from $S$ when simulating the commit and open phases for an uncorrupted party. When an uncorrupted party $P_i$ sends a message (commit, sid, $x_i,h_i,c_i$) on to $W(F_{CPFO})$, it simulates the actual NewGradRel protocol for $P_i$. In the open phase, it sends release messages as in the real-world experiment (instead of revealing random values).

(III) Simulated interaction with access to $F_{CRS}$: This is the same as (II), except that it differs from (II) in that it behaves like (I) in the commit phase, and also in the open phase
until step $m$. That is, it commits to random numbers until step $m$, and then commits to the “correct” values as in the real-world experiment.

(IV) Simulated interaction: This is the interaction of $Z$ with $S$, as described above.

Our aim is to show that interactions (I) and (IV) are indistinguishable to $Z$, or in other words that $Z$’s output at the end of interaction (I) deviates only negligibly from $Z$’s output at the end of interaction (IV). We prove this by showing that each consecutive pair of interactions are indistinguishable to $Z$. (We use the term “distribution i” to denote both “interaction i”, and “$Z$’s output from “interaction i”.)

The fact that distributions (I) and (II) are identical. This follows from the fact that the simulated ZK functionalities behave exactly the same as the actual ZK functionalities; and given perfect ZK functionalities, the outputs of the honest parties in (II) are exactly what they would be in (I), since the committed values that are opened in (I) must be exactly the same as what was extracted from the ZK functionalities (II).

Next, we consider the hybrid experiments (II) and (III). The difference between these experiments is as follows. In the (III) experiment, the first $(m - I)$ points of each uncorrupted party’s time-line are chosen by random quadratic residues, the last $(\kappa - m + 1)$ points of each time-line are real and consistent with the committed value of each uncorrupted party. In the (II) experiment, the produced points of time-line are real. So the difference lies in the prefixes of the first $(m - I)$ points of time-lines of the uncorrupted parties. Then one can easily reduce the indistinguishability between the two experiments to the strong pseudorandomness of time-lines (Lemma 1) via a standard hybrid argument.

Finally, distributions (III) and (IV) are identical. This follows from the fact that the distributions of $F_{CRS}$ and the simulated CRS functionality are the same, the distribution of $h_i,c_i$ values produced by simulated honest parties is the same, and that the values produced in the open phase are the same. ■

4 Conclusion

Based on the time-lines and sCS commitment, we show a more efficient resource-fair protocol securely realizing $F_{CPFO}$ functionality, which improves Garay et al. construction and allows commitment to value 0.

References


A. Proof of $c_i \equiv \alpha^x \in QR_{N^2}$

Note that $N = pq, \ p = 2p^i + 1, q = 2q^i + 1$ (p, q, p’, q’ are all primes), and $\alpha = 1 + N$.

(1) We know that $\alpha^x = 1 + x, N$.

(a) $(\alpha^x)^{(p-1)/2} \mod p = (1 + x, N)^{(2p+1)/2} = (1 + x, pq)^p \equiv 1 \mod p$, so $\alpha^x \in QR_p$.

(b) The same way, we can learn $\alpha^x \in QR_q$.

Then, we get that $\alpha^x \in QR_{N^2}$ ($N^2 = p^2 \cdot q^2$).

(2) $c_i$, which $S$ gets from $W(F_{CPPO})$ is the commitment of an uncorrupted party $P_i$ computes $c_i = \alpha^x \cdot \nu_s[k] = \alpha^x \cdot (u[k])^{y_1}$.
\[(a) (u[k])^\gamma_i = (g^{2^\gamma_i})^\gamma_i = ((g^{\gamma_i})^{2^\gamma_i})^2, \text{ so } (u[k])^\gamma_i \in QR_{N^2}\]

From (a) and (1), we get that \( c_i \in QR_{N^2}. \)

From (1) and (2), we get the conclusion that \( c_i / \alpha^x \in QR_{N^2}. \)