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

Ou Ruan [a,b], Cai Fu [a,*] , 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 F_{CPFO} 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"(F_{CPFO}) 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 ($y = i^2$ 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 $z = (u[\kappa])^{\alpha} \cdot x$ (=0) if x=0.

In this paper, we show a more efficient protocol securely realizing F_{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 commiments).

Setup:A trusted third party generates a safe RSA modulus N= pq, where p and q are *safe primes* such that p=q= $3 \bmod 4$, $p = 2p'+1, q = 2q'+1, |p| = |q|, p \neq q$, and p, q, p', q' are all primes, picks a random element $g' \in Z_{N^2}^{*}$ 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 $Z_{N^2}^{*}$, 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)[3] 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(mod N^2).

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

1.Compute the master time-line

(1) Compute $\mathrm{u_i} = \mathrm{g}^{2^{2^{\kappa}-2^{\kappa-i}}} \bmod N^2 = \mathrm{ReqSq}_{N^2,g}(2^{\kappa} - 2^{\kappa-i})$ for $0 \le i \le \kappa$. This computation can be performed by computing $\mathrm{a} = 2^{2^{\kappa}-2^{\kappa-i}} \bmod \varphi(N^2)$ and then $\mathrm{u_i} = \mathrm{g^a}$.

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

(3) Output L=(*N*, *g*, $\vec{u}$) (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 $\mathrm{h} = \mathrm{g}^{\alpha}$, and compute a derived time-line L'=(N, *h*, $\vec{v}$),where $v[i] = (u[i])^{\alpha}$ for $0 \le i \le \kappa$.

Lemma 1 (Strong Pseudorandomness)[1] :Let L=(*N*, *g*, $\vec{u}$), L'=(*N*, *h*, $\vec{v}$) and κ be as the above. Let δ 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***,

$$\left|\Pr\left[A(N^2, g, \vec{u}, h, \vec{w}, v[\kappa - l - 1]) = 1\right] - \Pr\left[A(N^2, g, \vec{u}, h, \vec{w}, R^2) = 1\right]\right| \le \varepsilon(\kappa).$$

3 A new protocol for F_{CPFO} functionality

3.1 The Commit-prove-fair-open Functionality

The formal definition of F_{CPFO} functionality is given in Figure 1,which combine the commit phase and prove phase of the F_{CPFO} functionality in [1].

Functionality F^{R}_{CPFO}

F^{R}_{CPFO} 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.

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

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

Figure 1: The commit-prove-fair-open functionality F_{CPFO} with relation R.

3.2 Realizing $W\left(F_{CPFO}\right)$: Protocol NewGradRel

Our new protocol, **NewGradRel,** securely realizes wrapped functionality $W(F_{CPFO})$ in the (F_{CRS}, F_{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 gereral *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_{ZK} functionality[1] for the following relations.

Diffie-hellman quadruple:**DH**={ $((N^2, g, h, x, y), \gamma) \mid h = g^{\gamma} \bmod N^2 \wedge y = x^{\gamma} \bmod N^2 \mid$}

sCS commitment equal:

ComEq= { $((N^2, g, h, u, c), (\gamma, m)) \mid h = g^{\gamma} \bmod N^2 \wedge \alpha^{m} = c / u^{\gamma} \bmod N^2$ }

The detailed protocol is as follow.

Setup: Compute the common reference string L=(*N*, *g*, $\vec{u} = \{\mathrm{u}_i\}_1^{\kappa}$) as the previous section time-lines.

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

(1) Pick a random value $\gamma_i \in [0, N/4]$.

(2) Set $h_i = g^{\gamma_i}$, and compute a derived time-line $L_i=(N, h_i, \vec{v_i})$,where $v[j]_i = u[j]^{\gamma_i}, 1 \le j \le \kappa$.

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

(4)Send message (**zk-prove**, *sid*,0, P_i, (N^2,g,h_i,$u[\kappa]$,c_i),(γ_i, x_i)) to the F_{ZK}^{ComEq} functionality.

(5)After receiving messages (Z**K-PROVE**, *sid*,0, P_i,(N^2,g,h_i, $u[\kappa]$,c_i)) from F_{ZK}^{ComEq},broadcast (**RECEIPT&PROOF**, *sid*, P_i,h_i,c_i).

Round $r = 2,\ldots,\kappa+1$(**Open phase**). Let $l = r-1$. For each party P_i, $1 \le i \le n$,does:

(1) Broadcast (**RELEASE**,*sid*, $v_i[l]$) and send message (**zk-prove**, *sid*,*r*, (N^2,g,h_i,$u[l]$, $v_i[l]$), γ_i) to the functionality F_{ZK}^{DH} .

(2) After receiving all *n* **RELEASE** and **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(κ+1), compute x_j as $m_j = (c_j \cdot (v_j[\kappa])^{-1})^2$, $m_j' = (m_j - 1)/N$ (over the integers), $x_j = m_j'/2 \bmod N$, for $1 \le j \le n$.Output(**DATA**, *sid*, $x_1,\ldots,x_n$) and terminate.

Panic mode: For each party P_i, $1 \le i \le n$,does:

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

- If the environment responds with (**dealaccept**, *sid*,ϕ), use $v_j[l-1]$ from the previous round to compute $v_j[\kappa]$ as $v_j[\kappa] = \mathrm{RepSq}_{N^2, v_j[l-1]}(2^{\kappa-l+1}-1)$,then compute x_j as the above method, for $1 \le j \le n$.Output(**DATA**, *sid*, $x_1,\ldots,x_n$) in round(κ+1) and terminate.

- Otherwise,output $\perp$ in round(κ+1) and terminate.

Theorem 2. Under the decision composite residuosity(DCR)[18], strong RSA and GBBS assumption, the above protocol **NewGradRel** securely realizes the ideal functionality W(F_{CPFO}) 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 **NewGradRel**. We construct an ideal adversary ***S*** with access to W(F_{CPFO}),which simulates a real execution of protocol **NewGradRel** with ***A*** such that for all *t*-bounded environment ***Z*** and adversary ***A***, we have

$$HYB_{NewGrad\,\mathrm{Re}l,A,Z}^{(F_{CRS},F_{ZK}^{DH},F_{ZK}^{ComEq})} \approx IDEAL_{W(F_{CPFO}),S^A(t),Z}\,.$$

First,we assume that W(F_{CPFO}) uses the same relation **ComEq** of **NewGradRel** as its parameterized relation R,and it has access to F_{CRS}.

Recall that ***S*** interacts with the ideal functionality W(F_{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_{ZK}^{DH} and F_{ZK}^{ComEq} functionalities. ***S*** proceeds as follows:

Initialization step: ***S*** simulates F_{CRS},***i.e.,*** ***S*** chooses the common reference string L=(*N*, *g*, $\bar{u}=\{\mathrm{u_i}\}_1^{\kappa}$) 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 (**RECEIPT&PROOF**, *sid*, P_i,h_i,c_i) From W(F_{CPFO}), it means that an uncorrupted party P_i has committed to a value. ***S*** then simulates the two broadcast message in the real world: (**COMMIT**,*sid*, P_i,h_i,c_i)from P_i,and (**ZK-PROVE**, *sid*,0, P_i,(N^2,g,h_i,$u[\kappa]$,c_i)) From F_{ZK}^{ComEq} .

Commitment and Proof by a corrupted party: When ***S*** sees a broadcast message (**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_{ZK}^{ComEq} functionality and expects the message ((**zk-prove**,*sid*,0,(N^2,g,h_i, $u[\kappa]$, c_i), (γ_i, x_i)).If it is received and verified,***S*** sends message (**commit**, *sid*, x_i,h_i,c_i) to W(F_{CPFO}) on behalf of P_i.

Simulating the open phase: In the open phase, ***S*** first sends message (***deliverat***, *sid*, (κ+1)) to W(F_{CPFO}). Then,***S*** simulates the gradual opening of the uncorrupted parties.Let $m=\kappa-\left\lfloor \log_2(\frac{t}{\delta}) \right\rfloor-1$. ***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 (**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 $v_i[l]\in QN_{N^2}$ (QN_{N^2} denotes the quadratic residues modulo N^2) and fakes two broadcast messages: (**RELEASE**,*sid*, P_i, $v_i[l]$) from P_i and (**ZK-PROVE**, *sid*,*r*, P_i,(N^2,g,h_i,$u[l]$, $v_i[l]$)) from ideal functionality F_{ZK}^{DH} .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 (**noinvest**, *sid*, 0) to W(F_{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_j , and then sends (**invest**, *sid*, 0, $n\cdot 2^{\kappa-m+1}$) to W(F_{CPFO}). It then immediately receives the opening of all the committed values in the message (**DATA**, *sid*, x_1, x_2, ..., 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/α^{x_i} which is a quadratioc 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 $w_i=\left(c_i/\alpha^{x_i}\right)^{(2^{1-2^{\kappa-m}})\bmod\Lambda}$, which is the $2^{2^{\kappa-m}-1}$th root of c_i/α^{x_i}. 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, $\mathrm{ReqSq}_{N^2,w_i}(2^{\kappa-m}-2^{\kappa-l})$)) 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, $n\cdot 2^{\kappa-m+1}-1$) 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 $\boldsymbol{Z}$ with $\boldsymbol{S}$, as described above.

Our aim is to show that interactions (I) and (IV) are indistinguishable to $\boldsymbol{Z}$, or in other words that $\boldsymbol{Z}$'s output at the end of interaction (I) deviates only negligibly from $\boldsymbol{Z}$'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 $(m - 1)$ 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 $(m - 1)$ 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

[1] J. Garay, P. MacKenzie, M. Prabhakaran, K. Yang,Resource Fairness and Composability of Cryptographic Protocols, in: S. Halevi and T. Rabin (Eds.), TCC 2006. LNCS,vol. 3876, Springer, Heidelberg ,2006, pp. 404-428.

[2] R. Canetti, Security and Composition of Multiparty Cryptographic Protocols, J CRYPTOL.13(1)(2000) 143-202.

[3] D. Boneh, M. Naor, Timed Commitments, in: Bellare M.(Ed.), CRYPTO 2000, LNCS,vol. 1880, Springer, Heidelberg,2000, pp. 236-254.

[4] J. Garay, M. Jakobsson, Timed-Release of Standard Digital Signatures, in: M. Blaze(Ed.), Financial Crypto'02. LNCS, vol.2357, Springer, Heidelberg,2002, pp. 168-182.

[5] Pinkas, B., Fair Secure Two-Party Computation, in: Biham, E. (Ed.), EUROCRYPT 2003, LNCS, vol. 2656, Springer, Heidelberg ,2003, pp. 87-105.

[6] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally Composable Two-party and

Multi-party Secure Computation, in: John H. Reif (Ed.) ,34th ACM Symposium on the Theory of Computing, ACM Press, New York, 2002, pp.494–503.

[7] R. Cramer, I. Damgård, J. Nielsen, Multiparty Computation from Threshold Homomorphic Encryption, in: Pfitzmann B. (Ed.), EUROCRYPT 2001, LNCS,vol.2045, Springer, Heidelberg ,2001, pp. 280–300.

[8] Mehmet S.Kiraz,Berry Schoenmakers,An Efficient Protocol for Fair Secure Two-Party Computation, in: T. Malkin (Ed.), CT-RSA 2008, LNCS, vol. 4964, Springer, Heidelberg,2008, pp.88-105.

[9] Stanisław Jarecki,Vitaly Shmatikov, Efficient Two-Party Secure Computation on Committed Inputs, in: Naor M. (Ed.), EUROCRYPT 2007,LNCS,vol.4515, Springer, Heidelberg ,2007, pp. 97-114.

[10] R. Cleve, Limits on the security of coin flips when half the processors are faulty, in: Juris Hartmanis (Ed.),Proceedings of the 18thAnnual ACM Symposium on Theory of Computing, , ACM Press, New York, 1986,pp. 364-369.

[11] Cachin C., Camenisch J.,Optimistic Fair Secure Computation, in: Bellare M.(Ed.), CRYPTO 2000. LNCS, vol. 1880, Springer, Heidelberg ,2000, pp. 93-111.

[12] J. Katz, On achieving the "best of both worlds" in secure multiparty computation, in: D.S. Johnson, U. Feige(Eds.), Proceedings of the 39 annual ACM symposium on Theory of computing. ACM Press,New York,2007,pp.11-20.

[13] T. Moran, M. Naor, G. Segev, An optimally fair coin toss, in:O. Reingold(Ed.), TCC 2009, LNCS, vol. 5444, Springer Heidelberg ,2009,pp.1-18.

[14] Amos Beimel,Eran Omri,Ilan Orlov, Protocols for Multiparty Coin Toss With Dishonest Majority, in: T. Rabin(Ed.), Crypto 2010, LNCS,vol. 6223, Springer, Heidelberg,2010,pp. 538-557.

[15] D.Gordon, J. Katz, Partial Fairness in Secure Two-Party Computation, in: H. Gilbert(Ed.), EUROCRYPT 2010, LNCS,vol.6110, Springer ,Heidelberg ,2010,pp.157-176.

[16] J. Camenisch , V. Shoup, Practical verifiable encryption and decryption of discrete logarithms, in: D. Boneh (Ed.),CRYPTO 2003. LNCS,vol. 2729, Springer, Heidelberg,2003, pp. 126-144.

[17] D. Boneh, The decision Diffie-Hellman problem, in: Buhler JP (Ed.),Proceedings of the Third Algorithmic Number Theory Symposium,LNCS,vol.1423, Springer, Heidelberg, 1998,pp.48–63.

[18] Paillier P,Public-key cryptosystema based on composite degree residue classes,in:Michael Wiener(Ed.),EuroCrypt'99, LNCS, vol. 1592,Springer, Heidelberg,1999,pp.223-238.

A. Proof of $c_i / \alpha^{x_i} \in QR_{N^2}$

Note that N = pq, $p = 2p'+1$, $q = 2q'+1$ (p, q, p', q' are all primes),and α =1+N.

(1) We know that $\alpha^{x_i} = 1 + x_i N$.

(a) $(\alpha^{x_i})^{(p-1)/2} \bmod p = (1+x_i N)^{((2p'+1)-1)/2} = (1+x_i pq)^{p'} = 1 \bmod p$, So, $\alpha^{x} \in QR_p$.

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

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

(2) c_i which $\boldsymbol{S}$ gets from W(F$_{\text{CPFO}}$) is the commitment of an uncorrupted party P$_i$ computes $c_i = \alpha^{x_i} \cdot v_i[\kappa] = \alpha^{x_i} \cdot (u[\kappa])^{\gamma_i}$.

(a) $\left(u[\kappa]\right)^{\gamma_i} = (g^{2^{2^{\kappa}-1}})^{\gamma_i} = ((g^{\gamma_i})^{2^{2^{\kappa}-2}})^2$,so $\left(u[\kappa]\right)^{\gamma_i} \in QR_{N^2}$

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

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