Collusion Free Protocol for Correlated Element Selection Problem

Amjed Shareef[1], Akshay Agrawal[2] and C. Pandu Rangan[1]
Department of Computer Science and Engineering,
[1] Indian Institute of Technology Madras, Chennai, India.
[2] National Institute of Technology Tiruchirappalli, India.
Email: {amjedshareef,akshay.agrawal1989}@gmail.com,
rangan@cse.iitm.ac.in

Abstract. A common problem in many markets is that competing firms cannot plan joint business strategies which are socially beneficial, as each firm has its own preferable business strategy which would yield higher profits for it and lower profits for the others. The solution to this problem becomes complex because each firm need not stick to its commitment to follow the pre-designated strategy. Game theory suggests to us a way to enforce this commitment, as when every player chooses his actions according to his observation of the value of a common public signal and, assuming that the others do not deviate, no player is willing to deviate from his recommended strategy. The players do not deviate from their recommended strategy as playing them would yield them a much higher expected pay-off than playing individually. The common public channel can be a trusted external mediator which may send each player his recommended strategy. This mediator can be simulated by a cryptographic protocol, which all the players agree to implement. This problem of suggesting the protocol is known as the *Correlated Element Selection Problem*. The first two-player protocol was proposed by Dodis et. al[1] in Crypto 2000. The extension of the two-player protocol to an n-player protocol is highly prone to collusions, as two firms can collude and cheat the rest of the firms. The main contribution of the paper is the first n-player collusion free protocol for the *correlated element selection problem* that does not use hardware primitives. We assume that players are honest but curious.

1 Introduction

The notion of correlated equilibrium was first introduced by Robert Aumann in 1974[2]. Each player chooses his action according to the mediator or the protocol. The distribution of the strategies to the players is called correlated if no player wants to deviate from the recommended strategy provided others do not deviate.

We consider an example for a two player game used by Vanessa Teague in [3]. The game has two competing firms in the same area. Each week, the CEOs of both the firms decide whether to have a sale or not. If one firm decides to have a sale and the other decides on having a no sale, then the first firm gets many customers and gains a high profit. At the same time the second firm gets much less customers and their profit goes down. Similarly, if both the firms decide on no sale, then the profit got by them is the usual one. But if both the firms decide to have a sale, then both of them get the same number of customers with the low prices. Hence, the profit of both the firms goes down significantly. The incentive from both the firms is to offer a sale but only when they are very confident that the other is not going to offer one. But if they do not have any information about the other firm, the state of (No sale, No sale) gives them the maximum average pay-off. The pay-off matrix for this game can be given as follows :

Pay-off table

	No Sale	Sale
No Sale	$(9, 9)$	$(5, 12)$
Sale	$(12, 5)$	$(0, 0)$

Probability Distribution

	No Sale	Sale
No Sale	$1/3$	$1/3$
Sale	$1/3$	0

As it is evident from the table, the strategy profile (Sale,Sale) is the worst for both the players and hence, they would like to avoid playing that strategy profile. But, as both the players are *rational* and would like to get maximum advantage, they would prefer that the other player plays No Sale so that he can play Sale. Thus, there is a problem in co-ordination between the players. This can be solved with the help of a trusted third party or, in other words, a mediator. The mediator will at random choose one of the three strategy profiles from (N Sale, No Sale), (Sale, No Sale), (No Sale, Sale). Thus, eliminating the chances of choosing the (Sale, Sale) strategy set. This assumes a uniform probability distribution as shown above. Thus, we increase the overall expected pay-off of each player. Each player does not know about the strategy of the other player. Therefore, there will be no tendency to deviate from his recommended strategy. This is a good example of correlated equilibrium. For a n player game, we refer to $\sigma^*_{-1}|a^*_1$ as the probability distribution on the action recommended to all the players other than p_1, subjected to the condition that the recommended action of player 1 is a^*_1. Thus, $u_1(a_1, \sigma^*_{-1}|a^*_1)$ is player 1's expected utility for choosing action a_1, assuming that all the other players act according to the distribution $\sigma^*_{-1}|a^*_1$. Similarly, for any player p_i, the expected utility for any strategy a_i is given by $u_i(a_i, \sigma^*_{-i}|a^*_i)$. Thus, we define the correlated equilibrium as follow :

Definition 1. ***(Correlated Equilibrium):*** *A correlated equilibrium for a game $\Gamma(n, L, U)$, is a strategy set $S^* = (\sigma^*_1, \sigma^*_2, ..., \sigma^*_n)$ such that for any $(a^*_1, a^*_2, ..., a^*_n) \in S^*$, for any $a_1 \in L_1,\ a_2 \in L_2, ..., a_n \in L_n$, all the following identities hold true:*

$$u_1(a^*_1, \sigma^*_{-1}|a^*_1) \geq u_1(a_1, \sigma^*_{-1}|a^*_1),\ u_2(a^*_2, \sigma^*_{-2}|a^*_2) \geq u_2(a_2, \sigma^*_{-2}|a^*_2), \ldots,$$
$$u_n(a^*_n, \sigma^*_{-n}|a^*_n) \geq u_n(a_n, \sigma^*_{-n}|a^*_n)$$

The question that arises from the above explanation is that, is there any other method to attain the increased expected pay-offs without the external mediator ? The concept of cryptography provides a solution to this question. If the players are allowed to communicate and coordinate among themselves before the start of the game, then there is a possibility that they agree on a correlated strategy profile without revealing any information about their own strategy. This can be helped by the execution of a pre-decided protocol by each player. The suggested protocol must be able to simulate the mediator. It is also a Nash Equilibrium for the players to execute the given protocol, as their expected pay-off will be better by executing the move recommended by the protocol. Thus, no player has a motivation to deviate from the protocol when all the other players also adhere to it. The problem of replacing the mediator with a set of rules is known as the correlated element selection problem[1].

The *correlated element selection problem*[1] is defined as follows. There are n players, $p_1, p_2 \ldots p_n$, with a common m list of tuples $\{(a_{i1}, a_{i2}, ..., a_{in})/i = 1 \textit{ to } m\}$ and they need to jointly choose a random tuple number r, such that player p_j only learns the value a_{rj}, not even the random tuple number r. It means that, every player p_j only learns the value a_{rj}. If some player learns the tuple number r, then it reveals to him the other players' strategies. This problem holds a huge significance, as we can find many analogies to this problem in the real world.

Motivation and significance of the problem:
When there are multiple players, some subset of the players can come together, collude, and try to get a significant advantage over the other players. Let us take the case of the previous example and consider that there are n firms and at any point of time only k, $k < n$, firms can announce a discount. The solution to the problem is to suggest, in a completely random manner, which set of players can announce a discount sale. If any two players collude and are able to get a chance to announce a discount sale, with some non-negligible probability, then the protocol is not collusion free. The only successful attempts to suggest a collusion free protocol are made in [4, 5] with the use of hardware primitives.

We assume, all the players agree to follow the protocol and anyone caught not following it can be punished by aborting the protocol. The players communicate via messages. The protocol shows how some basic *Cryptography*, *Blindable Encryption* and *Commitment* tools can be utilized to implement the correlated element selection problem. In the next section we present the related work towards solving the problem and our contributions. In section 3, we briefly narrate some basic game theory concepts and cryptographic primitives used in this paper. We then discuss some unsuccessful models to solve the problem in section 4. Our protocol is discussed in detail in section 5 with section 6 concluding the paper and showing a direction for further work.

2 Related Work and Contributions

The correlated element selection problem was first solved by Barany [6] for $n \geq 4$, where n is the number of players. The protocol has two players performing the role of the preparer. The protocol was efficient but it was not a collusion free protocol, i.e., if any two players collude, they can easily obtain moves of the other players. The other protocols [7, 8] are also in the same line. But all the above protocols ignored the issue of forming coalitions.

Recently, Dodis et. al[1] presented the first protocol for a 2-player game. The protocol assumed a uniform distribution of probability for all possible strategy profiles and the protocol had exponential communication complexity. Vanessa Teague[3] considered the problem with non uniform probability distribution of the strategies and presented better communication complexity protocol. The communication complexity is further improved by Mikhail et. al[9]. All the protocols [1, 3, 9] work for two players only. The correlated element selection problem was effectively solved with hardware primitives by Lepinksi et. al[4]. They have proposed a collusion free model for a n-player game. It provided a high level of security and efficiency. They have used hardware resources such as ballot boxes and envelopes, which makes the protocol not suitable in all the general cases. Along the similar lines, recently there is an another protocol by Izmalkov et. al[5] with hardware primitives.

Abraham et. al[10] has shown that if $k < n$, assuming the use of cryptographic primitives and also assuming that the utilities are known, there is a $k-$resilient multiparty computation protocol, i.e. simulating a mediator, with constant expected running time. They have extended their results with players having unexpected utility, and pre-play communication before the game starts, i.e cheap talk. When $2(k + t) < n$ and allowing cheap talk, they have shown that the mediator can be simulated and proved the existence of ϵ, (k, t) mechanism, where t represents the players with unexpected utilities. In fact, we have a given an example for the "mediator simulation for correlated element selection problem" and presented a protocol for the same.

In this paper we follow the uniform probability distribution, means every strategy profile is chosen with equal probability. Here, we present a collusion free n-player protocol for the correlated element selection problem. In other words, all the players want to collectively decide upon any one of the m tuples known to all. To the best of our knowledge, the protocol that we are presenting here is the first collusion free protocol, without any hardware primitive.

3 Preliminaries

We consider a n-player $(p_1, p_2, \dots p_n)$ correlated element selection problem with m correlated choice. Let $\Gamma(n, L, U)$ represents a n-persons game, where $L = \{L_1, \dots, L_n\}$ and L_i is the set of actions for each player p_i and $U = \{u_1, \dots, u_n\}$ is a utility function for each player, where $u_i : L \rightarrow R$. We consider one preparer and $n-1$ choosers. The choosers are numbered from 1 to $n-1$. The Preparer is denoted by P and the choosers are denoted by C_k, where k is in $\{1, 2, ..., n-1\}$. Let the set $\{1, 2, ..., n\}$ be denoted by $[n]$. We denote pk as the public key and sk as the secret key used by the preparer. We also denote the list of tuples (all the correlated choices) T as: $\{(a_{i1}, a_{i2}, ..., a_{in})/i = 1 \ to \ m\}$ which is a set of $(a_{i1}, a_{i2}, ..., a_{in})$ for all $i \in [m]$.

3.1 Basics of Game Theory

We define some basic terminologies of game theory in this section [11].
A *strategy* can be defined as a complete algorithm for playing the game; implicitly listing all moves and counter-moves for every possible situation throughout the game. And a *strategy profile* is a set of strategies for each player which fully specifies all actions in a game. A strategy profile must include one and only one strategy for every player. Let a_{-i} be a strategy profile of all players except for the player p_i. When each player p_i, $i \in \{1, \ldots, n\}$ chooses strategy $a_i \in L$ resulting in strategy profile $a = \{a_1, \ldots, a_n\}$, then player p_i obtains pay off $u_i(a)$. Note that the pay off depends on the strategy profile chosen, i.e., on the strategy chosen by player p_i as well as the strategies chosen by all the other players. Some basic notions in game theory are defined as follows:

Definition 2. ***(Weak Domination):*** *In a strategic game with ordinal preferences, player $p_i's$ action $a \in L_i$ weakly dominates his action $a' \in L_i$ if*

$$u_i(a, a_{-i}) \geq u_i(a', a_{-i})$$

for every list L_{-i} of the other players' actions. We say that the action $a^{'}$ is weakly dominated.

Definition 3. ***(Symmetric Games):*** *A game $\Gamma(N, L, U)$ is called a symmetric game if for each player p_i and for any permutation π we have :*

$$u_i(a_1, a_2, \ldots, a_n) = u_{\pi(i)}(a_{\pi(i)}, a_{\pi(i)}, \ldots, a_{\pi(n)})$$

In simple words, in a symmetric game, the pay-offs of playing a particular strategy are only determined by the other strategies employed and not by which player is playing them.

The protocol that we present in this paper is resilient to coalition, i.e., the protocol allows the player to tackle and overcome any coalition between players in the game. The outcome of the game will be the same even if some colluded players try to get inappropriate advantage. Thus, we introduce the term *Coalition Resilient Equilibrium.* Let C be a coalition set, where $C = \{P_1, P_2, \ldots, P_t\}$ and $t < n$. Let the vector $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ and $L_C = L_1 * L_2 * \ldots * L_t$ and $\sigma_C = (\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{it})$. Let σ_{-C} mean that all the players except the coalition members follow strategy σ. A strategy profile σ is a t-resilient equilibrium if for every coalition C of size at most t, no member of a coalition improves its pay-off by coordinating their actions and communicating messages. We can say k resilient equilibrium tolerates deviations by coalitions of size upto k, if any coalition C, of size $C \leq k$, deviates from the equilibrium, then no one in C can get any advantage. Thus, we define the coalition resilient equilibrium as:

Definition 4. ***(Coalition Resilient Equilibrium)[12]:*** *Let $\Gamma = (N, L, U)$ be a game. The joint strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ is a k-resilient equilibrium for $1 \leq k < n$, and for all possible non-empty sets C, $|C| \leq k$, $C \subseteq N$ and for every player $i \in C$ for any τ_C strategy profile of coalition players, then the following holds true:*

$$u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_C)$$

3.2 Cryptographic Primitives

The principle tool that we use in our protocol is the *Blindable Encryption Scheme.* Blindable encryption includes the algorithms for key-generation, encryption and decryption. We denote the blinding algorithm using the keyword *Blind* and also the encryption and the decryption function as *Enc* and *Dec* respectively. The formal description of the blinding and combining functions is:

Definition 5. ***(Blindable Encryption):*** *A public key encryption scheme ε with public key pk is blindable if there are some Blind and Combine algorithms such that for any message m_1 and every cipher text $c \in Enc_{pk}(m_1)$, we have: For any message β (the blinding factor), $Blind_{pk}(c, \beta)$ produces a random encryption of $(m_1 + \beta)$.*

$$Enc_{pk}(m_1 + \beta) \equiv Blind_{pk}(c, \beta)$$

Blinding twice can be achieved by using two random coins r_1 and r_2. Then for any two blinding factors m_2 and m_3, we have:

$$Blind_{pk}(Blind_{pk}(c, m_2; r_1), m_3; r_2) = Blind_{pk}(c, m_2 + m_3; Combine_{pk}(r_1, r_2))$$

Using the blindable encryption schemes anyone can covert an encryption c of m_1 to an encryption $\bar{c}$ of $(m + \beta)$ without the knowledge of the message or the secret key. Thus, only the person having the secret key can decrypt the message and the person who blinded it will know the original message signal. We have discussed about the working and the implementation of blindable encryption schemes in the appendix in detail.

Commitment schemes are the other cryptographic primitive that we use in this paper. They allow a player to commit to a value to the other players keeping them hidden at the same time. Thus, the player can later reveal the committed value which the other players can verify. Commitments are used to blind the value of a player so that he cannot adapt to other values to gain inappropriate advantage. More information about the commitment schemes is given in the appendix. The commitment scheme has two steps as defined in [13]:

Definition 6. ***(Commitment Scheme):***

1. *Commit. The sender sends the encrypted form of bit b which he wants to commit to, to the receiver.*
2. *Reveal or Open. The sender sends additional information to the receiver which enables him to recover the bit b.*

There are three requirements for this:

1. *Hiding Property.* The receiver does not learn anything about the committed value in the commit step. The receiver cannot distinguish between the commitment of any two discrete values.
2. *Binding property.* The sender cannot alter the committed value once he has committed to it in the commit phase. This requirement has to be satisfied, even if the sender tries to cheat.
3. *Viability.* If both the sender and receiver follow the protocol, the receiver will always recover the committed value.

4 Existing Protocol for 2 Players:

The 2-player protocol for the correlated element selection as given by Dodis [1] is given in table 1.

Protocol CES

Common inputs: List of the tuples $\{(a_i, b_i)/i = 1 \textit{ to } m\}$, Public key pk.
Preparer knows: Secret key sk

1. **Preparer**
   Picks the random permutation σ over $[m]$.
   Let $(c_i, d_i) = (Enc_{pk}(a_{\sigma(i)}), Enc_{pk}(b_{\sigma(i)}))$ for all $i \in [m]$.
   Send the list $\{(c_i, d_i)/i = 1 \textit{ to } m\}$ to the chooser.
2. **Chooser**
   Picks a random index $r \in [m]$, and a random blinding factor β.
   Let $(e, f) = (Blind_{pk}(c_r, 0), Blind_{pk}(d_r, \beta))$.
   Send (e, f) to the preparer.
3. **Preparer**
   Set $a = Dec_{sk}(e), \bar{b} = Dec_{sk}(f)$. Output a and send $\bar{b}$ to chooser.
4. **Chooser**
   Unblind the value and output, i.e., set $b = \bar{b} - \beta$. Outputs b as his strategy.

Table 1. Dodis 2-player protocol

The above protocol is formulated for a 2-player correlated element selection problem. One player acts like a Preparer (P) and the other as a chooser (C). The preparer in the first step permutes the pairs and then encrypts them using his public key. Preparer then sends these pairs to the chooser. The chooser then chooses a random pair number l, where $1 \leq l \leq m$. Chooser then blinds the first index with zero and the second index with his blinding factor β. After the blinding is done, the chooser sends the selected pair to the preparer. The preparer decrypts both the values. As the value at the first index is blinded only by 0, he gets the information about the original move, whereas, the preparer gets no information about the move at the second index as it is blinded by β. Thus, the preparer takes the decryption of the value at the first index as his move. The preparer returns the decryption of the value at the second index to the chooser. The chooser unblinds the received value by subtracting his blinding factor β and outputs the value as his move.

5 The n player Protocol

5.1 Unsuccessful Directions

In the extended n-player protocol, we have a single preparer (P) and $n - 1$ choosers (C_i) where $i \in [n-1]$. The preparer permutes and encrypts all the m

tuples and passes it to all the players. Here, a tuple is analogous to a pair and the tuple number analogous to the pair number of the 2-player game. We considered that all the choosers will collectively decide upon the tuple number to be chosen from the available m choices. Thus, only the choosers will have the information of the tuple number. But it is clearly seen that even if one chooser colludes with the preparer, he can inform the preparer about the tuple number. The preparer knows the tuple permutation and hence, can know all the moves of the players if he knows the selected tuple number. Thus, any collusion between the preparer and a chooser can cause the protocol to fail. To overcome this problem, let us have a particular chooser, C_k, who permutes and blinds all the tuples and sends them to all the choosers. The rest of choosers collectively decides on a tuple number. Even if any one of the choosers colludes with the preparer, they do not get any information about the original tuple. This is true as preparer's tuples are again permuted and blinded by another chooser C_k. The protocol seems to be working, but if the chooser C_k colludes with the preparer and with any of the players in the group, then they know the moves of all the other players. The same problem can be seen when two choosers also permute and blind the tuples..

Thus, to make tuple selection completely collusion free, every chooser, one by one, should permute and blind the tuples and pass it to other choosers. After all the choosers have finished blinding and permuting, the $(n-1)^{th}$ chooser randomly picks up a tuple number and informs all the other players. Then they jointly compute a permutation, which specifies which chooser takes which index from the tuple. For example, in the i^{th} tuple $(a_{i1}, a_{i2}, \ldots, a_{in})$ each player has to choose an index number which will determine the player's strategy. Thus, if the players agree upon a permutation $\{3, 2, 1, \ldots, n-1\}$ which signifies that the player one, p_1, takes the 3^{rd} index, p_2 takes the 2^{nd} index and the n^{th} player, p_n, takes the $(n-1)^{th}$ index and so on.

5.2 The Protocol

In an n-player game, we select one player to be the preparer(P) and all the other $n-1$ players as choosers. They collectively decide upon any one of the m correlated choices. A player's move is decided by the value at a distinct index (which no other player has taken) in the agreed upon tuple. All the $n-1$ choosers have two private blinding factors, i.e., every chooser C_i has β_i as its public blinding factor and α_i as its private blinding factor. The public key used by the preparer(P) is already made available to all the choosers. Thus, we present our protocol in the table 3.

At the beginning of the protocol, the preparer randomly permutes all the tuples of the game and encrypts them element wise using his public key. After the encryption is done, he sends the encrypted tuples to all the choosers. Next in a tuple, the users collectively select the permutation of index number, i.e., each player selects the index which will determine his move in the game. In other words, its result is a permutation which tells which player takes which index in a tuple. This is done by executing the sub-protocol (sub-protocol π_2). As we are considering a symmetric game, using its definition we can state that, irrespective

Protocol-NCES

Common inputs: List of the tuples $\{(a_{i1}, a_{i2}, ..., a_{in})/i = 1 \textit{ to } m\}$
Public key pk.
Preparer knows: Secret key sk

1. **Prepare**.
   Takes the m tuples and permutes them according to σ . Encrypts all the tuples using the public key as :
   $(b_{i1}, b_{i2}, ..., b_{in}) = (Enc_{pk}(a_{\sigma(1i)}), Enc_{pk}(a_{\sigma(2i)}), ..., Enc_{pk}(a_{\sigma(ni)}))$, for all $i \in [m]$.
   Sends the encrypted tuples, $\{(b_{i1}, b_{i2}, ..., b_{in})/i = 1 \textit{ to } m\}$ to all the choosers.
2. **Choosers (execution of the sub-protocol π_2)**.
   After the execution of this protocol, each chooser C_i agrees upon a distinct index number h_i in a tuple, i.e., the value at the index will determines the move of the player.
3. **Chooser 1 to** $(n-1)$
   - The permuted tuples are passed from one player to other in the following order: $C_1 \rightarrowtail C_2 \rightarrowtail ... \rightarrowtail C_{n-1} \rightarrowtail P$.
   - Each chooser $C_j, j \in [n-1]$ permutes all the tuples with his private permutation function σ_j, and blinds his index h_j in all the tuples with $(\alpha_j + \beta_j)$.
     $\{(\bar{b_{ih_j}})/i = 1 \textit{ to } m\} = \{(Blind_{pk}(b_{ih_j}, (\alpha_j + \beta_j))/i = 1 \textit{ to } m\}$.
   - Then the chooser blinds all the other values in the tuples with β_j.
     $(\bar{b_{i1}}, \bar{b_{i2}}, ..., \bar{b_{in}}) = (Blind_{pk}(b_{i1}, \beta_j), Blind_{pk}(b_{i2}, \beta_j), ..., Blind_{pk}(b_{in}, \beta_j))$ for all $i \in [m]$ and $i \neq h_j$.
   - Passes the tuples to the chooser $(j+1)$, till the $(n-1)^{th}$ chooser is reached.
   - The $(j+1)^{th}$ chooser also executes the step 3.
4. **Preparer.**
   - The preparer also blinds his index with $(\alpha_n + \beta_n)$ in all the tuples. Then blinds all the other value in tuples with β_n.
   - Let the tuples be denoted by $\{(e_{i1}, e_{i2}, ..., e_{in})/i = 1 \textit{ to } m\}$.
   - Sends these permuted tuples to all the players.
   - Preparer randomly selects a number $r \in [m]$, i.e., the tuple number and announces it to all.
5. **Choosers**.
   Every chooser now sends the value at the index allotted to him in the r^{th} tuple to the preparer, i.e., Player p_i sends $\bar{e_{rh_i}}$ to the preparer, where
   $\bar{e_{rh_i}} = Blind_{pk}(b_{rh_i}, (\alpha_i + \beta))$.
6. **Preparer**.
   - The preparer decrypts the values sent by the choosers and sends them back the decrypted message. Send $d_{rh_i} = Dec_{sk}(\bar{e_{rh_i}})$ back to the chooser i.
   - The preparer decrypts the value at the index allotted to him in the r^{th} tuple.
7. **All players**.
   Each player p_i publicly announces the choice of its β_i. Then each compute $\beta = \sum \beta_i$ for $i \in [n]$. p_i then secretly computes his move as : $e_i = d_{rh_i} - (\beta + \alpha_i)$. Output e_i and agree on the move e_i.

Table 2. n-player correlated element selection protocol.

of the index a player chooses, the overall pay-off will remain the same. This holds true because in a symmetric game the pay-off is determined only by the other strategies played and not by who plays it. Thus, after the execution of π_2, an index number is allotted to each player (including the preparer), i.e., every player p_i agrees upon the index number denoted by h_i.

In the next stage of the protocol, the original message is transmitted from one chooser to another in a particular order. Each chooser C_i first permutes all the tuples (only the tuples and not the values inside any tuple) with its permutation function σ_i and then blinds the element at his index in all the tuples with $(\alpha_i + \beta_i)$. Subsequently in the same step, the chooser blinds all the other values in each tuple with β_i. We call α_i as the private blinding factor of the player i. In the next succeeding steps, each chooser repeats the same steps till the $(n-1)^{th}$ chooser finishes.

Now after every chooser has permuted and blinded every tuple, the resulting tuple permutation is sent to the preparer. The preparer then performs the same blinding procedure in order to keep his strategy hidden from others. Then the preparer sends the final tuple combination to all the choosers. The $(n-1)^{th}$ chooser randomly chooses and announces a particular tuple number r such that $r \in [m]$. Consequently, each chooser sends the value at the index allotted to him in the agreed tuple to the preparer. The preparer then decrypts it and sends it back to all the choosers. The preparer does not get any information about the moves of the players as each choice is blinded by $(\alpha_i + \beta)$.

All the players p_i (including the preparer), announce their choices of β_i publicly. Henceforth, all the players individually calculate $\beta = \sum \beta_i$ for all $i \in [n]$. Then each player i individually subtracts the $(\alpha_i + \beta)$ from the decrypted value sent by the preparer and outputs his move.

5.3 Sub-Protocol π_2 (Random permutation selection)

In the sub-protocol random permutation selection, as described earlier, we need to decide upon the index number each player takes within a particular tuple. For making this sub-protocol uniform and distributed, we take in the individual choices of each player. Thus, each player has his own permutation function (ρ_i).

We first assume that each player is connected via a *simultaneous synchronous communication channel*, which means that all the players communicate simultaneously. As all the players announce their permutation function ρ_i simultaneously, then the computation of the random permutation selection is said to be trivial. All the players then agree on $\rho_n(\rho_{n-1}(\ldots(\rho_2(\rho_1(S))\ldots))$, where S is the sequence $\{1, 2, \ldots, n\}$.

But in the absence of the simultaneous channel, the players who have colluded with the preparer can wait to receive every other player's permutation function. Based on others' permutation functions, he can choose his permutation, such that it matches with the tuple permutation of their choice. To solve this problem we use the concept of *commitment schemes*. We use *Commit* to denote for committing to a particular value and *Reveal* to reveal the committed value. The commitment schemes hold a property that, if a player commits to a value then he can not deviate from it. If ever he does deviate, it is be noted by others. In this sub-protocol, every player p_i sends his committed value, $Commit(\rho_i)$, to all. After a chooser receives all the other commitments, he reveals ρ_i to all the other players. Finally, when every player has revealed his value, the permutation function is calculated as : $\rho_n(\rho_{n-1}(\ldots(\rho_2(\rho_1(S))\ldots))$ by all the players. The protocol is given in table 4.

Protocol-RPI

All the players execute the following steps :

- Each player p_i decides upon its permutation function ρ_i.
- Each player p_i computes its commitment, $Commit(\rho_i)$. He sends this value to all the other players and waits until he receives all other players' $Commit(\rho_j)$ value.
- After receiving every other players $Commit(\rho_j)$, each player p_i starts revealing its committed function, i.e., $\bar{\rho_i}$.
- After every value is revealed, each player p_i verifies whether :
  $Commit(\rho_j) = Commit(\bar{\rho_j})$.
  If it is not equal then the player quits from the protocol and announces the deviation by sending specific signal.
- Each player calculates the final permutation as :
  $\{g_1, g_2, ..., g_n\} = \rho_n(\rho_{n-1}(....(\rho_2(\rho_1(S))...))$,
  where S is the sequence $\{1, 2, 3, ..., n\}$.
- Each player p_i chooses the value at the i^{th} index of the final permutation, i.e., agrees upon h_i in the list of final permutation.

Table 3. Protocol for computation of random permutation of indices.

5.4 Proof of Correctness

Theorem 1. *The NCES protocol securely completes the function of the correlated element selection problem among the n players.*

Proof. The rational players in the above problem behave honestly, except that they want to know every other player's strategy, so that after completion of the execution of the protocol they can deviate. This is equivalent to honest-but-curious behaviour of players. Hence, as long as our encryption and blinding schemes are secure, the protocol *NCES* securely performs the function of the correlated element selection problem. We have not assumed the broadcast channels in the protocol *NCES* and *RPI* as player p_i cannot send different messages to different players. As it can result in a strategy profile which does not belong to the set of correlated choices, may result in a lower pay off to all the players. Let a be the strategy used by player p_i in which he sends the same message to all the players and a' represent the strategy in which he sends different messages to different players. Then, $u_i(a, a_{-i}) \geq u_i(a', a_{-i})$. So, sending the same message to all players weakly dominates sending different messages. Hence, our protocol results in a $(n-1)$ coalition resilient equilibrium. $\square$

6 Conclusion

We have considered the correlated element selection problem, where firms want to correlate their business strategies and agree upon them in an untrustworthy environment. We have proposed an n-player protocol for solving the correlated element selection problem. Thus, we have shown a successful way to simulate a mediator for achieving correlated equilibrium. As per our knowledge this is the first collusion free protocol to solve this problem without hardware primitives. We have used game theoretic concepts like weak dominance to prove the correctness of a cryptographic protocol (which are indeed used to solve a game theory problem). Thus, we have shown an interesting way to overlap different proof techniques. For the internet domains, in many problems like selfish routing[14], achieving correlated equilibrium has a significant importance[15]. The collusion free and efficient polynomial communication protocols of correlated element selection problem can be used to solve these problems and will have a significant impact on the total social cost. As a first attempt, we have proposed a collusion free protocol. Achieving both collusion free and polynomial communication are completely open and will have significant advantage. As our results are efficiently applicable to symmetric games, getting these results in asymmetric games is a good direction to extend the work.

References

1. Dodis, Y., Halevi, S., Rabin, T.: A cryptographic solution to a game theoretic problem. In: CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology, London, UK, Springer-Verlag (2000) 112–130
2. Aumann, R.J.: Subjectivity and correlation in randomized strategies. Journal of Mathematical Economics **1**(1) (1974) 67–96
3. Teague, V.: Selecting correlated random actions. In: In Financial Cryptography. (2004) 181–195
4. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely fair sfe and coalition-safe cheap talk. In: PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing. (2004) 1–10
5. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably secure devices. In: TCC. (2008) 273–301
6. Bárány, I.: Fair distribution protocols or how the players replace fortune. Math. Oper. Res. **17**(2) (1992) 327–340
7. Ben-Porath, E.: Correlation without mediation: Expanding the set of equilibrium outcomes by "cheap" pre-play procedures". Volume 80. (1998) 108 – 122
8. Gerardi, D.: Unmediated communication in games with complete and incomplete information. Journal of Economic Theory **114**(1) (2004) 104–131

9. Mikhail Atallah, Marina Blanton, K.F.J.L.: Efficient correlated action selection. In: Financial Cryptography and Data Security. Volume 4107 of LNCS., Springer (2006) 296–310
10. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, New York, NY, USA, ACM (2006) 53–62
11. Osborne, M.: *An Introduction to Game Theory.* Oxford University Press. (2004)
12. Bernheim, B., Peleg, B., Whinston, M.: Coalition-proof Nash equilibria. I. concepts. Journal of Economic Theory **42**(1) (1987) 1–12
13. Delfs, H., Knebl, H.: Introduction to Cryptography: Principles and Application. Springer, Berlin Heidelberg New York (2002)
14. Roughgarden, T., Tardos, E.: How bad is selfish routing? J. ACM **49**(2) (2002) 236–259
15. Blum, A., Even-Dar, E., Ligett, K.: Routing without regret: on convergence to nash equilibria of regret-minimizing algorithms in routing games. In: PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, New York, NY, USA, ACM (2006) 45–52
16. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2) (1984) 270–299
17. Blum, M.: Coin flipping by telephone a protocol for solving impossible problems. SIGACT News **15**(1) (1983) 23–27
18. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall Professional Technical Reference (2003)

A Appendix

A.1 Implementation of Blindable Encryption

We present two blindable encryption schemes namely ElGamal encryption and the Goldwasser-Micali encryption schemes. These schemes are briefly described below. **Goldwasser-Micali Encryption**
Goldwasser and Micali proposed this semantically secure scheme in[16]. The message is encrypted bit by bit. A random n is picked by a generation algorithm, where $n = pq$, a product of two k-bit primes together with any quadratic non-residue y with the Jacobi symbol l, the public key is n and y, and the secret key is p and q.

The encryption algorithm takes the message m and the random integer r as input and outputs $(y^m.r^2 mod\ n)$. To decrypt one can simply determine if the cipher text c is a square modulo n, i.e. modulo both p and q. The scheme is semantically secure under the quadratic residue assumption. Since $c = Enc(m; r)$ and $c_1 = Enc(m_1; r_1)$, then $c.c_1 = Enc(m + m_1; r.r_1)$. Thus, it can be clearly seen that this scheme is blindable. Now, as The blinding of any message m is done by using a randomness r. Thus, every time a person is blinding m value with different values of randomness r, the resulting cipher text will be different.

We can also use double blinding encryptions. Let m_2 be encrypted with the randomness r_2 and $c_2 = Enc(m_2; r_2)$. Thus, the following will hold good :

$$c.c_1.c_2 = Enc(m + m_1 + m_2; r.r_1.r_2)$$

Proving that the public key (n, y) is committing requires proving that y is a quadratic-non-residue modulo n, which can be done efficiently. Proving that c is an encryption of m can be done by proving quadratic residuosity.

ElGamal Encryption
In the ElGamal encryption scheme, a generation algorithm picks a random k-bit prime $p = 2q + 1$, such that q is a prime. And let g be the generator of the subgroup Q of the quadratic residues modulo p. Then it picks a random $x \in Z_q$ and the sets $E(m) =< g^r, h^r.m >$. The decryption $D(s,t)$ outputs $\frac{t}{s^x}$. This encryption scheme in semantically secure under the decisional Diffi-Hellman assumption. To blind the cipher text $< s,t >$ with the blinding factor $\bar{m}$, we compute $Blind(< s,t >, \bar{m}) =< s.g^{\bar{r}}, t.h^{\bar{r}}.\bar{m} >$, where $\bar{r}$ is chosen at random from Z_q. If $s = g^r, t = h^r.m$ (where r and m are some unknown), then $Blind(< s,t >, \bar{m}) =< g^{r+\bar{r}}, h^{r+\bar{r}}.(m\bar{m}) >$, which is a random encryption of $m\bar{m}$ and when $\bar{r}$ is random then $(r + \bar{r}\ mod\ q)$ is also random. Thus, we can see that $Combine(r_1, r_2) = (r_1 + r_2\ mod\ q)$.

We can prove the equality of the discrete-log of two known elements with respect to two known bases by using any one of the several known simple protocols. By proving the above equality we can prove that a cipher text $< s,t >$ is an encryption of a message m.

In our protocol, we have made use of the Goldwasser-Micali blindable encryption schemes. Similarly in the ElGamal encryption scheme, we blind the message with 1 instead of 0 to get back the original message. And if we have blinded the message with β, then we can obtain the original message by dividing the decrypted message by β instead of subtracting.

A.2 Commitment schemes

The concept of commitment schemes was first introduced and used by Manuel Blum in 1983[17]. Commitment schemes are cryptographic primitives that allows a player to commit to value while keeping it hidden from all the other players. The players can later reveal the committed value. Commitments are used to bind a player to a value so that later they cannot adopt to any other message in order to gain any inappropriate advantage over the other players. The commitment schemes are used in many other cryptographic protocols such as zero knowledge proofs and secure computation. The commitment schemes are executed in the following two steps :

– ***The commit phase :***
  In this phase a value is chosen by a player. The player then commits to this value to all the other players.

– ***The reveal phase :***
  In the reveal phase, the player reveals his original value. All the other players then check for the revealed value with the commitments they have received earlier.

The commit phase consists of a single message being sent from a player to the other players. It is important that none of the other player is able to get any information about the specific value held by the committing player. In the reveal phase, the player reveals the committed value to the other players. All the other players then perform a check upon the revealed and the committed value. The value chosen at the commit phase must be the only one that validates during the reveal phase.

The commitment schemes that we use are based on the *discrete logarithm problem (DL problem).* The discrete logarithm problem is defined in [18]. Suppose, the discrete logarithm of a is h. Finding the value of a from h is very difficult. finding the value of a is not like the taking the ordinary logarithms as the DL problem is defined in a discrete domain, the solution should be *exact.* Thus, based on this DL problem, a Commitment Scheme in which the sender commits to a message $m \in [q-1]$ is given in [13]. Thus, the protocol is:

Log Commitment Protocol

1. **System Set up.**
   The receiver randomly chooses large prime numbers p and q such that q divides $(p-1)$. Then he randomly chooses g and v from the subgroup G of order q in $\mathbb{Z}_q^*, g, v \neq 1$. Receiver then sends p, q, g and v to the sender.
2. **Commit.**
   Sender then verifies whether p and q are primes or not, that q divides $(p-1)$ and that g and v are elements of order q. To commit to a message $m \in [q-1]$, he chooses a random $r \in [q-1]$, sets $c := g^r v^m \; mod \; q$ and sends c to the receiver.
3. **Reveal.**
   Sender sends r and m to the receiver who verifies whether $c = g^r v^m \; mod \; q$.