Efficient Correlated Action Selection[*]

Mikhail J. Atallah, Marina Blanton, Keith B. Frikken, and Jiangtao Li

Department of Computer Science
Purdue University
{mja,mbykova,kbf,jtli}@cs.purdue.edu

Abstract. Participants in e-commerce and other forms of online collaborations tend to be selfish and rational, and therefore game theory has been recognized as particularly relevant to this area. In many common games, the joint strategy of the players is described by a list of pairs of actions, and one of those pairs is chosen according to a specified correlated probability distribution. In traditional game theory, a trusted third party mediator carries out this random selection, and reveals to each player its recommended action.In such games that have a correlated equilibrium, each player follows the mediator's recommendation because deviating from it cannot increase a player's expected payoff. Dodis, Halevi, and Rabin [1] described a two-party protocol that eliminates, through cryptographic means, the third party mediator. That protocol was designed and works well for a uniform distribution, but can be quite inefficient if applied to non-uniform distributions. Teague [2] has subsequently built on this work and extended it to the case where the probabilistic strategy no longer assigns equal probabilities to all the pairs of moves. Our present paper improves on the work of Teague by providing, for the same problem, a protocol whose worst-case complexity is exponentially better. The protocol also uses tools that are of independent interest.

1 Introduction

Many potentially beneficial collaborations over the Internet do not take place, even when both participants stand to gain from the interaction. One of the major reasons for this is the difficulty of finding a third party that they can both trust with not revealing (to their counterpart, or to outsiders) their private data or planned future actions and business moves. This reluctance to engage in apparently win-win collaborations results in organizations making lower-quality (and sometimes far-reaching) decisions and plans. Although this reluctance to collaborate causes large potential benefits to go unrealized, there are good reasons for it: the information learned by the third party mediator could be highly proprietary, help the competition, be inadvertently (or maliciously) leaked out and

cause embarrassment and lawsuits, be misused, etc. In addition to this, there are substantial costs to being a mediator, not only in terms of the electronic infrastructure but in other operational costs (such as liability insurance against accidental data disclosure). Cryptography has much to contribute in solving the problem, by obviating the need for a third-party mediator.This is why the recent work of Dodis, Halevi, and Rabin [1] and Teague [2], in getting rid of the need for a mediator, has such huge practical potential in addition to its intellectual content. As our work builds on these papers, we briefly review these and explain where our contribution lies.

The framework of this paper is the same as in [1, 2]: two entities want to coordinate their respective actions, implementing a strategy that is described as a set of m pairs of actions, with each pair having an associated probability of being selected (the action choices are correlated). If a pair is selected, the first (second) element of the pair is the first (second) entity's recommended action; no entity should learn the recommended action of the other (although, unavoidably inferences can be made from their knowledge of the public strategy and their own recommended action). Each party is incentivized to follow the recommendation given that an equilibrium exists, i.e., deviating from the recommended action cannot increase a party's expected payoff.

1.1 Related Work

Dodis, Halevi, and Rabin [1] described a two-party protocol that eliminates, through cryptographic means, the third party mediator: The protocol assumes a uniform distribution, selects at random and reveals to each party their respective selected action only (i.e., not the other party's action). Since cryptographic solutions have to be efficient, one might ask at what computational and communication cost this is achieved. The protocol of [1] works efficiently for a uniform distribution, but not if the distribution is non-uniform (particularly if a pair can have an associated probability much smaller than the probability of another pair). Teague [2] subsequently extended the work to non-uniform distributions, and gave a better (but still worst-case exponential) protocol for the case where pairs of moves can have widely differing probabilities. Other prior work that addresses the same problem without help from a third-party mediator includes [3–6]. All of the protocols of [3, 4, 1, 2, 5] may require communication exponential in the size of the binary description of the correlated equilibrium strategies. Our present paper improves on the work of Teague by providing, for the same problem, a protocol whose worst-case complexity is exponentially better. In addition, our protocol uses tools that are of independent interest and advantageously modify protocols recently presented in areas unrelated to the game-theoretic framework, such as [7].

Our work is not comparable to the polynomial solution given in [6], which does not apply to the important two-party case we consider here, and imposes assumptions (akin to ideal envelopes) on the physical channels used: they use general results to perform the computation for three or more parties, and then extend the protocols to achieve complete fairness during output recovery. Our

work is also similar to the cryptographic randomized response techniques [8]: the protocols in [8] allow a party to choose a value according to a probability distribution. The primary difference is that in [8] one party (the pollster) learns the result, but in our problem each of the two parties learns part of the result.

1.2 Notation

The rest of the paper uses the following notation. Let k denote a security parameter. The m action pairs are denoted as $\{(a_i, b_i)\}_{i=1}^m$. Each of these pairs is chosen with a certain probability q_i, such that their overall sum is equal to 1. Each q_i is given in the rational form (i.e., same form as in prior work in this area) as a pair of integers α_i, β_i such that $q_i = \alpha_i/\beta_i$. As our protocol will use a somewhat different representation, we next describe an input conversion that we thereafter assume has already taken place prior to the protocol.

Input Conversion Our algorithms require us to convert each q_i into an l-bit integer p_i such that $q_i = p_i / \sum_{j=1}^m p_j$. If we let L denote the least common multiple of all β_j's, then we can set $p_i = L \cdot q_i = \alpha_i(L/\beta_i)$, which implies that $\sum_{j=1}^m p_j = L$ and hence $q_i = p_i/L$. This conversion can be done in polynomial time and results in the p_i integers having a length l, which is polynomial in the number of bits in the original representation. To achieve worst-case polynomial time performance, it therefore suffices for our protocols to be polynomial in l. Let ℓ denote the integer such that $2^{\ell-1} < L \le 2^\ell$: if $L < 2^\ell$, we pad the probabilities with a "dummy" $p_{m+1} = 2^\ell - L$, so that $\sum_{i=1}^{m+1} p_i = 2^\ell$. Note that this is done only for ease of computation and the $(1+m)$th outcome is never chosen: if a protocol execution returns the $(1+m)$th outcome, the computation is restarted. The probability of restart is $p_{m+1}/(2^\ell) < 1/2$. In the rest of this paper we assume that the m tuples (a_i, b_i, p_i) contain a dummy element (whose action pair is a "restart protocol" recommendation to both parties) if necessary.

1.3 Comparison with Previous Work

Our results are the following: a protocol for the malicious (resp., honest-but-curious) model has computation and communication complexity $O(m\ell)$ (resp., $O(m + \ell \log m)$). See Table 1 for performance comparison with the prior work.

Note that secure function evaluation using generic garbled circuits constitutes a viable alternative to the solutions given in this work, especially since recent results (see, e.g., [9]) provide significant improvements over the initial results. Any solution using circuits, however, will require at least $O(m\ell)$ gates, while in this work we concentrate on finding solutions asymptotically as low as possible. In addition, any protocol that requires a majority of the players to be honest (which is the case in [9]) does not provide security against malicious behavior in the two-party case.

In the equal-probabilities case, the protocol of choice is that of [1]. Thus the following discussion is for the case of unequal probabilities. For the malicious model, our protocol is better than the previous approach of [2] in both

	Teague [2]	SFE [10, 11, 9]	Our Protocols
honest-but-curious	$O(\max\{m, 2^\ell\})$	$O(m\ell)$	$O(m + \ell \log m)$
malicious	$O(\sigma \cdot \max\{m, 2^\ell\})$	$O(m\ell)$	$O(m\ell)$

Table 1. Comparisons of *worst case* performance (computation and communication cost) of our and prior work. Here m is the number of action pairs, ℓ is the number of bits representing the probabilities, and σ is a security parameter for the cut-and-choose technique (i.e., the adversary can cheat with the probability no more than $1/\sigma$) that must be linear in the payoffs to make the expected gain from cheating negative.

asymptotic worst-case and in practical sense, as our protocol is polynomial *and* does not use the cut-and-choose technique as in [2]. For the honest-but-curious model, however, we can only claim an improvement in the worst-case asymptotic complexity, as there are inputs for which the approach of [2] is more practical, e.g., inputs where the number of bits (call it t) representing the smallest input probability is small enough that a complexity proportional to 2^t can compete with our $poly(\ell)$ complexity. Of course, the honest-but-curious model is of limited practical value in the kind of environments where these protocols are used, so one would almost always need to assume a stronger adversary model.

The rest of the paper is organized as follows. Section 2 gives preliminaries, our protocol, and security proofs for the semi-curious model. In section 3, we deal with malicious adversaries and provide additional cryptographic tools and our protocols for that setting.

2 A Protocol for the Honest-but-Curious Case

2.1 Security Model

Informally, we say that a two-party protocol Π *privately* computes function f if anything that can be obtained from a party's view during a semi-honest execution of Π could also be obtained from the input and the output of that party themselves. We use the standard model, and the following definition, similar to the one given in [11], formalizes our notion of security.

Definition 1. *Let $f_1(x, y)$ and $f_2(x, y)$ be the first and the second elements of $f(x, y)$, respectively. Let $\mathrm{VIEW}_1^\Pi(x, y)$ (resp., $\mathrm{VIEW}_2^\Pi(x, y)$) denote the view of the first (resp., second) party during an execution of Π on (x, y). The views are defined as $(x, r_1, m_1, \ldots, m_d)$ and $(y, r_2, m_1, \ldots, m_d)$ for the first and second parties, respectively, where r_1 (resp., r_2) is the outcome of internal coin tosses of the first (resp. second) player and $m_1, \ldots, m_d$ are the messages that it received during the protocol execution. Also let $\mathrm{OUTPUT}_1^\Pi(x, y)$ (resp, $\mathrm{OUTPUT}_2^\Pi(x, y)$) denote the first (resp., second) player's output after an execution Π on (x, y); and let $\mathrm{OUTPUT}^\Pi(x, y) = (\mathrm{OUTPUT}_1^\Pi(x, y), \mathrm{OUTPUT}_2^\Pi(x, y))$. Then Π privately computes f if there exist probabilistic polynomial-time algorithms M_1 and M_2 such that the ensembles $\{M_1(x, f_1(x, y)), f(x, y)\}_{x,y}$ and $\{\mathrm{VIEW}_1^\Pi(x, y), \mathrm{OUTPUT}^\Pi(x, y)\}_{x,y}$ and the ensembles $\{M_2(y, f_2(x, y)), f(x, y)\}_{x,y}$ and $\{\mathrm{VIEW}_2^\Pi(x, y), \mathrm{OUTPUT}^\Pi(x, y)\}_{x,y}$*

are computationally indistinguishable. Machine M_1 (resp., M_2) is called a simulator for the interaction of the first (resp., second) player with the second (resp., first) player.

2.2 Homomorphic Paillier Encryption

Our protocols in the honest-but-curious setting use the homomorphic Paillier encryption scheme [12, 13], which was first developed by Paillier [12] and then extended by Damgård and Jurik [13]. Let $n = pq$ be an RSA modulus, with $p = 2p'+1$ and $q = 2q'+1$ where p, q, p', and q' are primes. Given a message $M \in \mathbb{Z}_n$, we use $Enc^P(M)$ to denote encryption of M under the Paillier encryption scheme. By the homomorphic property, $Enc^P(a) \cdot Enc^P(b) = E(a + b \bmod n)$. It is easy to see that $Enc^P(a)^c = Enc^P(c \cdot a \bmod n)$. A homomorphic Paillier encryption scheme is *semantically secure* under the decisional composite residuosity assumption [12].

2.3 The Element Selection Protocol

As before, the (a_i, b_i) pairs are the move pairs in the joint strategy of the game, where the a_i's (resp., b_i's) are possible moves for Alice (resp., Bob). During the protocol, one of the indices $\{1, \ldots, m\}$ is selected randomly, where the probability of i being selected is $p_i/2^\ell$. The selected index (call it j) is not known to either Alice or Bob, who learn only their respective recommended moves: a_j for Alice, b_j for Bob. Note that, unavoidably, Alice's learning of her move does probabilistically reveal something about Bob's recommended move, and vice-versa (this comes from the game theoretic problem formulation and is true of any protocol, including [1, 2]).

Our protocol can be thought of as a secure version of the following naive (and in this form flawed) approach: Alice and Bob compute $P_i = \sum_{k=1}^{i} p_k$ for $1 \leq i \leq m$, and then generate a random value $r \in [0, 2^\ell - 1]$. Since the probability that $r \in [P_{i-1}, P_i)$ equals to $p_i/2^\ell$, Alice and Bob find the index i corresponding to the chosen r and choose actions a_i and b_i, respectively. Making the above simple idea work involves many challenges. Our protocol is presented next.

Setup: Alice generates a key pair (pk, sk) for the homomorphic Paillier encryption scheme such that $|n| = k$ and $n > 2^\ell + 1$, where k is a security parameter. We separate this step from the protocol itself, because in this application the correlated element selection may be executed by two parties on a regular basis, while it is sufficient to select the keys only once.

Input: Items $\{(a_i, b_i, p_i)\}_{i=1}^{m}$ are known to both parties; public key pk is known to both and secret key sk is known only to Alice.

Output: Alice obtains the value of a_j, and Bob obtains the value of b_j, where j is the index selected according to the probability distribution.

Protocol Steps:

1. Alice encrypts each item in each triplet obtaining $\{(Enc^P(a_i), Enc^P(b_i), Enc^P(p_i))\}_{i=1}^m$. She then picks a random permutation π_a over $[m]$ and permutes the encrypted triplets obtaining $\{Enc^P(a_{\pi_a(i)}), Enc^P(b_{\pi_a(i)}), Enc^P(p_{\pi_a(i)})\}_{i=1}^m$ and sends them to Bob.
2. Bob picks a random permutation π_b over $[m]$ and permutes the encrypted triplets received in the previous step. Let $(Enc^P(a_i'), Enc^P(b_i'), Enc^P(p_i'))$ denote $(Enc^P(a_{\pi_b(\pi_a(i))}), Enc^P(b_{\pi_b(\pi_a(i))}), Enc^P(p_{\pi_b(\pi_a(i))}))$ for $i = 1, \ldots, m$.
3. We use P_i' to denote $\sum_{k=1}^i p_k'$. For each $i = 1, \ldots, m$, Bob computes $Enc^P(P_i') = \prod_{k=1}^i Enc^P(p_k') = Enc^P(p_1' + \cdots + p_i')$.
4. For $i = 1, \ldots, m$, Bob uniformly generates a random value $y_i \xleftarrow{R} \mathbb{Z}_n$ and computes $Enc^P(P_i' - y_i) = Enc^P(P_i') \cdot Enc^P(-y_i)$. He sends $\{Enc^P(P_i' - y_i)\}_{i=1}^m$ to Alice.
5. Alice decrypts $\{Enc^P(P_i' - y_i)\}_{i=1}^m$ and obtains $\{P_i' - y_i \bmod n\}_{i=1}^m$. Let x_i denote $P_i' - y_i \bmod n$. At this point, Alice has $\{x_i\}_{i=1}^m$ and Bob has $\{y_i\}_{i=1}^m$, such that $x_i + y_i \bmod n = P_i'$.
6. Alice picks $r_a \xleftarrow{R} \{0,1\}^\ell$ and Bob picks $r_b \xleftarrow{R} \{0,1\}^\ell$. Let r denote $r_a \oplus r_b$. Clearly, r is a random ℓ-bit integer.
7. Alice and Bob jointly find the index of the value P_i' such that $r < P_i'$ and $r \geq P_{i-1}'$ (if P_{i-1}' exists) from the list $\{P_i'\}_{i=1}^m$ using the binary search protocol described in section 2.4. Let the outcome of the search be index j.
8. Bob chooses a random $\rho \xleftarrow{R} \mathbb{Z}_n$, computes $\{\gamma_1, \gamma_2\} = \{Enc^P(a_j'+0), Enc^P(b_j'-\rho)\}$, and sends the pair to Alice[1]. Alice decrypts $\{\gamma_1, \gamma_2\}$, obtains $\{a_j', b_j'-\rho\}$, and sends $b_j' - \rho$ back to Bob. In the end, Alice learns a_j' and Bob learns b_j'.

The complexity of this protocol is $O(m + \ell \log m)$ and the round complexity is $O(\log m)$. Note that any other known solution (e.g., using general circuit simulation results) is less efficient and requires at least $O(m\ell)$ computation.

2.4   Binary Search Protocol

This section gives an efficient search protocol for step 7 of the element selection protocol for semi-honest players. We use binary search to compute which action the randomly chosen r corresponds to, i.e., the index j such that $r \in [P_{i-1}', P_i')$.

Input: Alice has $\{x_i\}_{i=1}^m$ and Bob has $\{y_i\}_{i=1}^m$ such that $P_i' = x_i + y_i \bmod n$, $0 < P_i' \leq 2^\ell$, and $P_i' < P_{i+1}'$. Alice has r_a and Bob has r_b, such that $r = r_a \oplus r_b$.

Output: The index j such that $r < P_i'$ and $r \geq P_{i-1}'$ (if P_{i-1}' exists).

Protocol Steps: Alice and Bob execute the following recursive procedure on the list $\{P_i'\}_{i=1}^m$:

1. If the size of the current working set $|\{P_i', \ldots, P_j'\}| = 1$, return i.

---

[1] Here $Enc^P(a_j' + 0)$ is computed by $Enc^P(a_j') \cdot Enc^P(0)$. We intentionally randomize $Enc^P(a_j')$.

2. Otherwise, Alice and Bob run a scrambled circuit evaluation protocol [10, 11] to compute whether $r \geq P'_{\lceil \frac{j-i+1}{2} \rceil}$ (i.e., compute whether $r_a \oplus r_b \geq x_{\lceil \frac{j-i+1}{2} \rceil} + y_{\lceil \frac{j-i+1}{2} \rceil} \mod n$). Let c denote the outcome of the protocol that returns 1 if the condition holds, and 0 otherwise. Note that we can use the technique from [14] to reduce the communication and computation from being a function of $k = |n|$ to a function of $\ell + 1$ (i.e., the number of bits required to represent the value $P'_{\lceil \frac{j-i+1}{2} \rceil}$).

3. If $c = 1$, recurse on list $\{P'_{\lceil \frac{j-i+1}{2} \rceil+1}, \ldots, P'_j\}$; otherwise, recurse on list $\{P'_i, \ldots, P'_{\lceil \frac{j-i+1}{2} \rceil}\}$.

Step 2 takes $O(\ell)$ communication and computation and $O(1)$ rounds, therefore the overall complexity is $O(\ell \log m)$ and the round complexity is $O(\log m)$.

Lemma 1. *The protocol for binary search is secure against honest-but-curious adversaries.*

Proof (Sketch). The basic idea behind this proof is that from a particular index, i.e., from the output of the binary search, one can easily simulate the individual zigs and zags of the binary search. It is worth noting that this is similar to the proof of [15]. □

2.5 Security Proofs

To be able to show the correctness of the element-selection protocol, we first prove that, if both Alice and Bob follow the protocol, the output pair of actions is selected according to the probability distribution.

Lemma 2. *For any $i \in \{1, \ldots, m\}$, the probability that the randomly chosen $r \in \{0,1\}^\ell$ results in index i being returned, is equal to $p'_i/2^\ell$.*

Proof. Let us set $P'_0 = 0$. The probability that index i is returned, equals the probability that $r \in [P'_{i-1}, P'_i)$. This is equal to $(P'_i - P'_{i-1})/2^\ell = p'_i/2^\ell$. □

Recall that the binary search step of the element-selection protocol reveals the index, so we next prove that this index does not leak any information.

Lemma 3. *Let π be a random permutation over $[m]$ and let r be a random value in $\{0,1\}^\ell$. Given any set $\{p_i\}_{i=1}^m$ such that $\sum_{i=1}^m p_i = 2^\ell$, the probability of $r \in \left[\sum_{k=1}^{i-1} p_{\pi(k)}, \sum_{k=1}^i p_{\pi(k)}\right)$ is equal to $1/m$ for $i = 1, \ldots, m$.*

Proof. Let us fix i. We have

$$\mathbf{Pr}\left[r \in [\sum_{k=1}^{i-1} p_{\pi(k)}, \sum_{k=1}^i p_{\pi(k)})\right] = \frac{p_k \cdot \mathbf{Pr}[k = \pi(i)]}{2^\ell} = \frac{1}{2^\ell} \sum_{k=1}^m \frac{p_k}{m} = \frac{1}{2^\ell} \cdot \frac{2^\ell}{m} = \frac{1}{m}$$

In other words, if π and r are random, the output of the binary search in the element-selection protocol does not depend on $\{p_i\}_{i=1}^m$ and is uniformly distributed over $[1, m]$, i.e., it can be simulated by a random value in $[1, m]$. □

Theorem 1. *The element-selection protocol is secure against honest-but-curious adversaries.*

Proof. Correctness: Follows directly from Lemma 2.

Secrecy: To show that the element-selection protocol is secure, it is sufficient to show that there exists a simulator M_1 (resp., M_2) that, given Alice's (resp., Bob's) input and output, can simulate Alice's (resp., Bob's) interaction with Bob (resp., Alice) during the execution of the protocol, such the Alice's (resp., Bob's) view in real execution is computationally indistinguishable from the view produced by the simulator. That is, according to Definition 1:

$$\{M_1(x, f_1(x,y)), f(x,y)\}_{x,y} \stackrel{c}{\equiv} \{\text{VIEW}_1^\Pi(x,y), \text{OUTPUT}^\Pi(x,y)\}_{x,y}$$

$$\{M_2(y, f_2(x,y)), f(x,y)\}_{x,y} \stackrel{c}{\equiv} \{\text{VIEW}_2^\Pi(x,y), \text{OUTPUT}^\Pi(x,y)\}_{x,y}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability by families of polynomial-size circuits.

Consider the following simulator $M_1(\{(a_i, b_i, p_i)\}_{i=1}^m, a_j')$:

1. On receipt of the first message from Alice, for $i = 1, \ldots, m$ randomly select $x_i \stackrel{R}{\leftarrow} \mathbb{Z}_n$ and send $\{Enc^P(x_i)\}_{i=1}^m$ to Alice.

2. Select $r_b \stackrel{R}{\leftarrow} \{0,1\}^\ell$. At random select m distinct values from $\{0,1\}^\ell$, sort them in the increasing order obtaining $\{r_1, \ldots, r_m\}$, and set $y_i = Enc^P(r_i - x_i \mod n)$ for $i = 1, \ldots, m$. Engage in the execution of the binary search protocol with Alice using r_b and $\{y_i\}_{i=1}^m$ as input. At the end of the execution Alice receives a random index i as the outcome of the protocol.

3. Select a random $\omega \stackrel{R}{\leftarrow} \mathbb{Z}_n$, compute $\{Enc^P(a_j'), Enc^P(\omega)\}$, and send the pair to Alice.

According to Definition 1, Alice's view during an execution of the element-selection protocol Π is $\text{VIEW}_1^\Pi = (x, r_1, m_1, m_2, m_3)$. The distribution of x and r_1 remains the same for all possible input values, regardless of whether M_1 is used or a real protocol execution is performed. Next, we examine the messages that Alice receives. Let $M_1(x, f_1(x,y)) = (x', r_1', m_1', m_2', m_3')$.

Message m_1 is received in Step 4 of Π and is $m_1 = \{Enc^P(P_i' - y_i)\}_{i=1}^m$; message m_1' is received in Step 1 of simulation and is $m_1' = \{Enc^P(x_i)\}_{i=1}^m$. Due to the semantic security of the encryption scheme, encrypted values are uniformly distributed over the entire range resulting in identical distributions. After Alice decrypts the values, she still cannot distinguish between $P_i' - y_i$ and x_i because y_i's and x_i's are uniformly distributed over $\mathbb{Z}_n$.

Let us use Π_s to denote the binary search protocol of section 2.4. Then $m_2 = (\text{VIEW}_1^{\Pi_s}(x_s, y_s), i)$ and $m_2' = (\text{VIEW}_1^{\Pi_s}(x_s', y_s'), i')$, where $x_s = (\{x_i\}_{i=1}^m, r_a)$, $y_s = (\{y_i\}_{i=1}^m, r_b)$, $x_s' = (\{x_i'\}_{i=1}^m, r_a')$, and $y_s' = (\{y_i'\}_{i=1}^m, r_b')$, and all of $\{x_i\}_{i=1}^m$ and $\{x_i'\}_{i=1}^m$, $\{y_i\}_{i=1}^m$ and $\{y_i'\}_{i=1}^m$, r_a and r_a', and r_b and r_b' are pair-wise identically distributed. From Lemma 1 we obtain that the execution of Π does not leak any private information, and Lemma 3 tells us that i is uniformly distributed over $[1, m]$, and so is i'. Therefore m_2 and m_2' are also indistinguishable.

Lastly, $m_3 = \{Enc^P(a'_j), Enc^P(b'_j - \rho)\}$ and $m'_3 = \{Enc^P(a'_j), Enc^P(\omega)\}$. After Alice decrypts the values, the value of $b'_j - \rho$ is identically distributed to ω, and a'_j is the same in both messages. Also, no information can be gained from the encrypted values themselves. Thus m_3 and m'_3 are also indistinguishable.

Since we had $f(x, y) = \text{OUTPUT}^\Pi(x, y)$, we conclude Alice's view during an execution of Π is computationally indistinguishable from a simulation. The simulator M_2 for Bob's interaction can be constructed in a similar way and is omitted. Thus, Π privately computes the correlated action selection function. $\quad\square$

3 Handling Dishonest Behavior

In the previous section, we gave an efficient element-selection protocol for the honest-but-curious model. However, it is inefficient to make the preceding protocol secure against malicious adversaries, as the zero-knowledge proofs for certain steps of the protocol are very expensive. Instead, we present a new protocol for the malicious model, which uses two-party computation based on the conditional gate and relies on the use of threshold homomorphic ElGamal encryption.

3.1 Review of Cryptographic Tools Used

Homomorphic ElGamal Encryption Let G_q be a finite cyclic group of a prime order q, $|q| = k$, and g be the group's generator such that the Decision Diffie-Hellman (DDH) problem for G_q is assumed to be hard.[2] Given a published generator g, a public-private key pair for ElGamal encryption is generated as $(pk, sk) = (y, x)$, where $x \xleftarrow{R} \mathbb{Z}_q$ and $y = g^x$. Given a public key y and a message $M \in \mathbb{Z}_q$, encryption is performed as $Enc_y^G(M) = (\alpha, \beta) = (g^r, g^M y^r)$, where $r \xleftarrow{R} \mathbb{Z}_q$. Given the private key x, decryption of $(\alpha, \beta) = (g^r, g^M y^r)$ is performed by first computing $\beta/\alpha^x = g^M$ and then solving it for $M \in \mathbb{Z}_q$. This amounts to solving a discrete log problem and thus the message space must be small. In our protocols, the message space is $\{0, 1\}$ in most cases.

Such encryption is additively homomorphic, that is $Enc_y^G(a_1) \cdot Enc_y^G(a_2) = (g^{r_1} \cdot g^{r_2}, g^{a_1} y^{r_1} \cdot g^{a_2} y^{r_2}) = (g^{r_1+r_2}, g^{a_1+a_2} y^{r_1+r_2}) = Enc_y^G(a_1 + a_2)$. In addition, $Enc_{pk}^G(a)^b = Enc_{pk}^G(ab)$. Also, homomorphic ElGamal encryption is semantically secure assuming that the DDH problem is hard. When it is clear from the context or not essential to the discussion, we omit the encryption key from the notation and use $Enc^G(x)$ instead.

When Alice generates a ciphertext using homomorphic ElGamal encryption, she can prove that she knows the plaintext for the encryption using the techniques of [16]. She can make this a non-interactive proof of knowledge using Fiat-Shamir techniques [17]. Another proof of knowledge used in our protocols is a proof that a particular encryption is the encryption of 0 or 1. This protocol follows from the ability to prove the disjunction of two boolean values [18], and was given in [19].

---

[2] From this point on, arithmetic is assumed to be modulo q and operator $\mod q$ is implicit for each arithmetic operation.

Threshold Homomorphic ElGamal Encryption Homomorphic ElGamal encryption scheme can be used to construct (t, n)-threshold cryptosystem, where $0 < t \leq n$. In this case, the key is generated jointly by n parties, and decryption succeeds only if at least t parties participate. Encryption is performed in the traditional way, where anyone can use the public key y to encrypt messages.

Let $A_1, \ldots, A_n$ denote n players. As before, let the public key be y and let the private key be x with $y = g^x$. Then player A_i has a share x_i of the private key, where $y_i = g^{x_i}$ is public. Such shares can be generated using a secure distributed key generation protocol such as [20, 21], with communication complexity of $O(n^2 k)$ and a small hidden constant, where k is a security parameter.

To recover message M from its encryption (α, β), each player A_i computes a decryption share $d_i = \alpha^{x_i}$ and a proof that $\log_\alpha d_i = \log_g h_i$. Then having t correct decryption shares, M can be recovered from $g^M = \beta / \alpha^x$ by computing α^x from these shares using Lagrange interpolation. Decryption of private outputs is also possible in this framework, and it was shown in [7] how private output decryption used in RSA-like cryptosystem (such as Paillier's) can be modified to avoid having to decrypt an ElGamal encryption of a random messages in $\mathbb{Z}_q$. A non-interactive version of the protocol is also possible and can be found in [7].

Threshold homomorphic ElGamal cryptosystem is robust for $t < n/2$, but (non-robust) fairness can also be achieved for the two-party case using $(2, 2)$-threshold scheme. Note that neither party gains any advantage by quitting at an intermediate step of a protocol, and thus to achieve fairness, only the decryption phase of the protocols needs to be considered. This can be done using gradual release of information for a security parameter $k' < \log q$. See [7] for more detail. Note that allowing parties to prematurely quit during protocol execution will not allow us to finish the execution (and thus prove indistinguishability with the view in the ideal setting), and the protocol must be restarted.

Two-Party Computation Based on the Conditional Gate A recent work of Schoenmakers and Tuyls [7] introduced a new type of multiplication gate called *conditional gate* that permits efficient computation of two-party multiplication. In short, conditional gates permit efficient multiplication of x and y using homomorphic threshold ElGamal, where x is from a two-valued domain and y is unrestricted. In that work, conditional gates are also used to perform other types of secure computations such as XOR and different kinds of comparisons. In particular, the authors show how to perform comparison of two bitwise encrypted values x and y. Such operation requires ℓ rounds and $2\ell - 1$ conditional gates, where $|x| = |y| = \ell$, with the total of about 12ℓ modular exponentiations.

While individual operations are rather efficient and secure against malicious adversaries, the difficulty in applying these techniques to general function evaluation is in different representation of operands in such operations. That is, some operands are encrypted integers $x \in \mathbb{Z}_q$, while others are required to be encrypted in bitwise form, and there is no conversion procedure available between the two encryption formats.

Mixes One of the building blocks in our work is a mix, which was introduced in [22]. The parties "mix" a list of values by re-encrypting the values and permuting the order of the individual values. Furthermore, our protocols for the malicious model require that the mixing party be able to prove that the values were mixed properly. Also, we require that the protocols be able to mix vectors of values (where the vector consists of several encrypted values and the vector must be preserved). Examples of efficient mixes are [23, 24], and protocols for achieving a permutation of vectors can be found in [25].

3.2 The Element Selection Protocol

As before, we assume that $\sum_{i=1}^{m} p_i = 2^{\ell}$. We use $[a_i]_{\ell-1}\ldots[a_i]_0$ to denote the binary representation of a_i. For the purposes of this and subsequent sections, homomorphic ElGamal (2, 2)-threshold encryption is used.

Setup: Alice and Bob generate a key pair (pk, sk) for a security parameter k, where public key pk is known to both, but secret key sk is shared and is not known to either party.

Input: Items $\{(a_i, b_i, p_i)\}_{i=1}^{m}$ are known to both parties; public key pk is known to both and secret key sk is not known to either.

Output: Alice learns a_j, and Bob learns b_j, where j is the index selected according to the probability distribution.

Protocol Steps:

1. Alice encrypts tuples $\{(a_i, b_i, [p_i]_{\ell}, [p_i]_{\ell-1}\ldots[p_i]_0)\}_{i=1}^{m}$ with pk and then mixes them using a permutation π_a that she randomly generates. In the above, each of a_i and b_i are encrypted as an integer, but p_i's are encrypted bit by bit as $\ell + 1$ bit integers (i.e., the most significant bit is always 0).
   Alice proves in zero-knowledge that the output of this step was obtained using mixing π_a on the tuples $\{(a_i, b_i, p_i)\}_{i=1}^{m}$. Note that in order for Alice to prove proper mixing using known techniques, she first encrypts the list using no randomness (i.e., 0 in place of random values) and then proves that her output is a blinded permuted re-encryption of this list.
2. Bob blinds each of the items $(Enc^G(a_{\pi_a(i)}), Enc^G(b_{\pi_a(i)}), Enc^G([p_{\pi_a(i)}]_{\ell})$, $\ldots, Enc^G([p_{\pi_a(i)}]_0))$ by multiplying each value with $Enc^G(0)$ and mixes the tuples using a random permutation π_b. Let $(Enc^G(a_i'), Enc^G(b_i'), Enc^G(p_i'))$ denote $(Enc^G(a_{\pi_b(\pi_a(i))}), Enc^G(b_{\pi_b(\pi_a(i))}), Enc^G(p_{\pi_b(\pi_a(i))}))$ for $i = 1, \ldots, m$. Bob proves in zero-knowledge that his output was constructed by applying a random mix π_b to his input.
3. Alice and Bob compute $(Enc^G(a_i'), Enc^G(b_i'), Enc^G([P_i']_{\ell}), \ldots, Enc^G([P_i']_0))$, where $P_i' = \sum_{i=1}^{m} p_i'$. The description of this step (i.e., the addition operation) is given in section 3.3.
4. Alice picks $r_a \xleftarrow{R} \{0,1\}^{\ell}$, computes $\{Enc^G([r_a]_{\ell-1}), \ldots, Enc^G([r_a]_0)\}$ and sends it to Bob. She also proves in zero-knowledge that each $[r_a]_i$ in the encryptions corresponds to either 0 or 1. Similarly, Bob picks $r_b \xleftarrow{R} \{0,1\}^{\ell}$,

sends Alice $\{Enc^G([r_b]_{\ell-1}), \ldots, Enc^G([r_b]_0)\}$, and proves in zero-knowledge that each $[r_b]_i$ corresponds to a single bit.

5. Alice and Bob compute the bitwise encrypted value of $x = r_a + r_b \mod 2^\ell$ using the addition protocol of section 3.3. They prepend bitwise encrypted x with $Enc^G(0)$ to obtain $(\ell + 1)$-bit representation of x.

6. Alice and Bob jointly find the index of the value P'_i such that $x < P'_i$ and $x \geq P'_{i-1}$ (if P'_{i-1} exists) from the list $\{Enc^G([P'_i]_\ell), \ldots, Enc^G([P'_i]_0)\}_{i=1}^m$ using the binary search algorithm described in section 3.5. Let the outcome of the search be index j.

7. Having $Enc^G(a'_j)$ and $Enc^G(b'_j)$, Alice helps Bob to decrypt b'_j and Bob helps Alice to decrypt a'_j (see section 3.1 for detail).

Note that most of the work done in step 1 can be performed in advance (if the public key is available prior to protocol execution), by generating as many encryptions of 0's and 1's as needed. At the time of protocol execution, Alice then just selects the right combination of such encryptions to match the p_i's. Similarly, values for the zero-knowledge proof in that step and encryptions of 0 in step 2 can be pre-computed, thus reducing computational cost of asymptotically least efficient parts of the protocol.

The security proof of the protocol is omitted. One interesting direction for future work is to narrow the gap in the complexities between the semi-honest and malicious models.

3.3 Addition of Bitwise Encrypted Values

Here we first present an addition protocol with computational and round complexity of $O(\ell)$. After its description we show how its round complexity can be significantly lowered using standard techniques.

Input: Common input consists of encryptions $\{Enc^G([x]_{\ell-1}), \ldots, Enc^G([x]_0)\}$ and $\{Enc^G([y]_{\ell-1}), \ldots, Enc^G([y]_0)\}$.

Output: Alice and Bob obtain $\{Enc^G([z]_{\ell-1}), \ldots, Enc^G([z]_0)\}$, where $z = x + y \mod 2^\ell$.

Protocol Steps:

1. Alice and Bob compute encryptions of $[z]_0 = [x]_0 \texttt{ XOR } [y]_0$ and $c = x_0 \texttt{ AND } y_0$ as follows. Computation of $Enc^G(c) = Enc^G([x]_0 \cdot [y]_0)$ is performed using the conditional gate; then computation of $Enc^G([z]_0) = Enc^G([x]_0 + [y]_0 - 2[x]_0 \cdot [y]_0) = Enc^G([x]_0 + [y]_0 - 2c)$ is performed locally using common randomness.

2. For $i = 1, \ldots, \ell - 1$, Alice and Bob compute encryptions of $[z]_i = (([x]_i \texttt{ XOR } [y]_i) \texttt{ XOR } c)$ and $c = \texttt{MAJ}([x]_i, [y]_i, c)$ as follows:
   (a) Using the conditional gate, Alice and Bob compute $Enc^G(a_{xy}) = Enc^G([x]_i \cdot [y]_i)$, $Enc^G(a_{xc}) = Enc([x]_i \cdot c)$, $Enc^G(a_{yc}) = Enc^G([y]_i \cdot c)$, and $Enc^G(a_{xyc}) = Enc^G([x]_i \cdot [y]_i \cdot c)$.

(b) Using common randomness, Alice and Bob locally compute $Enc^G([z]_i) = Enc^G(4a_{xyc} - 2a_{xy} - 2a_{xc} - 2a_{yc} + [x]_i + [y]_i + c)$ and then $Enc^G(c) = Enc^G(a_{xy} + a_{xc} + a_{yc} - 2a_{xyc})$.

Logarithmic depth addition of two integers is carried out by the textbook carry-lookahead addition circuit [26] that has logarithmic depth and linear size (number of Boolean gates). Given $p_1, \ldots, p_m$, the prefix sum problem [27] is to compute all the sums $p_1 + \ldots + p_i$, $i = 1, \ldots, m$. It can be solved by a logarithmic depth circuit with a linear number of addition nodes [27]. If each addition node of the circuit of [27] is replaced by the circuit of [26], then the resulting Boolean circuit for the prefix problem for ℓ-bit numbers has $O(m\ell)$ gates and depth $O(\log m \log \ell)$. However, the use of the Wallace tree technique [28] is known to reduce the depth to $O(\log m + \log \ell)$ (see, e.g., [29]).

3.4 Constant-Round Comparison

Although we could carry out comparison in our model using the method given in [7], this would require $O(\ell)$ number of rounds. Below we give a constant-round comparison protocol, which is of independent interest.

Input: Alice and Bob each have encryptions $\{Enc^G([x]_{\ell-1}), \ldots, Enc^G([x]_0)\}$ and $\{Enc^G([y]_{\ell-1}), \ldots, Enc^G([y]_0)\}$.

Output: Alice and Bob obtain 1 if $x \geq y$, and 0 otherwise.

Protocol Steps:

1. Alice and Bob both locally compute $Enc^G(e_{\ell-1}) = Enc^G(x_{\ell-1} - y_{\ell-1})$ and then compute $Enc^G(e_i) = Enc^G(2e_{i+1} + x_i - y_i)$ for all $i \in \{\ell - 2, \ldots, 0\}$. Note that the value e_i will be 0 until the first difference between x and y.
2. Alice and Bob locally compute $Enc^G(f_{\ell-1}) = Enc^G(y_{\ell-1} - x_{\ell-1} - 1)$ and then $Enc^G(f_i) = Enc^G(3e_{i+1} + y_i - x_i - 1)$ for all $i \in \{\ell - 2, \ldots, 0\}$. Note that the value f_i will be 0 if the first $i - 1$ bits are equal and the ith bit of x is false and the ith bit of y is true. Thus if there is a single 0 entry (and there will be at most one) then $x < y$ and otherwise $x \geq y$.
3. Alice and Bob raise $Enc^G(f_i)$ to a random power (a protocol for doing this was described in [30]). Note that now the list of values will contain a 0 if $x < y$ and will be a set of random non-zero values otherwise.
4. Alice mixes the list and sends the mixed list to Bob along with a proof of proper mixing. Similarly, Bob mixes the list and sends the mixed list to Alice along with a proof of proper mixing.
5. Alice and Bob jointly decrypt the list and if a single entry is 0, then they output 0. If no entry is 0, then they output 1.

3.5 Binary Search

Here we give an efficient search protocol for step 6 of the main protocol. The overall complexity is $O(\ell \log m)$ and the round complexity is $O(\log m)$.

Input: A list of sorted bitwise encrypted m values $\{Enc^G([y_i]_{\ell-1}), \ldots, Enc^G([y_i]_0)\}_{i=1}^m$ and value x bitwise encrypted as $Enc^G([x]_{\ell-1}), \ldots, Enc^G([x]_0)$.

Output: The smallest index j such that $y_j > x$.

Protocol Steps: Alice and Bob execute the following recursive procedure on the bitwise-encrypted list $\{y_i\}_{i=1}^m$:

1. If the size of the current working set $|\{y_i, \ldots, y_j\}| = 1$, return i.
2. Otherwise, Alice and Bob execute the constant-round comparison protocol (see section 3.4) on (the encrypted values of) x and $y_{\lceil \frac{j-i+1}{2} \rceil}$ (i.e., check whether $x \geq y_{\lceil \frac{j-i+1}{2} \rceil}$). Let c denote the outcome of the protocol.
3. If $c = 1$, recurse on list $\{y_{\lceil \frac{j-i+1}{2} \rceil + 1}, \ldots, y_j\}$; otherwise, recurse on list $\{y_i, \ldots, y_{\lceil \frac{j-i+1}{2} \rceil}\}$.

Acknowledgments

References

1. Dodis, Y., Halevi, S., Rabin, T.: A cryptographic solution to a game theoretic problem. In: Advances in Cryptology – Crypto'00. (2000)
2. Teague, V.: Selecting correlated random actions. In: Financial Cryptography. Volume 3110. (2004) 181–195
3. Bárány, I.: Fair distribution protocols or how the players replace fortune. Mathematics of Operation Research **17** (1992) 327–341
4. Ben-Porath, E.: Correlation without mediation: Expanding the set of equilibria outcomes by "cheap" pre-play procedures. Journal of Economic Theory **80** (1998) 108–122
5. Gerardi, D.: Unmediated communication in games with complete and incomplete information. Journal of Economic Theory **114** (2004)
6. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely fair SFE and coalition-safe cheap talk. In: Symposium on Principles of Distributed Computing (PODC'04). (2004) 1–10
7. Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: ASIACRYPT'04. Volume 3329. (2004) 119–136
8. Ambainis, A., Jakobsson, M., Lipmaa, H.: Cryptographic randomized response techniques. In: Workshop on Theory and Practice in Public Key Cryptography (PKC'04). Volume 2947 of LNCS. (2004) 425–438
9. Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Advances in Cryptology – CRYPTO'05. Volume 3621 of LNCS. (2005) 378–411
10. Yao, A.: How to generate and exchange secrets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press (1986) 162–167

11. Goldreich, O.: The Foundations of Cryptography — Volume 2. Cambridge University Press (2004)
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology: EUROCRYPT '99. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 223–238
13. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography, Springer (2001) 119–136
14. Frikken, K., Atallah, M.: Privacy preserving route planning. In: Proceedings of the 3rd ACM Workshop on Privacy in the Electronic Society, Washington, DC, USA (2004) 8–15
15. Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the k th-ranked element. In: Advances in Cryptology – EUROCRYPT'04. Volume 3027 of LNCS. (2004) 40–55
16. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology **4** (1991) 161–174
17. Fiat, A., Shmair, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology – CRYPTO'86. Volume 263 of LNCS. (1986) 186–194
18. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in Cryptology – EUROCRYPT'94. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 174–187
19. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Advances in Cryptology – ASIASCRYPT'00. Volume 1976 of LNCS. (2000) 162–177
20. Pedersen, T.: A threshold cryptosystem without a trusted party. In: Advances in Cryptology – EUROCRYPT'91. Volume 547 of LNCS. (1991) 522–526
21. Gennaro, R., Jarecki, S., Krawzyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystem. In: Advances in Cryptology – EUROCRYPT'99. Volume 1592 of LNCS. (1999) 295–310
22. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. In: Communications of the ACM. Volume 24(2). (1981) 84–88
23. Jakobsson, M.: A practical mix. In: Advances in Cryptology – EUROCRYPT '98. Volume 1403. (1998) 448–461
24. Jakobsson, M., Juels, A., , Rivest, R.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX. (2002) 339–353
25. Golle, P., Jakobsson, M.: Reusable anonymous return channels. In: ACM Workshop on Privacy in the Electronic Society (WPES'03). (2003) 94–100
26. Ofman, Y.P.: On the algorithmic complexity of discrete functions. English translation of Soviet Physics Doklady **7** (1963) 589–591
27. Ladner, R., Fischer, M.: Parallel prefix computation. Journal of the Association for Computing Machinery (27) (1980) 831–838
28. Wallace, C.: A suggestion for a fast multiplier. IEEE Transactions on Electronic Computers **13** (1964) 14–17
29. Zheng, S., Yang, M., Masetti, F.: Constructing schedulers for high-speed, high-capacity switches/routers. International Journal of Computers and Applications **26** (2003) 4–271
30. Brandt, F.: Fully private auctions in a constant number of rounds. In: Financial Cryptography Conference (FC'03). Volume 2742 of LNCS. (2003) 223–238