Selecting correlated random actions

Vanessa Teague*

Stanford University, Stanford CA 94305, USA,
vteague@cs.stanford.edu

Abstract. In many markets, it can be beneficial for competing firms to coordinate their actions. However, such arrangements usually have the problem that nobody has an incentive to adhere to the agreement. In this paper we investigate a way for two companies to communicate together and agree on a coordinated strategy in such a way that both participants have an incentive to keep to the agreement.
We provide a more efficient solution to the game theoretic problem solved by Dodis, Halevi and Rabin in [DHR00]: two selfish rational parties want to select one of a list of pairs, according to some probability distribution, so that one party learns the first element and the other learns the second, and neither gains any other information. In game theory terms, the problem is to achieve a correlated equilibrium without the trusted mediator. Our solution is more efficient than [DHR00] in terms of the probability distribution.

1 Introduction

The Internet is composed of participants who are often neither malicious nor perfectly honest, but are selfish and (reasonably) rational just like the traditional game theory players. This paper is part of a growing body of work (including for example [AET01], [FPSS02], [FPS01], [NPS99], [NR01], [RT02]) that aims to consider both the agents' computational limitations and their incentives when solving problems. In this case, we solve an existing problem in game theory by making computational assumptions and using cryptography. Our solution is more efficient than the one in [DHR00] in terms of one of the parameters (a probability distribution).

Consider for example two competing furniture stores, selling roughly the same secondhand chairs from failed dot-coms for roughly the same prices. Each week, the CEO of each store may choose either to retain the usual prices or to have a special sale and discount the chairs substantially. They must choose in advance (so they can't just wait to see what the other does). If both decide to retain their usual higher prices then both make a moderate amount of money that week. If one retains the normal prices but the other goes on sale, then the latter will have far more customers and make more money than in a normal week, while the former will make less. However, if both decide to have a sale

* Supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR grant N00014-01-1-0795.

at the same time then this is a disaster because they will not get many more customers than usual but they will receive much less money per customer. An example of the possible payoffs is shown in Table 1, where the first number of each pair is the payoff of the row player and the second is that of the column player. The state with the highest average income (per store) is the one in which neither offers a sale. Unfortunately, there is a reason to expect that the stores will not remain in this state: if either side is confident that the other will not offer a sale, then it has an incentive to offer one itself, thus gaining 12 instead of 9. In game theory terms, (No Sale, No Sale) is not a Nash equilibrium. Indeed, there is no Nash equilibrium better than the ones where one player offers a sale and the other doesn't.

Payoffs

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

Probabilities

	No Sale	Sale
No Sale	$5/11$	$3/11$
Sale	$3/11$	0

Table 1. Payoffs and probabilities for the game

A Nash equilibrium might be *mixed*, meaning that each player randomizes among several different actions. In a (mixed) Nash equilibrium, the players' random choices must be independent. However, in some games (such as this one) the players can earn a higher average payoff by randomizing according to some correlated joint distribution. The idea of *correlated equilibrium* is that there is a trusted party who helps the players to do the correlation. It chooses a particular move for each player from a commonly-known distribution over pairs of actions, then it tells each player what action to take. (More precise definitions of Nash and correlated equilibria are given in section 2.)

It is natural for cryptographers (and game theorists) to ask whether the players can achieve the same payoff without the trusted party. That is, if the players are allowed to talk amongst themselves before the game, can they correlate their random choices without giving away any information that will destroy the equilibrium? In the game theory literature, Barany [Bar92] showed that this was possible for four or more players, but impossible for two computationally unbounded players (except in some uninteresting cases). Dodis, Halevi and Rabin [DHR00] have shown that by making computational assumptions, it is possible to solve this problem for two players using cryptography. It is a Nash equilibrium for the players to execute the cryptographic protocol that simulates the trusted party correctly and then choose the appropriate move. (There is also a solution by Urbano and Vila [UV02], but their cryptographic protocol has security flaws.) The contribution of this paper is to provide a protocol that solves the same problem as [DHR00] and is more efficient in terms of the probability distribution required.

Although the protocol presented in [DHR00] is very efficient in terms of the security parameter, it is designed for uniform distributions and is not efficient in terms of the probability distribution that the parties are supposed to be taking their moves from. The protocol presented in this paper is much more efficient in this parameter. (The form of the protocol that is designed for selfish agents is less efficient than [DHR00]'s in terms of the security parameter, which is itself a function of the end-of-game payoffs.) Although most examples of correlated equilibria in game theory textbooks (for example, [OR94] and [FT91]) do use a uniform distribution, there is no particular reason to expect that most correlated equilibria should be uniform or nearly uniform. For example, the game in Table 1 has a correlated equilibrium with the distribution given in the table. It also has a correlated equilibrium based on a uniform distribution among the same three pairs of actions, but that equilibrium has a lower average payoff. Section 2 contains proofs of these claims.

In the following section we give some background game theory and explain the problem more carefully. In section 3 we present a protocol that is secure against honest-but-curious players, and in section 4 we show how to compile the protocol for security against one malicious player. We then show that two selfish players would execute the protocol correctly.

2 Definitions and Game Theory Background

In this section we provide enough background to explain the problem that we are trying to solve.

Definition 1. *A* Game *for two players consists of:*

- *for each player* $i \in \{0,1\}$ *a nonempty set* A_i *of* actions*, and*
- *for each player* $i \in \{0,1\}$ *a* utility function $u_i : A_1 \times A_2 \to \mathbb{R}$.

The utility function represents how happy a player is with a certain outcome. The larger its value, the better-off the player. Players may randomize their actions. A *strategy* in a strategic game is a probability distribution over actions. (The notion of strategy will be broadened for extended games later in this section.) A *Nash equilibrium* is a self-enforcing agreement by each agent to choose a certain strategy. Given the agreement, neither player has an incentive to deviate from it unilaterally. The main restriction is that the random choices made by the two players must be independent.

Definition 2. *A* Nash equilibrium *of a game* G *is a pair of independent strategies* (σ_1^*, σ_2^*) *such that for any* $a_1 \in A_1$ *and* $a_2 \in A_2$*, we have* $u_1(\sigma_1^*, \sigma_2^*) \geq u_1(a_1, \sigma_2^*)$ *and* $u_2(\sigma_1^*, \sigma_2^*) \geq u_2(\sigma_1^*, a_2)$.

The idea of a correlated equilibrium is that there is a trusted third party who recommends an action to each player before the game. It chooses the pair of actions according to a probability distribution that is common knowledge. Hence the recommended action gives each player some information about its

opponent's action, since it knows the distribution of what will be recommended to its opponent conditioned on its own recommended action. Each player's recommended action should be its best strategy, given this information. This gives us a generalization of Nash equilibrium in which the players' random choices may be correlated.

We write $\sigma_2^*|a_1^*$ for the probability distribution on the action recommended to player 2, conditioned on action a_1^* being recommended to player 1. The utility $u_1(a_1, \sigma_2^*|a_1^*)$ is player 1's expected utility for choosing action a_1, assuming that player 2 acts according to the distribution $\sigma_2^*|a_1^*$. Similarly, $u_2(\sigma_1^*|a_2^*, a_2)$ is player 2's expected utility for a_2, given its information.

Definition 3. *A* correlated equilibrium *is a pair of strategies $s^* = (\sigma_1^*, \sigma_2^*)$ such that for any (a_1^*, a_2^*) in the support of s^*, for any $a_1 \in A_1$ and $a_2 \in A_2$, we have $u_1(a_1^*, \sigma_2^*|a_1^*) \geq u_1(a_1, \sigma_2^*|a_1^*)$ and $u_2(\sigma_1^*|a_2^*, a_2^*) \geq u_2(\sigma_1^*|a_2^*, a_2)$.*

It is easy to see that for any Nash equilibrium there is a correlated equilibrium with the same distribution on pairs of actions (and therefore the same average payoff). Correlated equilibrium payoffs can be outside the convex hull of Nash equilibrium payoffs. For example, consider the game in Table 1. We first show that the probabilities given in the table produce a correlated equilibrium. Since the game is symmetric, we need only consider the incentives of player 2, the column player, given what it has been told to play. The conditional probabilities for player 1's action given what has been recommended to player 2 are:

$$\begin{aligned} Pr\,(a_1 = \text{No Sale}|a_2 = \text{No Sale}) &= \frac{5/11}{3/11 + 5/11} \\ &= 5/8 \\ Pr\,(a_1 = \text{Sale}|a_2 = \text{No Sale}) &= 3/8 \\ Pr\,(a_1 = \text{No Sale}|a_2 = \text{Sale}) &= 1 \\ Pr\,(a_1 = \text{Sale}|a_2 = \text{Sale}) &= 0 \end{aligned}$$

Suppose the column player is told to play Sale. Then it can be certain that the row player has been told to play No Sale, so the column player gets its maximum possible payoff, namely 12, for doing as it is told. Suppose instead that the column player is told to play No Sale. Then its expected payoff for being obedient and playing No Sale is $9Pr\,(a_1 = \text{No Sale}|a_2 = \text{No Sale}) + 5Pr\,(a_1 = \text{Sale}|a_2 = \text{No Sale}) = 7\frac{1}{2}$. If it decides to disobey and play Sale instead, its expected payoff is $12Pr\,(a_1 = \text{No Sale}|a_2 = \text{No Sale}) + 0Pr\,(a_1 = \text{Sale}|a_2 = \text{No Sale}) = 7\frac{1}{2}$, so it has no incentive to deviate. Hence this is a correlated equilibrium.

There are two pure-strategy Nash equilibria, (No Sale, Sale) and (Sale, No Sale), both of which have an average payoff per player of $8\frac{1}{2}$. There is also a mixed Nash equilibrium in which each player plays No Sale with probability 5/8. This gives an average payoff of $7\frac{1}{2}$. The correlated equilibrium in the table gives the players an average payoff of $9 \times 5/11 + 12 \times 3/11 + 5 \times 3/11 = 8\frac{8}{11}$, which is better than any Nash equilibrium.

Furthermore, a correlated equilibrium with a non-uniform distribution can give a strictly higher average payoff than any with a uniform distribution in the same game. For example, in the game in Table 1, the uniform distribution over (Sale, Sale), (Sale, No Sale), (No Sale, Sale) also gives a correlated equilibrium, but this has an average payoff of $(5 + 12 + 9)/3 = 8\frac{2}{3}$ which is lower than the $8\frac{8}{11}$ achieved with the non-uniform distribution in the table.

It is interesting to wonder what happens for an arbitrary game when the trusted party is allowed to send more informative messages, rather than just informing each player of the action it is supposed to take. Upon first glance it seems that this might increase the set of equilibria, but it is a standard result in game theory (see [OR94] prop. 47.1) that it doesn't. For every equilibrium that is based upon some complicated messages from the trusted party, there is another that has the same distribution on action pairs (and therefore the same payoffs) in which the trusted party tells each player what action to choose and nothing else. This is why we adopt Definition 3 instead of a more complicated variant.

An *extended game* is a game preceded by a period of unrestricted communication among the parties. This is called "cheap talk" in the game theory literature, because the talk doesn't directly affect anyone's utility—the payoffs for the extended game are just the payoffs for the standard game that is played in the last step. A *strategy* in such a game consists of both a way of conducting the "cheap talk" (which may be randomized and may depend on messages received from others) and a choice of action afterwards.

One technical difficulty is defining such standard notions as Nash equilibrium given computational assumptions. We follow [DHR00] in assuming that players ignore any improvements in their utility that are negligible in the security parameter. For example, they don't bother trying to guess each other's private keys, because they know they have only a negligible probability of success.

Definition 4. *[DHR00, Definition 3] A* computational Nash equilibrium *of a game is a pair of independent strategies* (σ_1^*, σ_2^*) *such that*

- *both* σ_1^* *and* σ_2^* *are PPT computable.*
- *for any other PPT computable strategies* σ_1 *and* σ_2*, there exists a negligible function* μ *such that on security parameter* k*, we have* $u_1(\sigma_1^*, \sigma_2^*) \geq u_1(\sigma_1, \sigma_2^*) + \mu(k)$ *and* $u_2(\sigma_1^*, \sigma_2^*) \geq u_2(\sigma_1^*, \sigma_2) + \mu(k)$*.*

The idea now is to devise a cryptographic protocol that selects a pair of actions from some distribution and reveals to each player what action it should take and nothing else. We will call this the *correlated element selection* problem. It should be a computational Nash equilibrium of the extended game for each player to execute the protocol correctly during the "cheap talk" phase and then choose the action that is recommended by it.

Dodis *et al.* present an efficient protocol for solving the correlated element selection problem with the uniform distribution. They show how to augment this protocol to make it secure against malicious parties, which is enough to make it a computational Nash equilibrium for selfish parties to execute the protocol and

then take the action that they are supposed to, assuming the uniform distribution was indeed a correlated equilibrium. Their protocol is very efficient for uniform distributions, but the solution for non-uniform distributions is to repeat some of the pairs until choosing uniformly from the resulting list gives the correct distribution. This has an asymptotic efficiency on the order of the least common multiple of the denominators of the probabilities. Our contribution is a protocol for solving the same problem that is more efficient in terms of the probability distribution: its asymptotic efficiency is on the order of the inverse of the smallest non-zero probability. This is always smaller (except in the uniform case and some other simple cases, when it is equal), because the inverse of a number in $(0, 1]$ is smaller than or equal to the denominator, so the smallest of the inverses is smaller than or equal to the least common multiple of the denominators.

In both protocols, the longest message is a list sent in the first step. For example, for the game in Table 1 our protocol uses a list of length $\lceil 11/3 \rceil = 4$ while the one in [DHR00] uses one of length l.c.m $\{11, 11, 11\} = 11$. If we wanted a correlated equilibrium that consisted of choosing one pair of actions with probability $1/100$, one with probability $1/99$ and another with probability $1 - 1/99 - 1/100$, then Dodis *et al*'s protocol would use a list of 9900 messages, while our protocol would use 100. In the uniform case, the lists have the same length for both protocols. However, our messages are larger by a constant factor (about 3).

2.1 Summary: how the protocol fits in to the game

The traditional cryptographic assumption is that at most one player is *malicious*, meaning that it might deviate from the protocol in any computationally feasible way, while the other is completely honest. The protocol must either produce the correct answer and reveal no other information, or stop early without revealing information (See [Gol03] for a more precise statement). However, in this case we can't assume that one player is honest. The usual game theoretic assumption is that both players are *rational*. We assume that each player might deviate from the protocol (in any computationally feasible way) if this increases its expected utility (by a non-negligible amount).

In cryptography, it usually suffices to show that the honest player will notice if a malicious player tries to cheat. However, in our setting both players still have to choose an action after the protocol, so we must describe what action they should choose after detecting that the other has cheated. We follow [DHR00]: if player 1 detects that player 2 has cheated, it punishes 2 by choosing the action a_1 that minimizes $\max_{a_2}\{u_2(a_1, a_2)\}$. This is called player 2's minimax action. Likewise, if player 2 catches player 1 cheating, it minimaxes 1. It is shown in [DHR00] that the cheating player's minimax utility is always smaller than its correlated equilibrium payoff, so it has an incentive to be honest as long as its probability of being caught when cheating is sufficiently large. We use standard cryptographic techniques to make sure this is the case.

Consider the extended game defined by a period of "cheap talk" followed by the playing of a game. Let $\mathcal{P}$ be the protocol for correlated element selection

given in section 4, parameterized for a particular correlated equilibrium of the game. Let S_{honest} be the strategy, "follow $\mathcal{P}$ until you detect the other player cheating. If you reach the end without detecting cheating, choose the action output at the end of $\mathcal{P}$. Otherwise, punish your opponent to its minimax level."

A precise statement of our main result is:

Theorem 1. *It is a computational Nash equilibrium of the extended game for each player to follow strategy S_{honest}.*

This result means that two selfish players could be expected to implement the protocol correctly and then choose the recommended moves, thus achieving a correlated equilibrium without the trusted party.

While both players following strategy S_{honest} is a computational Nash equilibrium, it is not subgame perfect because it involves an "incredible threat". A player may have no incentive to punish the other even if it is caught cheating, since the minimax state may hurt the punisher as much as the cheater. Such incredible threats are allowed in Nash equilibria because we assume that each player chooses its strategy in advance and does not maintain control of the algorithm. However, for some games a subgame perfect equilibrium is possible: if the original (unextended) game has, for each player, a Nash equilibrium that gives that player a sufficiently small payoff, then that state can be used as punishment rather than a minimax. In that case, a rational player that detects the other cheating and can still control what move to make will have no incentive not to punish the other correctly.

3 Protocol for honest-but-curious players

We present a protocol for the correlated element selection problem that is efficient in terms of both the security parameter and the required probability distribution. This protocol has message size and computation time linear in the security parameter and linear in the inverse of the smallest probability in the distribution on pairs of actions. This protocol is secure in the honest-but-curious model, meaning that the players are allowed to record everything they know about the interaction and try to learn from it, but they are not permitted to deviate from the protocol in any other way, including the tossing of fair coins. In then next section, we will show how to make the protocol secure against one malicious player or two selfish players. There may be interesting applications for this protocol other than finding correlated equilibria.

We first describe a useful cryptographic primitive used in [DHR00] and also in our protocol. Blindable encryption allows someone who knows only the public key to transform a ciphertext of a message m into a random ciphertext of $m+m'$, for any value of m' that it chooses. (In the following definition, when we want to write the random inputs to a function explicitly, we separate them from the non-random inputs with a semicolon).

3.1 Blindable encryption

Definition 5. *[DHR00, Definition 4]. A public key encryption scheme $\mathcal{E}$ with public key pk is* blindable *if there exist (PPT) algorithms Blind and Combine such that for every message m and every ciphertext $c \in Enc_{pk}(m)$:*

- *For any message m' (also referred to as the "blinding factor"), $Blind_{pk}(c, m')$ produces a random encryption of $m + m'$.*
- *If r_1, r_2 are the random coins used by two successive "blindings", then for any two blinding factors m_1, m_2,*

$$Blind_{pk}(Blind_{pk}(c, m_1; r_1), m_2; r_2) = Blind_{pk}(c, m_1 + m_2; Combine_{pk}(r_1, r_2))$$

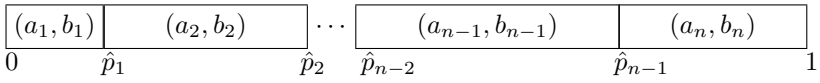
Two blindable encryption schemes are ElGamal and Goldwasser-Micali ([GM84]).

Blindable encryption provides an easy way for player 1 to ask player 2 to decrypt a message encrypted with 2's public key, without player 2 learning anything about what it has decrypted. Player 1 simply chooses a random blinding factor, blinds the ciphertext, asks player 2 for the decryption and then subtracts the blinding factor from the result. We will use this idea twice in this protocol. Blindable encryption also provides a useful way to make a given ciphertext unrecognizable. Suppose there are several encryptions of the same message. Someone who knows the public key can blind a ciphertext by zero, producing a random encryption of that message. Even someone who knows the private key cannot tell which encryption of the message the new ciphertext was derived from.

3.2 The protocol

Figure 1 describes the improved protocol for correlated element selection. There is a longer explanation in the following text. We assume that all numbers are expressed to $\hat{k} = \Theta(k)$ bits of precision, where k is the security parameter. Choosing a random number from a real interval means choosing randomly from among the (finitely many) $\hat{k}$-precision numbers in that interval.

The protocol is a variation on Dodis *et al.*'s protocol for correlated element selection. The common input is a list of triples $\{(a_i, b_i, p_i)\}_{i=1}^{n}$, meaning that the pair of actions (a_i, b_i) should be chosen with probability p_i. The output is an action a_i for the chooser and the corresponding action b_i for the preparer. Let $\hat{p}_i$ denote $\sum_{j=1}^{i} p_j$. We assume $\hat{p}_n = 1$. The probabilities divide the interval $[0, 1)$ into n intervals $[0, \hat{p}_1), [\hat{p}_1, \hat{p}_2), \ldots, [\hat{p}_{n-1}, 1)$, with each interval corresponding to a pair of actions.



Protocol CES

Common inputs: List of pairs of actions with probabilities $\{(a_i, b_i, p_i)\}_{i=1}^{n}$, public key pk, security parameter k.
Preparer knows: secret key sk.
Outputs: With probability p_i, C outputs a_i and P outputs b_i.
All calculations are done with $\Theta(k)$ bits of precision.

P: **1. Shift and encrypt**
Choose a random $r_0 \in [0, 1)$.
Choose minimum l s.t. $1/l \leq \min_i\{p_i\}$ and divide $[0, 1)$ into l equal-sized blocks.
Let $\hat{p}_i$ denote $\sum_{j=1}^{i} p_j$, and $frac(x)$ the fractional part of x
For $\lambda = 1 \ldots l$ make block λ as follows:
If there is an i with $frac(\hat{p}_i + r_0) \in [(\lambda - 1)/l, \lambda/l)$, make block
$((Enc_{pk}(a_i), Enc_{pk}(b_i)), Enc_{pk}(frac(l(\hat{p}_i + r_0))/l), (Enc_{pk}(a_{i+1}), Enc_{pk}(b_{i+1}))$
Otherwise, choose a random $q_\lambda \in [0, 1/l)$ and make block
$((Enc_{pk}(a_j), Enc_{pk}(b_j)), Enc_{pk}(q_\lambda), (Enc_{pk}(a_j), Enc_{pk}(b_j))$
(where $j = \min\{j' | frac(\hat{p}_{j'} + r_0) > (\lambda - 1)/l\}$)
Send the list of blocks to C.

C: **2. Choose and blind**
Choose a random block $((c_{a_1}, c_{b_1}), c_p, (c_{a_2}, c_{b_2}))$ from the list.
Choose a random blinding factor β_1
Send $c'_p = Blind_{pk}(c_p, \beta_1)$ to P.

P: **3. Blindly decrypt probability**
Send $p' = Dec_{sk}(c'_p)$ to C.

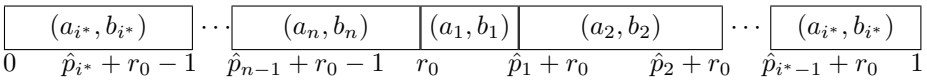
C: **4. Choose actions**
Let $p = p' - \beta_1$.
Generate a new random number β_2.
With probability lp, set $(e, f) = (Blind_{pk}(c_{a_1}, \beta_2), Blind_{pk}(c_{b_1}, 0))$
Otherwise set $(e, f) = (Blind_{pk}(c_{a_2}, \beta_2), Blind_{pk}(c_{b_2}, 0))$
Send (e, f) to P.

P: **5. Decrypt and output**
Set $\tilde{b} = Dec_{sk}(e)$. Send $\tilde{b}$ to C.
Output $Dec_{sk}(f)$.

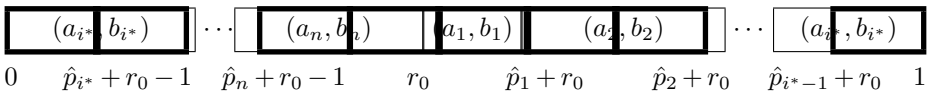
C: **6. Unblind and output**
Set $b = \tilde{b} - \beta_2$. Output b.

Fig. 1. A protocol for correlated element selection in the honest-but-curious model

Preparation The preparer generates a random number $r_0 \in [0,1)$ and shifts the probability intervals along by adding r_0 to each and "wrapping" back around to zero for those sums that are greater than 1. That is, let $i^* = \min\{i | r_0 + \hat{p}_i > 1\}$. The preparer creates new intervals $[0, \hat{p}_{i^*} + r_0 - 1), [\hat{p}_{i^*} + r_0 - 1, \hat{p}_{i^*+1} + r_0 - 1), \ldots, [\hat{p}_{n-1} + r_0 - 1, r_0), [r_0, \hat{p}_1 + r_0), \ldots, [\hat{p}_{i^*-1} + r_0, 1)$. It shifts the corresponding action pairs along with the intervals, so for $i < i^*$ the action pair (a_i, b_i) corresponds to the interval $[\hat{p}_{i-1} + r_0, \hat{p}_i + r_0)$, while for $i > i^*$, (a_i, b_i) corresponds to the interval $[\hat{p}_{i-1} + r_0 - 1, \hat{p}_i + r_0 - 1)$ and the action pair (a_{i^*}, b_{i^*}) is split over the two intervals $[0, \hat{p}_{i^*} + r_0 - 1)$ and $[\hat{p}_{i^*-1} + r_0, 1)$.



The preparer then chooses the smallest $l \in \mathbb{N}$ so that $1/l \leq \min_i\{p_i\}$, and divides the interval $[0,1)$ into l intervals of equal size, which will be called "blocks". (This is why the resulting message length is linear in the inverse of the smallest probability.) Some blocks fall entirely within one of the shifted probability intervals (and hence correspond to only one pair of actions), while others overlap the border between two (and hence correspond to two pairs of actions). No block corresponds to three or more pairs of actions because the size of a block is chosen to be smaller than the smallest interval that one pair occupies.



The idea is that the chooser will choose one of these blocks randomly and uniformly and will then choose randomly (but *not* uniformly) one of the actions represented by the block. Let $\mathit{frac}(x)$ denote the fractional part of x. The preparer prepares the blocks in the following way: if the λ-th block overlaps two shifted probability intervals (say that the block contains the value $\mathit{frac}(\hat{p}_i + r_0)$ for some i), it normalizes $\mathit{frac}(\hat{p}_i + r_0)$ by subtracting integer multiples of $1/l$ to get the value $\bar{p}_\lambda = \mathit{frac}(l(\hat{p}_i + r_0))/l$ in $[0, 1/l)$. It then separately encrypts $\bar{p}_\lambda$ and all four actions with its own public key, maintaining their pairing and order. Blocks that fall entirely within one shifted probability interval are easier to prepare, but must be made indistinguishable from the other kind of block: for these blocks, the preparer generates a random number $q_\lambda \in [0, 1/l)$ for the λ-th block, encrypts it and then encrypts the pair of actions twice with different randomness. When all blocks are prepared, the preparer sends the list of blocks to the chooser.

Decryption We use a semantically secure encryption scheme, so it is infeasible for the chooser to tell whether the two pairs of actions in a block are actually the same pair or not. It chooses one of the l blocks uniformly at random and gets the preparer to blindly decrypt its probability p (that is, the chooser blinds the encrypted p with a random value, asks the preparer to decrypt the result and then subtracts the blinding value). The chooser then chooses the leftmost pair of actions with probability lp and the rightmost otherwise. The chooser blinds

its own action (the first one in the chosen pair) with a random value and blinds the preparer's action with 0, then sends the result to the preparer. The preparer decrypts its own action (without knowing which of the blocks containing that action has been chosen) and the chooser's blinded action. It sends the blinded value back to the chooser (without knowing what the unblinded value is). The chooser subtracts the blinding value to get the result.

We need to prove that this protocol produces the right outputs with the correct probabilities, and that neither side gains any extra information.

Lemma 1. *Protocol CES securely computes the function of correlated element selection in the honest-but-curious model, with a negligible error in the probability of selecting any pair.*

Proof. **Correctness:** We will show that for all $i \in \{1, \ldots, n\}$, the action pair (a_i, b_i) is chosen by the protocol with probability p_i. This is true even given a fixed r_0. Suppose that $\hat{p}_i + r_0 < 1$. (The other cases are similar.) Let $int(x)$ denote the integer part of x. Then the number of blocks in which both pairs are (a_i, b_i) is $int(l(\hat{p}_i + r_0)) - int(l(\hat{p}_{i-1} + r_0)) - 1$. (Since $1/l < p_i$ this is always non-negative.) There is one block in which (a_i, b_i) is the leftmost pair only and another block in which it is the rightmost pair only. The probability of C choosing the leftmost pair in the former block is $frac(l(\hat{p}_i + r_0))$; the probability of it choosing the rightmost pair in the latter block is $1 - frac(l(\hat{p}_{i-1} + r_0))$. Therefore the chooser's probability of selecting (a_i, b_i) is

$$\begin{aligned} Pr\left(\text{Get } (a_i, b_i)\right) &= \left[int(l(\hat{p}_i + r_0)) - int(l(\hat{p}_{i-1} + r_0)) - 1\right]/l + frac(l(\hat{p}_i + r_0))/l \\ &\quad + \left[1 - frac(l(\hat{p}_{i-1} + r_0))\right]/l \\ &= \left[l(\hat{p}_i + r_0) - l(\hat{p}_{i-1} + r_0)\right]/l \\ &= p_i \end{aligned}$$

So if the protocol is followed correctly then it produces action pairs according to the correct distribution.

Secrecy: Informally, the preparer receives only three values: blinded encryptions of the chooser's action and the probability p_i that it is using, and a random encryption of its own action b_i. The first two provide no information because they are blinded by a random value that the preparer does not know. The last provides no information other than b_i, because any of the encryptions of b_i that were sent to the chooser are equally likely to have produced that ciphertext after blinding with zero. The chooser learns only four (unencrypted) values: the index of the block chosen, the probability of choosing the leftmost pair in that block, which pair (leftmost or rightmost) was chosen, and its output action a_i. The probability reveals no information, even combined with the index of the chosen block, because it is either a randomly chosen number independent of the input, or it is blinded by the factor r_0 which the chooser does not know. The choice of leftmost or rightmost pair also reveals no information because each pair of actions appears exactly the same number of times on the left and right sides.

More formally, the preparer's view can be simulated given its input and output b_i by sending it the encryptions of random values in place of the blinded action and probabilities, and sending it a random encryption of b_i in step 5. This produces a distribution identical to that of the protocol run.

The chooser's view can be simulated given its input and output a_i by sending it a "prepared list" of the right form which may actually include encryptions of anything. When it chooses a blinding factor β_1 for the probability, the simulator replies with a random value in $[0, 1/l)$. When the chooser chooses a blinding factor β_2 for its action, the simulator chooses an action pair (a_i, b_i) according to the correct distribution and sends $a_i + \beta_2$ to the chooser. This produces a distribution computationally indistinguishable from that of the protocol run. A distinguisher of the two distributions can defeat the semantically secure encryption scheme.

We now need to enhance the protocol to make it very unlikely that either player can cheat in a way that improves its utility without being detected by the other.

4 Protocol secure against one malicious or two selfish players

We use zero knowledge proofs adapted from [DHR00] and a cut-and-choose protocol to transform the protocol from section 3 into one secure against a malicious player. After each step of the honest-but-curious protocol, the sender of the last message "proves" to the receiver that it has constructed the message correctly. The cryptographers' view is that if one player is malicious it cannot gain an advantage over an honest opponent. More importantly for our purposes, it is an equilibrium for two selfish players to execute the protocol correctly, because any deviation is either unhelpful or likely to be detected and punished by the other.

4.1 Probability distribution is correct with one faulty player

The only protocol steps that can't be checked using zero-knowledge proofs or a cut-and-choose protocol are those that involve making some random choice according to a specified probability distribution. Since players can't prove to each other that they have done this correctly, we will use the standard trick of arguing that as long as one of the players makes its random choices correctly, the outcome will be correct (which in this case means that the final pair of actions is chosen according to the correct probability distribution). The game theory view is that it is an equilibrium for both players to toss their coins fairly, since unilateral deviation makes no difference to the outcome.

Lemma 2. *Suppose that the players are allowed to deviate from the protocol only by altering their random coin tosses. Then if at least one player tosses its coins correctly, the final pair of moves is selected according to the correct probability distribution.*

Proof. If the chooser chooses its randomness correctly, then it is easy to see that the resulting distribution is correct. For the other case, suppose that the preparer correctly chooses r_0 uniformly at random from $[0, 1)$ but that the chooser chooses its block according to a non-uniform distribution and chooses the leftmost pair of actions according to a (possibly randomized) function CH() that depends on the block chosen and the probability p decrypted by the preparer. CH() outputs either L (for left) or R (for right). Since the encryption algorithm used is semantically secure, the distribution on blocks must be independent of the choice of r_0. More importantly, CH() is independent of r_0 and the preparer's other coin tosses given p. We will show that for all p, for all choices of the block, the probability (given p) that the chooser chooses a particular pair of actions is exactly what it should be. Fix the value of p and the chooser's choice of block (call it λ). Let P_i be the probability that the i-th action pair (a_i, b_i) is chosen. We need to prove that $P_i = p_i$. Let r_P denote the random coin tosses of the preparer (apart from r_0) and let r_{CH} denote the random coin tosses of the CH() algorithm. Recall that all the computations are done to $\hat{k}$ bits of precision. Then

$$
\begin{aligned}
P_i &= Pr_{r_0, r_P, r_{\text{CH}}}[(\text{Left pair is } (a_i, b_i) \text{ and } \text{CH}(p, \lambda) = L) \text{ or } (\text{Right pair is } (a_i, b_i) \text{ and } \text{CH}(p, \lambda) = R)] \\
&= Pr_{r_0, r_P}(\text{Left pair is } (a_i, b_i))\, Pr_{r_{\text{CH}}}(\text{CH}(p, \lambda) = L) + Pr_{r_0, r_P}(\text{Right pair is } (a_i, b_i))\, Pr_{r_{\text{CH}}}(\text{CH}(p, \lambda) = R) \\
&= Pr_{r_0, r_P}(r_0 = (\lambda - 1)/l + p - \hat{p}_i \vee (r_0 \in [(\lambda/l - \hat{p}_i, (\lambda - 1)/l - \hat{p}_{i-1}) \wedge q_\lambda = p))\, Pr_{r_{\text{CH}}}(\text{CH}(p, \lambda) = L) \\
&\quad + Pr_{r_0, r_P}(r_0 = (\lambda - 1)/l + p - \hat{p}_{i-1} \vee (r_0 \in [\lambda/l - \hat{p}_i, (\lambda - 1)/l - \hat{p}_{i-1}) \wedge q_\lambda = p))\, Pr_{r_{\text{CH}}}(\text{CH}(p, \lambda) = R) \\
&= \frac{1/2^{\hat{k}} + (p_i - 1/l)l/2^{\hat{k}}}{l/2^{\hat{k}}} Pr_{r_{\text{CH}}}(\text{CH}(p, \lambda) = L) + \frac{1/2^{\hat{k}} + (p_i - 1/l)l/2^{\hat{k}}}{l/2^{\hat{k}}}(1 - Pr_{r_{\text{CH}}}(\text{CH}(p, \lambda) = L)) \\
&= p_i
\end{aligned}
$$

The first equation is given by the independence of CH() from r_0 and the other coin tosses of the preparer, given p. The second is just the definition of what it means to have (a_i, b_i) in the left or right part of the λ-th block. The third gives the probabilities for r_0 being in the required ranges, given p—the denominator is the *a priori* probability of getting a particular value for p.

This shows that, as long as one party tosses its coins correctly, the output probability distribution is correct.

4.2 Preventing deviation from the protocol

We use the same techniques as [DHR00] to prevent players from deviating from the protocol in ways other than making their random choices wrongly. After each round of the honest-but-curious protocol (except the first), the sender proves to the recipient in Zero Knowledge that it produced the message correctly. We can use the same proofs as [DHR00] for proving truthful decryption and proving that a certain ciphertext is truly a blinding of one from the original list. The only place where we must do this in an inefficient manner (differently to the method used in [DHR00]) is in the first message that the preparer sends to the

chooser. Here we use a "cut-and-choose" protocol: the preparer prepares several candidate lists and the chooser selects one of them to be used in the protocol. The preparer must decrypt the rest, thereby proving to the chooser that they were prepared correctly. Of course, if the prover makes one incorrect list there is some chance that it will succeed in fooling the chooser. There is a known upper bound on the profit that the preparer can make by cheating (its greatest payoff in the game minus its equilibrium payoff), so the number of candidate lists must be large enough that its expected gain from trying to cheat is negative. This means that the length of the first message is linear in the payoffs of the game. The protocol in [DHR00] is actually logarithmic in the payoffs, because their security parameter needs to be at least long enough that the probability of the cryptography failing (either in the zero knowledge proofs or in encryptions) is a constant fraction of the game's payoffs. Hence their methods for preventing protocol deviation are more efficient than ours in terms of the game's payoffs.

We can now prove Theorem 1, that it is a computational Nash equilibrium for everyone to execute the protocol correctly and minimax the other if it cheats. We have shown that a player that cheats (except by tossing its coins wrongly) will be detected and punished with high enough probability that it has no incentive to do so. We have also shown that a player that cheats by tossing its coins wrongly has no effect on the outcome if the other is honest, so there is no incentive for that kind of cheating either. Hence the "honest" strategy is a computational Nash equilibrium. Two competing, selfish and rational firms could use this to attain any correlated equilibrium payoff without using a trusted mediator.

5 Conclusion

There is a lot of interesting overlap between cryptography and game theory. In this paper we have given an efficient way of implementing a game theoretic solution concept that is impossible to implement without either cryptography or a trusted third party. This could be used by participants in all sorts of markets to coordinate their actions for their mutual benefit. We expect cryptography to be a useful tool for solving many other practical game theoretic problems that relate to the careful distribution of information.

6 Acknowledgements

Thanks to Yevgeniy Dodis for helpful feedback on an earlier version of this work, to Adam Cagliarini for thinking up examples of correlated equilibria with non-uniform distributions, to Bob McGrew for some pointers to literature, to John Mitchell for helpful comments on a draft of this paper, and to Andrew Conway for suggesting many interesting practical applications of this protocol, and for encouraging me to write this paper.

References

[AET01] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2001.

[Bar92] I. Barany. Fair distribution protocols or how the players replace fortune. *Mathematics of Operations Research*, 17(2):327–340, 1992.

[DHR00] Y. Dodis, S. Halevi, and T. Rabin. A cryptographic solution to a game theoretic problem. In *Proceedings of CRYPTO*, 2000.

[FPS01] Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001. Special issue on Internet Algorithms.

[FPSS02] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker. A bgp-based mechanism for lowest-cost routing. In *Proceedings of the 21st Symposium on Principles of Distributed Computing*, pages 173–182, New York, 2002. ACM Press.

[FT91] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, Cambridge, Massachusetts, 1991.

[Gol03] O. Goldreich. Draft of a chapter on general protocols: Extracts from a working draft for volume 2 of foundations of cryptography. Available at `http://www.wisdom.weizmann.ac.il/~oded/`, 2003.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Special issue of Journal of Computer and Systems Sciences*, 28(2):270–299, 1984.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conf. on Electronic Commerce*, 1999.

[NR01] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[OR94] M. Osborne and A. Rubinstein. *A course in Game Theory*. MIT Press, 1994.

[RT02] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM* 49(2):236–259, March 2002.

[UV02] A. Urbano and J. Vila. Computational complexity and communication: coordination in two-player games. *Econometrica*, 70(5):1893–1927, September 2002.