A Cryptographic Solution to a Game Theoretic Problem

Yevgeniy Dodis* Shai Halevi† Tal Rabin†

Abstract

Although Game Theory and Cryptography seem to have some similar scenarios in common, it is very rare to find instances where tools from one area are applied in the other. In this work we use cryptography to solve a game theoretic problem. The problem that we discuss arises naturally in the game theory area of two-party strategic games. In these games there are two players. Each player decides on a "move" (according to some strategy), and then the players execute the game, i.e. the two players make their moves simultaneously. Once these moves are played each player receives a payoff, which depends on both moves. Each player only cares to maximize its payoff.

In the game theory literature it was shown that higher payoffs can be achieved by the players if they use correlated strategies. This is enabled through the introduction of a trusted third party (a "mediator"), who assists the players in choosing their move. Though now the property of the game being a *two player* game is lost. It is natural to ask whether a game can exist which would be a two player game yet maintain the high payoffs which the mediator aided strategy offered.

We answer this question affirmatively. We extend the game by adding an initial step in which the two players communicate and then they proceed to execute the game as usual. For this extended game we can prove (informally speaking) the following: any correlated strategy for 2-player games can be achieved, provided that the players are computationally bounded and can communicate before playing the game.

We obtain an efficient solution to the above game-theoretic problem, by providing a cryptographic protocol to the following *Correlated Element Selection* problem. Both Alice and Bob know a list of pairs $(a_1, b_1) \dots (a_n, b_n)$ (possibly with repetitions), and they want to pick a *random* index i such that Alice learns only a_i and Bob learns only b_i. We believe that this problem has other applications, beyond our application to game theory. Our solution is quite efficient: it has constant number of rounds, negligible error probability, and uses only very simple zero-knowledge proofs.

The protocol that we describe in this work uses as a basic building block "blindable encryption schemes" (such as ElGamal or Goldwasser-Micali). We note that such schemes seem to be a very useful general primitive for constructing efficient protocols. As an example, we show a simple 1-out-of-n oblivious transfer protocol based on any such encryption scheme.

Key words. Game theory, Nash equilibria, Correlated equilibria, Element selection, Correlated coins, Coin Flipping, Blindable encryption, Oblivious transfer.

*Lab. of Computer Science, Massachusetts Institute of Technology, 545 Tech Square, Cambridge, MA 02139, USA. Email: `yevgen@theory.lcs.mit.edu`.

†IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598, USA. Email: {`shaih,talr`}`@watson.ibm.com`.

1 Introduction

The research areas of Game Theory and Cryptography are both extensively studied fields with many problems and solutions. Yet, the cross-over between them is surprisingly small: very rarely (if at all) are tools from one area borrowed to address problems in the other. In this paper we exhibit the benefits which arise in such a combined setting. Namely, we show how cryptographic tools can be used to address a natural problem in the Game Theory world. We hope that this work will encourage greater synergy between these classical fields.

1.1 Two Player Strategic Games

The game-theoretic problem that we consider in this work belongs to the general area of *two player strategic games*, which is an important field in Game Theory (see [18, 30]). In the most basic notion of a two player game, there are two players, each with a set of possible moves. The game itself consists of each player choosing a move from its set, and then both players execute their moves simultaneously. The rules of the game specify a *payoff* function for each player, which is computed on the two moves. Thus, the payoff of each player depends both on its move and the move of the other player. A *strategy* for a player is a (possibly randomized) method for choosing its move. The fundamental assumption of game theory is that each player is *selfish and rational*, i.e. its sole objective is to maximize its (expected) payoff.

A pair of players' strategies achieves an *equilibrium* when these strategies are *self-enforcing*, i.e. each player's strategy is an *optimal response* to the other player's strategy. In other words, once a player has chosen a move and believes that the other player will follow its strategy, its (expected) payoff will not increase by changing this move. This notion was introduced in the classical work of Nash [29].

In a *Nash equilibrium*, each player chooses its move *independently* of the other player. (Hence, the induced distribution over the pairs of moves is a product distribution.) Yet, Aumann [2] showed that in many games, the players can achieve much higher expected payoffs, while preserving the "self-enforcement" property, if their strategies are *correlated* (so the induced distribution over the pairs of moves is no longer a product distribution). To actually implement such a *correlated equilibrium*, the model of the game is modified and a "trusted third party" (called a *mediator*) is introduced. This mediator chooses the pair of moves according to the right distributions and privately tells each player what its designated move is. Since the strategies are correlated, the move of one player typically carries some information on the move of the other player. In a Correlated equilibrium, no player has an incentive to deviate from its designated move, even knowing this extra information about the other player's move.

1.2 Removing the Mediator

As the game was intended for two players, it is natural to ask if correlated equilibria can be implemented without the mediator. In the language of cryptography, we ask if we can design a two party game to eliminate the trusted third party from the original game. It is well known that in the standard cryptographic models, the answer is positive, provided that the two players can interact, that they are computationally bounded, and assuming some standard hardness assumptions ([21, 32]). We show that this positive answer carries over also to the Game Theory model. Specifically, we consider an *extended game*, in which the players first exchange messages, and then they choose their moves and execute them simultaneously. (The payoffs are still computed as a function of the moves, according to the same payoff function as in the original game.) Also, we define a *computational Nash equilibrium* as one where the strategies of both players are restricted to probabilistic polynomial time. Then, we prove the following:

Theorem 1 *If secure two-party protocols exist for non-trivial functions, then for any Correlated equilibrium s of the original game G, there exists an extended game G' with a computational Nash equilibrium σ, such that the payoffs for both players are the same in σ and s.*

In other words, any Correlated equilibrium payoffs of G can be achieved using a computational Nash equilibrium of G'. Thus, the mediator can be avoided if the players are computationally bounded and can communicate prior to the game.

We stress that although this theorem is quite natural from a cryptography point of view, the models of Game Theory and Cryptography are different, thus proving it in the Game Theory framework requires some care. In particular, two-party cryptographic protocols always assume that at least one player is honest, while the other player could be arbitrarily malicious. In the game-theoretic setting, on the other hand, *both players are selfish and rational*: they (certainly) deviate from the protocol if they benefit from it, and (can be assumed to) follow their protocol otherwise. Also, it is important to realize that in this setting we cannot use cryptography to "enforce" honest behavior. The only thing that the players are able to do is to choose their moves and execute them at the end of the game. Hence, even the most elaborate protocol would be useless if a "cheating player" can simply ignore the fact that it was "caught cheating" during the protocol, and nonetheless choose a move that maximizes its profit. We discuss these issues further in Section 3.

1.3 Doing it Efficiently

Although the assumption of Theorem 1 can be proven using tools of generic two-party computations [21, 32], it would be nice to obtain extended games (i.e. protocols) which are more efficient than the generic ones. In Section 4 we observe that for many cases, the underlying cryptographic problem reduces to a problem which we call *Correlated Element Selection*. We believe that this natural problem has other cryptographic application and is of independent interest. In this problem, two players, A and B, know a list of pairs $(a_1, b_1), \ldots, (a_n, b_n)$ (maybe with repetitions), and they need to jointly choose a random index i, so that player A only learns the value a_i and player B only learns the value b_i.[1]

We therefore dedicate a part of the paper to presenting an efficient cryptographic solution to the Correlated Element Selection problem. Our final protocol is very intuitive, has constant number of rounds, negligible error probability, and uses only very simple zero-knowledge proofs. We note that some of our techniques (in particular, the zero-knowledge proofs) are similar to those used for mixing networks (see [1, 24] and the references therein), even though our usage and motivation are quite different.

Our protocol for Correlated Element Selection uses as a tool *blindable encryption* (which can be viewed as a counterpart of blindable signatures [9]). Stated roughly, blindable encryption is the following: given an encryption c of an (unknown) message m, and an additional message m', a random encryption of $m + m'$ can be easily computed. This should be done without knowing m or the secret key. Examples of semantically secure blindable encryption schemes (under appropriate assumptions) include Goldwasser-Micali [22], ElGamal [16] and Benaloh [5]. (In fact, for our Correlated Element Selection protocol, it is sufficient to use a weaker notion of blindability, such as the one in [31].) Aside from our main application, we also observe that blindable encryption appears to be a very convenient tool for devising efficient two-party protocols and suggest that it might be used more often. To demonstrate that, we show in Appendix C a very simple protocol to achieve 1*-out-of-*n *Oblivious Transfer* ($\binom{n}{1}$-OT) protocol from any secure blindable encryption scheme.[2]

[1] A special case of Correlated Element Selection when $a_i = b_i$ is just the standard *coin-flipping* problem [7]. However, this is a degenerate case of the problem, since it requires no secrecy. In particular, none of the previous coin-flipping protocols seem to extend to solve our problem.

[2] It is known that oblivious transfer is complete for two-party secure computation [14, 26], hence blindable encryption schemes are sufficient for secure two-party computation.

1.4 Related Work

Game Theory. Realizing the advantages of removing the mediator, various papers in the Game Theory community have been published to try and achieve this goal. Barany [3] substitutes the trusted mediator with four (potentially untrusted) players. These players (two of which were the actual players who needed to play the game) distributively (and privately) computed the moves for the two active players. This protocol works in an information-theoretic setting (which explains the need for four players; see [6]). Of course, if one is willing to use a group of players to simulate the mediator, then the general multiparty computation tools (e.g. [6, 11]) can also be used, even though the solution of [3] is simpler and more efficient. The work of Lehrer and Sorin [25] describes protocols that "reduce" the role of the mediator (the mediator in this protocol computes some function on values which the players chose).

Cryptography. We already mentioned the relation of our work to generic two-party secure computations [21, 32], and to mix-networks [1, 24]. Additionally, encryption schemes with various "blinding properties" were used for many different purposes, including among others for secure storage [19], and secure circuit evaluations [31].

2 Background in Game Theory

Two-player Games. Although our results apply to a much larger class of two-player games[3], we demonstrate them on the simplest possible class of finite *strategic games* (with perfect information). Such a game G has two players 1 and 2, each of whom has a finite set A_i of possible *actions* and a *payoff function* $u_i : A_1 \times A_2 \mapsto R$ ($i = 1, 2$), known to both players. The players move simultaneously, each choosing an action $a_i \in A_i$. The *payoff* of player i is $u_i(a_1, a_2)$. The (probabilistic) algorithm that tells player i which action to take is called its *strategy*, and a pair of strategies is called a *strategy profile*. In our case, a strategy s_i of player i is simply a probability distribution over its actions A_i, and a strategy profile $s = (s_1, s_2)$ is a probability distribution over $A_1 \times A_2$. Game Theory assumes that each player is *selfish and rational*, i.e. only cares about maximizing its (expected) payoff. As a result, we are interested in strategy profiles that are *self-enforcing*. In other words, even knowing the strategy of the other player, each player still has no incentive to deviate from its own strategy. Such a strategy profile is called an *equilibrium*.

Nash equilibrium. This is the best known notion of an equilibrium [29]. It corresponds to a strategy profile in which players' strategies are *independent*. More precisely, the induced distribution over the pairs of actions, must be a product distribution, $s(A_1 \times A_2) = s_1(A_1) \times s_2(A_2)$. Deterministic (or *pure*) strategies are a special case of such strategies, where s_i assigns probability 1 to some action. For strategies s_1 and s_2, we denote by $u_i(s_1, s_2)$ the *expected* payoff for player i when players independently follow s_1 and s_2.

Definition 1 *A* Nash equilibrium *of a game G is an independent strategy profile (s_1^*, s_2^*), such that for any $a_1 \in A_1$, $a_2 \in A_2$, we have $u_1(s_1^*, s_2^*) \geq u_1(a_1, s_2^*)$ and $u_2(s_1^*, s_2^*) \geq u_2(s_1^*, a_2)$.*

In other words, given that player 2 follows s_2^*, s_1^* is an optimal response of player 1 and vice versa.

Correlated equilibrium. While Nash equilibrium is quite a natural and appealing notion (since players can follow their strategies independently of each other), one can wonder if it is possible to achieve higher expected payoffs if one allows *correlated* strategies.

[3]For example, *extensive games with perfect information and simultaneous moves*, see [30].

In a correlated strategy profile [2], the induced distribution over $A_1 \times A_2$ can be an arbitrary distribution, not necessarily a product distribution. This can be implemented by having a trusted party (called *mediator*) sample a pair of actions (a_1, a_2) according to some *joint* probability distribution $s(A_1 \times A_2)$, and "recommend" the action a_i to player i. We stress that knowing a_i, player i now knows a *conditional distribution* over the actions of the other player (which can be different for different a_i's), but knows *nothing more*. We denote these distributions by $s_2(\cdot \mid a_1)$ and $s_1(\cdot \mid a_2)$.

For any $a_1' \in A_1, a_2' \in A_2$, let $u_1(a_1', s_2 \mid a_1)$ be the expected value of $u_1(a_1', a_2)$ when a_2 is distributed according to $s_2(\cdot \mid a_1)$ (similarly for $u_2(s_1, a_2' \mid a_2)$). In other words, $u_1(a_1', s_2 \mid a_1)$ measures the expected payoff of player 1 if his recommended action was a_1 (thus, a_2 is distributed according to $s_2(\cdot \mid a_1)$), but it decided to play a_1' instead. As before, we let $u_i(s)$ be the expected value of $u_i(a_1, a_2)$ when (a_1, a_2) are drawn according to s.

Similarly to the notion of Nash equilibrium, we now define a *Correlated equilibrium*, which ensures that players have no incentive to deviate from the "recommendation" they got from the mediator.

Definition 2 *A* Correlated equilibrium *is a strategy profile $s^* = s^*(A_1 \times A_2) = (s_1^*, s_2^*)$, such that for any (a_1^*, a_2^*) in the support of s^*, and any $a_1 \in A_1$ and $a_2 \in A_2$, we have $u_1(a_1^*, s_2^* \mid a_1^*) \geq u_1(a_1, s_2^* \mid a_1^*)$ and $u_2(s_1^*, a_2^* \mid a_2^*) \geq u_2(s_1^*, a_2 \mid a_2^*)$.*

Given Nash (resp. Correlated) equilibrium (s_1^*, s_2^*), we say that (s_1^*, s_2^*) achieves *Nash (resp. Correlated) equilibrium payoffs* $[u_1(s_1^*, s_2^*), u_2(s_1^*, s_2^*)]$.

It is known that Correlated equilibria can give equilibrium payoffs *outside* (and better!) than anything in the convex hull of Nash equilibria payoffs, as demonstrated in the following simple example first observed by Aumann [2], who also defined the notion of Correlated equilibrium.

	C	D
C	4,4	1,5
D	5,1	0,0

"Chicken"

	C	D
C	1/4	1/4
D	1/4	1/4

Mixed Nash

	C	D
C	1/3	1/3
D	1/3	0

Correlated

Game of "Chicken". We consider a simple 2×2 game, the so called game of "Chicken" shown in the table to the right (much more dramatic examples can be shown in larger games). Here each player can either "dare" (D) or "chicken out" (C). The combination (D, D) has a devastating effect on both players (payoffs $[0, 0]$), (C, C) is quite good (payoffs $[4, 4]$), while each player would ideally prefer to dare while the other chickens-out (giving him 5 and the opponent 1). While the "wisest" pair of actions is (C, C), this is not a Nash equilibrium, since both players are willing to deviate to D (believing that the other player will stay at C). The game is easily seen to have three Nash equilibria: $s^1 = (D, C)$, $s^2 = (C, D)$ and $s^3 = (\frac{1}{2} \cdot D + \frac{1}{2} \cdot C, \frac{1}{2} \cdot D + \frac{1}{2} \cdot C)$. The respective Nash equilibrium payoffs are $[5, 1]$, $[1, 5]$ and $[\frac{5}{2}, \frac{5}{2}]$. We see that the first two pure strategy Nash equilibria are "unfair", while the last mixed equilibrium has small payoffs, since the mutually undesirable outcome (D, D) happens with non-zero probability in the product distribution. The best "fair" equilibrium in the convex hull of the Nash equilibria is the combination $\frac{1}{2}s^1 + \frac{1}{2}s^2 = (\frac{1}{2}(C, D) + \frac{1}{2}(D, C))$, yielding payoffs $[3, 3]$. On the other hand, the profile $s^* = (\frac{1}{3}(C, D) + \frac{1}{3}(D, C) + \frac{1}{3}(C, C))$ is a correlated equilibrium, yielding payoffs $[3\frac{1}{3}, 3\frac{1}{3}]$, which is better than any convex combination of Nash equilibria.

To briefly see it, consider the "row player" 1 (same works for player 2). If it is recommended to play C, its expected payoff is $\frac{1}{2} \cdot 4 + \frac{1}{2} \cdot 1 = \frac{5}{2}$ since, conditioned on $a_1 = C$, player 2 is recommended to play C and D with probability $\frac{1}{2}$ each. If player 1 switched to D, its expected payoff would still be $\frac{1}{2} \cdot 5 + \frac{1}{2} \cdot 0 = \frac{5}{2}$, making player 1 reluctant to switch. Similarly, if player 1 is recommended D, it knows that player 2 plays C (as (D, D) is never played in s^*), so its payoff is 5. Since this is the maximum payoff of the game, player 1 would not benefit by switching to C in this case. Thus, we indeed have a Correlated equilibrium, where each player's payoff is $\frac{1}{3}(1 + 5 + 4) = 3\frac{1}{3}$, as claimed.

3 Removing the Mediator

In this section we show how to remove the mediator using cryptographic means. We assume the existence of generic secure two-party protocols and show how to achieve our goal by using such protocols in the *game-theoretic* (rather than its designated cryptographic) setting. In other words, the players remain selfish and rational, even when running the cryptographic protocol. In Section 4 we give a specific efficient implementation for the types of cryptographic protocols that we need.

Extended Games. To remove the mediator, we assume that the players are (1) computationally bounded and (2) can communicate prior to playing the original game, which we believe are quite natural and minimalistic assumptions. To formally define the computational power of the players, we introduce an external security parameter into the game, and require that the strategies of both players can be computed in probabilistic polynomial time in the security parameter.[4]

To incorporate communication into the game, we consider an *extended game*, which is composed of three parts: first the players are given the security parameter and they freely exchange messages (i.e., execute any two-party protocol), then each player locally selects its move, and finally both players execute their move simultaneously.[5] The final payoffs u'_i of the extended game are just the corresponding payoffs of the original game applied to players' simultaneous moves at the last step.

The notions of a strategy and a strategy profile are straightforwardly generalized from those of the basic game. Similarly to the notion of a Nash equilibrium, we define the notion of a *computational Nash equilibrium* of the extended game, where the strategies of both players are restricted to probabilistic polynomial time. Also, since we are talking about a computational model, the definition must account for the fact that the players may break the underlying cryptographic scheme with negligible probability (e.g., by guessing the secret key), thus gaining some advantage in the game.

Definition 3 *A* computational Nash equilibrium *of an extended game G is an independent strategy profile (s^*_1, s^*_2), such that*
(a) both s^*_1, s^*_2 are PPT computable; and
(b) for any other PPT computable strategies s'_1, s'_2 there exists a negligible function μ such that on security parameter k, we have $u_1(s'_1, s^*_2) \leq u_1(s^*_1, s^*_2) + \mu(k)$ and $u_2(s^*_1, s'_2) \leq u_2(s^*_1, s^*_2) + \mu(k)$.

The idea of getting rid of the mediator is now very simple. Consider a Correlated equilibrium $s(A_1 \times A_2)$ of the original game G. Recall that the job of the mediator is to sample a pair of actions (a_1, a_2) according to the distribution s, and to give a_i to player i. We can view the mediator as a trusted party who securely computes a probabilistic (polynomial-time) function s. Thus, to remove it we can have the two players execute a cryptographic protocol P that securely computes the function s. The strategy of each player would be to follow the protocol P, and then play the action a that it got from P.

Yet, several issues have to be addressed in order to make this idea work. First, the above description does not completely specify the strategies of the players. A full specification of a strategy must also indicate what a player should do if the other player *deviates* from its strategy (in our case, does not follow the protocol P). While cryptography does not address this question, it is crucial to resolve it in our setting, since "the game must go on": No matter what happens inside P, both players eventually have to take simultaneous actions, and receive the corresponding payoffs (which they wish to maximize). Hence we must explain how to implement a "punishment for deviation" within the game-theoretic framework.

[4]Note that the parameters of the original game (like the payoff functions, the correlated equilibrium distribution, etc.) are all independent of the security parameter, and thus can always be computed "in constant time".

[5]The notion of an extended game is a special case of the well-studied notion of an extensive games see [30].

Punishment for Deviations. We employ the standard game-theoretic solution, which is to punish the cheating player to his *minimax level*. This is the smallest payoff that one player can "force" the other player to have. Namely, the minimax level of player 2 is $\underline{v_2} = \min_{s_1} \max_{s_2} u_2(s_1, s_2)$. Similarly, minimax level of player 1 is $\underline{v_1} = \min_{s_2} \max_{s_1} u_1(s_1, s_2)$. To complete the description of our proposed equilibrium, we let each player punish the other player to its minimax level, if the other player deviates from P (and is "caught"). Namely, if player 2 cheats, player 1 will play in the last stage of the game the strategy $\underline{s_1}$ achieving the minimax payoff $\underline{v_2}$ for player 2 and vice versa. First, we observe the following simple fact:

Lemma 1 *Let $[v_1, v_2]$ be the payoffs achieved by Correlated equilibrium s^*. Then, $v_i \geq \underline{v_i}$.*

Proof: Consider player 1. Let s_2^* be the marginal strategy of player 2 in the Correlated equilibrium s^*, and let s_1' be the best (independent) response of player 1 to s_2^*. (The strategy s_1' can be thought of as what player 1 should do if it knows that player 2 plays according to s_2^*, but it did not get any "recommendation" from the mediator.)

Since s^* is a Correlated equilibrium, it follows that $v_1 \geq u_1(s_1', s_2^*)$, since a particular deviation of player 1 from the correlated equilibrium is to "ignore" its recommendation and always play s_1', and we know that no such deviation can increase the payoff of player 1. Also, recall that s_1' is the best (independent) strategy in response to s_2^*, so we have $u_1(s_1', s_2^*) = \max_{s_1} u_1(s_1, s_2^*)$. Hence we get

$$v_1 \;\geq\; u_1(s_1', s_2^*) \;=\; \max_{s_1} u_1(s_1, s_2^*) \;\geq\; \min_{s_2} \max_{s_1} u_1(s_1, s_2) \;=\; \underline{v_1}.$$

The same holds for player 2. ■

We are now ready to prove Theorem 1, which we recall from the introduction:

Theorem 1 *If secure two-party protocols exist for non-trivial functions, then for any Correlated equilibrium s of the original game G, there exists an extended game G' with a computational Nash equilibrium σ, such that the payoffs for both players are the same in σ and s.*

Proof: (sketch) The extended game G' is the game G, preceeded with a cryptographic protocol P for securely computing the function s. (Such protocol exists by our assumption.) The computational Nash equilibrium σ consists of both players following their part in the protocol P, and executing the moves that they got from this protocol.

Clearly, this strategy achieves the same payoffs $[v_1, v_2]$ as the original correlated equilibrium. We only need to show that it is a computational Nash equilibrium. Indeed, if player i believes that the other player follows its designated strategy (i.e., behaves honestly in the protocol P), then the correctness of P implies that the probability of the this player "cheating without getting caught" is negligibly small. If we denote this probability of cheating by $\mu(k)$, and denote the highest payoff that player i can get in the original game by $\overline{v_i}$, then the expected payoff of a strategy that includes cheating in P is at most

$$\mu(k)\overline{v_i} + (1-\mu(k))\underline{v_i} \;=\; v_i + \mu(k)(\overline{v_i} - v_i) - (1-\mu(k))(v_i - \underline{v_i}) \;\leq\; v_i + \mu(k)(\overline{v_i} - v_i)$$

where the inequality follows from Lemma 1. Since $\overline{v_i} - v_i$ is just some constant which does not depend on k, this is at most negligibly larger than v_i. (We notice that a particular type of cheating (in the cryptographic setting) is early stopping. Since the extended game must always result in players getting payoffs, stopping is not an issue in game theory, since it will be punished by the minimax level as well.)

Finally, if no player has cheated in P, the privacy of P implies that we achieved exactly the same effects as with the mediator: each player only learns its move, does not learn anything about the other player's move except for what is implied by its move (and possibly except for a negligible additional advantage). Since s is a Correlated equilibrium, both players will indeed take the action they outputted in P. ■

A few remarks are in order: first, since the extended game produces some distribution over the moves of the players, then any Nash equilibrium payoffs achievable in the extended game are also achievable as Correlated equilibrium payoffs of the original game. Thus, Theorem 1 is the best possible result we could hope for.[6]

Also, one question that may come to mind is why would a player want to carry out a "minimax punishment" when it catches the other player cheating (since this "punishment" may also hurt the "punishing player"). However, the notion of Nash equilibrium only requires player's actions to be optimal *provided the other player follows its strategy*. Thus, it is acceptable to carry out the punishment even if this results in a loss for *both* players. ("the cheating player should have been rational and should not have cheated in the first place"). We note that this oddity (known as an "empty threat" in the game-theoretic literature) is not just a feature of our solution, but rather an inherent weakness of the notion of Nash equilibrium.

4 The Correlated Element Selection Problem

In most common games, the joint strategy of the players is described by a short list of pairs $\{(\mathsf{move1}, \mathsf{move2})\}$, where the strategy is to choose at random one pair from this list, and have Player 1 play $\mathsf{move1}$ and Player 2 play $\mathsf{move2}$. (For example, in the game of chicken the list consists of three pairs $\{(D, C), (C, D), (C, C)\}$.)[7]

Hence, to obtain an efficient solution for such games, we need an efficient cryptographic protocol for the following problem: Two players, A and B, know a list of pairs $(a_1, b_1), \ldots, (a_n, b_n)$ (maybe with repetitions), and they need to jointly choose a random index i, and have player A learn only the value a_i and player B learn only the value b_i. We call this problem the *Correlated Element Selection* problem. In this section we describe our efficient solution for this problem. We start by presenting some notations and tools that we use (in particular, "blindable encryption schemes"). We then show a simple protocol that solves this problem in the special case where the two players are "honest but curious", and explain how to modify this protocol to handle the general case where the players can be malicious.

4.1 Notations and Tools

We denote by $[n]$ the set $\{1, 2, \ldots n\}$. For a randomized algorithm A and an input x, we denote by $A(x)$ the output distribution of A on x, and by $A(x; r)$ we denote the output string when using the randomness r. If one of the inputs to A is considered a "key", then we write it as a subscript (e.g., $A_k(x)$). We use $pk, pk_1, pk_2, \ldots$ to denote public keys and $sk, sk_1, sk_2, \ldots$ to denote secret keys.

The main tool that we use in our protocol is *blindable encryption schemes*. Like all public key encryption schemes, blindable encryption schemes include algorithms for key-generation, encryption and decryption. In addition they also have a "blinding" and "combining" algorithms. We denote these algorithms by Gen, Enc, Dec, $Blind$, and $Combine$, respectively. Below we formally define the blinding and combing functions. In this definition we assume that the message space M forms a group (which we denote as an additive group with identity 0).

Definition 4 (Blindable encryption) *A public key encryption scheme $\mathcal{E}$ 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 $m + m'$. Namely, the distribution $Blind_{pk}(c, m')$ should be equal to the distribution*

[6] For example, there is no way to enforce the "desirable" outcome (C, C) in the Game of "Chicken", no matter which cryptographic protocols we design. Indeed, both players will want to deviate to D before making their final moves.

[7] Choosing from the list with distribution other than the uniform can be accommodated by having a list with repetitions, where a high-probability pair appears many times.

$Enc_{pk}(m+m')$.

$$Enc_{pk}(m+m') \equiv Blind_{pk}(c, m') \quad (1)$$

- *If* r_1, r_2 *are the random coins used by two successive "blindings", then for any two blinding factors* m_1, m_2,

$$Blind_{pk}(Blind_{pk}(c, m_1;\ r_1), m_2;\ r_2) = Blind_{pk}(c, m_1+m_2;\ Combine_{pk}(r_1, r_2)) \quad (2)$$

Thus, in a blindable encryption scheme anyone can "randomly translate" the encryption c of m into an encryption of $m+m'$, without knowledge of m or the secret key, and there is an efficient way of "combining" several blindings into one operation.

Both the ElGamal and the Goldwasser-Micali encryption schemes can be extended into blindable encryption schemes. We note that most of the components of our solution are independent of the specific underlying blindable encryption scheme, but there are some aspects that still have to be tailored to each scheme. (Specifically, proving that the key generation process was done correctly is handled differently for different schemes. See Section 4.4). The reader is referred to Section B for further discussion of the blindable encryption primitive.

4.2 A Protocol for the Honest-but-Curious Case

Let us recall the Correlated Element Selection problem. Two players share a public list of pairs $\{(a_i, b_i)\}_{i=1}^{n}$. For reasons that will soon become clear, we call the two players the "Preparer" (P) and the "Chooser" (C). The players wish to pick a random index i such that P only learns a_i and C only learns b_i.

Figure 1 describes the Correlated Element Selection protocol for the honest-but-curious players. We employ a semantically secure blindable encryption scheme and for simplicity, we assume that the keys for this scheme were chosen by a trusted party ahead of time and given to P, and that the public key was also given to C. We briefly discuss some key generation issues in Section 4.4.

The protocol is very simple: First the Preparer randomly permutes the list, encrypts it element-wise and sends the resulting list to the Chooser. (Since the encryption is semantically secure, the Chooser "cannot extract any useful information" about the permutation π.) The Chooser picks a random pair of ciphertexts (c_ℓ, d_ℓ) from the permuted list (so the final output pair will be the decryption of these ciphertexts). It then blinds c_ℓ with 0 (i.e. makes a random encryption of the same plaintext), blinds d_ℓ with a random blinding factor β, and sends the resulting pair of ciphertexts (e, f) back to the Preparer. Decryption of e gives the Preparer its element a (and nothing more, since e is a *random* encryption of a after the blinding with 0), while the decryption $\tilde{b}$ of f does not convey the value of the actual encrypted message since it was blinded with a random blinding factor. The Preparer sends $\tilde{b}$ to the Chooser, who recovers his element b by subtracting the blinding factor β.

It is easy to show that if both players follow the protocol then their output is indeed a random pair (a_i, b_i) from the known list. Moreover, at the end of the protocol the Preparer has no information about b other than what's implied by its own output a, and the Chooser gets "computationally no information" about a other than what's implied by b. Hence we have:

Theorem 2 *Protocol* CES-1 *securely computes the (randomized) function of the Correlated Element Selection problem in the honest-but-curious model.*

Proof omitted.

Protocol CES-1

Common inputs: List of pairs $\{(a_i, b_i)\}_{i=1}^n$, public key pk.
Preparer knows: secret key sk.

P : **1. Permute and Encrypt**.
Pick a random permutation π over $[n]$.
Let $(c_i, d_i) = (Enc_{pk}(a_{\pi(i)}),\ Enc_{pk}(b_{\pi(i)}))$, for all $i \in [n]$.
Send the list $\{(c_i, d_i)\}_{i=1}^n$ to C.

C : **2. Choose and Blind**.
Pick a random index $\ell \in [n]$, and a random blinding factor β.
Let $(e, f) = (Blind_{pk}(c_\ell, 0),\ Blind_{pk}(d_\ell, \beta))$.
Send (e, f) to P.

P : **3. Decrypt and Output**.
Set $a = Dec_{sk}(e)$, $\tilde{b} = Dec_{sk}(f)$. Output a.
Send $\tilde{b}$ to C.

C : **4. Unblind and Output**.
Set $b = \tilde{b} - \beta$. Output b.

Figure 1: Protocol for Correlated Element Selection in the honest-but-curious model.

4.3 Dealing with Dishonest Players

Generic transformation. Following the common practice in the design of secure protocols, one can modify the above protocol to deal with dishonest players by adding appropriate zero-knowledge proofs. That is, after each flow of the original protocol, the corresponding player proves in zero knowledge that it indeed followed its prescribed protocol: After Step 1, the Preparer proves that it knows the permutation π that was used to permute the list. After Step 2 the Chooser proves that it knows the index ℓ and the blinding factor that was used to produce the pair (e, f). Finally, after Step 3 the Preparer proves that the plaintext $\tilde{b}$ is indeed the decryption of the ciphertext f. Given these zero-knowledge proofs, one can appeal to general theorems about secure two-party protocols, and prove that the resulting protocol is secure in the general case of potentially malicious players.

We note that the zero-knowledge proofs that are involved in this protocol can be made very efficient, so even this "generic" protocol is quite efficient (these are essentially the same proofs that are used for mix-networks in [1], see Appendix A). However, a closer look reveals that one does not need all the power of the generic transformation, and the protocol can be optimized in several ways.

Proof of proper decryption. To withstand malicious players, the Preparer P must "prove" that the element $\tilde{b}$ that it send in Step 3 of CES-1 is a proper decryption of the ciphertext f. However, this can be done in a straightforward manner without requiring zero-knowledge proofs. Indeed, the Preparer can reveal additional information (such as the randomness used in the encryption of f), as long as this extra information does not compromise the semantic security of the ciphertext e. The problem is that P may not be able to compute the randomness of the blinded value f (for example, in ElGamal encryption this would require computation of discrete log). Hence, we need to device a different method to enable the proof.

The proof will go as follows: for each $i \in [n]$, the Preparer sends the element $b_{\pi(i)}$ and corresponding random string that was used to obtain ciphertexts d_i in the first step. The Chooser can then check that the element d_ℓ that it chose in Step 2 was encrypted correctly, and learn the corresponding plaintext.

Clearly, in this protocol the Chooser gets more information than just the decryption of f (specifically, it gets the decryption of all the d_i's). However, this does not effect the security of the protocol, as the Chooser now sees a decryption of a permutation of list that he knew at the onset of the protocol. This permutation of the all b_i's does not give any information about the output of the Preparer, other than what is implied by its output b. In particular, notice that if b appears more than once in the list, then the Chooser does not know which of these occurrences was encrypted by d_ℓ.

Next, we observe that after the above change there is no need for the Chooser to send f to the Preparer; it is sufficient if C sends only e in Step 2, since it can compute the decryption of d_ℓ by itself.

A weaker condition in the second proof-of-knowledge. Finally, we observe that since the security of the Chooser relies on an information-theoretic argument, the second proof-of-knowledge (in which the Chooser proves that it knows the index ℓ) does not have to be fully zero-knowledge. In fact, tracing through the proof of security, one can verify that it is sufficient for this proof to be *witness hiding* in the sense of Feige and Shamir [17]. The resulting protocol is described in Figure 2.

Remark. Notice that for the modified protocol we did not use the full power of blindable encryption, since we only used "blindings" by zero. Namely, all that was used in these protocols is that we can transform any ciphertext c into a *random* encryption of the same plaintext. (The zero-knowledge proofs also use only "blindings" by zero.) This is exactly the "random self-reducibility" property used by Sander et al. [31].

4.4 Key Generation Issues

In the protocols above we assumed that the encryption keys were chosen in an appropriate way ahead of time (say, by a trusted party). If we want to include the key generation as part of the protocols themselves (i.e., have the Preparer choose them and send the public keys to the Chooser in Step 1), we must ensure that the Preparer doesn't choose "bad keys".

A generic way to solve this problem is to have the Preparer prove in zero knowledge that it followed the key generation algorithm. We note, however, that for our application this is an overkill. Indeed, the security of the Chooser does not depend on the "hiding" properties of the encryption scheme. What the Chooser needs to verify is that this is a committing encryption (so that the Preparer cannot "open" the list of d_j's in more than one way), and that the output of the blinding operation is independent of which ciphertext of a given message was used as an input (i.e., that for all m and $c_1, c_2 \in Enc_{pk}(m)$, it holds that $Blind_{pk}(c_1, 0) \equiv Blind_{pk}(c_2, 0)$).

If we use ElGamal encryption to implement the blindable encryption, then checking these conditions is easy (assuming that the factorization of $p - 1$ is known, where p is the modulus used for the encryption). For the Goldwasser-Micali encryption, the first property requires proving that a given element y ($y = -1$ in the case of a Blum integer N) is a quadratic non-residue mod N (which can be done quite efficiently [23]), while the second property is automatically satisfied.

4.5 Putting It All Together

We now finally described all the components of our solution. Let us quickly re-cap everything and consider the efficiency of the protocol.

Protocol CES-2

Common inputs: List of pairs $\{(a_i, b_i)\}_{i=1}^n$, public key pk.
Preparer knows: secret key sk.

P : **1. Permute and Encrypt**.
Pick a random permutation π over $[n]$, and random strings $\{(r_i, s_i)\}_{i=1}^n$.
Let $(c_i, d_i) = (Enc_{pk}(a_{\pi(i)}; r_{\pi(i)}),\ Enc_{pk}(b_{\pi(i)}; s_{\pi(i)}))$, for all $i \in [n]$.
Send $\{(c_i, d_i)\}_{i=1}^n$ to C.

Sub-protocol Π_1: P proves in zero-knowledge that it knows the randomness $\{(r_i, s_i)\}_{i=1}^n$ and permutation π that were used to obtain the list $\{(c_i, d_i)\}_{i=1}^n$.

C : **2. Choose and Blind**.
Pick a random index $\ell \in [n]$.
Send to P the ciphertext $e = Blind_{pk}(c_\ell, 0)$.

Sub-protocol Π_2: C proves in a witness-hiding manner that it knows the randomness and index ℓ that were used to obtain e.

P : **3. Decrypt and Output**.
Set $a = Dec_{sk}(e)$. Output a.
Send to C the list of pairs $\{(b_{\pi(i)}, s_{\pi(i)})\}_{i=1}^n$ (in this order).

C : **4. Verify and Output**.
Denote by (b, s) the ℓ'th entry in this lists (i.e., $(b, s) = (b_{\pi(\ell)}, s_{\pi(\ell)})$).
If $d_\ell = Enc_{pk}(b; s)$ then output b.

Figure 2: Protocol for Correlated Element Selection.

0. Initially, the Preparer chooses the keys for the blindable encryption scheme, sends the public key to the Chooser and proves in zero-knowledge that the encryption is committing and has the blinding property. As we said above, this proof must be tailored to the particular encryption scheme that is used. Also, this step can be carried out only once, and the resulting keys can be used for many instances of the protocol. (Alternatively, we can use a trusted third party for this step.)

1. The Preparer encrypts the known list $\{(a_i, b_i)\}_{i=1}^n$ in some "canonical" manner, blinds with zero the list of ciphertexts, and permutes it with a random permutation π. It sends the resulting lists $\{(c_i, d_i)\}_{i=1}^n$ to the Chooser, and uses the ELC protocol to prove in zero-knowledge that it knows the permutation that was used.

2. The Chooser blinds with zeros the list of c_i's, and re-permutes it with a random permutation ρ. It sends the resulting list $\{e_i\}_{i=1}^n$ to the Prover, and again uses the ELC protocol to prove that it knows the permutation that was used. Here we can optimize the proof somewhat, since we later only use e_1, and also because the proof only needs to be witness hiding.

3. The Preparer decrypts the first ciphertext e_1, and outputs the corresponding plaintext a.

It also sends to the Chooser the list of the b_i's, permuted according to π, together with the randomness that was used to blind their "canonical encryption" to get the d_i's in Step 1.

4. The Chooser C sets $\ell = \rho^{-1}(1)$, and lets b, s denote the ℓ'th element and randomness, respectively, in the last list that it got from the Preparer. He checks that blinding with zero (and randomness s) of the "canonical encryption" of b indeed yields the ciphertext d_ℓ. If this is correct, C outputs b.

Although we can no longer use the general theorems about secure two-party protocols, the security proof is nonetheless quite standard. Specifically, we can prove:

Theorem 3 *Protocol* CES-2 *securely computes the (randomized) function of the Correlated Element Selection problem.*

Proof omitted.

Efficiency. We note that all the protocols that are involved are quite simple. In terms of number of communication flows, the key generation step (Step 0 above) takes at most five flows, Step 1 takes five flows, Step 2 takes three flows and Step 3 consists of just one flow. Moreover, these flows can be piggybacked on each other. Hence, we can implement the protocol with only five flows of communication, which is equal to the five steps which are required by a single proof. In terms of number of operations, the complexity of the protocol is dominated by the complexity of the proofs in Steps 1 and 2. The proof in Step 1 requires nk blinding operations (for a list of size n and security parameter k), and the proof of Step 2 can be optimized to about $nk/2$ blinding operations on the average. Hence, the whole protocol has about $\frac{3}{2}nk$ blinding operations.[8]

5 Epilogue: Cryptography and Game Theory

An interesting aspect of our work is the synergy achieved between cryptographic solutions and the game-theory world. Notice that by implementing our cryptographic solution in the game-theory setting, we gain on the game-theory front (by eliminating the need for a mediator), but we also gain on the cryptography front (for example, in that we eliminate the problem of early stopping). In principle, it may be possible to make stronger use of the game theory setting to achieve improved solutions. For example, maybe it is possible to prove that in the context of certain games, a player does not have an incentive to deviate from its protocol, and so in this context there is no point in asking this player to prove that it behaves honestly (so we can eliminate some zero-knowledge proofs that would otherwise be required).

More generally, it may be the case that working in a model in which "we know what the players are up to" can simplify the design of secure protocols. It is a very interesting open problem to find interesting examples that would demonstrate such phenomena.

References

[1] M. Abe. Universally Verifiable Mix-net with Verification Work Independent on the number of Mix-centers. In *Proceedings of EUROCRYPT '98*, pp. 437-447, 1998.

[8] We note that the protocol includes just a single decryption operation, in Step 3. In schemes where encryption is much more efficient than decryption – such as the Goldwasser-Micali encryption – this may have a significant impact on the performance of the protocol.

[2] R. Aumann. Subjectivity and Correlation in Randomized Strategies. In *Journal of Mathematical Economics*, 1, pp. 67-95, 1974

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

[4] M. Bellare, R. Impagliazzo, and M. Naor. Does parallel repetition lower the error in computationally sound protocols? In *38th Annual Symposium on Foundations of Computer Science*, pages 374–383. IEEE, 1997.

[5] J. Benaloh. Dense Probabilistic Encryption. In *Proc. of the Workshop on Selected Areas in Cryptography*, pp. 120-128, 1994.

[6] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

[7] M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. In *Advances in Cryptology – CRYPTO '81*. ECE Report 82-04, ECE Dept., UCSB, 1982.

[8] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *JCSS*, 37(2):156–189, 1988.

[9] D. Chaum. Blind signatures for untraceable payment. In *Advances in Cryptology – CRYPTO '82*, pages 199–203. Plenum Press, 1982.

[10] D. Chaum and H. Van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology — Crypto'89*, pages 212–217, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science No. 435.

[11] D. Chaum, C. Crépeau, and E. Damgård. Multiparty unconditionally secure protocols. In *Advances in Cryptology – CRYPTO '87*, volume 293 of *99 Lecture Notes in Computer Science*, pages 462–462. Springer-Verlag, 1988.

[12] D. Chaum and T. Pedersen. Wallet databases with observers. In E. Brickell, editor, *Advances in Cryptology — Crypto'92*, pages 89–105, Berlin, 1992. Springer-Verlag. Lecture Notes in Computer Science No. 740.

[13] R. Cramer, I. Damgard, and P. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. To appear in *2000 International Workshop on Practice and Theory in Public Key Cryptography*, January 2000, Melbourne, Australia.

[14] C. Crépeau and J. Kilian. Weakening security assumptions and oblivious transfer. In *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 2–7. Springer-Verlag, 1990.

[15] C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 409–418. ACM Press, 1998.

[16] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[17] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 416–426. ACM Press, 1990.

[18] D. Fudenberg, J. Tirole. Game Theory. MIT Press, 1992.

[19] J. Garay, R. Gennaro, C. Jutla, and T. Rabin. Secure distributed storage and retrieval. In *Proc. 11th International Workshop on Distributed Algorithms (WDAG '97)*, volume 1320 of *Lecture Notes in Computer Science*, pages 275–289. Springer-Verlag, 1997.

[20] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE, 1986.

[21] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[22] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[23] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[24] M. Jakobsson. A Practical Mix. In *Proceedings of EUROCRYPT '98*, pp. 448–461, 1998.

[25] E. Lehrer and S. Sorin. One-shot public mediated talk. Discussion Paper 1108, Northwestern University, 1994.

[26] J. Kilian. Founding Cryptography on Oblivious Transfer. In *Proc. of STOC*, pp. 20–31, 1988.

[27] P. MacKenzie. Efficient ZK Proofs of Knowledge. Unpublished manuscript, 1998.

[28] M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 573–590. Springer-Verlag, 1999.

[29] J.F. Nash. Non-Cooperative Games. *Annals of Mathematics,* 54 pages 286–295.

[30] M. Osborne, A. Rubinstein. A Course in Game Theory. The MIT Press, 1994.

[31] T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC1. In *40th Annual Symposium on Foundations of Computer Science*, pages 554–567. IEEE, 1999.

[32] A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, Nov. 1982.

A The Zero-Knowledge Proofs

In this section we provide efficient implementations for the sub-protocols Π_1 and Π_2. Recall that in Π_1, the Preparer needs to prove that the list of ciphertexts $\{(c_i, d_i)\}_{i=1}^n$ is a permuted encryption of the known list $\{(a_i, b_i)\}_{i=1}^n$, and in Π_2 the Chooser needs to prove that the ciphertext e was obtained from a blinding-with-zero of one of the ciphertexts c_ℓ.

Both protocols Π_1 and Π_2 are derived from a simple zero-knowledge proof for a problem which we call Encrypted List Correspondence. In this problem, which was studies in the context of mix-networks, Protocol ELC

Common inputs: $\{x_i\}_{i=1}^n$, $\{y_i\}_{i=1}^n$ (and pk).
P knows: π, $\{r_i\}_{i=1}^n$ s.t. $y_i = Blind_{pk}(x_{\pi(i)}, 0;\ r_i)$, for all $i \in [n]$.

P : Choose a random permutation ρ over $[n]$, and n random strings $\{s_i\}_{i=1}^n$.
Set $t_i = Combine(r_{\rho(i)},\ s_i)$
Set $z_i = Blind_{pk}(y_{\rho(i)}, 0;\ s_i)$ $\big(= Blind_{pk}(x_{(\pi\circ\rho)(i)}, 0;\ t_i)\big)$.
Send to C $\{z_i\}_{i=1}^n$.

C : Choose $\sigma \in_R \{0,1\}$ and send it to P

P : If $\sigma = 0$, reply with (ρ, $\{s_i\}_{i=1}^n$), else reply with ($\pi \circ \rho$, $\{t_i\}_{i=1}^n$).

C : If $\sigma = 0$, check $z_i = Blind_{pk}(y_{\rho(i)}, 0;\ s_i)$.
If $\sigma = 1$, check $z_i = Blind_{pk}(x_{(\pi\circ\rho)(i)}, 0;\ t_i)$.

Figure 3: A zero-knowledge proof-of-knowledge for Encrypted List Correspondence.

a prover P wants to prove in zero-knowledge that two lists of ciphertexts, $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, are permuted encryptions of the same list. More precisely, P wants to prove that one list was obtained by blinding-with-zero and permuting the other, and that he knows a permutation π and random coins $r_1, \ldots, r_n$ such that $y_i = Blind_{pk}(x_{\pi(i)}, 0; r_i)$, for all $i \in [n]$.[9] An efficient zero-knowledge proof for this problem was described by Abe [1]. (Although Abe's proof assumes ElGamal encryptions, it is easy to see that any blindable encryption will do.) For self-containment, we describe this proof in Figure 3. The protocol in Figure 3 achieves knowledge-error of 1/2. There are known transformations that can be used to derive a constant-round, negligible-error protocol from a three-round, constant-error one. (Specifically, we can get a 5-round, negligible-error zero-knowledge proof-of-knowledge for Encrypted List Correspondence.) We discuss this issue further in Appendix D.

A.1 The protocol Π_1

In sub-protocol Π_1, The Preparer needs to prove that the list of ciphertexts $\{(c_i, d_i)\}_{i=1}^n$ is a permuted encryption of the known list $\{(a_i, b_i)\}_{i=1}^n$. This can be done by having the prover encrypt the list $\{(a_i, b_i)\}_{i=1}^n$ in some "canonical" manner (say, using the all-zero random string) to generate a ciphertext list $\{(g_i, h_i)\}_{i=1}^n$, and then prove using the ELC protocol that the two lists $\{(c_i, d_i)\}_{i=1}^n$ and $\{(g_i, h_i)\}_{i=1}^n$ are encryptions of the same list. The only problem here is that the original list of ciphertexts $\{(c_i, d_i)\}_{i=1}^n$ was not obtained by blinding the list $\{(g_i, h_i)\}_{i=1}^n$, but rather by directly encrypting the plaintext list $\{(a_i, b_i)\}_{i=1}^n$. Hence, to use this protocol we slightly change the protocol CES-2 itself, and have the Preparer encrypt the plaintext list by first encrypting it in a "canonical" manner to get $\{(g_i, h_i)\}_{i=1}^n$, and then blind with zero and permute the latter list to get $\{(c_i, d_i)\}_{i=1}^n$. We stress that due to Equation (2) (from the definition of blindable encryption), this change in the protocol does not change the distribution of the ciphertext list $\{(c_i, d_i)\}_{i=1}^n$.

[9] We remark that a similar proof can be shown even when the two lists were not generated as blindings of each other, and even if the two lists were obtained using two different blindable encryption schemes.

A.2 The protocol Π_2

At first glance, the problem in Protocol Π_2 seems substantially different than the Encrypted List Correspondence problem. Furthermore, constructing a simple protocol that "the ciphertext e was obtained as blinding of some ciphertext c_ℓ" (without revealing ℓ) seems rather hard. Nonetheless, we can pose the problem of protocol Π_2 as an instance of the Encrypted List Correspondence. To this end, we again slightly modify the protocol CES-2, by having the Chooser blind with zero and re-permute the entire list (rather than just pick and blind one ciphertext), and then prove in zero-knowledge that it knows the corresponding permutation. Specifically, the Chooser selects a random permutation ρ over $[n]$, permutes the entire list of c_i's according to ρ, and then blinds with zero each of the ciphertexts. The entire new list (denoted $\{e_i\}_{i=1}^n$) is then sent to the Preparer. The preparer only uses a single, agreed upon, ciphertext from the list (say, e_1), in the rest of the protocol, but it uses the whole list to verify the proof. (Therefore, the effective index that is chosen by Chooser is $\ell = \rho^{-1}(1)$.) To prove that it chose the list of e_i's according to the prescribed protocol, the Chooser now proves that it knows the permutation ρ, which is exactly what is done in the ELC protocol.

Two optimizations. Since we only use the ciphertext e_1 in the CES-2 protocol (and "we don't really care" whether the other e_i's are obtained properly), we can somewhat simplify and optimize the proof in Π_2, as follows: The Chooser can send only e_1 (just as it is done in the protocol CES-2). Then, in the zero-knowledge proof, he prepares the full set of encryptions $z_1, ...z_n$ (as if he actually prepared and sent all the e_i's). Then, depending on the query bit, he either reveals the correspondence between the c_i's and all the z_i's, or the correspondence between e_1 and one of the z_i's.

Another optimization takes advantage of the fact that in the context of the CES-2 protocol, we only need this proof to be witness hiding, rather than fully zero-knowledge. It is therefore possible to just repeat the protocol from Figure 3 many times in parallel to get a negligible error (and we do not need to worry about adding extra flows of commitment).

B Implementation of Blindable Encryption

As we said above, possible implementations of blindable encryption include ElGamal encryption and the Goldwasser-Micali encryption scheme. Below we briefly describe these scheme.

B.1 ElGamal Encryption

The generation algorithm picks a random k-bit prime $p = 2q + 1$, where q is prime and a generator g for the subgroup Q of quadratic residues modulo p. Then it picks a random $x \in \mathbb{Z}_q$ and sets $h = g^x$. The secret key is x, the public key is h (and p).

To encrypt a message $m \in Q$, one picks a random $r \in \mathbb{Z}_q$ and sets $E(m) = \langle g^r, h^r \cdot m\rangle$. The decryption $D(s,t)$ outputs t/s^x. The encryption scheme is well known to be semantically secure under the decisional Diffie-Hellman assumption (DDH). To blind a ciphertext $\langle s,t\rangle$ with blinding factor m', we compute $Blind(\langle s,t\rangle, m') = \langle s \cdot g^{r'}, t \cdot h^{r'} \cdot m'\rangle$, where r' is chosen at random from $\mathbb{Z}_q$. We note that indeed if $s = g^r, t = h^r \cdot m$ (for some unknown r and m), then $Blind(\langle s,t\rangle, m') = \langle g^{r+r'}, h^{r+r'} \cdot (mm')\rangle$, which is a random encryption of mm' since $(r + r' \bmod q)$ is random when r' is. The $Combine(r_1, r_2)$ operation is just $(r_1 + r_2 \bmod q)$.

Checking that the public key (p, q, g) is "kosher" can be done by verifying that p, q are primes, $p = 2q + 1$, and g is an element of order q. Thus, no interaction is needed in the key generation phase. Proving that a ciphertext $\langle s,t\rangle$ is an encryption of a message m requires proving equality of the discrete-log of two known elements with respect to two known bases, which can be done using several known simple protocols [10, 12].

B.2 Goldwasser-Micali Encryption

This is the original semantically secure scheme introduced by Goldwasser and Micali [22]. The message is encrypted bit by bit. The generation algorithm picks a random $n = pq$, a product of two k-bit primes together with any quadratic non-residue y with Jacobi symbol 1 (in case n is a Blum integer, i.e. $p \equiv q \equiv 3 \bmod 4$, one can fix $y \equiv -1$). The public key is n and y, and the secret key is p and q.

To encrypt a bit $b \in \{0, 1\}$, one picks a random $r \in \mathbb{Z}_n^*$ and returns $(y^b \cdot r^2 \bmod n)$ (i.e. a random square for $b = 0$ and a random non-square for $b = 1$). To decrypt one simply determines if the ciphertext c is a square modulo n, i.e. modulo both p and q. The scheme is semantically secure under the quadratic residue assumption. The scheme is clearly blindable, since if $c = Enc(b; r)$ and $c' = Enc(b'; r')$, then $c \cdot c' = Enc(b + b'; r \cdot r')$ (where the addition $b + b'$ is done modulo 2, and all the other operations are done modulo n).

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

B.3 Verifying Decryption in Blindable Encryption Schemes

It is often useful when an encryption scheme has an efficient zero-knowledge proofs for claims of the form "c is an encryption of m". Although we do not know how to construct such efficient protocol based only on the blindable encryption property (other than using generic constructions), we can show a simple and efficient proof for the special case where the decryption algorithm also reveals the random coins that were used in the encryption (as in Goldwasser-Micali). The following simple 3-round protocol is such a proof with soundness error of $1/2$.

For this protocol, we assume without loss of generality that

$$Blind_{pk}(Enc_{pk}(m_1;\ r_1), m_2;\ r_2) = Enc_{pk}(m_1 + m_2;\ Combine(r_1, r_2)) \tag{3}$$

Indeed, let c_0 be any fixed encryption of 0 (the identity of M). We can then redefine the encryption process to be $Enc_{pk}(m; r) = Blind_{pk}(c_0, m; r)$. Equation (1) from the definition of blindable encryption shows that $Enc_{pk}(m; r)$ is indeed a random encryption of m, while Equation (2) now immediately implies Equation (3).

Protocol for Proving $c \in Enc_{pk}(m)$

Common inputs: m, c, pk.
Prover knows: sk.

P :	Compute r such that $c = Enc_{pk}(m, r)$. Pick random $m' \in M$ and r'. Let $c' = Blind_{pk}(c, m', r') = Blind_{pk}(Enc_{pk}(m, r), m', r')$ $\left(\overset{(3)}{=} Enc_{pk}(m + m', Combine_{pk}(r, r'))\right)$.
$P \longrightarrow V$:	Send m', c'.
$V \longrightarrow P$:	Choose $\sigma \in_R \{0, 1\}$; Send σ.
$P \longrightarrow V$:	If $\sigma = 0$, send r', else send $r'' = Combine_{pk}(r, r')$.
V :	If $\sigma = 0$, check $c' = Blind_{pk}(c, m', r')$, else check $c' = Enc_{pk}(m + m', r'')$.

C Blindable Encryption and Oblivious Transfer

We show below a very simple implementation of *1-out-of-n Oblivious Transfer* using blindable encryption. Recall that a 1-out-n Oblivious Transfer Protocol implements the following functionality. There are two players, P and C, who have some private inputs: P has a set of n strings $X = (x_1, \ldots, x_n)$, and C has an index $\ell \in [n]$. At the end of the protocol C should learn x_ℓ and nothing else, while P should learn nothing at all. There are other flavors of oblivious transfer, all known to be equivalent to the above.

We let P commit to his input X by encrypting each x_i, i.e. set $y_i = Enc_{pk}(x_i)$ (for all $i \in [n]$) and send $y_1, \ldots, y_n$ to C. Now we are in the situation that the Chooser wants the Preparer to decrypt y_ℓ without telling him ℓ. A simple solution that works in the honest-but-curious model, is as follows: C chooses a random blinding factor β, sets $z = Blind_{pk}(y_\ell, \beta)$, asks P to decrypt z, and subtracts β from the result to recover the correct x_ℓ. Since z is the encryption of a random element $x_j + \beta$, P indeed does not learn any information about ℓ. To adjust this protocol to work against malicious players, C needs to prove that he knows the index ℓ and blinding factor β, and P needs to prove that it decrypted z correctly. The proof of C is essentially the same problem as in the sub-protocol Π_2 in our CES-2 protocol, with the only difference being that now we also have the blinding factor β. Accordingly, the protocol for solving it is nearly identical to the ELC protocol, with the only difference being that the prover blinds the encrypted lists with random elements rather than with zeros (and shows the blinding factors when he is asked to "open" the blinding. Due to space limitations, we omit further details.

We note, though, that a small modification of the above protocol implements *random* 1-out-of-n oblivious transfer, where C should learn x_ℓ for a random ℓ. To implement that, P simply chooses a random permutation π in the first step and sets $y_i = Enc_{pk}(x_{\pi(i)})$.

D Reducing the Error in a Zero-knowledge Proof-of-knowledge

Below we describe a known transformation from any 3-round, constant-error zero-knowledge proof-of-knowledge into a 5-round, negligible error zero-knowledge proof-of-knowledge, that uses trapdoor commitment schemes. We were not able to trace the origin of this transformation, although related ideas and techniques can be found in [15, 27, 13].

Assume that you have some 3-round, constant-error zero-knowledge proof-of-knowledge protocol, and consider the 3-round protocol that you get by running the constant-error protocol many times in parallel. Denote the first prover message in the resulting protocol by α, the verifier message by β, and the last prover message by γ. Note that since the original protocol was 3-round, then parallel repetition reduces the error exponentially (see proof in [4]). However, this protocol is no longer zero-knowledge.

To get a zero-knowledge protocol, we use a trapdoor (or *Chameleon*) commitment schemes [8]. Roughly, this is a commitment scheme which is computationally binding and unconditionally secret, with the extra property that there exists a trapdoor information, knowledge of which enables one to open a commitment in any way it wants.

In the zero-knowledge protocol, the prover sends to the verifier in the first round the public-key of the trapdoor commitment scheme. The verifier then commits to β, the prover sends α, the verifier opens the commitment to β, and the prover sends γ and also *the trapdoor for the commitment*. The zero-knowledge simulator follows the one for the standard 4-round protocol. The knowledge extractor, on the other hand, first runs one instance of the proof to get the trapdoor, and then it can effectively ignore the commitment in the second round, so you can use the extractor of the original 3-round protocol.