Cryptography and Game Theory: Designing Protocols for Exchanging Information*

Gillat Kol and Moni Naor†

Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot 76100 Israel
{gillat.kol,moni.naor}@weizmann.ac.il

Abstract. The goal of this paper is finding *fair* protocols for the secret sharing and secure multiparty computation (SMPC) problems, when players are assumed to be rational.
It was observed by Halpern and Teague (STOC 2004) that protocols with bounded number of iterations are susceptible to backward induction and cannot be considered rational. Previously suggested cryptographic solutions all share the property of having an essential exponential upper bound on their running time, and hence they are also *susceptible to backward induction.*
Although it seems that this bound is an inherent property of every cryptography based solution, we show that this is not the case. We suggest *coalition-resilient* secret sharing and SMPC protocols with the property that after any sequence of iterations it is still a computational best response to follow them. Therefore, the protocols can be run any number of iterations, and are *immune to backward induction.*
The mean of communication assumed is a broadcast channel, and we consider both the *simultaneous* and *non-simultaneous* cases.

1 Introduction

1.1 Background and Related Work

The issue of fairness in multiparty computation has been actively investigated since the inception of the field. In fact, the goal of Yao's 1986 famous paper [33] (where Garbled Circuits were introduced) was to address this problem. In this work we consider the *rational*, game-theoretic version of the secure function evaluation problem, that is when the players are assumed to have utility functions they try to maximize.

Realizing the advantages of simulating an equilibrium without depending on an honest mediator, the Game Theory community began pursuing a similar goal to that of Yao's in Game Theoretic settings. The works [2, 5, 4, 30, 10, 16] tried to remove the mediator by allowing the players to have free communication (so-called "cheap talk") prior to playing the game. In [7] this problem was addressed using cryptographic tools.

* Research supported by a grant from the Israel Science Foundation.
† Incumbent of the Judith Kleeman Professorial Chair.

Recently, the Cryptography community started exploring cryptographic information exchange problems, such as secret sharing and secure multiparty computation (SMPC), in Game Theoretic settings. Recall that in the classical problem of m-out-of-n secret sharing a dealer issues shares of a secret and privately assigns them to n players, such that any subset of m or more players can reconstruct the secret, but a subset of less than m players cannot learn anything about the secret. An SMPC protocol enables a group of players to evaluate a function on private inputs, but does not reveal *any* additional information about the players' inputs, over what is already disclosed by the function.

Since rational players will only participate in information exchange protocols when having an initial incentive to collaborate, we need to assume that players prefer getting the designated value (the secret or the function's value) to not getting it. In some papers it was further assumed that players prefer that as few as possible of the other players get the value. Although our protocols work without this last assumption, in the following discussions we always use this extreme case as an example.

The main difficulty in designing such fair protocols in rational settings is the players' desire to keep silent in the last round, if they can identify it (e.g., if the protocol is bounded), since they do no longer fear future punishment. Then, using a *backward induction* argument it can be shown that players prefer to keep silent in every round (see discussion in Section 1.3).

Several protocols overcoming this hurdle were offered by Halpern and Teague [15], Gordon and Katz [14], Abraham et al. [1], and Lysyanskaya and Triandopoulos [23]. All protocols require *simultaneous* channels (either a broadcast channel, or secure private channels) and use the key idea that in any given round players do not know whether the current round is going to be the last round, or whether this is a just a test round designed to catch cheaters. To prevent players from finding out the type of the round before it is carried out, the protocols in [1, 23] used computational based cryptography.

We claim that those protocols have a weak point: they are still essentially bounded, since the cryptographic primitives used in the beginning of the protocols can surely be broken after an exponential number of rounds. Hence, they are also *susceptible to backward induction*. In a previous paper [19] we have offered a non-cryptographic protocol for rational secret sharing that is immune to backward induction. The protocol uses special formed shares taken from unbounded domains (we have shown that unbounded domains are necessary in this setting), and cannot be generalized to the case of rational SMPC.

In this work we show that new cryptographic tools can be used to get the best of all worlds. We start off by considering the case of a simultaneous broadcast channel (SBC), where all player broadcast messages at the same time (no rushing). We offer a fair, coalition-resilient rational secret sharing scheme that may use any set of shares (provided that they can be authenticated), and generalize our protocol to the case of rational SMPC. We then consider the case of a non-simultaneous broadcast channel (NSBC), where there is only a single sender per round. We show how to run the previous protocols using only an NSBC, at least when the function's range is small.[1] Unlike previously suggested cryptographic solutions, our protocols are *immune to backward induction*.

Another line of work was pursued by Lepinski et al. [20, 21] and Izmalkov et al. [17] in their recent sequence of papers. Roughly speaking, they were able to obtain fair, rational SMPC protocols, prevent coalitions, and eliminate subliminal channels. However, the hardware requirements needed for these operations, including ideal envelopes and ballot boxes, are very strict; it is not clear how they can be implemented for distant participants, if at all.

1.2 Rationality Concepts

In Game Theoretic settings players are assumed to be *rational*. A great deal of effort was invested in trying to capture the nature of rational behavior, resulting in a long line of stability concepts. The best known concept is that of a Nash equilibrium: a vector of players strategies is a Nash equilibrium if given that all the other players are following their prescribed strategy, no player can gain from deviating from his strategy. In a Nash equilibrium, each player's strategy is a *best response* to the strategies of the others.

A natural generalization of a Nash equilibrium is a $\mathcal{C}$-resilient equilibrium, where $\mathcal{C}$ is a collection of subsets of players (coalitions). In a $\mathcal{C}$-resilient equilibrium, for any $C \in \mathcal{C}$, *no* member of the coalition C can do better, even if the *whole* coalition C defects. A Nash equilibrium is a $\mathcal{C}$-resilient equilibrium, where $\mathcal{C}$ is the set of all coalitions of size 1.

A cryptographic protocol cannot be expected to be the best response for all possible situations, since a relatively benign player may be very lucky and discover how to break a cryptographic primitive. Therefore, the previously suggested cryptographic protocols, as well the protocols suggested in this paper, are not exact Nash equilibria. However, they are computational Nash equilibria, i.e., they are "close" to being Nash in the sense that no player has an *efficient* (polynomial) deviating strategy that yields a *non-negligibly* greater payoff than the equilibrium strategy. A computational $\mathcal{C}$-resilient equilibrium is defined similarly.

As pointed out by Halpern and Teague [15], when considering information exchange tasks, requiring protocols to induce a Nash equilibrium is not enough to ensure stability. For example, the famous m-out-of-n scheme due to Shamir [28], requiring players to broadcast their given shares, is a Nash equilibrium when $m < n$ and more than m players participate in the reconstruction, but is unstable since players prefer to keep silent rather than reveal their shares. This is due to the fact that silence strategy is never worse than the strategy of revealing the share, but it is sometimes strictly better. For example, if exactly $m - 1$ other players choose to reveal their shares.

[1] Quite a lot of effort was invested into approximating an SBC via an NSBC and obtaining fair protocols using cryptographic techniques of gradual release (see [6, 9, 25] for recent work). Note, however, that such results do not take into account the rationality consideration that we use in this paper. Incorporating rationality considerations into such protocols is an interesting challenge.

To rule out such behaviors, two different strengthenings of the notion of Nash equilibrium were used in [15, 14, 23, 1, 19]: equilibrium surviving iterated deletion of weakly dominated strategies and strict equilibrium. Such notions are not discussed in this paper: we find the notion of surviving iterated dominance problematic (see [19] for discussion), and the notion of strict equilibrium unsuitable for the computational case since it demands a *unique* best response.

1.3 The Backward Induction Process

As observed by Halpern and Teague, no information exchange protocol with bounded number of rounds can be regarded as stable in the rational setting: suppose that the protocol is bounded by b rounds. When round b is reached players no longer fear future punishment and prefer to keep silent. As mentioned before, the silence strategy is always at least as good as cooperation strategy, but is sometimes strictly better. Consequently, round $b-1$ is now essentially the last round, and players deviate from the same reason. The process continues in this way backwards in time, thus it is called backward induction, showing that players are better off keeping silent in rounds $b-2, b-3, ..., 1$ as well.

We sketch a basic version of the secret sharing schemes suggested in [23, 1], and show that a similar problem arises. We start by describing a version of the scheme that requires an "on-line dealer" (i.e., the dealer is involved in the reconstruction process), and then show how the on-line dealer was removed.

The scheme with an on-line dealer proceeds in a sequence of iterations. At the beginning of each iteration the dealer distributes new (Shamir) shares: with probability β (whose value is discussed later) the distributed shares are of the original secret, and with probability $1-\beta$ the shares are of a fictitious secret. Every player should then broadcast the last share given to him, as long as no player has deviated. If a deviation was detected, players abort the protocol.

When β is chosen to be small enough, as a function of the utility functions (the greater the ratio between the payoff for learning alone and learning with the others, the smaller β is), no player can improve his payoff by cheating. That is, the risk of deviating in a fake round and causing the others to abort overcomes the desire of getting a possibly higher payoff for deviating in a real round.

In order to remove the on-line dealer, players simulate the dealer using a (non-rational) SMPC protocol: the dealer only distributes initial shares of the secret, and in every iteration players run an SMPC protocol to compute the function that gets as input their initial shares and distributes new shares. It was shown in [1] that the described protocol is a computation $\mathcal{C}$-resilient equilibrium where $\mathcal{C}$ is the set of all coalitions of size smaller than the threshold.

We argue that a similar backward induction argument can be used to show the instability of the protocol without the on-line dealer, even in computational settings. To show our claim we first investigate the meaning of the phrase "*following a strategy*". We usually think of a strategy as a code of a program and say that player i follows the strategy σ_i if i runs the program σ_i line-by-line. However, the assumption that i runs the program σ_i, and not some other program σ_i' with the exact same "external functionality" (i.e., σ_i' broadcasts the same messages as σ_i), is not always realistic. *Therefore, we consider a strategy as satisfying the property X only if all possible* implementations *of it satisfy X.*[2] This approach of checking all possible "undetectable" deviating strategies resembles the "honest-but-curious" cryptographic approach.

Now suppose that players seem to be the running the protocol without the on-line dealer, but actually run an implementation of it for which each player works a polynomial "over time" in every iteration trying to crack information hidden about the shares from the SMPC used in the first iteration. This is done by checking one key in every iteration and storing the right key. Recall that in general an SMPC protocol only gives a computational protection, not information-theoretic one (this is certainly true when we want to be immune to arbitrary coalitions, or if we do not assume private lines). Therefore, after exponentially many iterations in the key size, even this new non-ambitious strategy will surely find the right key. This shows that there is an essential upper bound to the number of iterations this protocol can be run: if the K^{th} iteration is reached (where K is the number of possible keys), each player may be better off quitting and using his stored key to retrieve the secret and get a (non-negligible) extra payoff. From this point on, the same backward induction process can be applied.

The above example shows that the backward induction process in computational settings, where presumably we are not concerned with the protocol's stability in rare events, is as problematic as in the standard Game Theoretic settings, since it causes exponential events to be amplified: the instability of the protocol without the on-line dealer in the rare case that it runs for exponential number of iterations causes it to be unstable from round 1.

Although it seems that susceptibility to backward induction is an inherent property of every computational based cryptographic solution, this paper shows that this is not the case. Our protocols are not only computational $\mathcal{C}$-resilient equilibria, but satisfy the additional property that after *any sequence of iterations*, they still induce such equilibria. Thus, players will never have an incentive to deviate, and the backward induction argument cannot be used. We call such protocols computational $\mathcal{C}$-immune. Clearly, $\mathcal{C}$-immunity implies $\mathcal{C}$-resilience.[3]

---

[2] In classical Game Theory, where there are no computational limitations, the distinction between running σ_i and running σ'_i is insignificant: in both cases i's knowledge consists of his initial information and all previously selected actions. However, in settings such as ours, where resources are limited, the results of the calculations made by a player when running a *specific* program should also be considered as part of his knowledge, since it is not always possible for him to repeat them.

[3] We do not regard the $\mathcal{C}$-immunity property as a sufficient condition, ensuring the stability of information exchange protocols, as some unstable protocols satisfy it. For example, Shamir's m-out-of-n secret sharing scheme is $\mathcal{C}$-immune for the maximal possible set $\mathcal{C}$ (the set of all coalitions of size smaller than m), when $m < n$ and more than m players participate in the reconstruction, since its reconstruction protocol consists of a single communication round.

1.4 Organization and Summary

The main idea of our protocols is ensuring that no iteration until the last one contains *any* information, in the *information-theoretic sense*, about the players' private values. In order to so, we construct in Section 3 a new cryptographic tool called *meaningful/meaningless encryption* that has a special property: some public keys yield ciphertexts that cannot be decrypted (even with unbounded computational power). Such keys are called *meaningless*, while the other keys are called *meaningful* and provide semantic security. One can efficiently distinguish meaningful keys from meaningless ones only when given the private key.

In Section 4 we offer a rational secret sharing scheme for the SBC model that works for any kind of shares, provided that they can be authenticated. In every iteration of the scheme new private and public keys are created using a random seed via a (non-rational) SMPC. The public key is published and the seed is shared between the players. Players use the public key to encrypt their shares, and the ciphertexts are broadcasted. Then, the validity of the ciphertexts is verified by another SMPC. A key point is that the verification does not require knowledge of the original shares, thus leaks no information about the secret. After a successful verification the seed's shares are exchanged, allowing players regenerate the private key and check whether the public key is meaningful. If it is, the shares of the secret are retrieved from the ciphertexts, and the secret is regenerated. Otherwise, the protocol proceeds to the next iteration.

No information about the secret can be retrieved from the ciphertexts sent in iterations with meaningless keys, hence no coalition can benefit from deviating before the last iteration. Since players cannot efficiently identify this iteration before sending their encrypted share, they cannot prevent others from learning.

In Section 5 we offer a rational SMPC protocol, based on the secret sharing scheme. We first note that in a secret sharing scheme players are required to evaluate a "reconstruction function" on their shares in order to retrieve the secret. Since our secret sharing scheme works for any type of shares, it can be used to compute any reconstruction function. The main problem is that the computation is not secure, as players' shares are revealed during the last iteration. To protect players' inputs, the new rational SMPC protocol additionally creates a Garbled Circuit in each iteration, and requests players to encrypt their garbled strings instead of their original inputs.

Finally, in Section 6 we show how to get rid of the simultaneity assumption, at the price of causing the expected length of the protocol to depend (linearly) on the size of the function's range.

Our protocols are $\mathcal{C}$-immune for the maximal possible set of coalitions $\mathcal{C}$: the secret sharing scheme considers all coalitions of size smaller than the threshold. The SMPC protocols do not pose any new constraints on $\mathcal{C}$, over the ones already posed by the players ability to learn the function's value by colluding before the game starts. In general, we give no guarantee about the composability of our protocols with any other protocol.

Further details, as well as omitted proofs and definitions, can be found in the full version of this paper [18].

2 Definitions and Settings

2.1 Computing Games and Protocols

As discussed in Section 1.4, both rational secret sharing and rational SMPC require rational protocols allowing players to evaluate a function on their private values. Hence, we start off by describing a model for rational joint computation. This model is the computational analog of the one suggested in [19].

In rational joint computation a set of players $N = \{1, ..., n\}$ each holding an input are interested in evaluating an n-ary function $f : \mathbf{X} \to Y$ ($\mathbf{X} \subseteq \times_{i \in N} X_i$ for some sets X_i) with finite domain and range. Players are assumed to be rational, and try to maximize their utility function. Recall that utility functions associate numeric values to outcomes of the game, the value $u_i(o)$ is player i's payoff if outcome o was reached. In our case, an outcome consists of the players' inputs, and the sequence of actions taken by them.

Our input as protocol designers is the function f, the distribution over inputs $\mathcal{D}$, and players' utilities $(u_i)_{i \in N}$[4]. Actually, as discussed later, we only require partial information about the utilities and the distribution. We should then output a game and "rational" strategies allowing all players to "learn" $f(\mathbf{x})$.

We suggest a computing game for f (with respect to $(u_i)_{i \in N}$ and $\mathcal{D}$) that proceeds in a sequence of iterations, where each iteration may consist of multiple communication rounds. In every round players are allowed to broadcast *any* finite binary string of their choice and update their state (a private binary string). If an SBC is assumed, the broadcasts in every round are simultaneous. Otherwise, an NSBC is assumed, and only a single player may broadcast in every round. We make no assumptions regarding the NSBC's behavior when two or more players try to broadcast at the same time. In such cases, some players may get partial information about the messages. A player can leave the game in any round by broadcasting a quit sequence and outputting his guess of $f(\mathbf{x})$. Players observe the actions taken by the others in previous rounds, but do not view their guesses.

Throughout the paper we assume that players are computationally bounded and can only run efficient strategies to evaluate polynomial time computable functions. To define the computational power of the players, we introduce an external *initial security parameter* k into the game. The security parameter used in round t is $k + t$, and we require that the players' strategies can be computed in probabilistic polynomial time in the security parameter of the corresponding round. We assume that the parameters of the original game (like the payoff functions, the initial distribution over inputs, etc.) are all independent of the security parameter, and thus can always be computed "in constant time".

We say that strategy σ' implements strategy σ if they both choose the same action after witnessing the same transcript (sequence of messages broadcasted

[4] We regard the players' utility functions as given, and do not attempt to change them. Simpler solutions can be obtained by introducing a discounting factor to the utilities, causing them to decline over time. However, in such solutions when an advanced round is reached, the utilities assumed are very far from the original ones, thus do not properly reflect players preferences.

in previous rounds) when given the same input and random tape. Note that "implements" is a symmetric relation. A vector of strategies $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_n)$ is called a protocol, and we say that $\boldsymbol{\sigma}$ computes the function f if it almost always ends, and in every finite run of it all players output $f(\mathbf{x})$.

2.2 The $\mathcal{C}$-Immunity Property

In Game Theory, to show that an equilibrium $\boldsymbol{\sigma}$ is immune to backward induction, one needs to prove that it satisfies the following property: if players are running $\boldsymbol{\sigma}$, then after *any* history, following $\boldsymbol{\sigma}$ is still an equilibrium. Such equilibria are called subgame perfect or sequential equilibria. Note that if this property holds, then no player will ever have an incentive to deviate from σ_i, and thus no backward induction process can be applied.

However, since our protocols involve cryptographic tools, there may be histories for which the cryptographic primitives are broken, and we cannot expect the protocol to induce an equilibrium in such cases. In particular, since we deal with protocols that proceed in a sequence of iterations, executing cryptographic primitives in each, we can only hope to satisfy a slightly weaker property. Namely, that following the protocols is still a (computational $\mathcal{C}$-resilient) equilibrium after *any sequence of iterations*; i.e., after all histories that can be reached by $\boldsymbol{\sigma}$, after which a new iteration begins. As discussed in Section 1.3, we need to require this property to also hold when players are running an implementation of $\boldsymbol{\sigma}$, instead of $\boldsymbol{\sigma}$. We call protocols satisfying this demand computational $\mathcal{C}$-immune.

Definition 1 (computational $\mathcal{C}$-immune). *Let $\boldsymbol{\sigma}$ be an efficient protocol for a computing game, and $\mathcal{C}$ be a set of coalitions (subsets of players). Let R^t be the set of sequences of random tapes for the first t iterations that do not cause $\boldsymbol{\sigma}$ to end. A sequence $\mathbf{r} \in R^t$ is of the form $\mathbf{r} = (\boldsymbol{r}^1, ..., \boldsymbol{r}^t)$ where $\boldsymbol{r}^s = (r_1^s, ..., r_n^s)$ and r_j^s is the random tape used by player j in iteration s.*

The protocol $\boldsymbol{\sigma}$ is computational $\mathcal{C}$-immune *if for every coalition $C \in \mathcal{C}$, and every sequence of tapes $\mathbf{r}_0 = (\boldsymbol{r}_0^1, ..., \boldsymbol{r}_0^t) \in R^t$ used by the players in the first t rounds, there exists a negligible function $\varepsilon(k)$ such that for every player $i \in C$, every efficient (deviating) joint strategy $\boldsymbol{\sigma}'_C$ for players in C, and every efficient joint strategy $\boldsymbol{\tau}_{-C}$ for players in $N \setminus C$ implementing $\boldsymbol{\sigma}_{-C}$, it hold that:*

$$\mathbf{E}\left[u_i(\boldsymbol{\tau}_{-C}(k), \boldsymbol{\sigma}_C(k))\right] + \varepsilon(k) \geq \mathbf{E}\left[u_i(\boldsymbol{\tau}_{-C}(k), \boldsymbol{\sigma}'_C(k))\right]$$

The expectation is taken over all sets of random tapes for the players assigning them the tapes $\boldsymbol{r}_0^1, ..., \boldsymbol{r}_0^t$ for the first t iterations.

2.3 Settings for Rational Secret Sharing and Rational SMPC

We review the models for rational secret sharing and rational SMPC assumed in this paper.

Definition 2 (computational rational secret sharing scheme). *A* computational rational m-out-of-n secret sharing scheme *for a set of secrets Y, with respect to the distribution over secrets $\mathcal{D}$ and the utilities $(u_i)_{i \in N}$, consists of a dealer's algorithm for issuing shares, and a protocol allowing the players to reconstruct the secret. We require that:*

- *No subset C of less than m players can reveal any partial information about the secret before the game begins. I.e., the distribution over inputs given any shares of players in C is identical to the original distribution $\mathcal{D}$.*
- *The reconstruction protocol run by any group of at least m players is a computational $\mathcal{C}$-immune protocol for $\mathcal{C} = \{C \mid |C| \leq m - 1\}$ that computes the reconstruction function induced by the dealer's algorithm in the corresponding computing game.*

Definition 3 (computational $\mathcal{C}$-rational SMPC protocol). *Let $\mathcal{C}$ be a set of coalitions. A* computational $\mathcal{C}$-rational SMPC protocol *for f, with respect to a distribution over inputs $\mathcal{D}$ and utilities $(u_i)_{i \in N}$, is:*

- *A secure protocol in the cryptographic sense for the one shot case (see [11], Definition* 7.5.3*).*
- *A computational $\mathcal{C}$-immune protocol that computes f in the corresponding computing game.*

2.4 Assumptions on the Utilities and the Distribution Over Inputs

As mentioned in the Introduction, we must assume that players have initial motivation to participate in the computing games. As was done in previous papers, we assume that players prefer to learn the designated value. Formally, we say that a player learns the value when outcome o is reached, if according to o the player quits and outputs the right value. Our assumption is that for two possible outcomes o and o' it holds that $u_i(o) > u_i(o')$ whenever player i learns the value when o is reached, but does not learn when o' is reached.

In order to achieve $\mathcal{C}$-immune protocols, we additionally need to require that no coalition can guess the designated value or the last iteration of our protocol with a high enough probability. We denote by α an upper bound to the probability that a coalition $C \in \mathcal{C}$ can guesses the right value in advance, and by β' the probability (upto a negligible factor) that a coalition $C \in \mathcal{C}$ is able to identify the last iteration of the protocol before it is carried out. Note that in the protocol described in Section 1.3, as well as in our protocols, a value β determines the probability of proceeding to the next iteration and satisfies $\beta = \beta'$.

In the next sections we require $\alpha < \alpha_0$ and $\beta < \beta_0$, where α_0 and β_0 are functions of the utilities and of the set $\mathcal{C}$. The calculation of the functions is deferred to the full version of this paper [18]. As before, the greater the ratio between the payoff for learning the secret alone and learning with the others, the smaller α_0 and β_0 should be. Note that since players can always guess the value y with the highest probability according to $\mathcal{D}$, it holds that $\alpha \geq \mathcal{D}(y)$, and thus the requirement $\alpha < \alpha_0$ poses a condition on $\mathcal{D}$.

3 Cryptographic Tools

3.1 Standard Cryptographic Tools

Our protocols use several standard cryptographic tools:

A Commitment Scheme. We assume that $\mathsf{Commit}(x, r) = com$ generates a commitment for the value x using randomness r, and that the commitment is *perfectly binding*. We call (r, x) the *opening* of *com*.

A (Non-Rational) SMPC Protocol. We assume that the protocol allows the evaluation of randomized functions (in particular, we use it to select a random seed, and assume that the players cannot bias the result). In addition, we require that the SMPC protocol enables its participants to detect deviations with high probability. The protocol should work for an active adversary statically corrupting any number of parties ($\leq n-1$). We do not consider premature suspension of execution a violation of security, and do not assume fairness. Our application of the SMPC ensures that players have an incentive to carry it out, allowing everybody to get the output.

A 1-Out-Of-2 OT Protocol. We assume that the OT protocol works for the active adversary model and provides computational security to the sender, and information-theoretic protection to the receiver. That is: (i) if the sender's values are (s_0, s_1) and the receiver's input is $b \in \{0, 1\}$, then the OT protocol is an SMPC (again, in the sense of Definition 7.5.3 in [11]) of the function $f((s_0, s_1), b) = s_b$, (ii) for every behavior of the sender, he witnesses the same distribution over transcripts when the receiver's input is 1 and when it is 0.

Such protection is possible under standard assumptions such as *enhanced trapdoor permutations* [8, 11] and *Computational Diffie-Hellman* [3] for honest-but-curious players (the recent work [32] shows that OT is symmetric, thus a protocol that protects the sender information theoretically can be transformed to one that protects the receiver). In order to handle malicious behaviors, we use the compiler described in [11], with one change: the receiver uses a ZK *argument* with a *perfectly hiding* commitment ensuring information-theoretic security for its value in order to prove to the sender that he followed the protocol properly.

We assume that all the cryptographic primitives (the standard tools and the meaningful/meaningless encryption described next) are immune to non-uniform attacks. This assumption is needed in order to show that our protocols are stable after any number of iterations.

3.2 Meaningful/Meaningless Encryptions

In additional to the standard tools, we use a non-standard encryption scheme E called a meaningful/meaningless encryption. E has a special property: some public keys of it yield ciphertexts that cannot be decrypted (even with unbounded computational power). Such keys are called *meaningless*, while the other keys are called *meaningful*.

Definition 4 (meaningful/meaningless encryption). *An encryption scheme* $\mathsf{E}(pub\_key, random, plain) = cipher$ *is a* β-Meaningful/Meaningless Encryption *if it satisfies the following properties:*

Key Generation*: Polynomial time generation of a private key, priv_key, and a public key, pub_key, on a given seed.*

Encryption*: Computing* $c = \mathsf{E}(pub\_key, r, m)$ *can be done in polynomial time, given a public key pub_key, randomness* r*, and plaintext* m*.*

Meaningful and Meaningless Keys*: The public keys are partitioned into meaningful and meaningless sets. The probability, over the seeds, that the generated public key is 'meaningful' is* β*, and the probability of it being 'meaningless' is* $1 - \beta$*.*

If *pub_key* is meaningful*, then given* $c = \mathsf{E}(pub\_key, r, m)$ *and priv_key, the message* m *can be uniquely retrieved in polynomial time. Furthermore, for every ciphertext* c *there is only one plaintext* m *for which there exists a randomness* r *satisfying* $c = \mathsf{E}(pub\_key, r, m)$*. The encryptions are computationally indistinguishable: for any two messages* m *and* m'*, the distributions of* $\mathsf{E}(pub\_key, r, m)$ *and* $\mathsf{E}(pub\_key, r, m')$ *are computationally indistinguishable.*

If *pub_key* is meaningless*, then knowing* c *and priv_key yields no information about* m*. That is, for any two messages* m *and* m'*, the distributions of* $\mathsf{E}(pub\_key, r, m)$ *and* $\mathsf{E}(pub\_key, r, m')$ *are identical.*

Distinguishing Meaningful from Meaningless*: Given two public keys, one meaningful and one meaningless, guessing which is which cannot be done with a non-negligible advantage over* $\frac{1}{2}$ *by a probabilistic polynomial time tester. However, when supplied with the corresponding private key, the test is polynomial.*

Meaningful/meaningless encryption schemes can be constructed based on *Decisional Diffie Hellman*, using the construction in [24], on *Quadratic Residousity* [13], and on any *homomorphic encryption*[5].[6] For completeness we describe a construction of E that assumes the intractability of Quadratic Residousity, based on the scheme of Goldwasser and Micali [13].

Recall that in Goldwasser and Micali's scheme two distinct large prime numbers p and q are generated, and (p, q) is used as a private key. The public key generated is (N, x) where $N = pq$ and x is a quadratic *non*-residue of N ($x \neq z^2 \bmod N$) that has a Jacobi Symbol of $+1$. Each bit b_i of the message m is encrypted separately by choosing $y_i \in_R \mathbb{Z}_n^*$ and calculating $c_i = y_i^2 x^{b_i} \bmod N$.

[5] Homomorphic encryption is an encryption scheme with the additional special property: given two ciphertexts it is possible to generate a ciphertext for the sum (or multiplication) of the corresponding plaintexts.

[6] An interesting open problem is finding the minimal assumptions under which such a meaningful/meaningless encryption scheme can be constructed. The task requires non-trivial SZK: given a public key *pub_key* and two messages m and m' players should not be able to tell whether the two efficiently generated distributions $\mathsf{E}(pub\_key, r, m)$ and $\mathsf{E}(pub\_key, r, m')$ are identical or far apart. This problem was shown to be in SZK [27], and hence we must assume that there is a problem in SZK that is not in BPP.

The ciphertext is $(c_1, ..., c_n)$, and it can be decrypted using the private key (p, q): $b_i = 0$ iff c_i is a quadratic residue.

To construct a meaningful/meaningless encryption E we modify this scheme such that x is a random quadratic residue with probability $1 - \beta$, and a random quadratic non-residue with Jacobi Symbol of $+1$ with probability β. Note that if x is a quadratic residue, c_i is always a quadratic residue, and nothing can be learned about b_i, even when p and q are known.

Claim. The scheme E described above is a meaningful/meaningless encryption.

4 The Rational Secret Sharing Scheme

4.1 The Scheme

We describe an m-out-of-n rational secret sharing scheme for the SBC model.

The Dealer's Protocol. The scheme works for *any kind* of m-out-of-n shares the dealer may distribute (e.g. Shamir shares), provided that he additionally issues information-theoretic authentications for each share. For concreteness, it is assumed that the authentication information given to each player consists of a *tag* and a *hash function*. The hash function should allow the player to verify the authenticity of shares broadcasted by the others in probabilistic polynomial time and with error probability negligible in the security parameter. The tag should allow the player to prove the authenticity of the share he uses. The authentication information held by a group of players must not disclose any information about the other players' shares.[7]

The Players' Protocol. The reconstruction protocol is called `clean-slate` and it proceeds in a sequence of iterations. The protocol, like the one described in Section 1.3, uses a parameter β and has the property that after any sequence of iterations, the probability that the next iteration is the last one, revealing the secret, is β. Every iteration of the protocol consists of the following steps:

The *Key Generation* step. In each iteration new private and public keys for a β - meaningful/meaningless encryption are generated. This is done via a (non-rational) SMPC that takes no inputs, and generates the keys using a randomly chosen seed. The seed is shared between the players, and the public key, as well as a *perfectly binding* commitment to each of the seed's shares, are published.

If the public key generated is meaningful (which happens with probability β), we call the iteration *meaningful*, otherwise the iteration is *meaningless*. The

[7] For example, this can be done using the following method (see [31, 26]): if player i's true information is $x \in \mathbb{F}$, then $s_i, b_i \in \mathbb{F}$, $b_i \neq 0$, are chosen at random and we set $c_i = b_i \cdot x + s_i \in \mathbb{F}$. The value s_i (the tag) is given to i. The other players each get b_i and c_i (the hash function). Player i is required to broadcast s_i in order to prove that x is his true information. The other players can then verify with high probability by checking that $c_i = b_i \cdot x + s_i$.

protocol is designed not to reveal any information about the secret in meaningless iterations, and to allow the players to uncover the secret during the first meaningful iteration.

The *Encryption* and *Verification* steps. Players encrypt their share of the secret and authentication information (i.e., the tag and the hash function) using the meaningful/meaningless encryption with the public key generated in the last step. The ciphertexts are broadcasted and then validated by another SMPC.

The verification process takes as inputs the shares of the seed used to generate the keys, and additionally uses the broadcasted ciphertexts and the commitments published during the Key Generation step. It authenticates the seed's shares using the commitments, and uses them to regenerate the private key. Since the commitments are binding, the original private key is always the one generated, allowing the process to correctly determine whether the iteration is meaningful. If it is, the ciphertexts are decrypted and the retrieved authentication information is used to authenticate the retrieved shares of the secret, by verifying that all the tags and hash functions match.

The verification is considered to be successful if: (i) each seed share is a valid opening of the corresponding commitment, (ii) in case of a meaningful iteration, each ciphertext is valid encryption of a share of secret and a corresponding authentication.

A key point is that the verification process does not take the players' shares or authentication information as inputs, and when the public key is meaningless the ciphertexts it uses convey no information about the shares of the secret.

The *Exchange* step. If the verification process was successful, players *simultaneously* broadcast their shares of the seed. Each player then authenticates all seed's shares, regenerates the seed and determines by himself whether the iteration is meaningful. If it is, he decrypts the ciphertexts and uses the extracted shares of the secret to reconstruct the secret. Otherwise, the protocol proceeds to the next iteration.

Recall that players have only a small chance of discovering whether the key is meaningful before the seed's shares are revealed, since there is no efficient way of checking it. Thus, they are motivated to participate in the Exchange step. The complete protocol is described in Figure 1.

4.2 Scheme Analysis

We next argue that the suggested scheme is a computational rational secret sharing scheme. We first claim that `clean-slate` satisfies the following property, leading to its name: assuming that all players except (maybe) players in the coalition C are following the protocol, then no information about the secret is revealed before the last iteration (that is, every iteration "starts off with a clean slate"). The reason is that players' shares and authentications are only used by the protocol to create the encrypted messages. However, all iterations before the last one are meaningless, thus previous ciphertexts were created using meaningless keys and are simply random.

`clean-slate`$_i(share, authen)$

Let P be the set of players participating in the reconstruction, and denote $p = |P|$.

Repeat

If one of the following tests fail, or if a deviation was detected in one of the cryptographic schemes, quit.

Key Generation: Players run an SMPC of the function $GenarateKey$:

$GenarateKey$

- Choose p random strings, $(r_i)_{i \in P}$, of length $k + t$ where t is the iteration number and k is the initial security parameter.
- Generate public and private keys pub_key, $priv\_key$, for E using $\oplus_{i \in P} r_i$ as a seed.
- Choose p random strings, $(rand\_r_i)_{i \in P}$, of length $k + t$ and set $com\_r_i = \mathsf{Commit}(r_i, rand\_r_i)$.
- ***Public Output***: The public key pub_key, and the commitments $(com\_r_i)_{i \in P}$.
- ***Private Output***: The values r_i and $rand\_r_i$ are given to player i.

Encryption: Encrypt *share* and *authen* using E with parameter β and with the public key pub_key, and broadcast the encrypted message C_i.

Verification: Players run an SMPC of the function $Verify$ that takes $(r_i, rand\_r_i)_{i \in P}$ as inputs:

$Verify$

- Check that each input pair is a valid opening of the corresponding commitment. That is, verify $com\_r_i = \mathsf{Commit}(r_i, rand\_r_i)$.
- Regenerate $priv\_key$ using $\oplus_{i \in P} r_i$ as a seed, and use it to check whether pub_key is meaningful.
- If so, decrypt each C_i using $priv\_key$, and get the shares of the secret and authentication information of each player. Check that the shares are consistent with the authentications by verifying that all the tags and hash functions match.

Exchange:

- Broadcast r_i and $rand\_r_i$.
- Evaluate the first two stages of $Verify$ by yourself.
- If the pub_key is meaningful, reconstruct the secret using the retrieved shares (as done in the last step of $Verify$). Quit and Output the reconstructed secret.

Fig. 1. The rational secret sharing reconstruction protocol

To show that no coalition C of size at most $m - 1$ has an incentive to deviate after any sequence of iterations, we note that for any joint strategy players in C may follow, they cannot be worse-off (up to an exponentially small factor) by always following the Key Generation, Encryption, and Verification steps: Key Generation and Verification are done via an SMPC, and therefore cannot be broken with a non-negligible probability. As to broadcasting a valid ciphertext - in a meaningless iteration no information can be gained anyway, and in a meaningful iteration the verification step detects invalid ciphertexts with high probability. Thus, we may assume that players only deviate during the Exchange step by broadcasting a seed share that does not open the commitment published in the Key Generation step. Such deviations are always detected, since the commitments to the shares are perfectly binding.

We argue that a coalition can only gain from deviating in the Exchange step of a meaningful iteration: if it deviates in a meaningless iteration, then no information about the secret is revealed due to the clean slate property, and thus the players are forced to guess the secret. Recall that a coalition cannot efficiently distinguish between meaningful and meaningless iterations before the Exchange step, if all its players have broadcasted valid encryptions (which is what we assume). Therefore, if the coalition deviates in meaningful iterations with a certain probability, it must deviate in meaningless ones with almost the same probability. As before, for a sufficiently small β, the risk of deviating in a meaningless iteration and causing the game to end is too great.

Theorem 1. *Let $2 \leq m \leq n$, Y be a finite set of secrets, and* `dealer` *be an algorithm assigning m-out-of-n information-theoretic authenticatable shares. Assume that $\alpha < \alpha_0$ and $\beta < \beta_0$. The scheme* `(dealer, clean-slate)` *is a computational rational m-out-of-n secret sharing scheme for Y with expected number of iterations $O(1/\beta)$.*

5 The Rational SMPC Protocol

5.1 The Protocol

We present the protocol `secure-clean-slate`, a rational SMPC protocol for the SBC model, based on protocol suggested in Section 4. The new protocol, like the previous one, ensures that no information is leaked until the final iteration (in an information theoretical sense). However, it additionally protects the inputs (in a computational sense) during the last iteration. This is done by composing the meaningful/meaningless technique with *Yao's Circuit Garbling* method.[8]

Recall that a Garbled Circuit is an encrypted form of an original circuit. It allows the circuit to be evaluated, but reveals no information except the result of the evaluation. A Garbled Circuit consists of: two random (garbled) strings assigned to each input wire (the first corresponds to a 0 value, and the other to a 1), gates tables, and translation tables for outputs. To evaluate the original circuit on a specific input, the Garbled Circuit is evaluated for the corresponding garbled strings using the gates tables. Then, the output is translated using the outputs translation tables. For a detailed description of Garbled Circuits see [22]. The `clean-slate` protocol in changed in the following way:

[8] General techniques for (non-rational) SMPC do not offer information-theoretic protection for both sides, thus cannot be used directly. In models in which such protocols can be constructed, we can use the secret sharing scheme from the last section in order to allow players to fairly exchange the last messages sent by the protocols.

Adding the step of *Creating Garbled Circuit*. In every iteration the protocol constructs a new Garbled Circuit from the circuit representing f. The gates tables and translation tables are made public, and commitments to both garbled strings corresponding to each input wire are published *in an arbitrary order* (the reason for the arbitrary order will be made clear later). However, players are not given both garbled strings assigned to each of their input wires, since this will allow player i to learn $f(\mathbf{x}_{-i}, x'_i)$ for every x'_i. Instead, a share of an n-out-of-n secret sharing of each garbled string assigned to an input wire is given to every player, and commitments to all shares are published.

Adding the step of *Obtaining Garbled Inputs*. Each player obtains *one* of the garbled strings chosen for each of his input wires according to the value assigned to the wire by his input. Player i gets all the shares of each such garbled string by engaging in a 1-out-of-2 OT protocol with every player j. When running the OT protocol, player j is the sender and his values are the shares of the two garbled strings chosen for i's input wire. Player i is the receiver, and his goal is to learn the value corresponding to his input bit. As discussed in Section 3, the OTs give information-theoretic protection to the receiver regarding the value he received, and computational security to the sender about the other value. This kind of protection is crucial, since we want to ensure that no information about i's input is leaked during meaningless iterations.

For ease of exposition we say that the sender (player j) sends encryptions of his two values to the receiver (player i) when the OT protocol is carried out. We require j to supply an additional ZK proof to convince i that both encryptions are valid. That is, after sending the encryptions, j must prove to i that each encryption contains a value that opens the corresponding commitment published during the Creating Garbled Circuit step.

Revising the steps of *Encryption* and *Verification*. Players encrypt their *garbled strings*, instead of their original inputs, using the β - meaningful/ meaningless encryption with the public key generated in the Key Generation step.

The verification process is changed: in a meaningful iteration it decrypts the ciphertexts and retrieves the garbled strings for each input bit. It then verifies that each extracted garbled string indeed opens *one* of the corresponding commitments. Note that since the commitments to the garbled strings corresponding to the same input wire were published in an arbitrary order when the Garbled Circuit was created, no information about the real value of this input wire is revealed to the other players.

During the Exchange step of a meaningful iteration the garbled strings are retrieved from the ciphertexts, allowing all players to learn the function's value, but protecting the original inputs. In a meaningless iteration, no information about the garbled strings encoding the real inputs is revealed, and hence no information about the real inputs is disclosed either. The complete protocol is described in Figure 2.

`secure-clean-slate`$_i(input)$

Repeat

If one of the following tests fail, or if a deviation was detected in one of the cryptographic schemes, quit.

Key Generation: As in `clean-slate`$_i$ (with $P = N$).

Creating Garbled Circuit: Players run an SMPC of the function: $CreateGarbledCiruit$

- Create a Garbled Circuit of the evaluated function f. The garbled string assigned to wire q and bit b is denoted W_q^b.
- Choose random strings $rand\_W_q^b$ of length $k + t$ where t is the iteration number and k is the initial security parameter. Denote $V_q^b = (W_q^b, rand\_W_q^b)$.
- Randomly select shares $V_q^{b,1}, ..., V_q^{b,n}$ such that $V_q^b = \oplus V_q^{b,i}$, and strings $rand\_V_q^{b,i}$ of length $k + t$.
- ***Public Output***: (i) Tables for the garbled gates and translation tables for the outputs. (ii) The commitments $com\_W_q^b = \mathsf{Commit}(W_q^b, rand\_W_q^b)$. *For every input wire* q, *the commitments* $com\_W_q^0, com\_W_q^1$ *are output in an arbitrary order.* (iii) The commitments $com\_V_q^{b,i} = \mathsf{Commit}(V_q^{b,i}, rand\_V_q^{b,i})$.
- ***Private Output***: The values $V_q^{b,i}$ and $rand\_V_q^{b,i}$ are given to player i.

Obtaining Garbled Inputs: If player i holds the q^{th} input bit of f and its value is b, he engages in a 1-of-2 OTs (perfectly protecting player i) with every other player j, in order to get $V_q^{b,j}$ and $rand\_V_q^{b,j}$. When running an OT protocol, after player j sends encryptions of his two pair of values, $(V_q^{0,j}, rand\_V_q^{0,j})$ and $(V_q^{1,j}, rand\_V_q^{1,j})$, to player i, he supplies a ZK proof to convince i that each encryption contains a pair that is a valid opening the corresponding commitment ($comm\_V_q^{0,j}$ or $comm\_V_q^{1,j}$). Player i then reconstructs V_q^b using the received shares.

Encryption: Player i encrypts all V_q^b acquired during the previous step using E with parameter β and public key pub_key, and broadcasts the ciphertext C_i.

Verification: As in done in `clean-slate`$_i$, a $Verify$ procedure is run via an SMPC. The previous procedure is changed: if the key is meaningful, it decodes every C_i and checks that for every input bit q, the retrieved value $V_q^b = (W_q^b, rand\_W_q^b)$ is an opening of one of the commitments $com\_W_q^0$ or $com\_W_q^1$.

Exchange: As in `clean-slate`$_i$ with the exception that if the public key is meaningful, the function's value is obtained by evaluating the garbled circuit using the gates tables on the garbled strings extracted from the ciphertexts, and then translating the output using the outputs translation tables.

Fig. 2. The rational SMPC protocol

5.2 Protocol Analysis

We next argue that `secure-clean-slate` is a computational rational SMPC protocol. As discussed before, the protocol is secure (in the cryptographic sense), since no information about the inputs is revealed before the last iteration, and due to the fact that the Garbled Circuit created in the last iteration protects players' inputs computationally. To show that the protocol is also $\mathcal{C}$-immune, we must first assume that players in every coalition $C \in \mathcal{C}$ have an initial incentive to use their true inputs when running a protocol that computes f. Note that although non-rational SMPC protocols allow players to change their inputs, we must rule out such behaviors since our utility functions only reward players for learning the value of f evaluated on the *original* inputs.

One way of ensuring such incentives is to assume that players in C would have reported their true inputs had a trusted mediator been running the computation. That is, by using fictitious inputs, players in C are unlikely to be able to change the output of the calculation and still deduce the designated value (see discussion in [29]).[9] An alternative way is to assume the presence of an authenticator that produces authentication information for the inputs (as was done in the secret sharing scheme of Section 4). If one of the above options holds, we say that players in C *have an initial incentive to use their true shares*. When such an incentive is assumed, the described protocol can be shown to be $\mathcal{C}$-immune using the arguments made for the `clean-slate` protocol.

Theorem 2. *Let f be a polynomial time computable function, and let $\mathcal{C}$ be a set of coalitions. Assume that players in every coalition $C \in \mathcal{C}$ have an initial incentive to use their true shares, and that $\alpha < \alpha_0$ and $\beta < \beta_0$. The protocol* `secure-clean-slate` *is a computational rational SMPC protocol for f with expected number of iterations $O(1/\beta)$.*

6 The Rational SMPC Protocol for the NSBC Model

We describe the protocol `NSBC-secure-clean-slate`, a rational SMPC protocol for the NSBC model, based on the protocol suggested in Section 5. We first note that the trivial way of dividing every simultaneous round of the previous protocol into n non-simultaneous rounds fails: the last player to broadcast his share of the seed in the Exchange step of the meaningful iteration has already learned the value, and thus has no incentive to cooperate. We construct a new protocol in which players can retrieve the value even if the last player deviated, since the needed information is revealed by the number of the round he deviated in. The previous protocol is changed in the following way:

Revising the step of *Key Generation*. The new Key Generation step generates $|Y|$ pairs of keys, instead of just one. The set of public keys generated in every iteration has the property that at most one is meaningful. An iteration containing a meaningful key is called meaningful, and the others are called meaningless. As before, no information about the inputs is revealed in meaningless iterations, and players uncover the value during the first meaningful iteration.

[9] For example, suppose that the players' inputs are bit strings and they wish to calculate the strings' XOR. A player benefits from using a fictitious input string, even if the computation is done by a trusted mediator: the other players will get a false value, but the deviating player will be able retrieve the real value by XORing the result with both his fictitious and real strings.

Revising the steps of *Encryption* and *Verification*. In the Encryption step, players are required to encrypt their inputs $|Y|$ times using each of the public keys, and broadcast the ciphertexts one-by-one.

The verifications process is changed: in addition to validating the ciphertexts, it also outputs a permutation of the public keys. In a meaningless iteration the published permutation is completely random. But, in a meaningful iteration the permutation places the (only) meaningful key in position y, where y is the designated value, and randomly orders the rest of the keys. Note that the verification process can obtain y by evaluating the Garbled Circuit on the garbled strings retrieved from the ciphertexts, and then translating the output.

Revising the step of *Exchange*. The Exchange step is partitioned to $|Y| \cdot n$ non-simultaneous communication rounds in which shares of the seeds used to generate the keys are revealed *one by one*. First the shares of seed 1 are revealed in the first n rounds (call it cohort 1) with player j sending his share in round j, and so on for each of the $|Y|$ seeds. If a player deviates (e.g. refuses to reveal his share of the seed), and this is the last round of the y^{th} cohort, the other players conclude that he already learned f's value, and hence it must be y.

Note 1. The described protocol is susceptible to existence of a malicious player: such a player can cause the others to output a wrong value by simply aborting prematurely. However, the deviating player will not be able to learn the secret himself. Since we assume that all players are rational individuals that prefer to learn above all else, there will never be an incentive to such behavior.

Theorem 3. *Let f be a polynomial time computable function, and let $\mathcal{C}$ be a set of coalitions. Assume that players in every coalition $C \in \mathcal{C}$ have an initial incentive to use their true shares, and that $\alpha < \alpha_0$ and $\beta < \beta_0$. The protocol* `NSBC-secure-clean-slate` *is a computational rational SMPC protocol for f with expected number of communication rounds $O\left(\frac{|Y|n}{\beta}\right)$.*

References

1. I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation. *PODC,* pages 53-62*,* 2006.
2. I. Barany. Fair distribution protocols or how the players replace fortune. *Mathematics of Operations Research*, volume 17, pages 327-340, 1992.
3. M. Bellare, and S. Micali, Non-Interactive Oblivious Transfer. *CRYPTO*, pages 547-557, 1989.
4. E. Ben-Porath. Cheap talk in games with incomplete information. *Journal of Economic Theory*, volume 108, pages 45-71, 2003.
5. E. Ben-Porath. Correlation Without Mediation: Expanding the Set of Equilibria Outcomes by "Cheap" Pre-Play Procedures. *Journal of Economic Theory*, volume 80, pages 108-122, 1998.
6. D. Boneh and M. Naor, Timed commitments. *CRYPTO*, *Springer LNCS* 1880, pages 236-254, 2000.

7. Y. Dodis, S. Halevi and T. Rabin, A Cryptographic Solution to a Game Theoretic Problem. *CRYPTO*, pages 112-130, 2000.
8. S. Even, O. Goldreich and A. Lempel, A Randomized Protocol for Signing Contracts. *Communications of the ACM*, volume 28, No. 6, pages 637-647, 1985.
9. J. Garay and M. Jakobsson, Timed Release of Standard Digital Signatures. *In Proceedings of Financial Cryptography*, LNCS 2357, pages 168-182, Springer, 2002.
10. D. Gerardi. Unmediated communication in games with complete and incomplete information. *Journal of Economic Theory*, volume 114, pages 104-131, 2004.
11. O. Goldreich. Foundations of Cryptography, volume 2: Basic Applications. *Cambridge University Press*, 2004.
12. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. *STOC*, pages 218-229, 1987.
13. S. Goldwasser, and S. Micali, Probabilistic Encryption. *Journal of Computer and System Sciences*, volume 28, page 270-299, 1984.
14. S. D. Gordon and J. Katz. Rational Secret Sharing, Revisited. *SCN,* pages 229-241, 2006.
15. J. Halpern and V. Teague. Rational Secret Sharing and Multiparty Computation. *STOC,* pages 623-632, 2004.
16. Y. Heller. A coalition-proof cheap-talk protocol. *Manuscript*, 2005.
17. S. Izmalkov, S. Micali, and M. Lepinski. Rational Secure Computation and Ideal Mechanism Design. *FOCS*, pages 585-595, 2005.
18. G. Kol and M. Naor, Cryptography and Game Theory: Designing Protocols for Exchanging Information, full version:
`www.wisdom.weizmann.ac.il/%7Enaor/PAPERS/crypto_games.html`.
19. G. Kol and M. Naor, Games for Exchanging Information. *Manuscript*, 2007.
20. M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely Fair SFE and Coalition-Safe Cheap Talk. *PODC*, pages 1-10, 2004.
21. M. Lepinski, S. Micali, and A. Shelat. Collusion-Free Protocols. *STOC,* pages 543-552, 2005.
22. Y. Lindell and B. Pinkas. A Proof of Yao's Protocol for Secure Two-Party Computation. *ECCC*, Report TR04-063, 2004.
23. A. Lysyanskaya and N. Triandopoulos. Rationality and Adversarial Behavior in Multi-Party Computation. *CRYPTO,* pages 180-197, 2006.
24. M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. *SODA*, pages 448-457, 2001.
25. B. Pinkas. Fair Secure Two-Party Computation. *Eurocrypt,* pages 87-105, 2003.
26. T. Rabin and M. Ben-Or, Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. *STOC*, pages 73-85, 1989.
27. A. Sahai and S. Vadhan. A Complete Problem for Statistical Zero Knowledge. *Journal of the ACM*, volume 50, pages 196-249, 2003.
28. A. Shamir. How to share a secret. Communications of the ACM, volume 22, pages 612-613, 1979.
29. Y. Shoham and M. Tennenholtz. Non-Cooperative Computation: Boolean Functions with Correctness and Exclusivity. *TCS*, 343 (2), pages 97-113, 2005.
30. A. Urbano and J. Vila. Computational Complexity and Communication: Coordination in Two-Player Games. *Econometrica*, volume 70, pages 1893-1927, 1992.
31. M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *JCSS*, volume 22, pages 265-279, 1981.
32. S. Wolf and J. Wullschleger. Oblivious Transfer is Symmetric. *Eurocrypt*, pages 222-232, 2006.
33. A. Yao. How to Generate and Exchange Secrets. *FOCS*, pages 162-167, 1986.