Rational Secure Computation and Ideal Mechanism Design

Sergei Izmalkov
Dept of Economics
MIT

Silvio Micali
CSAIL
MIT

Matt Lepinski
CSAIL
MIT

Abstract

Secure Computation essentially guarantees that whatever computation n players can do with the help of a trusted party, they can also do by themselves. Fundamentally, however, this notion depends on the honesty of at least some players.

We put forward and implement a stronger notion, Rational Secure Computation*, that does not depend on player honesty, but solely on player* **rationality.** *The key to our implementation is showing that the* ballot-box*—the venerable device used throughout the world to tally secret votes securely—can actually be used to securely compute* **any** *function.*

Our work bridges the fields of Game Theory and Cryptography, and has broad implications for Mechanism Design.

1 The Case for Rational Security

Secure Computation. The general notion of Secure Computation was put forward and first exemplified by Goldreich, Micali and Wigderson [8], building on earlier two-party results of Yao [17]. Given a joint computation among n players and a trusted party, Secure Computation aims at removing the trusted party without suffering any correctness or privacy loss. A bit more precisely, all prior secure-computation work—by now quite extensive— adopts the original *ideal/real* paradigm, illustrated below in the crucial, special case of a *secure function evaluation* (SFE for short).

An *ideal evaluation* of a (possibly probabilistic) n-input, n-output function f consists of the following process. Each player i has a private input, x_i, and is assumed to be *honest* or *malicious*. An honest i simply confides his original x_i to a trusted party. Malicious players may instead perfectly coordinate their actions, so as to compute and report to the trusted party alternative inputs x'_j for every malicious player j. The trusted party then evaluates f on all reported inputs, so as to generate a vector of outputs $(y_1, \ldots, y_n)$, and privately hands out y_i to each player i.

A *real evaluation* of f consists of a communication protocol executed by the n players alone: that is, the players (without any external help) evaluate f on their private inputs by exchanging messages back and forth using a specified communication channel. Honest players always send their messages according to their prescribed protocol instructions. Malicious players may instead deviate from their instructions in an arbitrary manner, and even perfectly coordinate their communication strategies before and during the protocol.

Finally, an SFE of f is a real evaluation of f that guarantees the same privacy and correctness as an ideal evaluation. In essence, the ideal/real paradigm consists of *security preservation.* The ideal setting captures the desired correctness and privacy achievable in the presence of malicious players, and SFE (more generally, Secure Computation) guarantees that whatever malicious players could accomplish in a real evaluation they could also accomplish in the ideal one.

Game Theory. In Game Theory, strategic interactions among n players are formally modeled as *games.* In this paper, we are concerned with finite versions of a very general class of *mediated games of incomplete information.* In their normal form, each player has his own private type t_i, where $(t_1, \ldots, t_n)$ are drawn from some publicly known, joint distribution. Players do not communicate with each other —conceptually, they are isolated in separate rooms. Instead, each player i privately sends a report m_i (an element of his finite message space M_i) to a trusted mediator. The mediator then evaluates a specified, possibly probabilistic, finite function on the received reports so as to compute a public outcome y, and each player i's payoff is defined, via his specific utility function u_i, as $u_i(y, t_i)$ —more generally, as $u_i(y, t_1, \ldots, t_n)$.

How will such a game be played? For this as for all other kinds of games, Game Theory predicts that *rational players* will end up in *equilibrium.* Rational players are those always acting to maximize their expected payoffs. An equilibrium essentially consists of an n-tuple of strategies (i.e., ways to choose actions —in the case at hand, reports to be sent to the mediator), such that no player will receive a greater payoff if he deviates from his strategy: provided that all other players stick to theirs (in a *Nash* equilibrium), or no matter what the other players might do (in a *dominant-strategy* equilibrium).[1] But while Game Theory predicts that an equilibrium is how rational players will play, it is agnostic as to which equilibrium will actually be played when a game has multiple equilibria. (In general, this final selection will depend on a variety of exogenous circumstances.)

The Need for a Bridge. Game Theory provides a wonderfully general notion of a mediated game. In practice, however, trusted mediators are hard to find, and Game Theory does not address whether the players alone might be able to replace the trusted mediator, so as to play by themselves.

Secure Computation, on the other hand, provides powerful notions and techniques for replacing trusted parties with specially designed communication among the players alone. As we explain below, however, such replacements only apply to very simple settings: essentially, when there are no payoffs.

Without payoffs, *it is rational to be honest.* In an ideal evaluation where no payoffs are associated to the outputs of f, a player has no incentive to report a "false" private input to the trusted party in the ideal evaluation. This continues to be true if payoffs are assigned but are always the same, or if the payoff structure is sufficiently simple (as in Halpern and Teague [9]). But if general payoffs are associated to f's outputs, then *honesty may be irrational,* both in an ideal and in a real evaluation of f. In an ideal evaluation, rather than honestly reporting his original input, it may be rational for a player to affect f's output by reporting a different input, so as to improve his payoff.[2] Similarly, in a real evaluation, rather than blindly following his prescribed instructions, it is rational for a player to deviate from them when doing so enables him to receive a higher payoff. Of course, honesty is often enforced by morality; but if deviating causes a player to suffer moral agony, then this price should be explicitly modeled in the player's *true* payoffs!

[1]Indeed, there are many possible kinds of equilibria. See [6].

[2]Notice that reporting a false input to the trusted party is rather easy. Since each input is private, no one can tell whether a player has switched his. Even within the constraints of a publicly known distribution for all private inputs, a player would still have substantial freedom in reporting a different input without being caught.

2 Rational Secure Computation

We strongly advocate a new direction for Secure Computation. Currently, Secure Computation does not model incentives for the players. Yet, if its correctness and privacy will used for any real-world purpose (rather than for "pure fun"), then the players will prefer some of its outputs to others, and will thus rationally respond to their incentives. For this reason, having security depend on honesty is dangerous when being honest is being irrational. To be safe,

> *Security must be guaranteed even when all players pursue their selfish interests.*

This is the very goal of our paper. To achieve it, we put forward the new notion of *Rational Secure Computation* (in particular, that of a *Rational SFE*) and provide its first implementation.

The new notion may also be cast in the ideal/real paradigm, by explicitly adding incentives to both settings. In essence, the ideal setting becomes a mediated game Γ and the real one an unmediated game G, and security means that rational players must find the two games totally equivalent.

More specifically, Γ is a finite, normal-form, mediated game of incomplete information. Again, the n players have private types $t_1, \ldots, t_n$, and Γ is played in two stages: first every player i privately communicates a report $m_i \in M_i$ to the mediator; then the mediator announces an outcome y obtained by evaluating a pre-specified function f on the received reports. The payoff of every i is $u_i(y, t_i)$—more generally, $u_i(y, t_1, \ldots, t_n)$. However, since the mediator is trusted but the players are not, for completeness, an outcome must be defined in all contingencies, including when a player *aborts*, that is, acts outside his specified boundaries by not sending any report or by sending a report outside his message space —e.g., an insult to the mediator. We thus assume that, when player i aborts, a predetermined, substantial monetary fine F is imposed on i, and that no information about the players' reports is revealed.[3]

On the other hand, G is an unmediated, *extensive-form* game of incomplete information (having the same players, type sets, outcome set, and utility functions as Γ). The players of G jointly compute y by exchanging messages in several rounds, over a given communication channel. Any player i can abort G's communication prematurely, in which case the same predetermined outcome Y_i is realized as when i aborts in Γ.

[3]Thus aborting is not part of any equilibrium. Without loss of any generality, we can think that the communication with the mediator is sequential, so that it is clear who aborts first.

Because rationality yields equilibria, to capture the intuition that rational players have no preference between playing Γ or G, we make the following

Definition (Informal): *G rationally and securely simulates* Γ if the following two conditions hold:

1. For any equilibrium of Γ, of any strength (e.g., Bayesian-Nash, dominant strategy, ex post, etc.), there exists an equilibrium of G that has the same strength and induces the same payoffs for every player—and vice versa for any equilibrium of G.

2. The privacy of the players in Γ is exactly preserved in G: no group of players (whether collaborating or not) may acquire more information about other players' types or reports in G than they may in Γ.

(As we shall see, our implementation actually achieves a more demanding notion of Rational Secure Computation. However, the above formulation is rather simpler and potentially allows for many more implementations.) Rational Secure Computation is actually achievable in a reasonable communication model. Namely, we show the following

Main Theorem (Informal): Any mediated game of incomplete information Γ can be rationally and securely simulated by a ballot-box game G.

A ballot-box game is an extensive-form game of incomplete information. It includes physical actions such as sealing a message into an envelope, packing up to 5 envelopes into a single, larger one, and randomizing the order of a set of envelopes via a bingo-like device, *the ballot box*. It ends with a special *swap* operation, in which all the players simultaneously exchange specified sets of sealed envelopes. Each player then learns the outcome by opening all the envelopes in his possession.

The ballot box is perhaps Humanity's oldest and best "security assumption." Indeed, it has been relied upon for thousands of years for securely computing a very specific function: the *tally function*.[4] In proving our Main Theorem, we actually show that the security power of the ballot box is *universal*: with minimal changes it can be used to privately and correctly compute any function.

As for other universal results in computation (Recursive Functions, Gödel numberings, etc.) the key to our result is "equating subjects and objects." In our case, we represent data and operations as permutations, physically represented as sequences of envelopes, and then use the ballot box in order to effectively interpret a sequence of envelopes (data) as an algorithm (acting on other data). In order to do so, we shall make fundamental use of the original encoding of [8], indeed its first use after almost 20 years!

Perfection First. Cryptographic notions, such as Zero Knowledge, often possess both perfect and computational versions. In this paper we introduce the perfect version of Rational Secure Computation. That is, each computation is correct with probability 1, and is private in an information-theoretic sense (i.e., it is not affected by the computational power of the players). Furthermore, neither property can be disrupted by any coalition of players, no matter how large. Finally, it enjoys "perfect composibility." Indeed, it satisfies the information-theoretic conditions of Dodis and Micali [5]. Therefore, in the language of Canetti [4], it automatically enjoys universal composibility in the information theoretic setting. In addition, it also enjoys other valuable properties, such as synchronous reducibility. All these composibility properties imply that, although we focus in this extended abstract on simulating "one-shot" mediated games, our notions and techniques generalize to simulating mediated games involving several rounds of interaction with the trusted mediator.

We believe that other perfect implementations of Rational Secure Computation will soon be available. But we also believe that relaxing our stringent security requirements may become necessary to yield still meaningful notions that possess more practical implementations. Let us remark, for instance, that the construction of [13] can yield a computationally bounded version of Rational Secure Computation, albeit in a more "liberal" ideal setting.

[4]To compute $y = tally(b_1, \ldots, b_n) = \sum_i b_i$, each player i seals his own private binary vote b_i into a ballot and inserts it into the ballot-box. After this, the ballot-box "returns a random permutation of the inserted ballots," and all ballots are publicly opened and read, enabling all players to compute y without revealing individual b_i's. When paper was expensive, pebbles and tablets of clay were used in place of envelopes. Ballot-box tallying of votes is the golden standard of secure computation: it works no matter what computational power "bad" players may have, and no matter how many bad players there may be. The worst bad players can do is abstaining from "voting" or "inputting votes different from the ones they originally had in mind." At first glance, however, it would appear that the ballot-box mechanism is so specifically tailored to securely computing the tally function that it cannot be used for securely computing other functions. If $tally(b_1, \ldots, b_n)$ were defined to return only the "majority bit" and not the actual number of votes, then computing it privately and correctly would no longer be as easy.

3 The Intuitive Version of Our Notion

Prior to *informally* sketching what it means for G to simulate Γ rationally and securely, we find it useful to highlight the difficulties that must be addressed.

Two Main Difficulties. First, in Γ each player sends a single message to the mediator, without seeing the messages of other players. Quite differently, in G, players send many messages in response to the messages they receive. One would thus expect that many more equilibria would arise from this much richer structure, making it impossible to deliver on our promised equilibrium-preservation property. Second, our players have private types and it may sometimes be in their interest to *signal* information about their types, as illustrated by the following

Example. Consider a mediated 2-player game Γ of incomplete information, in which (only) Player 1 can be of two types, t_1 or t'_1, each equally likely. There are two possible messages that can be sent by each player: $\{U, D\}$ for Player 1 and $\{L, R\}$ for Player 2. Payoffs of the players depend on Player 1's type and are given by the following two matrices:

t_1	L	R
U	$2,2$	$0,0$
D	$0,0$	$0,0$

t'_1	L	R
U	$0,0$	$0,0$
D	$0,0$	$1,1$

Since the players can send messages only to the mediator, there is no way for Player 2 to know which type of Player 1 he is facing. Thus there are essentially two equilibria here; namely, the following Bayesian-Nash equilibria: (E_1) Player 2 plays L, Player 1 of type t_1 plays U, Player 1 of type t'_1 can play anything; and (E_2) Player 2 plays R, both types of Player 1 play D.[5] Note that equilibrium E_1 is the best for Player 2, and gives him an expected payoff of 1.

For the above example's Γ, our definition in particular requires that any communication game G that rationally and securely simulates Γ must have only two equilibria, equivalent to E_1 and E_2. But, on the surface, this requirement seems impossible to achieve: if Player 1 were allowed to send even a single message to Player 2, he could surely signal the one bit of information corresponding to his type. Such a signal immediately introduces a new equilibrium: namely, depending on whether Player 1 signals t_1 or t'_1, Player 2 respectively plays L or R.[6] This new equilibrium gives Player 2 an expected payoff of 1.5, higher than he could get in any equilibrium of Γ.

[5]If we were to eliminate weakly dominated strategies, only equilibrium E_1 would survive (with Player 1 of type t'_1 playing D).

Two Questions and Two Answers. Before sketching our notion of security and how it addresses the above difficulties, we would like to answer two questions that the above example is likely to raise.

First: *Why should the players not benefit from some additional equilibria?* The answer is that such additional equilibria may be detrimental to "the larger context." For instance, in mechanism design, the whole point is to find a game whose equilibria implement a desired function, typically deemed to be beneficial to Society. If the players were able to introduce additional equilibria that benefit solely them, Society may suffer.

Second: *Can signaling be prevented in G by just restricting the messages players are allowed to send?* The answer is no. Preventing players in G from saying outright "I am of type X" is not enough. In order to guarantee privacy, players' communication strategies in G must be probabilistic (see [7]). But even a minimal amount of entropy in a communication strategy enables *steganography*, that is, subliminal communication between two players that is provably undetectable by any one else (see [1]).

The Essence of Rational Secure Computation. Assume that in Γ the mediator evaluates a function $f : M_1 \times \cdots \times M_n \to Y$. We say that G perfectly simulates Γ if the following properties hold:

1. At any time before G ends, the information available to the players coincides with that available to them before the game begins.
2. G begins with an *input stage* in which each player i, independently, commits to one of his possible inputs. (Any later attempt of i to change his committed input results in an abort.)
3. G is *forced-action*: namely, (1) the players act in predetermined order (by acting out of turn, a player aborts); (2) during the input stage, for any possible input, there is a *single sequence of actions* a player may take for committing it without aborting the game; and (3) after the input stage, when it is a player's turn to act, there is *a single action* he may take without aborting.
4. If no player aborts G then all players receive outputs distributed identically to those produced by a random evaluation of f on the committed inputs.

[6]Notice that since the players' incentives are perfectly aligned, there is no issue of whether Player 2 believes Player 1. Player 1 has no incentive to lie and thus Player 2 can trust the signal.

REMARK. Our forced-action property is *absolutely crucial* to the notion of Rational Secure Computation. It is *very demanding,* but not impossible to implement. Let us clarify that forced-action does not mean that each player's action is predictable by the other players, nor that it is immediately apparent when a player chooses an "alternative action." For example, in a ballot-box game, player i's only "non-aborting" action may consist of sealing a specific value v into a specific envelope E. But the other players may not know v: G's being forced-action only guarantees them that if i does not seal the "right" value, G will eventually abort. By sealing in E a value v' instead, i would be committing to his aborting G, but his abort may become apparent only later on —e.g., when opening E will reveal a content different from that of another, specific envelope.

Let us now sketch how the above properties collectively capture our desired equivalence of Γ and G for rational players.

First of all, notice that, for any player i, aborting in G is fully equivalent to aborting in Γ. Though i may decide to abort at any time, Property 1 guarantees that when making this decision he has absolutely no additional information about other players's types or inputs; nor any idea of what the outcome of the game will be. Thus, i may decide to abort in G with exactly the same information as in Γ. To be sure, i can abort Γ only at the very beginning. In G, instead, he can choose the communication round at which to abort. This choice provides him with the ability to signal information about his own type, if he so desires. But *this signaling does not affect any payoffs*, because whenever i aborts the outcome always is the default value Y_i.

Second of all, notice that for any player i, *provided that no player aborts the game,* sending a report m_i to the mediator in Γ is fully equivalent to committing m_i in the input stage of G. In either case, m_i cannot be changed any more: in Γ because there is no other action available to player i, and in G because of Property 2. Furthermore, in either case, f will be evaluated correctly: in Γ thanks to the trusted mediator, in G thanks to Property 4.

Finally, Property 3 implies that there are no strategies in G available to player i other than "abort" or "commit to an input m_i and then follow the prescribed strategy until the very end." Therefore, as shown above, these strategies perfectly correspond to those available to i in Γ: namely "send a report $\notin M_i$" and "send a report $m_i \in M_i$."

Let us now show how to implement this compelling notion with the help of envelopes and a ballot box.

4 Our Intuitive Communication Model

In this section we *informally* present the communication network in which ballot-box games are played.

Conceptually, we envisage a group of players, seating far apart around a large table, communicating in two ways. The first way is via *broadcasting:* a player stands up and loudly utters a given message. The second way is via identical, opaque *envelopes* (and *super-envelopes*) and a *ballot box.* Informally, a player can choose a message, write it on a piece of paper and seal it into a new, empty envelope. So long as it is not opened, the envelope totally hides and guarantees the integrity of its content. Only the owner of a sealed envelope can open it, either *privately* (in which case —though all other players are aware that he is opening it— he will be the only one to read its content) or *publicly* (in which case all players will learn its content). A player *owns* a sealed envelope if it is physically close to him. By definition, the player originally sealing a new envelope owns it. After that, ownership of an envelope can be transferred to another player by tossing it to him.

A player can also publicly put up to 5 of his envelopes into a new super-envelope E, in which case none of them can be opened before E. All super-envelopes are again opaque and identical to each other, but have a slightly larger size than (ordinary) envelopes. A super-envelope E thus tightly packs its *sub-envelopes,* so that their relative order (counting —say— from E's front) does not change when E is moved about. The rules of ownership for super-envelopes are the same as for envelopes. There is only one possible way for a player i to open a super-envelope E of his: all players observe that i has opened E, and E's sub-envelopes become "exposed" again, and can thus be manipulated (e.g., opened or transferred) individually. Such sub-envelopes always guarantee the integrity and privacy of their contents.

Envelopes and super-envelopes always stay above the table and their transfers are always tracked by the players. The players can thus "mentally assign" to each envelope or super-envelope an identifier, j, insensitive to any possible change of ownership. The only exception is when a player i publicly puts some of his envelopes or super-envelopes into a ballot box: when they are taken out, their contents will remain unchanged and private, but their identities are randomly permuted, in a way that is unpredictable to all players. We shall use the ballot box to permute either (1) only envelopes, or (2) only super-envelopes with the same number of envelopes inside.

5 Structure of Our Protocol

The high-level structure of our protocol is essentially that of [8], with minimal adjustments demanded by our communication model. It is only in the implementation of this high-level structure that we make essential use of our ballot box and depart from [8].

Computing with Permutations. The following is our rendition of Barrington's way [2] to compute with permutations in S_5, the symmetric group of 5 elements.

SIX CONSTANTS. There exist six permutations in S_5, denoted by $Id, a, b, [a \rightarrow b], [a^{-1} \rightarrow a]$, and $[aba^{-1}b^{-1} \rightarrow a]$ which satisfy the following properties:

- Id is the identity permutation.
- $aba^{-1}b^{-1} \neq Id$.
- $[a \rightarrow b]^{-1}a[a \rightarrow b] = b$.
- $[a^{-1} \rightarrow a]^{-1}a^{-1}[a^{-1} \rightarrow a] = a$.
- $[aba^{-1}b^{-1} \rightarrow a]^{-1}aba^{-1}b^{-1}[aba^{-1}b^{-1} \rightarrow a] = a$.

Proof: $a = 1\,2\,4\,5\,3$, $b = 2\,5\,3\,4\,1$, $[a \rightarrow b] = 3\,4\,1\,2\,5$, $[a^{-1} \rightarrow a] = 1\,2\,4\,5\,3$, and $[aba^{-1}b^{-1} \rightarrow a] = 1\,3\,2\,4\,5$.

THREE OPERATORS. Let " — " , " ′ " and " ∗ " be operators on S_5 defined as follows: for any $\sigma \in S_5$,

- $\overline{\sigma} = [a \rightarrow b]^{-1}\sigma[a \rightarrow b]$
- $\sigma' = [aba^{-1}b^{-1} \rightarrow a]^{-1}\sigma[aba^{-1}b^{-1} \rightarrow a]$
- $\sigma^* = [a^{-1} \rightarrow a]^{-1}\sigma[a^{-1} \rightarrow a]^{-1}$

BIT REPRESENTATION. 0 is represented by Id, and 1 by a.

AND AND NOT. If σ_1 and σ_2 represent bits b_1 and b_2, respectively, then

$$\neg b_1 = (\sigma_1 a^{-1})^*$$

$$b_1 \wedge b_2 = (\sigma_1\overline{\sigma_2}\sigma_1^{-1}\overline{\sigma_2^{-1}})'$$

Enveloped Permutations. We need a way to encode permutations in S_5 for use in our communication model.

If $\sigma \in S_5$, its corresponding *enveloped permutation*, denoted by the bold character $\boldsymbol{\sigma}$, is a sequence of five envelopes whose contents are, respectively, $\sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5)$. Viceversa, if $\boldsymbol{\sigma}$ is an enveloped permutation, we denote by σ its corresponding permutation.

Circuits and Permutation-Labeling. We assume that the finite function to be evaluated is represented as a combinatorial logic circuit, $C : (\Sigma_k)^n \rightarrow \Sigma_k$.

Evaluating C on specific inputs $x_1, \ldots, x_n$ yields a specific *bit value* for each wire w of C. Whenever the inputs in question are clear, we simply denote w's bit value by $bit(w)$.

A *permutation-labeling* of C is a function mapping each wire w of C to a label $(\vec{P}, p)$, where $p \in S_n$ is a permutation of the n players and $\vec{P}$ is a sequence of n permutations in S_5: $\vec{P} = (P_1, \ldots, P_n) \in (S_5)^n$.

A permutation labeling of C is *valid*, relative to inputs $x_1, \ldots, x_n$, if the label $(\vec{P}, p)$ of each wire w satisfies the following

FUNDAMENTAL INVARIANT:

1. The permutations $\vec{P}$ are random and $(n-1)$-wise independent elements of S_5.
2. The product $P_1 \cdots P_n$ equals permutation a if $bit(w) = 1$ and Id otherwise.
3. If w is not an input wire for Player i, then Player i knows permutation $P_{p(i)}$ and has no information about any permutation $P_{p(j)}$ for all $j \neq i$. If w is an input wire for Player i, then Player i knows all permutations $P_1, \ldots, P_n$.

Overview of the Joint Computation. For all possible private inputs $x_1, \ldots, x_n$, the players compute a valid permutation labeling of C by the following conceptual steps.

1. *Initial Step.* For each input wire w of Player j, Player j computes a valid label $(\vec{P}, p)$, and gives to each player i the enveloped permutation $\boldsymbol{P}_{p(i)}$.

2. *Inductive Step.* For each gate g, and for each output wire w of g, the players use their enveloped permutations for g's input wire(s) to jointly compute enveloped permutations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$ such that each player i owns $\boldsymbol{P}_{p(i)}$, and $(\vec{P}, p)$ is a valid label for w.

3. *Final Step*
After processing all gates as in the Inductive Step, each output wire w has a valid label $(\vec{P}, p)$ such that the product $P_1 \cdots P_n$ is Id if $bitw = 0$, and a otherwise. However, each enveloped permutations $\boldsymbol{P}_j$ belongs to a different player. Therefore in the final step, the players swap envelopes such that for all output wires w, if $bit(w)$ is a bit of Player i's output then Player i now owns the enveloped permutations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$. After the swap, he can open all relevant envelopes to learn $P_1, \ldots, P_n$ and thus compute the bit carried by w.[7]

[7]This envelope swap is a somewhat unusual operation in requiring a simultaneous transfer of envelopes among multiple players. Nonetheless it is simple, and is a publicly observable operation: indeed anyone can observe whether all envelopes have been transferred to the proper recipients prior to be opened. As such, swaps can be realized in at least three ways. First, swaps could be performed by a physical device. One can imagine, for example, an n player version of the "revolving door style"-devices

6 Implementing The Initial Step

In this extended abstract we lack sufficient space to implement in detail the above high-level structure. Therefore, we have chosen to discuss informally our techniques for implementing just the Initial Step. The Inductive Step is somewhat harder, but relies on similar ideas. The Final Step is instead quite straightforward.

A Naive Approach. Assume that input wire w belongs to Player i. Naively, one may think that i may randomly generate a valid label for w as follows. First, he randomly selects permutations $P_1, \ldots, P_{n-1}$. Second, he computes P_n so that $P_1 \cdots P_n = a$ if his input bit is 1 and Id otherwise. Third, he prepares enveloped permutations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$. Fourth, he gives a suitable zero-knowledge proof that indeed the product of their underlying permutations is either Id or a. Fifth, he transfers each enveloped permutation $\boldsymbol{P}_j$ to Player j. Finally, each player j opens (and reseals) $\boldsymbol{P}_j$, to learn P_j and fulfill the last requirement of the Fundamental Invariant.

This simple procedure, however, suffers of several drawbacks. The most important one is that the above procedure is not forced action. (In particular, Player i has total freedom in selecting P_j and therefore — since there are 120 permutations in $\mathcal{S}_5$— he could trivially signal $\log 120 \approx 6$ bits of information to Player j through his selection of P_j. In addition, through a traditional zero-knowledge proof, Player i has the opportunity of additional random choices, which he can use to steganographically signal even more information to other players.) Furthermore, the above procedure is not correct with probability 1, as required by our perfect notion of security. Indeed, a traditional zero-knowledge proof admits a positive, though negligible, soundness error.

(The reader should realize that using the ballot box to choose $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-1}$ and then computing $\boldsymbol{P}_n$ so that $P_1 \cdots P_n = a$ or Id, is equally naive, though for more subtle reasons!)

employed in cash-for-goods exchanges in banks, gas-stations, etc. Second, swaps could be performed by a referee. That is, any agent that can perform public actions and is fully accountable for such actions. This is very different from a mediator who is trusted to perform secret computations on private data, because a referee is caught whenever he deviates, he could be punished for breach of contract by a court system. Thirdly, swaps could be handled by incentives. For instance, a weaker version of swap —suitable for many applications— could be implemented by having all players give their to-be-transferred sets of envelopes to a designated player i, and then instructing i to distribute such sets to the proper recipients without opening any envelope. If a punishing outcome for Player i is automatically realized whenever i does not implement the swap correctly, then it will be in i's interest to properly transfer the sets of envelopes.

Input Commitment. To avoid the above drawbacks, we have Player i perform the Initial Step for input wire w as follows. First, Player i must commit to his input bit b. He does this by privately creating an enveloped permutation ρ such that $\rho = Id$ if his bit is 0 and $\rho = a$ otherwise. Player i then lays these five envelopes on the table where he cannot change them.

Shortly, Player i will need a second copy of $\boldsymbol{\rho}$. Therefore, he *clones* $\boldsymbol{\rho}$ as follows. First, he publicly creates three copies of $\mathbf{Id}$: $\mathbf{Id}_1$, $\mathbf{Id}_2$ and $\mathbf{Id}_3$. (That is, he publicly writes the values "1", "2", "3", "4" and "5" on five pieces of paper and seals them, in order, into a sequence of five envelopes. He then does the same for a.) Next, he creates five new super envelopes so that super envelope k contains the kth envelopes of $\mathbf{Id}_1$, $\mathbf{Id}_2$ and $\mathbf{Id}_3$. He places these five super envelopes into the ballot-box and "shakes it" so as to randomly reorder the five super envelops. Next, Player i opens the five super envelopes, and from their exposed sub-envelopes he obtains three new enveloped permutations, $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3$, as follows: the kth envelope of every $\boldsymbol{\sigma}_j$ is one of the three sub-envelopes of the kth super envelope returned by the ballot box. Notice that by using the ballot box in this fashion, three properties hold: (1) everyone is guaranteed that $\sigma_1 = \sigma_2 = \sigma_3 = \sigma$, (2) σ is a random element of S_5, and (3) no one — not even i — knows σ. At this point, Player i privately opens, reads and reseals each envelope of $\boldsymbol{\sigma}_3$ so that now he is the only one to know σ. He then i computes and broadcasts π, a permutation in S_5 such that $\pi\sigma = \rho$. Note that since σ is random and independent of ρ, so is π. Therefore, broadcasting π provides no information about ρ —and thus about i's private input bit b— to any other player. Finally, Player i publicly reorders the envelopes of $\boldsymbol{\sigma}_1$, $\boldsymbol{\sigma}_2$ and $\boldsymbol{\sigma}_3$ according to π so as to obtain three new enveloped permutations $\boldsymbol{\tau}_1$, $\boldsymbol{\tau}_2$ and $\boldsymbol{\tau}_3$. All other players monitor that each $\boldsymbol{\sigma}_j$ is indeed reordered according to the broadcast permutation π and thus all other players are guaranteed that $\tau_1 = \tau_2 = \tau_3 = \pi\sigma = \tau$. However, they have no reason to believe that $\tau = \rho$. Therefore, Player i proves $\tau = \rho$ in "perfect zero-knowledge" as follows. First, Player i creates five super envelopes such that the k^{th} super envelope contains the k^{th} envelope of $\boldsymbol{\tau}_3$ and the k^{th} envelope of $\boldsymbol{\rho}$. Second, he uses the ballot box to randomly reorder these five super envelopes. Finally, he publicly opens each super envelope E returned by the ballot box, and then E's two sub-envelopes so as to publicly show that they contain a pair of equal values. This concludes, i's cloning of $\boldsymbol{\rho}$: Everyone knows that $\tau_1 = \tau_2 = \rho$, but nobody has gained any information about ρ. In particular, therefore, no one has evidence that ρ is actually equal to a or Id. We shall denote the two clones produced by this procedure by $\boldsymbol{\rho}$ and $\boldsymbol{\rho}'$.

Player i proves, in "perfect zero-knowledge", that "$\rho = a$ or $\rho = Id$" as follows. First, he publicly creates two enveloped permutations: $\mathbf{Id}$ and $\boldsymbol{a}$. Next, he packs the five envelopes of $\mathbf{Id}$, in order, into a single super envelope and similarly he puts the envelopes of $\boldsymbol{a}$, in order, into a second super envelope. Next, he places both super envelopes into the ballot-box, that returns them in random order. Player i then opens both returned super envelopes to expose two enveloped permutations, call them $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, such that no one knows which is $\boldsymbol{a}$ and which is $\mathbf{Id}$. At this point, Player i privately opens, reads and reseals each envelope of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, so he is now the only one who knows which is a and which is Id.[8] Without loss of generality assume that Player i's committed permutation ρ is equal to α. Player i then announces that $\rho = \alpha$ and proves it in "perfect zero-knowledge" as when proving $\tau = \rho$. At this point, everyone knows that $\rho = \alpha$ and everyone knows that α is either Id or a, therefore, Player i has successfully proven that ρ properly encodes a bit.

Permutation Labeling. At this point, Player i must compute from ρ, in a forced-action manner, enveloped permutations $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$ such that $(\vec{P}, Id_n)$ is a valid label of his input wire w, where $\vec{P} = P_1, \ldots, P_n$ and Id_n is the identity of S_n. Doing so requires several steps.

To begin with, for $j = 1$ to $n-1$, Player i generates $\boldsymbol{P}_j$ by publicly creating an enveloped permutation $\mathbf{Id}$ and then using the ballot box to randomly permute these five envelopes. Next, Player i privately opens, reads and reseals each envelope of $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-1}$ so as to learn $P_1, \ldots, P_{n-1}$. Finally, he computes P_n so that $P_1 \cdots P_n = \rho$ and generates the enveloped permutation $\boldsymbol{P}_n$.

At this point, all other players are convinced that $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-1}$ are enveloped permutations, but they do not know whether $\boldsymbol{P}_n$ is an enveloped permutation, much less that it is the particular enveloped permutation such that $P_1 \cdots P_n = \rho$. Player i in essence provides a "perfect zero-knowledge proof" of both facts as follows. First, for each j, he uses the same cloning procedure of Input Commitment to produce a clone $\boldsymbol{P}'_j$ of $\boldsymbol{P}_j$. (Notice that, if $\boldsymbol{P}_n$ is not an enveloped permutation, then our cloning procedure is guaranteed to abort. Thus, if Player i succeeds in cloning every $\boldsymbol{P}_j$, he has *a fortiori* proved in perfect zero-knowledge that $\boldsymbol{P}_n$ is an enveloped permutation.) Next, Player i uses $n-1$ times the following procedure (for multiplying two enveloped permutations) to produce $\boldsymbol{P}$ from $\boldsymbol{P}'_1$, ..., $\boldsymbol{P}'_n$.

Procedure EnvMult. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be two enveloped permutations owned by and known to the same player i. To produce an enveloped permutation $\boldsymbol{\gamma}$ and prove in perfect zero knowledge that $\gamma = \alpha\beta$, Player i acts as follows. First, he publicly creates an enveloped permutation $\mathbf{Id}$. Next he creates five new super envelopes, $E_1, \ldots, E_5$. He packs into each E_k a pair of envelopes: the first is the kth envelope of $\mathbf{Id}$, and the second is the kth envelope of $\boldsymbol{\beta}$. He then uses the ballot box to obtain $E'_1, \ldots, E'_5$, a random reordering of $E_1, \ldots, E_5$. Next, he publicly opens $E'_1, \ldots, E'_5$, and organizes the "exposed" envelopes into two new enveloped permutations, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$: the kth envelope of $\boldsymbol{\mu}$ (respectively, $\boldsymbol{\nu}$) is the first (respectively, second) envelope from E'_k. Therefore, everyone is guaranteed that μ is a random element of S_5 (unknown to everyone), and that $\nu = \mu\beta$. At this point, Player i privately opens, reads and reseals each envelope of $\boldsymbol{\mu}$, thus becoming the only player to know μ. Then, he computes and broadcasts π, a permutation in S_5 such that $\pi\mu = \alpha$. Note that since μ is random and independent of α and β, so is π; thus broadcasting π provides no additional information about α or β. Finally, Player i publicly reorders the envelopes of $\boldsymbol{\mu}$ according to π so as to obtain a new enveloped permutation $\boldsymbol{\phi}$. Similarly, he reorders the envelopes of $\boldsymbol{\nu}$ according to π to obtain $\boldsymbol{\gamma}$. All other players monitor that both $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are indeed reordered according to the broadcast permutation π, and are thus guaranteed that $\gamma = \pi\nu$ and $\phi = \pi\mu$. Therefore, since $\nu = \mu\beta$, all players know that $\gamma = \phi\beta$. However, all players other than i have no reason to believe that $\phi = \alpha$. Therefore, Player i gives a perfect zero-knowledge proof of this fact using the same "prove equal" procedure of Input Commitment. After this proof all players are convinced that $\gamma = \alpha\beta$ as desired.

After Player i finishes using EnvMult $n-1$ times, all players are guaranteed that $P = P'_1 \cdots P'_n$, and yet gain no additional information about any P'_j. Because each $\boldsymbol{P}'_j$ is a clone of $\boldsymbol{P}_j$, they are also guaranteed that indeed $P = P_1 \cdots P_n$.

At this point, Player i proves in perfect zero knowledge that $\boldsymbol{P} = \boldsymbol{\rho}'$, thus convincing all other players that the product of $P_1, \ldots, P_n$ is indeed ρ.

Finally, Player i transfers each enveloped permutation $\boldsymbol{P}_j$ to Player j, who privately opens, reads and reseals each of its envelope to learn P_j. Thus, $(\vec{P} = (P_1, \ldots, P_n), Id_n)$ is a valid permutation labeling of w.

[8]Even without the ability to reseal an envelope, we could still achieve Rational Secure Computation. However, doing so would require making additional copies of every enveloped permutation in our protocol. We find this makes describing our protocol cumbersome and so for convenience we assume that envelopes can be resealed.

7 Implications for Mechanism Design

A game can be viewed as having two parts: *a context* and *a mechanism.* Informally, the context describes the n players, their type space $T = T_1 \times \cdots \times T_n$, the joint distribution p over T, the set of possible outcomes Y, and the utility function $u_i : T \times Y \to \mathbb{R}$ for each player i. The mechanism describes the set of pure strategies available to the players and the function g that selects the outcome of the game based on the strategies actually chosen by the players.

Ultimately, in a game G, the outcome of an equilibrium E is a (probabilistic) function of the players' actual types, $t_1, \ldots, t_n$. Letting μ be such a function, we say that E *implements* μ. We further say that G *fully implements* μ if all equilibria of G implement μ.

A fundamental problem of Mechanism Design is the following: Given a context C and a *social-choice* function μ, find a mechanism M such that the resulting game G (fully) implements μ.[9]

In essence, the designer seeks to ensure that the players find it in their own interest to reach the outcomes he desires: if he successfully "extends" a context C to a game G, then, by playing rationally (i.e., at equilibrium), the players produce outcomes that depend on their private types in the desired manner (i.e., as specified by μ). Given this goal, it is no surprise that Mechanism Design is crucial to so many human activities! Allocation of private goods, provision of public goods, design of markets (e.g., of eBay auctions), of voting procedures, social planning, etc., all stand to benefit from it.[10]

Notice that, though the distribution of players' types is widely known, only the players themselves know their actual types. Therefore, not even a dictator could select and impose an outcome $y = \mu(t_1, \ldots, t_n)$, unless the players truthfully reveal $t_1, \ldots, t_n$. Thus, it is unsurprising that Mechanism Design—ensuring that the socially desired outcomes arise solely by exploiting the players' self interest—is very hard.

Simplifying Mechanism Design. We guarantee that, given a context C, it suffices for a mechanism designer to find just an *ideal mechanism*, that is one that can count on the help of a trusted mediator. Our perfect simulation of ideal games straightforwardly implies that such an ideal mechanism can automatically be transformed to an equivalent "ordinary" one, that still fully implements the desired μ. Since designing ideal mechanisms is certainly easier, our result simplifies the general Mechanism-Design problem.

In addition, whenever multiple equilibria are not a concern, our result fully dispenses with the need for a mechanism designer. Let us explain. The famous *Revelation Principle* tells us that, if any given μ is implementable at all, then it is implementable by a very simple ideal mechanism: the *Direct Mechanism* D_μ in which every player i *truthfully* reports his actual type t_i to the trusted mediator, who then announces $y = \mu(t_1, \ldots, t_n)$. Our construction automatically generates an ordinary mechanism implementing μ, if any such mechanism exists, by replacing the mediator of D_μ. Unfortunately, this may not be a full implementation of μ, because D_μ may have additional equilibria not implementing μ. However, if μ is *dominant-strategy incentive compatible* (i.e., if truth-telling is a dominant-strategy equilibrium), then the presence of additional equilibria becomes "practically irrelevant:" if a player i has a strategy that is best for him no matter what other players might do, he has no reason to play any other strategy! Dominant-strategy equilibria are arguably the strongest equilibria, and the ones most sought after in Mechanism Design. Indeed, the problem of Mechanism Design is often defined in terms of dominant-strategy equilibria. We thus make Mechanism Design fully automatic in a very important special case.

Privacy and Mechanism Transparency. We hold these truths to be self evident: (1) People care about Privacy; (2) Ideal mechanisms preserve Privacy better than any other mechanisms; and (3) Our construction preserves perfectly the Privacy of ideal mechanisms.

Privacy is indeed a crucial aspect to our simulation of ideal games. Beyond simplifying the design of future mechanisms, we also "increase the Privacy of existing mechanisms." Let us explain.

In the context of an auction of a single item, the most popular social-choice function is the famous *second-price* function. In essence, this is the function μ demanding that the player who most values the item should win it, and that the price he should pay is the second highest valuation. A typical mechanism for implementing μ consists of having the players seal their bids into envelopes, and then publicly opening all of them. The winner is the player with the highest bid, and the price is the second highest bid. Note that, for this simple mechanism, choosing a bid coinciding with your own private valuation for the item is a dominant-strategy equilibrium. Note too that this simple mechanism is essentially unmediated. Unfortunately, at equi-

[9]The term "social-choice function" derives from the fact that, typically, the designer wishes to produce outcomes that are desirable for Society at large.

[10]For an introduction to the subject see [12] and [11], for an excellent overview see [16]. On mechanism design in incomplete information environments see [14] and [10].

librium, it obliges all the players to reveal their private types (in this context, their private valuations of the item for sale). Thus, some players might be reluctant to use such a mechanism.

An alternative, but mediated, mechanism for the same context is the following: the players give their sealed bids to the trusted mediator (i.e., the auctioneer), who then privately opens them and announces the outcome (i.e., winner and price) together with all losing bids (in a random order). Though mediated, this mechanism is reasonably "transparent." That is, it does not totally rely on the honesty of the mediator: anyone whose losing bid is not announced will surely cry foul. However, it still causes a substantial privacy loss, and (since the winner's bid is not announced) it enables a dishonest auctioneer to collude with a given player and declare him the winner, regardless of his bid.

Another mediated mechanism is that in which the players submit sealed bids to the mediator, who then only announces the winner and the price (i.e., the second highest bid). Clearly, this mechanism enjoys the maximum possible privacy, but it is the least transparent of the three, and is rarely used in practice [15].

Perhaps, our instinctive preference for more transparent mechanisms might result in choosing mechanisms and social-choice functions that tend to reveal more information than strictly necessary. In any case, our result succeeds in "disentangling Trust and Privacy" in auctions and all other contexts. *It dispenses with the need to trade-off between the amount of information that is publicly revealed and the amount of trust bestowed on the mechanism.* In particular, we provide optimal solutions to various problems in *secure auctions*, that have recently attracted a lot of attention (in particular, see [3] and references therein). For instance, our construction enables the players *alone* to simulate perfectly the auctioneer of the second mediated mechanism discussed above. Thus, *we implement the efficient second-price social-choice function with both maximum privacy and maximum transparency.*

More generally, our result enables us to implement social-choice functions with "private outcomes." In particular, assuming that actual transactions can be kept private, we can implement second-price auctions where the losers learn only that they have lost, the winner learns that he has won and at what price, and the seller learns only the winner and the price.

Privacy and Modular Mechanism Design. Privacy has not received explicit attention in Mechanism Design. In Game Theory in general, information loss is studied only as far as it is relevant for the game at hand, under the assumption that its players will not interact again. In our eyes, *Privacy is a strategic interest of the players.* The players' desire for Privacy stems from their utilitarian concerns about *future* interactions. In a forthcoming paper, we generalize our present results so as to simulate perfectly any *sequence of (correlated) mediated games.* Such more general simulations will enable us to achieve *Modular Mechanism Design.*

Modularity —in essence, the ability of breaking the design of complex systems into a multiplicity of simpler components— is central to Science, Engineering, and many other human activities, but was so far unavailable to Mechanism Design.

References

[1] L. von Ahn, N. Hopper and J. Langford. Provably secure steganography. In *Crypto '02*, 2002.

[2] D.A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proceedings of STOC '86.* ACM, 1986.

[3] F. Brandt and T. Sandholm. (Im)possibility of unconditionally privacy-preserving auctions. In *Proc of 3rd AAMAS.* ACM, 2004.

[4] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc of 42nd FOCS*, 2001.

[5] Y. Dodis and S. Micali. Parallel reducibility for information-theoretically secure computation. In Crypto '00, 2000.

[6] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, Massachusetts, 1991.

[7] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28(2), 1984.

[8] O. Goldreich, S. Micali and A. Wigderson. How to play any mental game. In *Proc. of STOC '87.* ACM, 1987.

[9] J. Halpern and V. Teague. Rational secret sharing and multiparty computation. In *Proc of 36th STOC*, 2004.

[10] M. O. Jackson. Bayesian implementation. *Econometrica*, 59(2):461–477, 1991.

[11] M. O. Jackson. Mechanism Theory. *Optimization and Operations Research.* EOLSS Publishers, Oxford, UK, 2003.

[12] M. O. Jackson. A crash course in implementation theory. *Social Choice and Welfare*, 18(4):655—708, 2001.

[13] M. Lepinski, S. Micali and A. Shelat. Collusion-free protocols. In *Proceedings of STOC '05*, 2005.

[14] T. R. Palfrey, and S. Srivastava. *Bayesian Implementation.* Harwood Academic Publishers, Chur, Switzerland, 1993.

[15] M. H. Rothkopf, T. J. Teisberg and E. P. Kahn. Why are Vickrey Auctions Rare? *Journal of Political Economy*, 98:94–109, 1990.

[16] T. Sjöström and E. Maskin. Implementation Theory. *Handbook of Social Choice and Welfare.* 237–288, North-Holland, 2002.

[17] A. Yao. Protocols for secure computations. In *Proc. of FOCS '82.* IEEE, 1982.