Perfect Implementation

By Sergei Izmalkov, Matt Lepinski, and Silvio Micali*

January 7, 2010

Abstract

Privacy and trust affect our strategic thinking, yet they have not been precisely modeled in mechanism design. In settings of incomplete information, traditional implementations of a normal-form mechanism —by disregarding the players' privacy, or assuming trust in a mediator— may fail to reach the mechanism's objectives. We thus investigate implementations of a new type.

We put forward the notion of a perfect implementation of a normal-form mechanism $\mathcal{M}$: in essence, a concrete extensive-form mechanism exactly preserving all strategic properties of $\mathcal{M}$, *without relying on a trusted mediator or violating the privacy of the players.*

We prove that *any* normal-form mechanism can be perfectly implemented by a *verifiable mediator* using envelopes and an envelope-randomizing device (i.e., the same tools used for running fair lotteries or tallying secret votes). Differently from a trusted mediator, a verifiable one only performs prescribed public actions, so that everyone can verify that he is acting properly, and that he never learns any information that should remain private.

1 Introduction

A game consists of a *context* —describing the outcomes, the players (including their types and beliefs), and the players' preferences over the outcomes— and a *mechanism* (or a game form), describing the actions available to the players, and the way in which actions lead to outcomes. The usual goal of mechanism design consists of finding a mechanism that, for a given class of contexts, defines a game whose equilibria yield a desired outcome. While contexts can be arbitrarily complex, mechanism design strives to find simple mechanisms. A *normal-form* mechanism consists of a set of *reports* (or messages) for each player, and an *outcome function* mapping these reports to outcomes.

Normal-form mechanisms can enjoy valuable theoretical properties. As defined, however, they are *abstractions*. Outcome functions do not spontaneously evaluate themselves on players' reports. To be useful in real strategic settings, normal-form mechanisms are concretely implemented with the help of a *mediator*. But such concrete implementations present theoretical difficulties that are far from being understood. Indeed, we argue that

> *Unless they satisfy several new goals,* not achieved and often not addressed so far, *concrete mediated implementations of normal-form mechanisms not only are not very meaningful, but fail to enjoy the very properties achieved by their abstract counterparts.*

We contend that without a rigorous and sufficiently powerful theory of concrete implementation, the practical meaningfulness of normal-form mechanisms is at risk. Providing such a theory is the very purpose of this paper. Let us now articulate our two main goals.

*We would like to thank participants to numerous seminars, and especially Ran Canetti, Dino Gerardi, Sergiu Hart, Bart Lipman, and Dave Parkes, for their comments. We are grateful to the NSF grant SES-0551244 for financial support.

1.1 The Goal of Maximum Privacy (and Minimum Trust)

Among auctions in the private-value setting, the famous second-price sealed-bid mechanism is a normal-form mechanism whose reports consist of numerical bids —one for each player— and whose outcome function returns the player with the highest bid as the *winner,* and the value of the second-highest bid as the *price.* The characteristic property of this mechanism is *economic efficiency achieved in dominant strategies.* Indeed, because it is optimal for each player to report his true valuation no matter what the others may report, one can expect that the player with the highest valuation wins the item for sale. We shall use this mechanism to illustrate the problems of privacy and trust when abstract normal-form mechanisms are concretely implemented with the help of a mediator.

Trusted vs. Verifiable Mediators. Consider the following two mediated implementations, $\mathcal{M}_1$ and $\mathcal{M}_2$, of the second-price sealed-bid mechanism.

$\mathcal{M}_1$ *(With a Trusted Mediator)*: The players seal their bids into envelopes and hand them to the mediator, who then privately opens them, privately evaluates the outcome function, and finally publicly announces the winner and the price.

$\mathcal{M}_2$ *(With a Verifiable Mediator)*: The players seal their bids into envelopes and hand them to the mediator, who then publicly opens all of them, so as to enable everyone to compute the winner and the price.

Notice that the mediators of these two implementations are asked to perform different types of actions. The mediator of $\mathcal{M}_1$ is asked to perform *private actions*, so that only he knows whether he has performed the right actions. By contrast, the mediator of $\mathcal{M}_2$ only performs *public actions*, so that everyone can verify that the right actions have been performed. Accordingly, these two implementations are at opposite ends with respect to privacy and trust. On the one extreme, implementation $\mathcal{M}_1$ reveals nothing more than the announced outcome, but requires total trust in the mediator. Indeed, the players cannot verify whether the mediator announces the correct outcome, nor whether he will keep all bids secret after the auction is over.[1] On the other extreme, $\mathcal{M}_2$ guarantees the correctness of the outcome, but makes the entire players' reports public knowledge. In sum, $\mathcal{M}_1$ requires total trust and offers total privacy, while $\mathcal{M}_2$ requires no trust and offers no privacy.

To be sure, a verifiable mediator is still trusted, but in a different, "public" sense: he is trusted to really perform whatever public actions he is asked to perform. For instance, in implementation $\mathcal{M}_2$ he is trusted not to open just the first half of the envelopes and destroy the other half. Clearly such trust is much milder: because any deviation from prescribed public actions is immediately observed, a public mediator can be kept accountable.

Privacy and Trust as Strategic Problems. Privacy-valuing players may avoid participating in implementation $\mathcal{M}_2$, and distrustful players in implementation $\mathcal{M}_1$. Such reduced participation not only causes the seller to fetch lower prices, but negatively affects economic efficiency. Indeed, the requirement that, in an auction, "the item for sale be won by the player who values it the most" applies to all potential bidders, not just the ones trusting a given mediator. Thus, no implementation that deters some players from bidding can be economically efficient in a general sense. In addition, whenever (for lack of other ways of getting a desired item) distrustful players participate in $\mathcal{M}_1$ or privacy-valuing players participate in $\mathcal{M}_2$, *neither mechanism can be economically efficient, even with respect to just the participating players.* Let us explain.

- In $\mathcal{M}_1$, a player who *truly distrusts* the mediator may fear that the price will always be artificially close to the winner's bid.[2] Such a player will not find it optimal to report his true valuation to the mediator;

[1] According to Rothkopf, Teisberg and Kahn (1990), such concerns explain why second-price auctions are rarely used in practice.

[2] If player i bids \$1000 dollars and the mediator announces that i is the winner and must pay \$999 dollars, it is impossible for i to know if there was really a bid of \$999. Certainly, the mediator has incentives to manipulate the outcome (e.g., if he receives a percentage of the generated revenue for his services) and to betray a player's secret (e.g., if he is bribed for disclosing it).

instead, he will have a strong incentive to "underbid." Thus, the item for sale might very well go to another player, who values it less but trusts the mediator more. Therefore, concrete implementation $\mathcal{M}_1$ may not be efficient.

- In $\mathcal{M}_2$, a player *truly valuing* the privacy of his valuation receives, *by definition,* some negative utility from its publicity. But then the only way for him to prevent his true valuation from becoming public is to bid a different value, perhaps in a randomized fashion. This distortion may make concrete implementation $\mathcal{M}_2$ inefficient as well.

More generally, privacy and trust affect the players' strategic thinking, making it unclear whether mediated implementations satisfy the strategic properties of the abstract mechanisms they purportedly implement.

Novelty and Relative Nature of the Problem. In the second-price auction context, to preserve $\mathcal{M}_2$'s economic efficiency, one may be tempted to handle privacy-valuing players by (a) monetizing —somehow— privacy loss; (b) revising the players' valuations accordingly; and then (c) relying on the traditional Vickrey machinery. In so doing, however, the resulting auction would at best be efficient in the "revised" valuations, while Society's interest is that it be economically efficient in the "original" valuations. Integrating privacy and strategic concerns in the design of concrete implementations of normal-form mechanisms is a *new* problem, and requires new techniques and conceptual frameworks.

At a closer look, not even the *abstract* second-price auction itself may be perfectly efficient in a private-value setting. For instance, even in a magic world where the outcome function evaluates itself on players' reports, everyone is bound to learn that the winner's valuation is higher than the price; and this small loss of the winner's privacy may suffice to miss the target of efficiency in its purest form. But if the abstract second-price auction itself does not achieve perfect efficiency, what goal can we set for its concrete implementations? We answer this question with the strongest possible relativism:

> A concrete implementation of the second-price mechanism should always enjoy economic efficiency to exactly the same degree (whatever it may be) as the abstract mechanism.

More generally, any normal-form mechanism implies some loss of privacy. This is the case because the outcome itself (being a function of the players' reports) contains information about these reports. We regard this loss of privacy (which potentially affects all kinds of strategic concerns) as *inherent*, and do not wish to rectify or modify this —or any other— property of a normal-form mechanism. In sum, we advocate that

> *A mediated implementation of a normal-form mechanism $\mathcal{M}$ should preserve exactly all of $\mathcal{M}$'s theoretical properties (including its privacy properties!) without trusting the mediator at all.*

1.2 The Goal of Strategic Equivalence

Even with respect to strategic aspects alone, we wish to "raise the bar" for concrete implementation. Mechanisms are typically designed to guarantee a given property *at equilibrium*, and equilibria are typically defined with respect to "single-player deviations". Accordingly, researchers have often considered the following minimal requirement for a concrete implementation $\mathcal{M}'$ of an abstract mechanism $\mathcal{M}$: "ensuring that $\mathcal{M}'$ possesses an equilibrium E' equivalent to a given mechanism E of $\mathcal{M}$." Notably, this is the very requirement of all prior *pre-play* implementations of *correlated equilibrium*. Unfortunately, such a notion of implementation is quite weak: since $\mathcal{M}'$ is free to introduce additional equilibria (that is, equilibria not having any counterparts in $\mathcal{M}$), a play of $\mathcal{M}'$ may be quite different from a play of $\mathcal{M}$.

A more demanding notion (often referred to as "full implementation") requires that each equilibrium of $\mathcal{M}'$ is pay-off equivalent to some equilibrium of $\mathcal{M}$, and viceversa. However, even when $\mathcal{M}'$ fully implements $\mathcal{M}$, a play of $\mathcal{M}$ may still be dramatically different from a play of $\mathcal{M}'$. A main reason for this is due to collusion. In particular, one may not worry about collusion in the normal-form mechanism $\mathcal{M}$, because the players by definition do not have the means to coordinate their strategies. By contrast, $\mathcal{M}'$ may provide two or more players with a golden opportunity to collude successfully. And if such players have the means

and the incentives to jointly deviate from their equilibrium strategies, the actual play of $\mathcal{M}'$ may not be an equilibrium at all. So, beyond just preserving all equilibria, we advocate that

> *A concrete implementation of a normal-form mechanism should not give any additional power to any subset of players.*

1.3 The Goal of Efficiency Preservation

Consider the following mediated implementation of a *deterministic* two-player normal-form mechanism, whose outcome function g has k-bit inputs. The implementation aims at maximizing privacy while minimizing trust.

Concrete Implementation *Naive*

1. A mediator publicly generates a $2^k \times 2^k$ matrix of envelopes as follows. For each possible pair of reports (m_i, m_j), a mediator publicly seals $g(m_i, m_j)$ into a separate envelope, and makes it the (i, j) item of the matrix. The mediator then hands the entire "envelope matrix" to Player 1.
2. Player 1 secretly permutes the rows, so that the jth row of envelopes becomes the first row if his secret report is j, and hands the permuted matrix to Player 2.
3. Player 2 secretly permutes the columns, so that his intended report corresponds to the first column, and hands the further permuted matrix to the mediator.
4. The mediator publicly opens the top-left corner envelope of the matrix to reveal the outcome.

Although conceptually very simple and relying on a verifiable mediator, *Naive* requires 2^{2k} envelopes and a matching number of envelope operations, whether or not g is easy to compute. When $k = 150$ (less than the number of bits required to describe the players' "types" in many practical contexts), even though g could be computed in, say, $2k = 300$ bit operations, *Naive* requires 2^{300} envelopes. Since 2^{300} is higher than the total number of elementary particles in the universe, there simply is not enough "matter" to manufacture so many envelopes. This efficiency loss is typical of most pre-play implementations, as they treat an outcome function g as the *ordered list* of all its possible outputs, rather than as an *algorithm*. Treating functions this way is as conceptually trivial as operationally hopeless. Without ruling out such concrete implementations, any theory of concrete implementation would remain just ... "theory."

What to do? Of course, no mechanism $\mathcal{M}'$ concretely implementing an abstract normal-form mechanism $\mathcal{M}$ can hope to be more efficient than $\mathcal{M}$, but we should prevent it to be much worse. That is, once more, we have a "relative" goal: namely,

$\mathcal{M}'$ should preserve $\mathcal{M}$'s efficiency, whatever it may be, as much as possible.

1.4 The (Informal) Notion of a Perfect Implementation

Let us now explain how our notion of a perfect implementation dovetails with our goals.

Informally, we say that an (abstract) normal-form mechanism $\mathcal{M}$ is perfectly implemented by a (concrete) mechanism $\mathcal{M}'$ with a verifiable mediator (i.e., one taking only public actions, so that all can verify that he takes the right ones), if the following properties hold:

- *Strategy Equivalence:* There exist an outcome-preserving bijection between the players' strategies in $\mathcal{M}$ and the players' strategies in $\mathcal{M}'$.
- *Privacy Equivalence:* For any strategy profile σ of $\mathcal{M}$, and any subset of the players, the information learnable by these players in an execution of $\mathcal{M}$ under σ coincides with that learnable by the same players in an execution of $\mathcal{M}'$ under the corresponding strategy profile σ'.
- *Complexity Equivalence:* There exists an integer constant $c > 0$ such that, if the outcome function of $\mathcal{M}$ requires K bit-operations to be evaluated, then an execution of $\mathcal{M}'$ requires at most $c \cdot K$ elementary operations.

Consequently, from a strategy/privacy/complexity perspective, individual players as well as arbitrary subsets of the players are indifferent, for any context $\mathcal{C}$, between playing the games $G = (\mathcal{C}, \mathcal{M})$ and $G' = (\mathcal{C}, \mathcal{M}')$. In particular, the two games enjoy identical solutions. Indeed, although $\mathcal{M}'$ may be of extensive-form, strategy equivalence states that $\mathcal{M}$ and $\mathcal{M}'$ have identical —up to renaming and reordering of strategies— normal-form representations. Of course, all traditional equilibrium concepts (such as Nash, Bayesian, dominant strategy, ex-post, undominated Nash, and trembling-hand Nash equilibria) and set-valued solution concepts (such as rationalizability and iterated elimination of dominated strategies) are invariant to isomorphic transformations of normal-forms. In particular, therefore, perfect implementations preserve all traditional equilibria and set-valued solutions.

1.5 A Macroscopic View of Our Construction

We constructively prove that every normal-form mechanism $\mathcal{M}$ is perfectly implemented by a concrete mechanism $\mathcal{M}'$ with a verifiable mediator. Let us provide here the very high-level view of our construction.

Our Physical Operations. Mechanism $\mathcal{M}'$ works on *ballots* of one of two sizes. Ballots of the smaller size are also called "envelopes," and ballots of the bigger size are also called "super-envelopes." Envelopes are indistinguishable among each other, and super-envelopes are undistinguishable among each other. (Envelopes are quite commonly used in the pre-play literature, and super-envelopes have been used by Ben-Porath (1998) and Krishna (2006).) In our case, an envelope contains an integer between 1 and 5, while a super-envelope contains at most 5 envelopes. Each ballot preserves both the integrity and secrecy of its content.

For such ballots we envisage traditional elementary operations (such as making a ballot, opening it, and destroying it), but we add the following one that indeed characterizes or construction:

Randomly permuting a given sequence of ballots (via a bingo-like machine which we call the ballot box*).*

A 3-Stage Construction. Our construction consists of 3 stages.

1. *The input Stage.* Here each player simultaneously hands over to the verifiable mediator a sequence of envelopes whose contents encode —in a very special way— his intended message.
2. *The computation stage.* Here the verifiable mediator performs a uniquely determined sequence of public operations on the received sequences of envelopes so as to (a) obtain a final sequence of envelopes whose contents are guaranteed to encode the correct outcome, and (b) reveal absolutely nothing about the original messages of the players.
3. *The output stage.* Here the verifiable mediator opens all final envelopes so as to enable everyone to learn what the outcome is (and solely that).

Thus a perfect implementation enjoy the same "logical structure" of the traditional interaction of the players with a trusted mediator, but avoids the correctness, privacy, and strategy problems of the latter interaction. This logical structure in particular implies that the players cannot signal. Indeed, after they simultaneously submit the envelopes containing their messages, the take no further actions.

1.6 A Microscopic View of Our Construction

The above high-level description of our construction suffices to get an idea of how strategy and privacy equivalence might be achieved, but it does not provide any clue about complexity equivalence. Let us now fill this gap.

A Complexity Theoretic Fact. It is well known —for instance, see Goldreich (2008), Section 1.2.4.1— that each probabilistic finite function f can be computed by a finite sequence of NOT, AND, $DUPLICATE$ and $COIN$ operations, where NOT maps a bit b to the bit $\bar{b} = 1 - b$; AND maps two bits b_1 and b_2 to

1 if $b_1 = b_2$ and to 0 otherwise; $DUPLICATE$ maps a bit b to the pair of bits (b, b); and $COIN$ is the elementary probabilistic function that, on no input, generates a random bit b.

Accordingly, f is efficient if it is computable by a sufficiently short sequence of NOT, AND, $DUPLICATE$ and $COIN$ operations. (Disregarding constant factors, f's efficiency does not depend on this specific "basis" of elementary functions.[3])

Our Use of This Fact. For each elementary function $X \in \{NOT, AND, DUPLICATE, COIN\}$, we prove that there exists an "enveloped version" of X. For example, in the case of AND, we show that there exists a sequence S_{AND} of verifiable ballot operations satisfying the following properties. Given two sequences of 5 envelopes each, the contents of the first encoding a secret bit b_1 and the contents of the second encoding a secret bit b_2, the ballot operations of S_{AND} progressively transform these initial 10 envelopes into a final sequence of just 5 envelopes whose contents are guaranteed to encode the bit $b_1 \wedge b_2$. Moreover, although they include publicly opening some envelopes, the ballot operations in S_{AND} do not reveal any information about b_1 and/or b_2. Constructing the "enveloped versions" of our four bit-functions is the heart of our construction. The rest is easy. In fact, the input, computation, and final stages of our construction can be conceptually expanded as follows:

1. Assume that in the normal-form mechanism $\mathcal{M}$, each player i wishes to select a binary strategy (i.e., string) s_i. Then, in the input stage of $\mathcal{M}'$, for each bit b in s_i, player i hands to the verifiable mediator a sequence of 5 envelopes whose content encodes b.
2. Let f be the outcome function of mechanism $\mathcal{M}$ and $\bar{f}$ a sequence of NOT, AND, $DUPLICATE$, and $COIN$ operations computing f. Then, in the computation stage of $\mathcal{M}'$, the mediator publicly performs the verifiable ballot operations of the "enveloped version" of each elementary function X in $\bar{f}$, each time making sure to use the proper envelopes. At the end, therefore, for each bit c in in $f(s_1, \ldots, s_n)$, the mediator has verifiably produced a sequence of 5 envelopes whose content is guaranteed to be c. At the same time, since no enveloped version of our basic operations reveals any information about its inputs or its outputs, no information about the strategy profile $(s_1, \ldots, s_n)$ is revealed.
3. In the final stage, the verifiable mediator publicly opens all final envelopes, so that, after decoding every bit, every one learns the desired (and correct) outcome $f(s_1, \ldots, s_n)$.

Intuitive Achievement of Strategy, Privacy, and Complexity Equivalence. Assume, without loss of generality as we shall argue, that in the normal-form mechanism $\mathcal{M}$ we want to implement each player strategy consists of a k-bit string. Then, in our construction, each player delivers to the verifiable mediator $5k$ sealed envelopes. Let us now sketch why our construction is a perfect implementation of $\mathcal{M}$.

- Strategic equivalence holds for the following intuitive reason. In our construction, the strategy of a player i consists solely of the contents of his $5k$ envelopes. In fact, no player takes any action after the input stage. Assume now that each player i hands to the mediator $5k$ envelopes correctly encoding a k-bit string. Then, there obviously is a one-to-one correspondence between the strategies of i in $\mathcal{M}$ and in our construction.

  Note. Realistically, however, nothing prevents a player i to put into his $5k$ envelopes contents that do not correspond to any encoding to a k-bit string, so that i also has a much larger number of "illegal" strategies. But we shall prove —in section 3.2— that such behavior can be formally and practically dealt with. In essence, our final construction ensures that any illegal strategy of a player i can be detected without compromising the privacy of any other player.

[3] Of course, rather than NOT, AND, $DUPLICATE$, and $COIN$, one may choose a different finite basis B of elementary functions from which to compute any finite function f. A proper choice of B may reduce the number of new elementary operations required to compute a given f, but not by more than a constant factor C. In fact, by themselves being a basis, NOT, AND, $DUPLICATE$, and $COIN$ can be used to implement any function in B, and thus C is just the minimum number of NOTs, ANDs, $DUPLICATE$s, or $COIN$s sufficient to implement any function in B.

- Privacy equivalence holds for three reasons. First, no privacy is lost in the input stage, because each player i is the only one who knows the contents of his own envelopes: any one else only sees $5k$ identical envelopes. Second, in the computation stage the verifiable mediator publicly executes a sequence of enveloped versions of NOT, AND, $DUPLICATE$, and $COIN$ operations, and each one of them (as we shall prove) reveals no information about its inputs and/or outputs. Third, in the output stage the mediator publicly opens the final envelopes, but these (as we shall prove) only contain an encoding of the proper outcome, that needs to be learned by definition.

- Complexity equivalence holds because each enveloped version of NOT, AND, $DUPLICATE$, and $COIN$ consists of a fixed number of elementary ballot operations, and thus if the outcome function can be computed by k of our bit functions, our construction requires at most $O(k)$ elementary ballot operations.

1.7 Pros and Cons of the Ballot Box

Our construction can be viewed as replacing trust in a mediator with trust in a ballot randomizer. It should be appreciated that this is a considerable advantage, not only from a theoretical point of view (the one relevant for this paper), but also from a *practical* point of view. Indeed, trusting a mediator requires trusting a lot of different things. To begin with, we must trust that he is actually capable of correctly performing a prescribed computation on the data he receives. Next, we must trust that he is capable of performing his computations in a private environment. (This actually entails a plethora of separate trusts: for instance, (1) if he uses pen and paper, we must trust that he does not perform his computations next to a window; (2) if he uses a computer, we must trust that he uses a computer that is virus-free and not connected to the Internet; and (3) we must trust that he is capable of identifying and erasing all tracks of his computations —hitting the "delete key" on a computer hardly qualifies. The list could actually go on.) Finally, we must trust that until his death he will never reveal the data he learned. By comparison, trusting a ballot randomizing device is a much simpler conceptual requirement. Moreover, from a practical point of view, plenty of players routinely trust such devices. Indeed high-stake lotteries routinely use bingo-like machines to select winners, and casinos routinely use card-shufflers in high-stake card games.[4]

On the "cons" side, the physical nature of the ballot box requires the players to be physically present in the same location. This said, we note that this same requirement arises in any implementation requiring envelopes —and thus in most of the pre-play literature— since the players must watch that no one sneakily opens them.[5]

1.8 Relation to Prior Work

Pre-Play. Pre-play aims at constructing a concrete communication game G', using a suitable communication channel, having an equilibrium payoff-equivalent to a special equilibrium of an abstract game G. (Much of the pre-play literature has been devoted to implementing correlated equilibrium, see in particular the works of Bárány (1992), Forges (1990), Ben-Porath (1998), Aumann and Hart (2003), Urbano and Vila (2002), Ben-Porath (2003), Gerardi (2004), Dodis, Halevi and Rabin (2000), Gerardi and Myerson (2007), Krishna (2006).) Pre-play, therefore, does not aim at nor achieves the strategic equivalence of G and G'. (It is thus not surprising that G' may end up having equilibria with no counter-parts in G.)

[4]Note that cards essentially are ballots with an identical upper side, safekeeping a given value on their other side.

[5]In addition, to be overly precise, physical presence cannot be easily dismissed. In particular, the abstract notion of a normal-form mechanism requires the final outcome y to be common knowledge, and broadcasting y —e.g., via e-mail or via radio— does not quite work if the goal is to avoid trust. In fact a player receiving y via e-mail does not know for sure that the others received y too. Even receiving y by radio one may worry to be the single target of a rather "dedicated" broadcast. In the end, witnessing y in the presence of the other players may remain the best practical approximation of making y common knowledge.

Zero Knowledge and Secure Computation. The study of privacy and correctness without trust started two decades ago, in theoretical computer science, with the zero-knowledge proofs of Goldwasser, Micali and Rackoff (1985). Closer to our concerns is *secure computation*, as introduced by Goldreich, Micali and Wigderson (1987), improving on earlier results of Yao (1986). In essence, a secure computation of a finite function f with n inputs consists of a communication protocol P_f such that, *whenever the majority of n players honestly stick to P_f's instructions, f* can be evaluated by the players alone so as to match the privacy and correctness achievable with the help of a trusted mediator. The problem, however, is that P_f makes any subset of the players with cardinality $> n/2$ "omnipowerful," that is capable of arbitrarily and undetectably forcing the output of f to be any value in f's range of their choice.[6] Thus implementing the outcome function f of a normal-form mechanism with P_f does not, in particular, satisfy strategic equivalence.

Spreading trust. Some works, in particular that of Naor, Pinkas and Sumner (1999), rather than putting trust on a single mediator, distribute it onto multiple mediators. (Thus, correctness and privacy hold only in so far these mediators do not collude with each other, nor signal information that they are not supposed to.) By contrast, our emphasis is on *removing all trusted parties.*

Impossibility results. Whether or not a trusted mediator can be replaced by an unmediated interaction of the players alone crucially depends on the means of interaction available to the players. Because such a replacement is counter-intuitive, we informally expect it to be impossible in most interaction models. Indeed, this replacement has been proved impossible, in a formal sense, in many specific interaction models, even for a restricted class of contexts and outcome functions. Notably, Aumann and Hart (2003) prove that two players cannot reach any non-trivial *correlated equilibrium* via "cheap talk," so long as the players communicate essentially by broadcasting messages. Brandt and Sandholm (2004) argue the impossibility of "unconditionally privacy-preserving second-price auctions" in many interaction models. By contrast, we prove that *there exists a reasonable model of interaction* (via ballots and a ballot box) in which dispensing with a trusted mediator is possible *for all outcome functions and all contexts.*

2 Verifiable Ballot Computation

In this section we show that a verifiable mediator can, given a sequence of envelopes containing by hypothesis (the encoding of) a secret input to a function g, produce a sequence of envelopes guaranteed to contain (the encoding of) g's correct output, without anyone learning anything about g's inputs or outputs.

2.1 Working with Physical Ballots

Ballots. We envisage the players and the verifiable mediator to seat around a sufficiently large table, and having at their disposal a sufficiently large number of initially empty *ballots* of two kinds: *envelopes* and *super-envelopes.* Externally, all ballots of the same kind are identical, but super-envelopes are slightly larger than envelopes. Each envelope may contain an integer between 1 and 5, and each super-envelope may contain a sequence of at most 5 envelopes. Once made, that is, filled with a proper content for the first time, an envelope perfectly hides and guarantees the integrity of the integer it contains, until it is opened. Once made, a super-envelope tightly packs the envelopes it contains, and thus keeps them in the same order in which they were inserted.

The *content of a sequence of envelopes* $E = E_1, \ldots, E_k$ is defined to be be the concatenation of the contents of all E_i. By saying that the string $c_1 \cdots c_k$ is the content of E we imply that c_i is the integer contained in E_i.

[6] A weaker notion of secure computation, also introduced by Goldreich et al. (1987), does not any subset of $< n$ players to control the outcome, but restricts the players to efficient computation only, and enable a deviating player in P_f to be the only one to learn the correct output of f.

Ready Ballots. A properly made ballot ready to be operated upon is placed on a portion of the table never previously occupied, where it can be observed by everyone. Each time that a ballot becomes ready for the first time, or ready again, it automatically receives a new, unique, and public identifier. Such an identifier can be thought of as a specification of the ballot position on the table.

Note that only ballots directly in touch with the table surface are considered ready. If inside a super-envelope S on the table, an envelope cannot be directly operated upon, and does not have its own identifier. But, once S is opened, each of its inner envelopes is put back on the table, receives a new identifier, and becomes ready once more.[7]

Ballot Operations. We need 5 types of (atomic) operations on ready ballots.

1. Publicly make a sequence of 5 new envelopes $E_1, \ldots, E_5$ with content $c_1 \cdots c_5$.
   (Each such envelope E_i is now ready to be operated upon.)
2. Publicly open a sequence of 5 envelopes $E_1, \ldots, E_5$ so as to reveal its content $c_1 \cdots c_5$ to all players.
   (Each such E_i is no longer ready. We refer to $c_1 \cdots c_5$ as a *public record*.)
3. Publicly make a new super-envelope S containing a sequence $E_1, \ldots, E_k$ of envelopes, where $2 \leq k \leq 5$.
   (S is now ready, but $E_1, \ldots, E_k$ no longer are.)
4. Publicly open a super-envelope S to expose its inner envelopes $E_1, \ldots, E_k$.
   (S is no longer ready, but each E_i now is.)
5. *Ballot-box* a sequence $B_1, \ldots, B_k$ of at most 5 ballots of the same kind to obtain the sequence of ballots $B'_1 \ldots, B'_k$.
   (Essentially, there exists a secret —randomly selected by Nature— bijection β between $B_1, \ldots, B_k$ and $B'_1 \ldots, B'_k$. The content of the former B_i and the current $\beta(B_i)$ coincide for each i. The ballots $B_1, \ldots, B_k$ cease to be ready, while $B'_1 \ldots, B'_k$ become ready.)

For simplicity only, we further envisage three additional types of operations on ready ballots:

6. Publicly reorder a sequence of 5 ballots $B_1, \ldots, B_5$ of the same kind to get the new ready ballots $B'_1, \ldots, B'_5$.
   (Essentially, the positions on the table of the ballots B_i are publicly switched. Since in effect these ballots are not becoming ready or ready again just now, they do not receive new identifiers. Rather, their identifiers are "renamed" via a publicly chosen permutation π.)
7. Publicly destroy a ballot B.
   (B, and if it is a super-envelope each of its inner envelopes, no longer is ready and never will be.)
8. Do nothing.
   (All ballots, their contents, and their identifiers remain the same.)

The *identifier* of any operation O includes O's type and the sequence of the (identifiers of) the ballots O acts upon. If O is of type 1, O's identifier further includes the content $c_1 \cdots c_5$ of the new sequence of envelopes; if it is of type 6, the permutation π. (If O is of type-2, its public record is an "effect," not an identifier.)

Operations of type 1, 2, and 6 involve exactly 5 ballots only to simplify describing our constructions.

2.2 Verifiable Ballot Mediators and Verifiable Ballot Computers

A verifiable ballot mediator essentially is an elementary program to be run on a sequence of envelopes. This program is "straight-line," in the sense that it does not make use of loops (such as "until X is true, do Y"):

[7] A ballot B can be formalized as a triple (i, b, c); where i, B's identifier, is a positive integer; b, B's kind, is a bit (0 in case of an envelope, and 1 in case of a super-envelope); and c, B's content, is an integer between 1 and 5 if B is an envelope, and a sequence of envelope identifiers if B is a super-envelope. Our ballot operations can be formalized as acting on a proper set of such triplets, a sequence of public records, and a sequence of secret records, with the help of a few global variables —such as the maximum identifier in existence so far. We prefer however to stick to a physically intuitive treatment of ballots and ballot operations in this already quite technical paper.

it consists of a fixed-length sequence of verifiable ballot operations, each chosen based on the contents of the envelopes opened so far. A verifiable ballot mediator thus is a rather syntactic object. A verifiable ballot computer is instead a verifiable ballot mediator satisfying a semantic constraint: namely, given a sequence on envelopes containing a string x, it produces a sequence of envelopes containing a string y guaranteed to be a pre-specified function of x. Let us now be more formal.

Definition 1. *A (m-operation)* verifiable ballot mediator *M is a sequence of m functions, $M = f_1, \ldots, f_m$, where each f_i, on inputs*

1. *(the identifiers of) a sequence S of ready ballots;*
2. *a sequence E of ballot identifiers; and*
3. *a (possibly empty) string R of public records,*

outputs (the identifier of) a ballot operation O_i involving solely ballots in S.

Letting E correspond to a sequence of initially ready envelopes, M is executed as follows: for $i = 1, \ldots, m$,

(a) Set $O_i = f_i(S_i, E, R)$, where

- *S_i coincides with E if $i = 1$, and otherwise with the currently ready ballots —in the order in which they have become ready— that*
  - *(a) have become ready after the performance of the ballot operation O_1, or*
  - *(b) have identifiers in E.*
- *R is the sequence of public records generated by performing $O_1, \ldots, O_{i-1}$*

(b) Perform O_i on its specified ballots.

We say that such an M is *p-to-q* if, at the end of any execution of M in which the envelope sequence E has length p, all the (currently) ready ballots that have been generated by M consist solely of envelopes, and these, considered in the order in which they have been made ready, form a *final sequence* of length q.

We say that a verifiable ballot mediator M is *no-ballot-left-behind* for envelope sequences of length x if, at the end of any execution of M on a sequence E of x envelopes, none of the (currently) ready ballots has been generated by M.

Remarks.

- Any spectator can verify whether the execution of a verifiable ballot mediator M on a sequence of envelopes E has been properly carried out. Indeed, at any point in time, any spectator knows which types of operations have been performed on which ballots, and which public records (if any) have been produced. And from all such information he can easily compute what the next operation of M should be.[8]
- A verifiable ballot mediator may have multiple executions. Let us explain. If the operation returned by one of its functions f_i consists of ballot-boxing k ballots, then there are $k!$ ways of reordering the given ballots. Although no visible difference can be immediately detected, the execution may continue in visibly different ways. For instance, if the k ballots in question were envelopes, and some of these envelopes are later on publicly opened, then it is possible for different public records to be observed. And different public records may, in turn, cause different operations to be subsequently executed.
- A verifiable ballot mediator, executed of a sequence of envelopes E part of a larger sequence of ballots B, will never touch a ballot of B that is not in E. This property facilitate the modular composition of such mediators, on which we indeed rely in our construction.

[8]The fact that no one observes the permutation actually involved in a ballot-box operation is not a problem. Since a ballot-box is assume to be an act of Nature, to verify that a sequence of ballots has been randomly permuted it suffices to observe that the proper ballots have been inserted in the bingo-like machine, and that this has been "cranked up" the prescribed number of times.

Definition 2. *Let $f : X^s \to Y$ be a probabilistic finite function (where $X, Y \subset \{1, \ldots, 5\}^k$ for some integer $k > 0$), and M a sk-to-k verifiable ballot mediator. We say that M is a* verifiable ballot computer for *f if there exists a distribution $\mathbb{D}$ of sequences of public records such that, in a random execution of M on a sequence S of sk envelopes, the first k containing $x_1 \in X$, the second k containing $x_2 \in X$, and so on, the following properties hold:*

- Correctness. *The content of M's final sequence of k envelopes is distributed as in $f(x_1, \ldots, x_s)$; and*
- Privacy. *The sequence of public records generated by M is distributed as in $\mathbb{D}$.*

The correctness property guarantees that, "under the envelopes' cover, M properly computes f all the time." In particular, consider the case when f is deterministic. In this case, fixing $(x_1, \ldots, x_s) \in X^s$, there is a single value $y = f(x_1, \ldots, x_s)$. Yet, a verifiable ballot computer M for such f may still have multiple executions (because M may need —and in fact *must*, in order to satisfy the privacy property— use the ballot box). But if even a single one of these executions did not have y as the content of the final sequence of envelopes F, the distribution of F's content could not coincide with "y with probability 1."

On the other hand, the privacy property guarantees that no information about the content of the envelope sequence S may be revealed. In fact, since M is fixed and public knowledge, the only information that may be gained in a random execution of M consists of the sequence of public records produced. But this information is randomly drawn according to the same, fixed distribution D, independently of which element in X^s the sequence of envelopes S may contain. In other words, the privacy property guarantees that all information revealed during M's correct computation is "random and independent noise."

Note that M does not have any correctness/privacy constraints when S's content does not belong to X^s.

2.3 Verifiable Ballot Computers for Three Special Functions

Our Representation of Permutations of 5 Elements. The symmetric group of 5 elements, $\mathbb{S}_5$, is central to our construction. If $\sigma \in \mathbb{S}_5$, then we identify the permutation σ with the string of integers $\sigma(1) \cdots \sigma(5)$. (For example, the string 13542 is identified with the permutation mapping 1 to 1, 2 to 3, 3 to 5, 4 to 4, and 5 to 2; in symbols, $1 \to 1, 2 \to 3, 3 \to 5, 4 \to 4, 5 \to 2$.) Accordingly, 12345 is the identity element of $\mathbb{S}_5$. If σ and τ are members of $\mathbb{S}_5$, then by $\sigma\tau$ we denote the product of σ and τ, that is, the permutation mapping each integer $i \in \{1, \ldots, 5\}$ to $\sigma(\tau(i))$.

In this subsection we show that verifiable ballot computers indeed exist for the following 3 functions:

1. *Permutation Inverse,* mapping a permutation $p \in \mathbb{S}_5$ to p^{-1}.
2. *Permutation Product,* mapping a pair of permutations $(p, q) \in \mathbb{S}_5 \times \mathbb{S}_5$ to pq; and
3. *Permutation Clone,* mapping a permutation $p \in \mathbb{S}_5$ to the pair of permutations (p, p).

Lemma 1. *There exists a 14-operation verifiable ballot computer for permutation inverse.*

Proof. Consider the following verifiable ballot mediator INV. Given a sequence of envelopes $A_1, \ldots, A_5$:

(1) Publicly make a sequence of 5 envelopes $B_1, \ldots, B_5$ whose content is 12345.
(2) For $\ell = 1$ to 5: make a new super-envelope S_ℓ containing the pair of envelopes (A_ℓ, B_ℓ).
(3) Ballot-box the sequence of super-envelopes $S_1, \ldots, S_5$ to obtain the new sequence $S'_1, \ldots, S'_5$.
(4) For $\ell = 1$ to 5: open super-envelope S'_ℓ to expose the envelope pair (A'_ℓ, B'_ℓ).
(5) Publicly open the sequence of envelopes $A'_1, \ldots, A'_5$ to reveal the content α.
(6) Publicly reorder $B'_1, \ldots, B'_5$ according to α^{-1}, if $\alpha \in \mathbb{S}_5$, and according to the identity permutation otherwise, to produce the final envelope sequence $B''_1, \ldots, B''_5$.

Clearly INV is a 14-operation 5-to-5 verifiable ballot mediator. (Only for clarity is INV described by 6 conceptual steps rather than a sequence of 14 functions, $f_1, \ldots, f_{14}$.) Let us now prove that it is a verifiable ballot computer for permutation inverse as desired. Assume that $A_1, \ldots, A_5$ contain a permutation $\sigma \in \mathbb{S}_5$.

By definition, the sequence of envelopes $S'_1, \ldots, S'_5$ is obtained in Step 3 by permuting the super-envelopes $S_1, \ldots, S_5$ of Step 2 according to a random and secret permutation $\rho \in \mathbb{S}_5$. "Because inside the bingo-like machine the inner envelopes A_i and B_i traveled together within S_i (keeping their original relative order)," once the super-envelopes S'_i are opened in Step 4, we deduce that the sequences of envelopes $A'_1, \ldots, A'_5$ and $B'_1, \ldots, B'_5$ have been obtained by reordering the respective envelope sequences $A_1, \ldots, A_5$ and $B_1, \ldots, B_5$ according to the *same* random and secret permutation ρ. Since the content of $B_1, \ldots, B_5$ originally was the identity permutation 12345, the content of $B'_1, \ldots, B'_5$ necessarily is the permutation ρ. And since the content of $A_1, \ldots, A_5$ originally was, by hypothesis, the permutation σ, the content of $A'_1, \ldots, A'_5$ necessarily is the permutation $\rho\sigma$; that is, $\alpha = \rho\sigma$. Since $\rho\sigma$ is the only public record of an entire execution of INV, and since ρ is a random and secret permutation, then, no matter what σ might be, this public record consists of a randomly and independently selected permutation in $\mathbb{S}_5$. Thus, letting $\mathbb{D}$ be the uniform distribution over $\mathbb{S}_5$, it is easily seen that INV enjoys the required privacy property of a ballot computer. Finally, because the sequence of envelopes $F_1, \ldots, F_5$ has been obtained by publicly permuting $B'_1, \ldots, B'_5$ according to $\alpha^{-1} = (\rho\sigma)^{-1} = \sigma^{-1}\rho^{-1}$, the content of INV's final sequence of evenvelopes $B''_1, \ldots, B''_5$ necessarily is $\sigma^{-1}\rho^{-1}\rho = \sigma^{-1}$. Thus INV enjoys the required correctness property of a verifiable ballot computer for permutation inverse.

Lemma 2. *There exists a 27-operation verifiable ballot computer for permutation product.*

Proof. Consider the following verifiable ballot mediator $MULT$. Given a sequence of 10 envelopes, where $A_1, \ldots, A_5$ are the first 5 envelopes and $B_1, \ldots, B_5$ the next five:

(1) Execute the ballot computer INV (of the proof of Lemma 1) on $A_1, \ldots, A_5$ so as to obtain a final sequence of envelopes $C_1, \ldots, C_5$ (which therefore are guaranteed to contain a permutation in $\mathbb{S}_5$).

(2) For $\ell = 1$ to 5: make a new super-envelope S_ℓ containing the pair of envelopes (B_ℓ, C_ℓ).

(3) Ballot-box the sequence of super-envelopes $S_1, \ldots, S_5$ to obtain the new sequence $S'_1, \ldots, S'_5$.

(4) For $\ell = 1$ to 5: open super-envelope S'_ℓ to expose the envelope pair (B'_ℓ, C'_ℓ).

(5) Publicly open the sequence of envelopes $C'_1, \ldots, C'_5$ to reveal the content $\alpha \in \mathbb{S}_5$.

(6) Publicly reorder $B'_1, \ldots, B'_5$ according to α^{-1} to produce the "output" envelope sequence $B''_1, \ldots, B''_5$.

It is clear that $MULT$ is a 27-operation 10-to-5 verifiable ballot mediator. (Only for clarity is $MULT$ described by 6 conceptual steps rather than a sequence of 27functions, $f_1, \ldots, f_{27}$.) Let us now prove that it is a verifiable ballot computer for permutation product as desired. Assume that the content of $A_1, \ldots, A_5$ is $\sigma \in \mathbb{S}_5$ while that of $B_1, \ldots, B_5$ is $\tau \in \mathbb{S}_5$. Then by construction and by Lemma 1, at the end of Step 1 the sequence of envelopes $C_1, \ldots, C_5$ contains the permutation σ^{-1}. By definition, the sequence of envelopes $S'_1, \ldots, S'_5$ is obtained in Step 3 by permuting the super-envelopes $S_1, \ldots, S_5$ of Step 2 according to a *new*, random, independent, and secret permutation $\rho \in \mathbb{S}_5$. Thus, also the sequences of envelopes $B'_1, \ldots, B'_5$ and $C'_1, \ldots, C'_5$ have been obtained by reordering the respective envelope sequences $B_1, \ldots, B_5$ and $C_1, \ldots, C_5$ according to ρ. Accordingly, the content of $B'_1, \ldots, B'_5$ necessarily is the permutation $\rho\tau$ and the content of $C'_1, \ldots, C'_5$ necessarily is the permutation $\rho\sigma^{-1} = \alpha$. The latter permutation is revealed in Step 4, and actually is the only public record generated by $MULT$ in Steps 2 through 6. This public record is thus a permutation randomly and independently selected in $\mathbb{S}_5$. In particular, it is independent of the only other public record of an execution of $MULT$: the single public record of the execution of INV of Step 1 (which itself was randomly and independently selected member of $\mathbb{S}_5$). Thus, letting $\mathbb{D}$ be the uniform distribution over $\mathbb{S}_5$, $MULT$ enjoys the required privacy property. Finally, because $B'_1, \ldots, B'_5$ contained $\rho\tau$ and have been reorder in Step 6 according to $\alpha^{-1} = (\rho\sigma^{-1})^{-1} = \sigma\rho^{-1}$, the content of $MULT$'s final sequence of envelopes $B''_1, \ldots, B''_5$ necessarily is $\sigma\rho^{-1}\rho\tau = \sigma\tau$, as required for $MULT$ to enjoy the correctness property.

Lemma 3. *There exists a 30-operation verifiable ballot computer for permutation clone.*

Proof. Consider the following verifiable ballot mediator $CLONE$. Given a sequence of envelopes $A_1, \ldots, A_5$,

(1) Execute the verifiable ballot computer INV (of the proof of Lemma 1) on $A_1, \ldots, A_5$ so as to obtain a final sequence of envelopes $B_1, \ldots, B_5$ (which therefore contains a permutation in $\mathbb{S}_5$).

(2) Publicly make two sequence of 5 envelopes $C_1, \ldots, C_5$ and $D_1, \ldots, D_5$ with content 12345.

(3) For $\ell = 1$ to 5: make a new super-envelope S_ℓ containing the triple of envelopes (B_ℓ, C_ℓ, D_ℓ).

(4) Ballot-box the sequence of super-envelopes $S_1, \ldots, S_5$ to obtain the new sequence $S'_1, \ldots, S'_5$.

(5) For $\ell = 1$ to 5: open super-envelope S'_ℓ to expose the triple of envelopes pair $(B'_\ell, C'_\ell, D'_\ell)$.

(6) Publicly open the sequence of envelopes $D'_1, \ldots, D'_5$ to reveal the content $\delta \in \mathbb{S}_5$.

(7) Publicly reorder $B'_1, \ldots, B'_5$ and $C'_1, \ldots, C'_5$ according to δ^{-1} to produce the two final envelope sequences $B''_1, \ldots, B''_5$ and $C''_1, \ldots, C''_5$.

It is clear that $CLONE$ is a 30-operation 5-to-10 verifiable ballot mediator. (Only for clarity is $CLONE$ described by 7 conceptual steps rather than a sequence of 30 functions, $f_1, \ldots, f_{30}$.) Let us now prove that it is a verifiable ballot computer for permutation clone as desired. Assume that $A_1, \ldots, A_5$ contained $\sigma \in \mathbb{S}_5$. Then, by construction and by Lemma 1, at the end of Step 1 the sequence of envelopes $B_1, \ldots, B_5$ contains the permutation σ^{-1}. By definition, the sequence of envelopes $S'_1, \ldots, S'_5$ is obtained in Step 4 by permuting the super-envelopes $S_1, \ldots, S_5$ of Step 3 according to a *new*, random, independent, and secret permutation $\rho \in \mathbb{S}_5$. Thus, also the sequences of envelopes $B'_1, \ldots, B'_5$, $C'_1, \ldots, C'_5$, and $D'_1, \ldots, D'_5$ have been obtained by respectively reordering $B_1, \ldots, B_5$, $C_1, \ldots, C_5$, and $D_1, \ldots, D_5$ according to ρ. Accordingly, the contents of $B'_1, \ldots, B'_5$, $C'_1, \ldots, C'_5$ and $D'_1, \ldots, D'_5$ necessarily and respectively are ρ, ρ, and $\rho\sigma^{-1} = \delta$. The latter permutation is revealed in Step 6, and actually is the only public record generated by $CLONE$ in Steps 2 through 7. This public record is thus a permutation randomly and independently selected in $\mathbb{S}_5$. In particular, it is independent of the only other public record of an execution of $CLONE$: the single public record of the execution of INV of Step 1 (which itself was randomly and independently selected member of $\mathbb{S}_5$). Thus, letting $\mathbb{D}$ be the uniform distribution over $\mathbb{S}_5 \times \mathbb{S}_5$, $CLONE$ enjoys the required privacy property. Finally, because $B'_1, \ldots, B'_5$ and $C'_1, \ldots, C'_5$ contained ρ and have been reorder in Step 7 according to $\delta^{-1} = (\rho\sigma^{-1})^{-1} = \sigma\rho^{-1}$, the content of both $B''_1, \ldots, B''_5$ and $C''_1, \ldots, C''_5$ necessarily is $\sigma\rho^{-1}\rho = \sigma$, as required for $CLONE$ to enjoy the correctness property.

2.4 Verifiable Ballot Computers for (the Encodings of) Four Basic Binary Functions

We show that, *relative to a special encoding*, there exists a verifiable ballot computer for each of the elementary binary functions discussed in Section 1.6: namely, NOT, AND, $DUPLICATE$, and $COIN$.

$\mathbb{S}_5$ Encodings. *The $\mathbb{S}_5$ encoding of a binary string $z = b_1, \ldots, b_k$, denoted by $\underline{z}$, is $\underline{b_1} \cdots \underline{b_k}$, where*

$$\underline{0} = 12345; \quad \underline{1} = 12453.$$

The $\mathbb{S}_5$ encoding of a set X of binary strings is the set $\underline{X}$ defined as follows: $\underline{X} = \{\underline{x} : x \in X\}$.
The $\mathbb{S}_5$ encoding of a function $h : \{0,1\}^a \to \{0,1\}^b$ is a function $\underline{h} : \underline{\{0,1\}^a} \to \underline{\{0,1\}^b}$ defined as follows:

$$\underline{h}(\underline{z}) = \underline{h(z)} \text{ for all } z \in \{0,1\}^a.$$

If $s = \underline{z}$ for some binary string z, the $\mathbb{S}_5$ decoding of s, is defined to be z.

The following two identities of Barrington (1986) enable one to evaluate the $\mathbb{S}_5$ encodings of the functions NOT and AND by means of a fixed sequence of inverse and product operations.

Barrington's Identities. For all bits a and b:

- $\underline{NOT}(\underline{a}) = 12354\ \underline{a}\ 12435$
- $\underline{AND}(\underline{a}, \underline{b}) = 13245\ \underline{a}\ 34125\ \underline{b}\ 34125\ \underline{a}^{-1}34125\ \underline{b}^{-1}\ 24135$.

Lemma 4. *There exists a 56-operation verifiable ballot computer for $\underline{NOT}$.*

Proof. The lemma follows from Barrington's first identity and our Lemmas 1 and 2. Indeed, the desired verifiable ballot computer for $\underline{NOT}$ can be informally described as follows.

Given a sequence x of 5 envelopes containing the $\mathbb{S}_5$ encoding of a bit a:

(1) publicly make two sequences of 5 envelopes, the first containing the permutation 12354 and the second the permutation 12435 (for a total of 2 ballot operations), then

(2) apply twice, to the proper sequences of envelopes, $MULT$, our 27-operation verifiable ballot computer for permutation product, (for a total of 54 ballot operations) so as to compute a final sequence of 5 envelopes whose content is guaranteed to be 12354 $\underline{a}$ 12435, and thus the the $\mathbb{S}_5$ encoding of the negation of bit a. ■

Lemma 5. *There exists a 309-operation verifiable ballot computer for $\underline{AND}$.*

Proof. The lemma follows from Barrington's second identity and our Lemmas 1, 2, and 3. Indeed the desired verifiable ballot computer for $\underline{AND}$ can be informally described as follows.

Given a sequence of 10 envelopes, the first five containing $\underline{a}$ and the next five containing $\underline{b}$:

(1) publicly make 5 sequences of 5 envelopes each, respectively containing the permutations 13245, 34125, 34125, 34125, and 24135 (for a total of 5 ballot operations);

(2) use twice $CLONE$, our 30-operation verifiable computer for permutation clone, in order to generate two sequences of 5 envelopes both containing $\underline{a}$, and two sequences of 5 envelopes both containing $\underline{b}$ (for a total of 60 ballot operations);

(3) use twice INV, our 14-operation verifiable ballot computer for permutation inverse, in order to transform one of the envelope sequence containing $\underline{a}$ into an envelope sequence containing $\underline{a}^{-1}$, and one of the sequences containing $\underline{b}$ into a sequence containing $\underline{b}^{-1}$ (for a total of 28 ballot operations); and finally

(4) use 8 times $MULT$, our 27-operation verifiable ballot computer for permutation product, in order to compute a sequence of envelopes whose content is guaranteed to coincide with

$$13245\ \underline{a}\ 34125\ \underline{b}\ 34125\ \underline{a}^{-1}34125\ \underline{b}^{-1}\ 24135$$

(for a total of 216 ballot operations). ■

Lemma 6. *There exists a 30-operation verifiable ballot computer for $\underline{DUPLICATE}$.*

Proof. A trivial corollary of Lemma 3: because $\underline{DUPLICATE}$ is a restriction of permutation clone, any verifiable ballot computer for permutation clone also is a verifiable ballot computer for $\underline{DUPLICATE}$. ■

Lemma 7. *There is a 7-operation verifiable ballot computer for $\underline{COIN}$.*

Proof. The following fixed sequence of steps is a compact description of the desired verifiable ballot computer:

(1) Publicly make two envelope sequences $A_1, \ldots, A_5$ and $B_1, \ldots, B_5$ with respective contents $\underline{0}$ and $\underline{1}$.

(2) Publicly make 2 new super-envelope S and T respectively containing $A_1, \ldots, A_5$ and $B_1, \ldots, B_5$.

(3) Ballot-box S and T to obtain super-envelopes S' and T'.

(4) Publicly destroy S' and publicly open T' to expose the final sequence of envelopes $R_1, \ldots, R_5$.

The computer's correctness is trivial, and, because it does not generate any public record, so is its privacy.

Verifying the total number of ballot operations is trivial. ■

2.5 Verifiable Ballot Computers for (the $\mathbb{S}_5$ Encoding of) Any Binary Finite Functions

Computation is very "local." That is, the computation of every function f can be broken into that of properly chosen elementary functions. Theoretically —see, for instance, Goldreich (2008), Section 1.2.4.1— it suffices to consider the above discussed four elementary functions: NOT, AND, $DUPLICATE$, and $COIN$ (if f is probabilistic). A bit more precisely,

Basic Fact For any probabilistic finite function $f : (\{0,1\}^a)^n \to \{0,1\}^a$, there exists a sequence of functions $\tilde{f} = \tilde{f}_1, \ldots, \tilde{f}_k$ (where each $\tilde{f}_i$ is either NOT, AND, $DUPLICATE$, or $COIN$) such that:

- Each input bit of $\tilde{f}_i$ coincides with (a) one bit of f's inputs or (b) one output bit of some function $\tilde{f}_h$ for $h < i$;
- Each input bit of f (each output bit of each $\tilde{f}_i$) appears at most once as an input of some $\tilde{f}_j$; and
- Exactly a outputs of the functions $\tilde{f}_i$ are not inputs of any $\tilde{f}_j$ and (after all functions $\tilde{f}_i$ have been orderly evaluated on their proper inputs) each one of them coincides with one output bit of f.

We refer to such $\tilde{f}$ as a *Boolean circuit* for f. We say that f has *complexity (at most) c* if f has a Boolean circuit of length c.

Theorem 1. *For every probabilistic, finite, binary function f of complexity c, there exists a 309-operation verifiable ballot computer for $\underline{f}$.*

Proof. Let $\tilde{f} = \tilde{f}_1, \ldots, \tilde{f}_c$ be a Boolean circuit for f. Then a desired verifiable ballot computer for f can be obtained by replacing each $\tilde{f}_i$ (that is, each occurrence of NOT, AND, $DUPLICATE$, and $COIN$ in $\tilde{f}$) by its corresponding verifiable ballot computer for $\underline{NOT}$, $\underline{AND}$, $\underline{DUPLICATE}$, and $\underline{COIN}$ of, respectively, Lemma 4, Lemma 5, Lemma 6, or Lemma 7, ensuring that each one of them operates on the right sequences of envelopes. Since the number of ballot operations of each of these four (sub)computers is at most 309, the total number of ballot operations of the so constructed verifiable ballot computer for f is at most $309c$. (To ensure that the number of operations is exactly $309c$, one can always resort to our "do nothing" operation.) ■

3 Perfect Ballot Implementations of Normal-Form Mechanisms

We are now ready to state and prove that any normal-form mechanism can be perfectly implemented by a ballot-box mechanism. We do so in two steps: first, in an idealized setting, where the players of a ballot-box mechanism are assumed to use only "legitimate strategies," and then in a realistic setting, where the players are also free to use "illegitimate strategies" if they so want.

3.1 The Idealized Setting

We start by recalling the classical notion of a normal-form mechanism in a form convenient to our goals.

Definition 3. *A n-k* idealized normal-form mechanism *$\mathcal{M}$ has*

- *$\{1, \ldots, n\}$ as its set of players;*
- *$X = \{0,1\}^k$ as the strategy set of each player;*
- *$Y = \{0,1\}^k$ as its outcome set; and*
- *a function $g : (\{0,1\}^k)^n \to \{0,1\}^k$ as its outcome function.*

Such a mechanism is played as follows:

1. *each player i, simultaneously with the others, privately chooses a strategy m^i;*
2. *g is privately and correctly evaluated on the strategy profile $(m^1, \ldots, m^n)$; and*
3. *the outcome $y = g(m^1, \ldots, m^n)$ is publicly announced.*

The complexity *of $\mathcal{M}$ is defined to be c if g has complexity c.*

Remark. In general, in a finite normal form mechanism, a player i is assumed to have his own, arbitrary set M_i of binary strings, and the outcome function g maps $M_1 \times \cdots \times M_n$ to an arbitrary set of binary strings Y. Yet it is easy to see that our formulation does not cause any loss of generality.[9]

Let us now define ballot-box mechanisms in the idealized setting.

Definition 4. *A n-k* idealized ballot-box mechanism $\mathcal{M}'$ *has*

- $\{1, \ldots, n\}$ *as its set of players;*
- $\underline{\{0,1\}^k}$ *as the strategy set of each player;*
- $Y = \{0,1\}^k$ *as its outcome set;*
- *a verifiable ballot mediator G that is no-ballot-left-behind for sequences of $5kn$ envelopes;*
- *an interpretation function I mapping arbitrary strings to Y.*

Such a mechanism is played as follows:

1. *Each player i privately chooses a strategy $\underline{m^i}$. Correspondingly, a sequence E^i of $5kn$ envelopes is privately made having $\underline{m^i}$ as its content.*
2. *G is publicly executed on the the $5kn$-envelope sequence E —whose first $5k$-envelope subsequence is E^i, the next is E^2, and so on— producing a sequence of public records R.*
3. *The outcome is declared to be the string $y = I(R)$.*

The complexity *of $\mathcal{M}'$ is defined to be k if G is k-operation.*

In the definition above, we of course interpret the envelopes in each E^i as made by i himself, and G as run by a "verifiable third party" in front of the players. Of course too, $\mathcal{M}'$ is idealized because nothing prevents a player i from making a sequence of $5k$ envelopes that do not contain the $\mathbb{S}_5$ encoding of a bit. (We shall lift this assumption in the next subsection).

We insist on "no-envelope-left-behind" to ensure that all concerns about privacy are restricted to the sequence of public records R generated by the envelopes opened by G.

Notation If s is a strategy profile of a mechanism $\mathbb{M}$, of any form, then by $\mathbb{M}(s)$ we denote the distribution over $\mathbb{M}$'s outcomes generated by playing $\mathbb{M}$ under s.

Definition 5. *We say that an n-k idealized ballot-box mechanism $\mathcal{M}'$* perfectly implements *a n-k idealized normal-form mechanism $\mathcal{M}$ if, for each player i, there exists a bijection ψ_i between i's strategies in $\mathcal{M}$ and i's strategies in $\mathcal{M}'$ such that, for all strategy profiles $(m^1, \ldots, m^n)$ of $\mathcal{M}$, the following properties hold:*

- Strategy Equivalence: $\mathcal{M}(m^1, \ldots, m^n) = \mathcal{M}'(\psi_1(m^1), \ldots, \psi_n(m^n))$.
- Privacy Equivalence: *For any subset S of the players, the information available to S, about the strategies of the other players, is the same in a random play of $\mathcal{M}$ under $(m_1, \ldots, m_n)$ and in a random play of $\mathcal{M}'$ under $(\psi_1(m^1), \ldots, \psi_n(m^n))$.*
- Complexity Equivalence. *If the complexity of $\mathcal{M}$ is c, then that of $\mathcal{M}'$ is (at most) $309c + 5k$.*

[9] Indeed, any such general-looking mechanism can be put in our format as follows. Let z be the cardinality of the largest set among $M_1, \ldots, M_n$, and Y. Choose $k = \lceil \log(z) \rceil$. Letting c_i be the cardinality of message set M_i, encode the elements of M_i as the lexicographically first c_i strings in $\{0,1\}^k$ and consider any string $x \in \{0,1\}^k$ lexicographically greater than c_i as an alternative encoding of the first element of M_i. Analogously encode the outcome set Y as elements of $\{0,1\}^k$. Define now $g' : (\{0,1\}^k)^n \to \{0,1\}^k$ to be the following function: if x'_i is an encoding of $x_i \in M_i$ for all $i \in N$, and if y' is an encoding of $y \in Y$, then $g'(x'_1, \ldots, x'_n) = y'$ if and only if $g(x_1, \ldots, x_n) = y$.

Remarks.

- *A specific definition.* Note that we have formalized our notion of perfect implementation for ballot-box mechanisms, rather than for "generic" concrete mechanisms. This is actually made necessary by the fact that, without precisely knowing how a concrete mechanism operates, one cannot express properties such as privacy equivalence in a formal way. One may, however, apply the essence of our definition to any other sufficiently specified class of concrete mechanisms. (Ensuring that the definition is achievable of course puts additional constrains on concrete mechanisms.)
- *About strategy equivalence.* Note that requiring a bijection ψ between the strategy profiles of $\mathcal{M}$ and those of $\mathcal{M}'$ is a less stringent and less satisfactory requirement than our "player-by-player bijection." Consider asking a player i about to play $\mathcal{M}$ with a strategy m^i whether he would mind playing $\mathcal{M}'$ instead. If $\mathcal{M}'$ satisfied our stringent condition, he would have no substantial objections: he could easily switch from playing m^i in $\mathcal{M}$ to playing $\psi_i(m^i)$ in $\mathcal{M}'$ and be guaranteed to be in the same strategic position. But if $\mathcal{M}'$ satisfied instead the above, less stringent condition, because the player does not know what the strategies of the others may be, he could not take advantage of the existence of an "overall bijection" ψ to figure out what strategy to switch to in order to be in an equivalent strategic position.
- *About privacy equivalence.* A weaker notion of privacy consists of stating that no *single* player learns anything about the strategies of the others beyond what is implicitly revealed by the outcome itself. We find such a notion insufficient because we realistically assume that people collude whenever there is any advantage to be gained. And violating another player's privacy may be important enough to collude.
- *About complexity equivalence.* Complexity equivalence aims at guaranteeing that the "physical complexity of $\mathcal{M}'$ is essentially comparable to the computational complexity of $\mathcal{M}$." Of course, there is nothing magic about the expression "$309c + 5k$". (This expression is actually far from being optimized on one hand, and dependent on our specific choice of ballot operations on the other.) But in light of the exponential blow ups of previous implementations, it is important, and perhaps surprising, that the relation between the two complexity can be linear.

Theorem 2. *For every n-k idealized normal-form mechanism $\mathcal{M}$ there is a n-k idealized ballot-box mechanism $\mathcal{M}'$ perfectly implementing $\mathcal{M}$.*

Proof. Let g be the outcome function of $\mathcal{M}$, and $\tilde{g}$ a Boolean circuit for g of length c. Then, let $\mathcal{M}'$ be an n-k idealized ballot-box mechanism having

- a verifiable ballot mediator consisting of the concatenation of two mediators: (1) a verifiable ballot computer G for $\underline{g}$ constructed, as in our proof of Theorem 1, using $\tilde{g}$ as the underlying Boolean circuit for g; and (2) a verifiable ballot mediator O that opens all the envelopes left unopened by G; and
- an interpretation function consisting of a function that, when given a sequence of public records R, computes the substring R' comprising the last $5k$ records (i.e., the contents of the final envelopes of G that are subsequently opened), and then $\mathbb{S}_5$ decodes R' (i.e., computes a k-bit string y such that $\underline{y} = R'$).

Such an $\mathcal{M}'$ exists in virtue of Theorem 1; of the fact that, for any initial sequence of envelopes and any execution of G, the identifiers of all final envelopes are always the same (else, the desired O would be hard to come by); and the fact that the concatenation of two verifiable ballot mediators is a verifiable ballot mediator. Let us now prove that $\mathcal{M}'$ perfectly implements $\mathcal{M}$. (To begin with, note that, by construction, the mediator of $\mathcal{M}'$ is indeed no-ballot-left-behind when executed on a sequence of $5kn$ envelopes.)

Strategy equivalence follows straightforwardly from the fact that G is a verifiable ballot computer for $\underline{g}$ and the definitions of $\mathbb{S}_5$ encoding and decoding. To see this, just let each ψ_i be the function mapping any k-bit string s to $\underline{s}$.

Privacy equivalence holds because the total information available to a subset S of the players in a random execution of $\mathcal{M}$ under $(m^1, \ldots, m^n)$ consists of (1) their own strategy subprofile m^S, and (2) an outcome y drawn from the distribution $g(m^1, \ldots, m^n)$; while, in a random execution of $\mathcal{M}'$ under $(\underline{m^1}, \ldots, \underline{m^n})$, consists

of $(1')$ their own strategy profile $\underline{m^S}$ and thus, equivalently, m^S, $(2')$ an outcome y drawn from the distribution $g(m^1, \ldots, m^n)$, and $(3')$ "random noise", as guaranteed by G's privacy property.

Complexity equivalence holds because Theorem 1 guarantees that G is $309c$-operation, and $\mathcal{M}'$ requires only $5k$ additional ballot operations in order to open all the final envelopes generated by G. ■

3.2 The Realistic Case

Normal-form mechanisms would remain an abstraction even if outcome functions would magically and spontaneously evaluate themselves on the players' strategies as soon they are chosen in the players' minds. This is so because it is very hard in real life to constrain a player to choose a string in a predetermined set: typically there are plenty of "alternative strategies." For instance, a player could kill another player, or prefer to commit suicide rather than choosing a k-bit string as expected from him. Presumably when modeling a real strategic situation as a normal-form mechanism, one feels entitled to disregards such alternative strategies because they lead to "bad outcomes," so that no rational player would ever need to consider them. Yet, without envisaging what would happen in these alternative scenarios, a concrete mechanism would be under-specified and could not be properly analyzed, let alone proved equivalent to a normal-form one that does not have any room for any "alternatives." Indeed, even if one were willing to implement a normal-form mechanism with the help of a trusted mediator, he would have to decide what would happen if a player, rather then handing to the mediator a k-bit string as prescribed, handed him no string or a longer string.

We refer to any "alternative strategy" as an *aborting* strategy. To enable to measure meaningfully and precisely how close ballot-box mechanisms may get to normal-form ones, we include in the latter mechanisms a single, abstract, aborting strategy. That is, we chose to enlarge the strategy set of each player of a normal-form mechanism with a distinguished string, `abort`, having the following effect. If no player chooses `abort`, then the outcome is chosen as usual, else the outcome only consists of identifying the set of all players who chose `abort`. This choice does not compromise the privacy of the other players, and leaves room for dissuading players from aborting in a variety of ways —for instance, by imposing a suitably large fine to the aborting players.

We stress that, while players of a normal-form mechanism can abort in a quite abstract and controlled way, the players of a concrete ballot-box mechanism can abort in any way they want. Yet, we shall state and prove that perfect implementation of "slightly more realistic" normal-form mechanisms by means of "much more realistic" ballot-box ones is still possible.

Note that aborting strategies in a ballot-box mechanism do not present a great difficulty when a player shoots another player of displays some other *public* deviant behavior. When "public aborting strategies" are employed, it is trivial to identify the aborters, halt the ballot-box mechanism outputting the set of all aborting players, and thus enforce (the properly enhanced versions of) strategic, privacy and complexity equivalence.

What is more difficult to handle is a player that privately prepares an envelope sequence that does not contain the $\mathbb{S}_5$ encoding of a k-bit string. For instance, such a player may make a subsequence of 5 envelopes containing the string 22233, or the string 54321.[10] Such "secret alternative strategies" are in fact not detectable right away, and left unchecked may in principle cause some ballot-box mechanisms to be ultimately quite different from their still quite abstract counterparts, and possibly reward the so deviating players.

For the reasons put forward in the above two paragraphs, in the remainder of this section, we shall consider only "secret alternative strategies."

3.2.1 Formalization

Definition 6. *A n-k* realistic *normal-form mechanism $\mathcal{M}$ has*

[10] Moreover, as we shall soon see, we do not need to consider the possibility of making an envelope with a content, like "7" or "$", that is not an integer between 1 and 5. Such deviations do not present any additional problems, except notational ones.

- $\mathcal{N} = \{1, \ldots, n\}$ *as its set of players;*
- $\{0,1\}^k \cup \{\texttt{abort}\}$ *as the strategy set of each player;*
- $\{0,1\}^k \cup \{(\texttt{abort}, S) : S \subset \mathcal{N}\}$ *as its outcome set; and*
- *a function* $g : (\{0,1\}^k)^n \to \{0,1\}^k$ *as its outcome function.*

Such a mechanism is played as follows:

1. *Each player* i*, simultaneously with the others, privately chooses a strategy* m^i*;*
2. *The set of players* $S = \{i : m^i = \texttt{abort}\}$ *is computed.*
3. *An outcome is publicly announced as follows:* (abort,S)*, if* S *is non-empty, and* $y = g(m^1, \ldots, m^n)$ *otherwise.*

The complexity *of* $\mathcal{M}$ *is defined to be the complexity of* g*.*

Definition 7. *A* n*-*k realistic *ballot-box mechanism* $\mathcal{M}$ *has*

- $\{1, \ldots, n\}$ *as its set of players;*
- $\{1, \ldots, 5\}^{5k}$ *(that is,* $5k$ *integers between 1 and 5) as the strategy set of each player;*
- $Y = \{0,1\}^k \cup \{(\texttt{abort}, S) : S \subset \mathcal{N}\}$ *as its outcome set;*
- *a verifiable ballot mediator* G *that is no-ballot-left-behind for envelope sequences of* $5kn$ *envelopes;*
- *an interpretation function* I *mapping arbitrary strings to* Y*.*

Such a mechanism is played as follows:

1. *Each player* i *privately chooses a strategy* s^i*. Correspondingly, a sequence* E^i *of* $5k$ *envelopes is made with* s^i *as its content.*
2. G *is publicly executed on the the* $5kn$*-envelope sequence* E *—whose first* $5k$*-envelope subsequence is* E^i*, the next is* E^2*, and so on— producing a sequence of public records* R*.*
3. *The outcome is declared to be the string* $y = I(R)$*.*

The complexity *of* $\mathcal{M}'$ *is defined to be the number of ballot operations in* G*.*

Definition 8. *We say that an* n*-*k *realistic ballot-box mechanism* $\mathcal{M}'$ perfectly implements *a* n*-*k *realistic normal-form mechanism* $\mathcal{M}$ *if, for each player* i *and each string* $\mathtt{x}_i \in \{1, \ldots, , 5\}^{5k} \setminus \underline{\{0,1\}^k}$*, there exists a bijection* $\psi_i^{\mathtt{x}_i}$ *between* $\{0,1\}^k \cup \{\texttt{abort}\}$ *and* $\underline{\{0,1\}^k} \cup \{\mathtt{x}_i\}$ *such that, for all strategy profiles* $(m^1, \ldots, m^n)$ *of* $\mathcal{M}$*, the following properties hold:*

- Strategy Equivalence: $\mathcal{M}(m^1, \ldots, m^n) = \mathcal{M}'(\psi_1^{\mathtt{x}_1}(m^1), \ldots, \psi_n^{\mathtt{x}_n}(m^n))$.
- Privacy Equivalence: *For any subset* S *of the players, the information available to* S*, about the strategies of the other players, is the same in a random play of* $\mathcal{M}$ *under* $(m_1, \ldots, m_n)$ *and in a random play of* $\mathcal{M}'$ *under* $(\psi_1^{\mathtt{x}_1}(m^1), \ldots, \psi_n^{\mathtt{x}_n}(m^n))$*.*
- Complexity Equivalence. *If the complexity of* $\mathcal{M}$ *is* c*, then that of* $\mathcal{M}'$ *is at most* $309c + 151nk$*.*

3.2.2 Solution

The key to prove that every realistic normal-form mechanism is perfectly implementable by a realistic ballot-box mechanism is constructing the following special type of a verifiable ballot mediator.

Definition 9. *We say that a 5-to-5 verifiable ballot mediator* M *is a* bit-checker *if there exist a predicate* $\mathbb{P}$*, mapping sequences of public records to* $\{0,1\}$*, and a distribution* $\mathbb{D}$*, over sequences of public records, such that, respectively denoting by* E *and* F *the initial and the final envelope sequence of* M*, the following two properties hold:*

1. Correctness. *For any bit* b*, in any execution of* M *in which* E*'s content is* $\underline{b}$*,* F*'s content is also* $\underline{b}$*.*

2. Privacy. *For any bit b, in a random execution of M in in which E's content is $\underline{b}$ the the final sequence of public records is distributed according to $\mathbb{D}$.*

3. Public exposure. *Let R_1 (respectively, R_0) be the final sequence of public records of an execution of M in which E's content is (respectively, is not) the $\mathbb{S}_5$ encoding of a bit. Then $\mathbb{P}(R_1) = 1$ and $\mathbb{P}(R_0) = 0$.*

Consider now the following verifiable ballot mediator

$\mathbb{BC}$

Given a sequence of envelopes $A_1, \ldots, A_5$, produce a final sequence of envelopes $K_1, \ldots, K_5$ as follows:

(A) Execute on $A_1, \ldots, A_5$ the verifiable ballot computer INV of the proof of Lemma 1 to obtain a final sequence 5 envelopes, $F_1, \ldots, F_5$.

Claim A: *If $A_1, \ldots, A_5$ originally contained a permutation $\sigma \in \mathbb{S}_5$, then (1) $F_1, \ldots, F_5$ now contains σ^{-1}, and (2) the public record so far consists of a sequence of permutations in $\mathbb{S}_5$. Else, one of the public records produced in Step a is not a permutation in $\mathbb{S}_5$.*

Comment: *In light of Claim 1, in all the steps and claims below we assume that $A_1, \ldots, A_5$ contained a permutation in $\mathbb{S}_5$, from now on consistently denoted by σ.*

(B) Execute INV on $F_1, \ldots, F_5$ to obtain a final sequence of envelopes $G_1, \ldots, G_5$.

Claim B: *$G_1, \ldots, G_5$ contains the same permutation σ originally contained in $A_1, \ldots, A_5$.*

(C) Replace $G_1, \ldots, G_5$ with the final sequences of envelopes $I_1, \ldots, I_5$, $J_1, \ldots, J_5$, and $K_1, \ldots, K_5$ obtained by executing the verifiable ballot computer $CLONE$ of the proof of Lemma 3 twice: first on $G_1, \ldots, G_5$, to generate the envelope sequences $H_1, \ldots, H_5$ and $I_1, \ldots, I_5$, and then on $H_1, \ldots, H_5$, to generate $J_1, \ldots, J_5$ and $K_1, \ldots, K_5$.

Claim C: *$I_1, \ldots, I_5$, $J_1, \ldots, J_5$ and $K_1, \ldots, K_5$ contain the same permutation $\sigma \in \mathbb{S}_5$ originally contained in $A_1, \ldots, A_5$.*

(D) Execute the verifiable ballot computer for $\underline{NOT}$ described in the proof of Lemma 4 on $I_1, \ldots, I_5$ so as produce the envelope sequence $L_1, \ldots, L_5$.

Claim D: *If the permutation σ originally contained in $A_1, \ldots, A_5$ was $\underline{b}$ for some bit b, then both $J_1, \ldots, J_5$ and $K_1, \ldots, K_5$ contain $\underline{b}$, and $L_1, \ldots, L_5$ contain $\underline{1-b}$. Else, none of $J_1, \ldots, J_5$, $K_1, \ldots, K_5$, and $L_1, \ldots, L_5$ contain the $\mathbb{S}_5$ encoding of a bit.*

(E) Publicly make a new super-envelope S containing the sequence of envelopes $K_1, \ldots, K_5$, and a new super-envelope T containing $L_1, \ldots, L_5$.

Ballotbox S and T to obtain the super-envelopes S' and T'.

Publicly open S' to expose the sequence of envelopes $X_1, \ldots, X_5$, and publicly open T' to expose the envelope sequence $Y_1, \ldots, Y_5$.

Finally, publicly open each envelope in $X_1, \ldots, X_5$ to reveal a content x and each envelope in $Y_1, \ldots, Y_5$ to reveal a content y.

Claim E: *If x and y are the $\mathbb{S}_5$ encodings of opposite bits, then: $A_1, \ldots, A_5$ contained $\underline{b}$ for some bit b, and the same $\underline{b}$ is now the content of the final envelope sequence $J_1, \ldots, J_5$. Else, $A_1, \ldots, A_5$ did not contain the $\mathbb{S}_5$ encoding of any bit.*

Lemma 4: $\mathbb{BC}$ is a 151-operation bit-checker.

Proof. To begin with, note that $\mathbb{BC}$ indeed is a 5-to-5 verifiable ballot mediator. (This is so because it is the concatenation of a few verifiable ballot mediators, chosen from INV, NOT, and $CLONE$, which can always

be executed so as to produce their prescribed final sequences of envelopes no matter what the contents of their initial envelope sequences are.) Moreover, it is immediately seen that $\mathbb{BC}$ indeed transforms an initial sequence of 5 envelopes to a final sequence of 5 envelopes. Let us now prove that each of Claims A through E holds.

Proof of Claim A. If $A_1, \ldots, A_5$ contained a permutation $\sigma \in \mathbb{S}_5$, then $F_1, \ldots, F_5$ contain σ^{-1} because INV is a verifiable ballot computer for permutation inverse.

If the content of $A_1, \ldots, A_5$ is not a permutation in $\mathbb{S}_5$, then neither is the content α publicly revealed in Step 5 of INV. □

Thus, the predicate $\mathbb{P}$, demanded by the definition of a bit-checker, can utilize α to differentiate the sequences of public records arising when the initial envelopes $A_1, \ldots, A_5$ contain the $\mathbb{S}_5$ encoding of a bit from those arising when $A_1, \ldots, A_5$ do not contain such an encoding. Accordingly, as per our comment, we can indeed assume below that $A_1, \ldots, A_5$ originally contained a permutation in $\mathbb{S}_5$, denoted by σ. By Claim A, therefore, $F_1, \ldots, F_5$ contain σ^{-1}.

Proof of Claim B. Trivially true, since INV is a verifiable ballot computer for permutation inverse. □

Proof of Claim C. Trivial, in light of the fact that $CLONE$ is a verifiable ballot computer for $DUPLICATE$. □

Proof of Claim D. The "if part" follows trivially from the fact that we are running a verifiable ballot computer for $\underline{NOT}$. Let us now argue the "else part." To this end, recall that we have already argued that the sequences $I_1, \ldots, I_5$, $J_1, \ldots, J_5$, and $K_1, \ldots, K_5$ all contained the same permutation σ originally contained in $A_1, \ldots, A_5$. Thus if σ is not the $\mathbb{S}_5$ encoding of a bit, neither is the content of $J_1, \ldots, J_5$ or $K_1, \ldots, K_5$. The same is true for the content of the sequence $L_1, \ldots, L_5$. In fact, as we have already noted, our verifiable ballot computer for $\underline{NOT}$ is the product of *specific* permutations, dictated by the first Barrington identity, and thus itself a bijection between $\mathbb{S}_5$ and $\mathbb{S}_5$. And since this bijection maps $\{12345, 12453\}$ into $\{12345, 12453\}$, it cannot map any other other permutation in $\mathbb{S}_5$, including our σ, to either 12345 or 12453. □

Proof of Claim E. Since Step e leaves the envelope sequence $J_1, \ldots, J_5$ alone, and ultimately opens $K_1, \ldots, K_5$ and $L_1, \ldots, L_5$, Claim E is trivially implied by Claim D. □

In sum, therefore, the correctness property of $\mathbb{BC}$ follows trivially from Claims A through E.

The public-exposure property follows from Claims A and E. In fact, the predicate $\mathbb{P}$, demanded by the definition of a bit-checker, can be trivially constructed by using the content of α of $K_1, \ldots, K_5$ and $L_1, \ldots, L_5$ to distinguish execution of $\mathbb{BC}$ in which the initial sequence of envelopes contained a permutation in $\{\underline{0}, \underline{1}\}$ from those whose initial envelopes contained a permutation in $\mathbb{S}_5 \setminus \{\underline{0}, \underline{1}\}$.

The privacy property holds too. To this end notice that, whenever $A_1, \ldots, A_5$ contain the $\mathbb{S}_5$ encoding of a bit b, then, no matter whether $b = 0$ or $b = 1$, every public record of a random execution of $\mathbb{BC}$ consists of a permutation randomly and independently selected in $\mathbb{S}_5$, except for the public-record pair (x, y) of Step e, which however is guaranteed to be $(\underline{0}, \underline{1})$ with probability 1/2, and $(\underline{1}, \underline{0})$ with probability 1/2. Thus, whenever the initial 5 envelopes contain the $\mathbb{S}_5$ encoding of a bit b, the distribution over the final sequence of public records is independent of b, as demanded for a bit-checker. ■

Theorem 3. *For every n-k realistic normal-form mechanism $\mathcal{M}$ there is a n-k realistic ballot-box mechanism $\mathcal{M}'$ perfectly implementing $\mathcal{M}$.*

Proof. The proof mimics that of Theorem 2, after properly incorporating a bit-checker into the verifiable ballot mediator of $\mathcal{M}'$. Let g be the outcome function of $\mathcal{M}$, $\tilde{g}$ a Boolean circuit for g of length at most c, and $\mathcal{M}'$ the n-k realistic ballot-box mechanism with the following verifiable ballot mediator B' and interpretation function I':

B' : Given a sequence E^i of $5k$ envelops for each player i, the verifiable ballot mediator of $\mathcal{M}'$ works as follows. First, conceptually, it subdivides each E^i into k contiguous (sub)sequences of 5 envelopes each, $E^i_1, \ldots, E^i_k$. Then, it executes the bit-checker $\mathbb{BC}$ on each E^i_j to produce a sequence of public records R^i_j and a 5-long sequence of final envelopes F^i_j.

If the predicate $\mathbb{P}$ of $\mathbb{BC}$ evaluated on some R^i_j indicates that the corresponding envelope sequence E^i_j was not the $\mathbb{S}_5$ encoding of a bit, then the remaining operations of B' consist of a suitable number of "do-nothing" operations followed by number of "destroy-ballot" operations so that no unopened ballot is left while the required total number of operations is reached.

Else, the remaining operations of B' consist of executing on the obtained sequences $F^1_1, \ldots, F^n_k$ a verifiable ballot computer G for $\underline{g}$ constructed, as in our proof of Theorem 1, using $\tilde{g}$ as the underlying Boolean circuit for g; and finally a verifiable ballot mediator that opens all the envelopes left unopened by G.

I' : If, for some player i there exists at least one (bit-position) j such that $\mathbb{P}(R^i_j) = 0$, then, letting S be the subset of all such players i, I' returns $(\texttt{abort}, S)$. Else, letting R' be the subsequence consisting of the last $5nk$ public records produced by B', I' returns the $\mathbb{S}_5$ decoding of R'.

To see that strategy equivalence holds, for each player i and each $\mathtt{x}_i \in \{1, \ldots, 5\}^{5k} \setminus \underline{\{0,1\}^k}$, define $\psi_i^{\mathtt{x}_i}$ as follows: $\psi_i^{\mathtt{x}_i}(m^i) = \underline{m^i}$ if $m^i \in \{0,1\}^k$, and $\psi_i^{\mathtt{x}_i}(m^i) = \mathtt{x}_i$ if $m^i = \texttt{abort}$.

To see that privacy equivalence holds, notice that, for each player i and each $\mathtt{x}_i \in \{1, \ldots, 5\}^{5k} \setminus \underline{\{0,1\}^k}$, the total information available to a subset S of the players in $\mathcal{M}'$ under the strategy profile $(\psi_1^{\mathtt{x}_1}(m^1), \ldots, \psi_n^{\mathtt{x}_n}(m^n))$ consists of the information available to the players in S in $\mathcal{M}$ under $(m^1, \ldots, m^n)$ plus some "random noise."

To see that complexity equivalence holds, notice that the total number of ballot operations of $\mathcal{M}$ consists of the sum of the ballot operations of G, (which is at most $309c$) and the total number of ballot operations of $\mathbb{BC}$, which is at most $151nk$, since $\mathbb{BC}$ is 151-operation and is executed is executed nk times, that is for each strategy bit the players have in $\mathcal{M}$. ■

4 From Normal-Form to Arbitrary Mechanisms

In a very recent conference paper, Izmalkov, Lepinski and Micali (2008), we generalize the present verifiable-mediator results in a variety of ways. To begin with, we show how to perfectly simulate a richer form of normal-form mechanisms, where, after each player i simultaneously with the others submits a report x^i, not only there is a (public) outcome function g that determines the publicly announced outcome $y = g(x_1, \ldots, x_n)$, but also a *private* outcome function f that determines the profile of private outcomes $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, and each player i learns (in addition to y) just his own y_i. In particular, this result enables one to perfectly simulate, with ballots and without trusting anyone, a more private version of the second-price mechanism in which (in addition to the seller, if so desired) only the winner learns that she has won and the price she has to pay, while all other players learn only that they did not win.

More importantly, after changing some of our ballot operations with other ones just as intuitive, we show that it is possible to perfectly simulate a general "interactive mechanism." Such a mechanism proceeds in a sequence of stages. The jth stage is played as follows:

- Each player i, simultaneously with the others, submits a private message x^j_i; and the mechanism has available a private message z^j_0 (z^1_0 is typically the empty string).
- An outcome function F^j is privately evaluated on the sequence of all stage-j messages to compute a public outcome y^j, a profile of private outcomes $(y^j_1, \ldots, y^j_n)$, and a mechanism private message z^{j+1}_0.
- Outcome y^j is publicly announced, message z^{j+1}_0 is kept private by the mechanism, and each player i privately receives outcome y^i_j.

(Essentially, such a general mechanism can be conceptualized as a sequence of enriched normal-form mechanisms, passing prescribed private information not only to the players, but to each other.[11])

[11] For instance, in implementing two auctions one after the other, the first mechanism may announce the winner and the second highest bid, but privately pass on to the second mechanism some aggregate information of all bids —e.g., the average of all bids, which may be used to set the reserve price of the second auction.

The ballot-box remains a crucial ingredient in perfectly implementing such an $\mathcal{M}$, but it becomes more tricky to use. In fact, in simulating normal-form mechanisms, we were not concerned that the execution of a ballot-box mechanism made a random string (i.e., the sequence of public records) common knowledge. Of course, the availability of a public random string can affect the play of a mechanism.[12] But such a random string cannot cause any strategic effect when the players seeing it have no further play ahead of them, as it is the case in a ballot-box mechanism implementing a normal-form one. By contrast, an interactive mechanism essentially envisages a sequence of interrelated plays, and to enable a ballot-box mechanism to simulate it perfectly such a sequence, we must make sure that the generated sequence of public records contains no randomness at all (else, strategic equivalence could not hold in its purest form).

Ensuring that this additional condition is the main technical contribution of our latest construction.

5 Final Remarks

Equating Subjects and Objects. As for other constructions in logic and computation, the universality of ours ultimately arises from "equating data and programs." In our case, this is achieved by a very simple idea: namely encoding a 5-element permutation p by the string of integers $p(1)p(2)p(3)p(4)p(4)p(5)$, and then putting $p(j)$ into envelope j, so as to have a 5-envelope representation of p. The convenience of this simple representation is that such a sequence of envelopes is both a piece of (really physical!) *data,* as well as an *algorithm*, which via the ballot box is capable of operating on other data. Indeed the crucial subroutine of our construction (because it is the one that "brings together" different permutations, and thus, ultimately, different bits) is a procedure that, given two sequences of 5 envelopes —the first sequence standing a permutation p and the second for a permutation q— interprets the first sequence as the program "multiply by p" and returns (without revealing any information about p or q) a sequence of 5 envelopes encoding the product permutation pq, that is, the permutation mapping each i between 1 and 5 to $p(q(i))$.

It should be noted that other classical ways to represent permutations —such as the "cycle representation" that encodes the permutation encoded by us as "35412" with "(1,3,4)(2,5)"— although very convenient for other purposes, do not seem to amenable to equating subjects and objects, at least in a straightforward way, in our ballot context.

Complementing Mechanism Design. As remarkable as they may be, the solutions offered by mechanism design are, most of the time, abstract normal-form mechanisms, which may not retain their properties when straightforwardly played by players who value privacy or do not trust anyone.[13] Thus, while we do not help a designer in engineering new mechanisms, by perfectly implementing whatever abstract mechanisms he finds, we do enable him to ignore issues of privacy and trust in his work.

Meaningfulness and Abstraction. Abstractions are certainly powerful. But the meaningfulness of an abstraction ultimately depends on whether concrete implementations that adequately approximate it exist. In a sense, therefore, our contributions guarantee the practical meaningfulness of the very notion of a normal-form mechanism: no matter how "delicate," its theoretical properties will continue to hold intact for at least one concrete implementation (i.e., its ballot-box implementation).

[12] In particular, it enables the players of a normal-form game now achieve, without additional external help, the payoffs of the "convex closure" of the initial set of Nash equilibria.

[13] Sjöström and Maskin (2002) provide a comprehensive survey of mechanism design and implementation theory literature. Normal-form mechanisms are also extensively used in more applied fields, such as auction theory and contract theory, see Krishna (2002), Bolton and Dewatripont (2005). Often the problems of privacy and trust are by-passed by explicit additional assumptions. For instance, it is typically assumed that the seller (and similarly the principal) can fully commit to the mechanism she offers to the buyers (and similarly to the agents)—and, since buyers know that, their rationality dictates they must trust the seller.

A Fact, a Prediction, and a Wish. People care about privacy. And to develop more accurate models, privacy should intrinsically inform any general study of human interactions. In particular, we believe and hope that a rigorous and comprehensive treatment of privacy will become an integral part of game theory.

New Economic School, Nakhimovsky prosp. 47, room 17-21 Moscow, Russia, 117418; sizmalkov@gmail.com;

BBN Technologies, 10 Moulton Street, Cambridge, MA 02138; mlepinski@bbn.com;

and

Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 32 Vassar Street, Room G644, Cambridge, MA 02139; silvio@csail.mit.edu.

References

Aumann, R. J. and Hart, S.: 2003, Long cheap talk, *Econometrica* **71**(6), 1619–1660.

Bárány, I.: 1992, Fair distribution protocols or how the players replace Fortune, *Mathematics of Operations Research* **17**, 329–340.

Barrington, D. A.: 1986, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *Proceedings of the 18th Symposium on Theory of Computing*, ACM, pp. 1–5.

Ben-Or, M., Goldwasser, S. and Wigderson, A.: 1988, Completeness theorems for fault-tolerant distributed computing, *Proceedings of the 20th Symposium on Theory of Computing*, ACM, pp. 1–10.

Ben-Porath, E.: 1998, Correlation without mediation: Expanding the set of equilibrium outcomes by cheap pre-play procedures, *Journal of Economic Theory* **80**, 108–122.

Ben-Porath, E.: 2003, Cheap talk in games with incomplete information, *Journal of Economic Theory* **108**(1), 45–71.

Bolton, P. and Dewatripont, M.: 2005, *Contract Theory*, MIT Press, Cambridge, Massachusetts.

Brandt, F. and Sandholm, T.: 2004, (Im)possibility of unconditionally privacy-preserving auctions, *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, IEEE, pp. 810–817.

Dodis, Y., Halevi, S. and Rabin, T.: 2000, A cryptographic solution to a game theoretic problem, *Advances in Cryptology — CRYPTO 2000, LNCS*, Vol. 1880, Springer-Verlag, pp. 112–130.

Forges, F. M.: 1986, An approach to communication equilibria, *Econometrica* **54**(6), 1375–85.

Forges, F. M.: 1990, Universal mechanisms, *Econometrica* **58**, 1341–1364.

Gerardi, D.: 2004, Unmediated communication in games with complete and incomplete information, *Journal of Economic Theory* **114**(1), 104–131.

Gerardi, D. and Myerson, R. B.: 2007, Sequential equilibria in bayesian games with communication, *Games and Economic Behavior* **60**(1), 104–134.

Goldreich, O. 2008, *Computational Complexity*, Cambrdige University Press.

Goldreich, O., Micali, S. and Wigderson, A.: 1987, How to play any mental game, *Proceedings of the 19th Symposium on Theory of Computing*, ACM, pp. 218–229.

Goldwasser, S., Micali, S. and Rackoff, C.: 1985, The knowledge complexity of interactive proof-systems, *Proceedings of the 17th Symposium on Theory of Computing*, ACM, pp. 291–304. Final version in *SIAM Journal on Computing*, 1989, 186–208.

Izmalkov, S., Lepinski, M. and Micali, S.: 2005, Rational secure computation and ideal mechanism design, *Proceedings of the 46th Symposium on Foundations of Computer Science*, IEEE, pp. 585–594.

Izmalkov, S., Lepinski, M. and Micali, S.: 2005, Verifiably Secure Devices, *Proceedings of the 5th Theory of Cryptography Conference*, Springer, pp. 273-301.

Krishna, R. V.: 2006, Communication in games of incomplete information: Two players, *Journal of Economic Theory* . forthcoming.

Krishna, V.: 2002, *Auction Theory*, Academic Press, San Diego, California.

Lepinski, M., Micali, S., Peikert, C. and Shelat, A.: 2004, Completely fair sfe and coalition-safe cheap talk, *Proceedings of the 23rd annual Symposium on Principles of distributed computing*, ACM, pp. 1–10.

Lindell, Y. and B. Pinkas: 2004, A Proof of Yao's Protocol for Secure Two-Party Computation, available at: `http://www.cs.biu.ac.il/~lindell/abstracts/yao_abs.html`.

Myerson, R. B.: 1982, Optimal coordination mechanisms in generalized principal-agent problems, *Journal of Mathematical Economics* **10**(1), 67–81.

Naor, M., Pinkas, B. and Sumner, R.: 1999, Privacy preserving auctions and mechanism design, *Proceedings of the 1st conference on Electronic Commerce*, ACM.

Osborne, M. J. and Rubinstein, A.: 1997, *Game Theory*, MIT Press, Cambridge, Massachusetts.

Rothkopf, M. H., Teisberg, T. J. and Kahn, E. P.: 1990, Why are Vickrey auctions rare?, *Journal of Political Economy* **98**, 94–109.

Sjöström, T. and Maskin, E.: 2002, *Handbook of Social Choice and Welfare*, Vol. 1, North-Holland, chapter Implementation Theory, pp. 237–288.

Urbano, A. and Vila, J. E.: 2002, Computational complexity and communication: Coordination in two-player games, *Econometrica* **70**(5), 1893–1927.

Wilson, R.: 1987, Game-theoretic analyses of trading processes, *in* T. F. Bewley (ed.), *Advances in Economic Theory, Fifth World Congress*, Cambridge University Press, Cambridge, UK, pp. 33–70.

Yao, A.: 1986, Protocol for secure two-party computation, never published. The result is presented in Lindell and Pinkas (2004).