

PERFECT IMPLEMENTATION OF NORMAL-FORM MECHANISMS

BY SERGEI IZMALKOV, MATT LEPINSKI, AND SILVIO MICALI

October 31, 2006

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 not be realistic and 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, an extensive-form mechanism exactly preserving all strategic properties of \mathcal{M} , *without relying on a trusted party or violating the privacy of the players*.

We prove that *any* normal-form mechanism can be perfectly implemented via envelopes and an envelope-randomizing device (i.e., the same tools used for running fair lotteries or tallying secret votes).

1. INTRODUCTION

1.1. The Problem of Implementing Normal-Form Mechanisms

The game theoretic notion of a normal-form mechanism consists of a set of *reports* (or messages) for each player, and an *outcome function* mapping these reports to outcomes. For example, in the famous second-price auction mechanism, the reports consist of numerical bids —one for each player— and the outcome function returns the player with the highest bid as the *winner*, and the value of the second-highest bid as the *price*. Normal-form mechanisms can be designed so as to enjoy valuable theoretical properties. The characteristic property of the second-price mechanism is *efficiency*: because it is a dominant strategy for a player to bid his true valuation of the item for sale, in equilibrium the winner is the player with the highest true valuation.

Of course, however, normal-form mechanisms are *abstractions*. Outcome functions do not spontaneously evaluate themselves on players' reports. To be of use in concrete strategic settings, a normal-form mechanism must be *implemented*, but then its theoretical properties may suffer.

PRIVACY VS. TRUST. An implementation of a normal-form mechanism is called *mediated* if its players rely on an external trusted party (the *mediator*), and *unmediated* otherwise. Every normal-form mechanism has implementations of both types; in particular, the following ones:

M_1 : (Mediated Mechanism) The players confide their reports to the mediator, who then secretly evaluates the outcome function and publicly announces the result.

M_2 : (Unmediated Mechanism) The players seal their reports into envelopes, which are then publicly opened so as to enable everyone to compute the outcome.¹

¹For convenience, the players could employ an external party, who first collects their envelopes and then publicly opens them, without making the mechanism a mediated one. In fact, while a mediator is blindly trusted, such an external party only performs public actions, and thus all players can verify that he performs them correctly.

These two general and concrete mechanisms are at opposite ends with respect to privacy and trust. Consider using them to implement a second-price auction in a *private-value* setting. On one hand, as long as the mediator is honest, M_1 reveals nothing more than the correct outcome. (However, the players cannot verify the mediator’s honesty: nothing guarantees that he announces the correct outcome, or that he will keep their bids secret after the auction is over.²) On the other hand, M_2 guarantees the correctness of the outcome, but reveals much more: it makes the players’ reports public knowledge in their entirety. In sum, M_1 requires total trust and offers total privacy, while M_2 requires no trust and offers no privacy.

PRIVACY AND TRUST AS A STRATEGIC PROBLEM. Other implementations may fall in between the above two extremes, but some loss of privacy appears to be unavoidable in unmediated mechanisms, and some reliance on trust unavoidable in mediated ones. Accordingly, privacy-valuing players may avoid participating in unmediated mechanisms, and distrustful players in mediated ones. In an auction, such reduced participation not only causes the seller to fetch lower prices, but negatively affects efficiency. Indeed, the requirement that “the item be won by the player who values it the most” applies to all potential bidders. Thus, no implementation that deters some players from bidding can be efficient in a general sense. In addition, whenever (for lack of other ways of getting a desired item) distrustful players participate in M_1 or privacy-valuing players participate in M_2 , *neither mechanism can be efficient, even with respect to just the participating players.*

In M_1 , a player who *truly distrusts* the mediator may fear that the price will always be artificially close to the winner’s bid.³ Such a player will not find it optimal to report his true valuation to the mediator; 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. In this setting, therefore, M_1 is not efficient.

In 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 thus make M_2 inefficient as well.

More generally, privacy and trust affect the players’ strategic thinking, making it very hard for *concrete* mechanisms —whether mediated or unmediated— to satisfy the strategic properties of the *abstract* mechanisms they purportedly implement.

NOVELTY AND RELATIVE NATURE OF THE PROBLEM. To preserve efficiency in M_2 in the second-price auction context, 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 the Society’s interest is that it be efficient in the “original” valuations. Integrating privacy and strategic concerns in the design of concrete implementations of normal-form mechanisms is a *new* problem, requiring new techniques and conceptual frameworks.

At a closer look, however, 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

²According to Rothkopf, Teisberg, and Kahn (1990), such concerns explain why second-price auctions are rarely used in practice.

³If player i bids \$1000 dollars and the mediator (auctioneer) announces that i is the winner and must pay \$999 dollars, it is impossible for i to know if there were really a bid of \$999. Certainly, an auctioneer has incentives to manipulate the outcome (e.g., if he is paid a percentage of the generated revenue for his services) and to betray a player’s secret (e.g., if he is offered a bribe for his disclosure).

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: *they should always enjoy efficiency (or any other desired properties) to exactly the same degree (whatever it may be) as the abstract second-price auction.*

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.⁴ That is: *we cast the problem of implementing a normal-form mechanism as that of matching exactly all of its privacy and strategic properties.*

1.2. Our Contributions

We contribute to a rigorous treatment of privacy and trust in mechanism design as follows:

1. We introduce *ballot-box mechanisms*, a new class of unmediated, extensive-form mechanisms concretely playable via ballots and a ballot-box.

Ballots and ballot-boxes are the same devices used, from time immemorial, for running fair lotteries and tallying secret votes. Such venerable devices have previously been part of many specific mechanisms, but today we demonstrate their power by showing that they can be used as universal implementation devices for normal-form mechanisms.

2. We put forward a rigorous definition of what it means for a ballot-box mechanism \mathcal{B} to *implement perfectly* a mediated normal-form mechanism \mathcal{M} in the classical setting.⁵

As we shall see, perfect implementation is a strong but technical notion that captures many a desideratum. In particular, whenever \mathcal{B} perfectly implements \mathcal{M} , the following two main properties are guaranteed:

Strategic Equivalence. For each player i , there is one-to-one correspondence between his strategies in \mathcal{B} and his strategies in \mathcal{M} , and the outcome of any profile of strategies in \mathcal{B} is identical to the outcome of the profile of the corresponding strategies in \mathcal{M} .

Privacy Equivalence. No set of players can gain more information (about the other players' strategies) in \mathcal{B} than they can in \mathcal{M} , where the only information available to them coincides with what is deducible from the mechanism outcome.

3. We prove that *every* mediated normal-form mechanism is perfectly implementable by a ballot-box mechanism.

In general, proving the existence of certain objects may not be helpful in finding them.⁶ Our

⁴When dealing with a sequence of normal-form mechanisms, a carefully designed loss of privacy may actually help to satisfy the strategic goals of the mechanisms yet to be played.

⁵By “classical setting” we mean a study of a single mechanism in isolation from any other interaction—concurrently or in the future. We stress, however, that no restrictions are placed on the players' preferences. The players' beliefs (of any order) about anything relevant for the present as well as future payoffs are not limited in any way.

⁶For instance, Nash equilibria provably exist for any finite normal-form game, but how to find them efficiently remains an open problem.

proof, instead, is constructive in a very strong sense. We exhibit a *universal mechanism translator*: namely, a *single* algorithm that, given the description of *any* normal-form mechanism, *efficiently* outputs the description of a ballot-box mechanism perfectly implementing it.

A GENERAL NOTION. Although presented via ballot-box mechanisms, the notion of a perfect implementation is essentially independent of them. Ballot-box mechanisms are simply attractive because they work in a most intuitive manner and enable us to measure precisely the flow of information during play (which is necessary for the rigorous treatment of privacy). What makes them unique, right now, is that they are universal: namely, a perfect implementation of *any* normal-form mechanism can be found within their class. We do conjecture, however, that other classes of concrete mechanisms capable of universal perfect implementation will be discovered.

PRIVATE OUTCOMES. The notion of perfect implementation and our theorems are stated and proven for normal-form mechanisms with *public outcome functions*, where the outcome of the mechanism is publicly revealed. A more general case is that of *private outcome functions*, where, in addition to the public outcome, each player privately learns his own “piece” of the outcome. For example, a second-price auction can be conducted so that only the winner and the seller learn that they transact and the price, while all other players learn only that they did not win, and neither the identity of the winner nor the price.

The notion of perfect implementation and our ballot-box constructions extend straightforwardly to the case of private outcomes. Because such an extension requires some additional formalization, which, in principle, may depend on how exactly the private information is communicated to the players and additional desiderata on such communication, we chose to formalize it separately. One such reasonable formalization is presented in Appendix I.

A CONTEXT-FREE RESULT. A mechanism \mathcal{M} studied in a *context* \mathcal{C} forms a *game* \mathcal{G} , whose solution ultimately defines \mathcal{M} ’s properties. Contexts describe everything about the players that is relevant to solving the game; in particular, the players’ preferences over the outcomes, and their beliefs about each other. The solutions of the same mechanism in different contexts may enjoy different properties. (For example, the unique Bayesian-Nash equilibrium of the first-price sealed-bid auction is efficient in the symmetric independent private-values setting, and not efficient in settings with any degree of asymmetry.) In principle, therefore, an automatic and universal procedure for perfect implementation might very well consist of an algorithm that, on input an arbitrary mechanism-context pair $(\mathcal{M}, \mathcal{C})$, outputs a concrete mechanism $\mathcal{M}'_{\mathcal{C}}$ “equivalent” to \mathcal{M} in context \mathcal{C} . The applicability of such a procedure, however, would be quite limited: while \mathcal{M} is a simple object (a set of messages and an outcome function), \mathcal{C} may not be. For instance, as part of \mathcal{C} , the distribution on the players’ types may be prohibitively complex to specify.

Our translator is instead *context-free*. Namely, on input an arbitrary normal-form mechanism \mathcal{M} , it generates a concrete mechanism \mathcal{M}' such that, for all possible contexts \mathcal{C} , the players are indifferent between playing \mathcal{M} and playing \mathcal{M}' . In essence, therefore, our translator achieves for mechanism implementation what Wilson (1987) advocates for mechanism design: namely, that one should strive to find solutions minimizing dependence on the details of the problems at hand (the “Wilson’s Doctrine”).

EXACT PRESERVATION OF EQUILIBRIA. Assume that a normal-form mechanism \mathcal{M} has been designed so that, for all contexts \mathcal{C} (of a given class), the resulting game $\mathcal{G} = (\mathcal{M}, \mathcal{C})$ has a specific set of Bayesian-Nash equilibria. Then, when choosing a concrete implementation \mathcal{M}' of \mathcal{M} , we should

aim at preserving such a set of Bayesian-Nash equilibria in the concrete game $\mathcal{G}' = (\mathcal{M}', \mathcal{C})$. In addition, we should aim at preserving all kinds of solutions and not only of a specific type—even if it is a dominant strategy equilibrium—to be able to claim that the resulting outcomes of play of \mathcal{G} and \mathcal{G}' would have been identical. Indeed, Saijo, Cason, and Sjöström (2003) present two games possessing the same set of dominant-strategy but different sets of Nash equilibria that are actually played quite differently in experiments. Thus, ideally, \mathcal{G}' *should not lose any original equilibria and should not introduce any additional ones*.

Perfect implementation does indeed guarantee the exact preservation of all equilibria. Strategic equivalence states that a mediated mechanism \mathcal{M} and its perfect implementation \mathcal{M}' have identical (up to renaming and reordering of strategies) normal-form representations. And traditional equilibrium concepts (such as Nash, Bayesian-Nash, 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. Therefore, if a concrete mechanism \mathcal{M}' perfectly implements a normal-form mechanism \mathcal{M} , then, for all contexts \mathcal{C} and all traditional solution concepts \mathcal{E} , the games $\mathcal{G} = (\mathcal{M}, \mathcal{C})$ and $\mathcal{G}' = (\mathcal{M}', \mathcal{C})$ are guaranteed to have the same set of \mathcal{E} -solutions.⁷

A SIGNALLING-FREE SOLUTION. In a normal-form mechanism, players cannot signal information to one another. By contrast, signalling seems unavoidable in an extensive-form mechanism. Anytime a player A can choose between two or more actions that cause another player B to observe different consequences before having a choice of actions himself, there is an opportunity for A to communicate to B so as to affect the play and usher in new strategic opportunities. To guarantee strategic equivalence, we thus construct our perfect implementations to be signalling-free. Technically, we ensure that the action set available to any player at any point of our ballot-box mechanisms consist of either (1) a single action, or (2) a pair of actions a and b which generates identical observables for all other players. (In the second case, the player must make a choice, but this choice will only affect the final outcome, at which point no choice of actions is available to any player.)

COMPLETE COLLUSION CONTROL. Our strategic and privacy equivalence properties together guarantee that all subsets of players (including the set of all players) have the same strategic opportunities and have the same information in a play of a normal-form mechanism and in a play of its perfect implementation. Therefore, perfect implementations provide no additional incentives for any coalition of players—of any size!—to form.

When considering an abstract normal-form mechanism \mathcal{M} the most a coalition of players C can do is to send a joint sub-profile of messages m_C . Whether players in C can coordinate on such a joint message, whether such a possibility is to be taken into account when solving the game, and what joint strategy players in C would follow ultimately depends on the context in which \mathcal{M} is to be applied. In many a context, having a particular solution concept in mind, such as a dominant strategy equilibrium, the opportunities that are available to coalitions may not even be described as a part of the context. Such ignorance is often supported by an assumption (or fact) that running coalitions is not profitable—i.e., that the costs of running or a threat of punishment if discovered exceed the potential surplus from coordinated actions. A concrete implementation of \mathcal{M} that keeps all dominant strategy equilibria intact, but enriches the options available to coalitions

⁷An example of a solution that is not normal-form invariant is a specific variation of a trembling-hand equilibrium notion where the trembles (mistakes) are specified for *actions* (and not strategies) and are required to be equally likely for all available actions. This will lead to different distributions of trembles on normal-form strategies.

may be highly inadequate, since, it may provide incentives for coalitions to form, thus, changing the outcome of play.

As an example, consider a government that wishes to auction off rights to an oil field using a second-price auction, and suppose that there are five interested buyers B_1, B_2, \dots, B_5 with valuations in between 40 and 60 million dollars. The most the coalition of any three players, say B consisting of B_1, B_2 , and B_3 , can achieve in such an auction is to lower the winning price, if wins, from the second highest valuation to the highest valuation among B_4 and B_5 , which would still exceed 40 million. It may not even be possible for B to achieve this, if no communication among its members is available outside of the auction. Suppose, instead, that in a concrete implementation of M such a coalition B can completely take control of an auction—while “playing by the rules”—guaranteeing that the object is allocated to one of its members at a price of \$1.⁸ Clearly, one may and, perhaps, should expect such an outcome.⁹

(to be removed perhaps) Preserving the power of a coalition is actually an important property, introduced in a weaker sense by Lepinski, Micali, Peikert, and Shelat (2004).

1.3. Complementing Mechanism Design

The practical meaningfulness of any abstraction depends on whether concrete implementations that adequately approximate it exist. In a sense, therefore, our contributions enhance the practical meaningfulness of the very notion of a normal-form mechanism: no matter how “delicate,” the theoretical properties of a normal-form mechanism will continue to hold intact for at least its ballot-box implementation.

We view our results as *complementary* to 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.¹⁰ 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.

(to be added, perhaps) Our results can help simplify analysis of problems in settings of incomplete information, especially in applied fields, where suitable mediators are not readily available

⁸Indeed, based on a particular secure computation model, it is possible to construct such an implementation. See also the discussion on excessive coalitional power in Section 1.4.

⁹As evidence of how additional opportunities available to the players can be found and exploited, as well as the evidence of how the signaling can change the outcome, consider the German GSM spectrum auction of 1999. It has resulted in the outcome that was hardly expected by the authorities (see Grimm, Riedel, and Wolfstetter (2003) for the comprehensive account): two main participants, Mannesmann and T-Mobil, split the available 10 licenses at a very low price. The failure can be directly attributed to the design flow (it was a specific simultaneous ascending price auction format presumably chosen to implement the efficient allocation and Vickrey prices), that let the two buyers coordinate in the process of bidding on the outcome. In the first round, Mannesmann bid for all 10 licences: its bids on licences 1-5 were slightly lower so that if one were to make a minimal raise in the next round the resulting bids would match the Mannesmann’s bids on licences 6-10. In the second round, T-Mobil did exactly that, and no more bids were submitted, resulting in both firms winning 5 licences. Later, T-Mobil commented the were no agreement between firms prior to the auction, and and that the intentions of Mannesmann were clear from its bidding pattern.

¹⁰Sjöström 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.

(e.g., in pairwise bargaining) or, at the other extreme, are available but the conventional analysis is problematic (e.g., an informed principal problem, where the choice of the mechanism by the principal may reveal some of his private information, that, in particular, renders the Revelation Principle invalid (ref)). It might be easier to solve such problems in the ideal setting, where there is an access to a universally trusted mediator, and then, using technics suggested here, to remove such a mediator.

1.4. Prior Work on Privacy and Trust

ZERO KNOWLEDGE AND SIMULATORS. The study of privacy without trust started two decades ago in theoretical computer science; specifically, with the zero-knowledge proofs of Goldwasser, Micali, and Rackoff (1985). Assume that a prover P wishes (1) to prove a mathematical statement S to a verifier V , but also (2) to keep private any detail of the proof. Saying “ S is true” is not convincing, and a classical proof of S , though convincing, would reveal much more than just “ S is true”.¹¹ In contrast, a zero-knowledge proof is an interactive process that enables P to convince any distrusting V that S is indeed true but *does not reveal any additional piece of knowledge*. Zero-knowledge proofs are formalized via the notion of a *simulator*, a conceptual tool that is also used in our paper.

SECURE COMPUTATION. A direct predecessor of our notion of perfect implementation is *secure computation*, as defined (and computationally achieved) by Goldreich, Micali, and Wigderson (1987), improving on a two-party result of Yao (1986). They showed that n parties, each possessing a secret input, can *securely evaluate any function f* on their inputs. That is, the n parties can talk back and forth so as to compute the output of f on their secret inputs with essentially the same correctness and privacy as if they privately handed their inputs to a trusted party, who would then secretly evaluate f and announce the result. Such privacy and correctness should hold even if some players deviate from their prescribed communication instructions. Specifically, secure computation protects against malicious *monolithic coalitions* — i.e., groups of perfectly coordinated players. To model the perfect coordination of a group of players C , secure computation envisage a single entity A , *the adversary*, that controls the members of C , before, during, and after the evaluation. (In particular, each message received by a member of the coalition C , during the communication protocol, is actually received by A , and any message sent by a member of C is actually chosen by A .) The way secure computation bounds the power of a monolithic coalition C is by guaranteeing that —as long as the players *not in C* stick to their communication instructions— whatever C can achieve in a secure evaluation of f (i.e., whatever influence C can have on f 's output, and whatever knowledge C can gather about the inputs of the other players) the same C can achieve in a mediated evaluation of f .

Secure computation has been achieved in two main models of communication. In the first one, the original one of Goldreich, Micali, and Wigderson (1987), the players communicate by ordinary broadcast, but use encryption to guarantee privacy. (Thus, their security properties hold only for computationally bounded adversaries, who cannot crack such encryptions.) In the second one, put forward by Ben-Or, Goldwasser, and Wigderson (1988) and Chaum, Crépeau, and Damgård (1988),

¹¹For instance, a proof starting with 0 reveals that S has a proof starting with 0, which needed not be the case. A bit more concretely, providing two large primes p and q whose product equals n is a proof of the statement $S = “n$ is the product of two primes.” But such proof appears to contain much more knowledge than the mere statement “ n is the product of two primes” (whatever they may be)!

the players communicate via perfectly private channels. That is, it is envisaged that between each pair of players i and j there exists a dedicated “lead pipe” that enable i and j to exchange messages so that no one else can see, alter, or gain any information about what they say to each other, or whether they are communicating at all. (The privacy guaranteed in this model is information theoretic, rather than computational.) In the first model security is guaranteed against coalitions of any size except of the grand coalition, while in the second model only against coalitions comprising less than a third of the players. Rabin and Ben-Or (1989) show that any coalition containing less than half of the players can be protected against in a communication model comprising both broadcast and private channels.

STRATEGIC LIMITATIONS OF SECURE COMPUTATION. A protocol for securely evaluating a function f is, essentially, a concrete, extensive-form mechanism. It is thus very natural to consider implementing a normal-form mechanism \mathcal{M} by the following concrete mechanism \mathcal{M}_{sc} : the players first choose their reports, and then securely compute on them the outcome function of \mathcal{M} .

Unfortunately, this appealingly simple approach does not work since, as it turns out, all such concrete mechanisms are strategically richer than the ideal normal-form ones. The more specific reasons, some of which are potentially correctable, are listed below.

Honesty. The very notion of secure computation is stated relative to the honesty of some players. A honest player is one always following his instructions, and secure computation bounds the power of a coalition C , only if the other players are honest. Honesty is clearly at odds with rationality: a rational player will surely deviate from his prescribed instructions whenever it is advantageous for him. Nonetheless, though explicitly part of the definition of secure computation, honesty by itself is only superficially problematic to the above mechanism \mathcal{M}_{sc} . For instance, in the secure-computation protocol of Goldreich, Micali, and Wigderson (1987), each player is required, for any encrypted message he sends, to give a (zero-knowledge) proof that the underlying clear-text message is actually in compliance with his prescribed strategy. If a player fails to provide such a proof, then he is immediately identified. Accordingly, if \mathcal{M}_{sc} is enriched so that a sufficiently high fine is imposed on such a player, it will be rational for him to comply with his prescribed instructions.

Signaling. Very crucially and very intrinsically, mechanism \mathcal{M}_{sc} , *no matter on which of the secure-computation protocols known today it based upon*, fundamentally alters the strategic opportunities available to the players in \mathcal{M} . Despite the fact that secure computation bounds coalitional power, all known secure computation protocols allow *uncontrolled and undetected signaling* among the players.

This is self-evident in all secure-computation protocols relying on private channels: by assumption such channels allow for free and undetectable communication among the players. Uncontrolled signalling still occurs, but more subtly, in all other secure-computation protocols, because their players execute probabilistic strategies and exchange messages with positive “entropy.” Whenever this is the case, Hopper, Langford, and von Ahn (2002) prove that any pair of players can implement a covert channel of communication, so called *steganographic communication*. That is, while exchanging their prescribed messages, any two players may also exchange additional information without anyone else (even a computationally unbounded observer) noticing it. By means of these covert channels, the players can signal to each other, and thus gain additional strategic opportunities not present in \mathcal{M} . The very inability of detecting such signalling rules out any possibility of punishing signalling players (so as to make it irrational to engage in steganographic communi-

tion). In sum, no implementations of normal-form mechanisms based on known secure-computation protocols can satisfy our notion of strategic equivalence.

The only way to control signaling in secure computation is provided by the recent work of Lepinski, Micali, and Shelat (2005). Roughly, while they cannot prevent signalling in the secure computation of an outcome function, they can ensure that it is “no more powerful than a pre-play cheap-talk conversation.” In essence, they show how simulate the following 4-stage mediated mechanisms: (1) the players freely engage in cheap talk; (2) each player, now locked in his own room, chooses his report and submits it to the mediator; (3) the mediator applies the outcome function to the submitted reports; and (4) the mediator broadcasts the public outcome, and privately delivers to each player his private outcome.

Excessive coalitional power. All of the existing secure-computation protocols allow some coalitions to take complete control of the protocol. In protocols that rely on private channels any coalition consisting of a majority of the players can act so that to select specific messages for all of the players, even those not in the coalition. (In the context of a second-price auction this would mean selecting bids for all the bidders in the auction.) In addition, in all protocols, the grand coalition, consisting of all the players, can reach any feasible outcome. It is the assumption of honesty of some players that makes such coalitions infeasible. If honesty is substituted by rationality, such an excessive power to some coalitions becomes problematic, both as a security flaw and of a strategic concern—a coalition of players that can reach an outcome that can potentially be much preferred by each of its members to anything such a coalition can obtain in an ideal mediated computation cannot simply be ignored!

This paper can be interpreted as providing a new and stronger concept of secure computation that is based solely on rationality of the players and offers protection against coalitions of any size.¹²

1.5. Other Related Work

Privacy of information was certainly recognized in earlier economics papers, but from a normative perspective, not one of mechanism design.¹³ Closer to our work, trusted-party simulation has further been investigated in the following three directions.

SPECIAL CASES. All the literature on pre-play communication games achieving correlated equilibrium, communication equilibrium, or Bayesian-Nash equilibrium, can be viewed as replacing the mediator in the evaluation of *a specific class of probabilistic functions* f . See in particular, Barany (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 (2005), Krishna (2006). In addition, the emphasis of all these papers is on preserving a specific equilibrium of the original game, without concerns about other equilibria that may be generated in the process.

Halpern and Teague (2004) —see also Gordon and Katz (2006) and Lysyanskaya and Triandopoulos (2006)— use secure function evaluation to replace the mediator, for any function but *for a specific class of incentives*. (In essence, they require that the preferences are such that the function

¹²Izmalkov, Lepinski, and Micali (2005) cast an earlier version of our results in terms of secure computation.

¹³The special issue “The Law and Economics of Privacy” of The Journal of Legal Studies (1980, Vol. 9(4)) and Posner (1981) address the question of and under which circumstances should private information be collected and publicly released. Stigler (1980) provides several examples of economic phenomena that are linked to privacy concerns: reputation, blackmail, industrial secrecy and espionage.

to be evaluated is dominant-strategy incentive compatible.) Again, no consideration is given to the introduction of additional equilibria.

By contrast, we replace the mediator *for all functions* and *for all possible preferences* of the players, and *keep all equilibria intact*.

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 with an unmediated interaction of the players alone crucially depends on the task at hand and the means of interaction available to the players. Popular means of interactions include broadcasting messages, privately sending messages by phone, encrypted channels, or via envelopes. A more sophisticated type of interaction is the simultaneous broadcast (so that, if i and j broadcast at the same time, neither one of them can change his message mid-stream depending on what he hears from the other).

Replacing a mediator for all possible outcome functions and in all contexts is somewhat counter-intuitive. Thus, we should expect that for most reasonable models of interaction, such a general replacement is not possible. In fact, this replacement has been proved impossible in a formal sense, in many specific interaction models, *even for a restricted class of outcome functions*. Notably, Aumann and Hart (2003) prove that two players cannot reach any non-trivial *correlated equilibrium* via cheap talk (essentially, via ordinary broadcast). 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 replacing the mediator is possible for all outcome functions and all contexts.

2. THE INTUITIVE NOTION OF A BALLOT-BOX MECHANISM

Ballot-box mechanisms are extensive-form, imperfect-information mechanisms with Nature. Accordingly, to specify them we must specify who acts when, the actions and the information available to the players, when the play terminates, and how the outcome is determined upon termination.

A ballot-box mechanism ultimately is a mathematical abstraction, but possesses a quite natural physical interpretation. The physical setting is that of a group of players, seated around a large table, acting on a set of *ballots* with the help of a randomizing device, the *ballot-box*. Within this physical setting, one has considerable latitude in choosing natural actions available to the players. In this paper, we make a specific choice, sufficient for our present goals. In future papers, one may make different choices, so as to achieve different goals, and still deal with a ballot-box mechanism.

BALLOTS. There are two kinds of ballots: *envelopes* and *super-envelopes*. Externally, all ballots of the same kind are identical, but super-envelopes are slightly larger than envelopes. An envelope may contain a symbol from a finite alphabet, and a super-envelope may contain a sequence of envelopes. (Our constructions actually needs only envelopes containing an integer between 1 and 5, and super-envelopes capable of containing at most 5 envelopes.) An envelope perfectly hides and guarantees the integrity of the symbol it contains until it is opened. A super-envelope tightly packs the envelopes it contains, and thus keeps them in the same order in which they were inserted. Initially, all ballots are empty and in sufficient supply.

BALLOT-BOX ACTIONS. There are 8 classes of actions in a ballot-box mechanism. Members of the first 6 classes are *public actions* and members of the 7th class are *private actions*, all such actions are taken by the players. Actions of the 8th class are taken by Nature, and are thus referred to as *actions of Nature*.

Public actions are performed in plain view, and every component of them is observable by all players. The classes of public actions are: (1) publicly writing a symbol on a piece of paper and sealing it into a new, empty envelope; (2) publicly opening an envelope to reveal its content to all players; (3) publicly sealing a sequence of envelopes into a new super-envelope; (4) publicly opening a super-envelope to expose its inner envelopes; (5) publicly reordering a sequence of envelopes; (6) publicly renaming an envelope. (The last two types of actions are purely a syntactic convenience for describing our protocols.) Notice that a public action has the same effects, no matter which player performs it.

Private actions involve a component that is solely known to the player performing them. For every pair of ballots of the same kind (i.e., either two envelopes or two super-envelopes containing the same number of envelopes), there are two possible private actions. To perform the 0-action, a player i picks up both ballots, hides them behind his back, and then returns them to the table in the same order. To perform the 1-action, i picks up both ballots, hides them behind his back, and then returns them in the opposite order. Because the other players cannot tell whether the order has been changed, i is effectively choosing a secret bit. We call two private actions *complementary* if they are the 0-action and 1-action for the same pair of ballots.

An action of Nature consists of the ballot-box secretly randomizing the order of a publicly chosen sequence of ballots, and thus involves a permutation hidden from all players.

PUBLIC AND PRIVATE INFORMATION. To keep track of the ballots, each ballot on the table is associated with a unique identifier j , a positive integer that is common information to all players. These identifiers correspond to the order in which the ballots are placed on the table for the first time, or returned to the table after being picked up and permuted —by the ballot box, by a private action, or by a public reordering. A player can publicly change a ballot's identifier as one of his actions.

Each action, when played, generates a publicly available string. In the case of a public action A on ballots j, k, l, \dots , this string consists of " A, j, k, l, \dots ". A private action additionally generates a secret bit available only to the acting player. If player i secretly re-orders ballots j and k , then the public string consists of "a re-ordering of ballots j and k has occurred" and i 's secret bit corresponds to the specific re-ordering he chose. The *public record* is the concatenation of the public strings generated by all actions played thus far. Similarly, for each player i , i 's *private record* is the concatenation of the bits corresponding to all private actions played by i .

MECHANISM PLAY. In a ballot-box mechanism, players and Nature act one at a time, in a pre-specified order. The information available to the acting player consists of the current public record and his current private record. (Because players act in a fixed order, the public record implicitly determines which action was taken by which player.) The actions available to the acting player are a fixed function of the public record.

The mechanism terminates when all players in the fixed sequence have acted, and the outcome is a pre-specified function of the final public record.

3. THE INTUITIVE NOTION OF A PERFECT IMPLEMENTATION

A *perfect implementation* of a mediated normal-form mechanism \mathcal{M} is defined to be a ballot-box mechanism \mathcal{B} that preserves the logical structure of \mathcal{M} . By exactly capturing such a structure we guarantee that \mathcal{B} is strategically and privacy equivalent to \mathcal{M} .

3.1. A Structural Definition

In this paper we deal only with *standard mediated mechanisms*; that is, mediated normal-form mechanisms whose outcome function has finite domain and range, and whose possible messages have the same length. (If they do not, we can artificially pad them so that they become of the same length.)

A standard mediated mechanism consists of 3 conceptual stages: (1) a *commitment stage*, in which each player—independently and simultaneously with the others—sends to the mediator his message string; (2) a *computation stage*, in which the mediator evaluates the outcome function on the received messages; and (3) a *revelation stage*, in which the mediator publicly announces the resulting outcome.

Accordingly, a ballot-box mechanism \mathcal{B} perfectly implementing a standard mediated mechanism \mathcal{M} consists of 3 *ballot-box protocols* (i.e., 3 components), respectively named the *ballot-box committer*, the *ballot-box computer*, and the *ballot-box revealer*, corresponding to the 3 stages of \mathcal{M} .¹⁴ When \mathcal{M} 's messages are L -bit long and its outcome function is g , \mathcal{B} 's ballot-box protocols are required to have the following functionalities.

1. The ballot-box committer is executed first and results in each player i having an input m_i . The players are free to choose whatever inputs they like, but are then “committed” to them: by the end of the stage, for each player i , there is a separate sequence of envelopes whose contents encode m_i .
2. The ballot-box computer is executed next and transforms the envelopes encoding m_1, \dots, m_n into envelopes encoding the outcome $g(m_1, \dots, m_n)$.
3. The ballot-box revealer is executed last and solely consists of opening the envelopes whose contents encode the outcome: by decoding these now public contents, each player individually computes $g(m_1, \dots, m_n)$.

Further, \mathcal{B} 's ballot-box protocols must satisfy the following properties.

Minimum Information. The first two protocols reveal no information about the inputs; and the third one reveals only the outcome. (This information requirement matches that of \mathcal{M} .)

Minimum Choice. In the first protocol of \mathcal{B} , for each player i , there are exactly L rounds (corresponding to the L bits of i 's message) in which i is active and his action set consists of two complementary private actions. In all other rounds of \mathcal{B} , the action set consists of a *single* action: either a public one or an action of Nature. (This requirement matches the fact that in \mathcal{M} each player chooses his own L bits and then makes no other strategic choices.)

¹⁴Ballot-box protocols essentially are ballot-box mechanisms which may start with a non-empty set of ballots, or produce no outcome.

Bit-By-Bit Encoding. All three protocols use the same encoding scheme, which encodes a binary string by concatenating the encodings of its individual bits. (This technical requirement matches the fact that the players and the mediator of \mathcal{M} “talk in the same language”, and in this language a message is the concatenation of its individual bits.)

Perfect implementation is formalized in our appendix, where we also prove the following result.

THEOREM 1: *For every standard mediated mechanism \mathcal{M} , there exists a ballot-box mechanism \mathcal{B} that perfectly implements \mathcal{M} .*

3.2. Strategic and Privacy Equivalence

Let us now sketch what it means for two mechanisms to be strategically and privacy equivalent.

Informally, an extensive-form (imperfect-information) mechanism \mathcal{B} is *strategically equivalent* to a standard mediated mechanism \mathcal{M} if “there exist isomorphisms between the players’ strategies in the two mechanisms that preserve the outcomes.” That is, no matter which strategies the players may select in \mathcal{B} , they are guaranteed to have corresponding strategies in \mathcal{M} that yield exactly the same outcomes. Viceversa, for each possible way to play \mathcal{M} , the players are guaranteed to have a corresponding way to play \mathcal{B} which produces an identical outcome. Importantly, this correspondence is not between strategy profiles, but between the strategies of each *individual* player. That is, every player i can “translate” his own strategy from one mechanism to the other without any cooperation from the others. Thus the players truly are strategically indifferent between playing \mathcal{M} or \mathcal{B} .

Assuming that a ballot-box mechanism \mathcal{B} is strategically equivalent to a standard mediated mechanism \mathcal{M} , we further say that \mathcal{B} is *privacy equivalent* to \mathcal{M} if, for any subset P of the players and any strategy profile s , the sequences of decision nodes the players in P go through in an execution of \mathcal{B} , with profile s , can be generated from just the final outcome and s_P , the strategy sub-profile of P .¹⁵ This technical property can be interpreted as follows. At the end on an execution of \mathcal{M} , the players of P collectively know: (1) the strategies they used (i.e., the messages they sent to the mediator) and (2) the final outcome. At the end of an execution of \mathcal{B} the players of P collectively know: (1′) the strategies they used and (2′) the sequences of decision nodes they went through.¹⁶ Because \mathcal{B} is strategically equivalent to \mathcal{M} , (1′) and (2′) suffice to reconstruct the outcome, but in principle could contain *additional* information. However, when \mathcal{B} is privacy equivalent to \mathcal{M} they cannot. Therefore, all players are indifferent between playing \mathcal{M} and \mathcal{B} also from a privacy perspective.

Strategic and privacy equivalence are formalized in our appendix, where we also prove the following result.

THEOREM 2: *If a ballot-box mechanism \mathcal{B} perfectly implements a standard mediated mechanism \mathcal{M} , then \mathcal{B} is strategically and privacy equivalent to \mathcal{M} .*

¹⁵Actually we prove a much stronger property when \mathcal{B} is a perfect ballot-box implementation of \mathcal{M} : informally, that there exists a probabilistic algorithm that on just the outcome information —without any knowledge of the total profile of strategies s that generated it— reproduces a “fictitious” public record with the same probability distribution with which s would generate it.

¹⁶Note that, due to the strategic equivalence of \mathcal{M} and \mathcal{B} , (1) and (1′) are actually equivalent.

4. ADDITIONAL PROPERTIES OF OUR IMPLEMENTATION

4.1. Modularity

Modularity enables us to understand and to design complex systems by breaking them into a multiplicity of simpler components. As such, it is central to Science and Engineering. In Science, the usefulness of modularity is eloquently exemplified by our extensive use of lemmas in long proofs. As for Engineering, the very computer used in typesetting this paper has been manufactured by integrating together hundreds of individual components, often produced by separate firms.

Perfect implementation extends modularity to mechanism design as well. Informally, this means that a perfect implementation of a standard mediated mechanism \mathcal{M} can always be obtained from perfect implementations of simpler mechanisms. Notice that designing ballot-box committers is not a problem: after finding a way to commit to a single bit, repeating it L times enables the players to commit to L bits. Designing ballot-box revealers is not a problem either: they simply consist of opening a fixed sequence of envelopes. The only challenge is that of designing ballot-box computers. And it is here that modularity is crucial.

Modularity is formalized in our appendix, where we also prove the following result.

THEOREM 3: *If g is a modular composition of functions g_1, \dots, g_k , where each g_j has a ballot-box computer G_j , then g has a ballot-box computer G that is a modular composition of G_1, \dots, G_k .*

4.2. Universality and Computational Efficiency

In principle, the existence of a ballot-box mechanism perfectly implementing a mediated one may be highly non-constructive. If this were the case, finding a perfect implementation would require a new ad hoc construction for each mediated mechanism of interest. We dispel this possibility by providing a *universal method* of finding perfect ballot-box implementations. That is, we exhibit a single algorithm \mathcal{C} converting (the description of) any standard mediated mechanism into (the description of) a ballot-box mechanism perfectly implementing it.

Such a universal method, however, would be practically useless if it were computationally infeasible to carry out. Traditionally, players' resources (for reasoning and computing) are assumed to be unbounded in game theory, but bounded in cryptography. Our construction, however, achieves the best of both worlds. Namely, *perfect implementation applies to players with unlimited resources, but is achievable by players with limited ones*. For any standard mediated mechanism \mathcal{M} , not only do ballot-box mechanisms perfectly implementing \mathcal{M} exist, but ours are also (1) *very easy to find*, and (2) *very easy to play*. That is, our converting algorithm \mathcal{C} is very efficient, and so are the ballot-box mechanisms that it produces.

Mechanism efficiency and representation are formalized in our appendix, where we also prove the following result.

THEOREM 4: *There exists a linear-time algorithm \mathcal{C} that, on input any standard mediated mechanism \mathcal{M} , outputs a linear-time ballot-box mechanism \mathcal{B} that modularly implements \mathcal{M} .*

THE SECRET OF UNIVERSALITY. Our universal construction has roots in the earlier ones of Gödel and Turing, in logic and computation respectively. In all these contexts, universality stems from the ability of “equating subjects and objects.” In the case of Gödel numberings, for example, the coherence of Peano's arithmetics is itself encoded as a sentence in such arithmetics. In our case, we represent data and operations as permutations, physically embodied as sequences of envelopes.

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 is a procedure that, given two sequences of 5 envelopes —the first sequence encoding a permutation of 5 elements, p , and the second another permutation, q — interprets the first sequence a multiplication program and returns (without revealing any information about p or q) a sequence of 5 envelopes encoding the product permutation pq (i.e., the permutation mapping each i between 1 and 5 to $p(q(i))$).

5. CONCLUSIONS AND EXTENSIONS

People care about privacy. To make more accurate predictions and to develop more meaningful models, this fact should intrinsically inform any general study of human interactions. By disregarding privacy we may fail to reach our objectives or severely restrict the applicability of our frameworks. In particular, therefore, we sincerely hope that privacy will be investigated as an integral part of game theory.

In this paper, we have started such an investigation by implementing any *single* normal-form mechanism without altering its strategic and privacy properties. Human interactions, however, are much richer. In a forthcoming paper, we generalize the present results so as to implement perfectly *multiple* mediated mechanisms. Such mechanisms may be more complex: that is, their mediators might produce private outcomes in addition to or instead of a public outcome.¹⁷ Such mechanisms may also be interdependent in an arbitrary way: that is, they can be executed simultaneously or sequentially, and their mediators may privately communicate with each other.¹⁸

Department of Economics, Massachusetts Institute of Technology, 50 Memorial Drive, E52-252C, Cambridge, MA 02142; izmalkov@mit.edu;

Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 32-G678, Cambridge, MA 02142; lepinski@csail.mit.edu;

and

Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 32-G644, Cambridge, MA 02142; silvio@csail.mit.edu.

A. NOTATION

Basics. We denote by \mathbb{R}^+ the set of non-negative reals; by Σ the alphabet consisting of English letters, arabic numerals, and punctuation marks; by Σ^* the set of all finite strings over Σ ; by \perp a symbol not in Σ ; by SYM_k the group of permutations of k elements; by $x := y$ the operation that assigns value y to variable x ; by ϕ the empty string; and by \emptyset the empty set. If A is a set, by A^0 we denote the empty set, and by A^k we denote the Cartesian product of a set A with

¹⁷In particular, our extended results immediately yield second-price auctions in which “winner and price” are solely revealed to the winner —and the seller— and “you lost” to all other players.

¹⁸For instance, in implementing two auctions one after the other, the first mediator may announce the winner and the second highest bid, but pass on to the second mediator 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.

itself k times. If x is a sequence, we denote by x^i the i th element of x . If x is a sequence of k elements and i and j are positive integers such that $i < j \leq k$, then by $x^{(i,j]}$ and $x^{[i,j]}$ we respectively denote the sequences x^j, \dots, x^k and x^{j+1}, \dots, x^k . If x is a sequence of k integers, and m is an integer, by $x + m$ we denote the sequence $x^1 + m, \dots, x^k + m$. If x and y are sequences, respectively of length j and k , we denote by $x \circ y$ the sequence of $j + k$ elements obtained by appending y to x (i.e., the i th element of $x \circ y$ is x_i if $i \leq j$, and y_{i-j} otherwise). If x and y are strings, we denote the concatenation of x and y by either xy or $x|y$.

Players and profiles. We always denote by N the (finite) set of players, and by n its cardinality. If i is a player, $-i$ denotes the set of the other $n - 1$ players, that is, $-i = N \setminus \{i\}$. Similarly, if $C \subset N$, then $-C$ denotes $N \setminus C$. A profile is a vector indexed by N . If x is a profile, then, for all $i \in N$ and $C \subset N$, x_i is i 's component of x and x_C is the sub-profile of x indexed by C ; thus: $x = (x_i, x_{-i}) = (x_C, x_{-C})$. We say that a profile x satisfies property P if each component of x satisfies P .

Probability distributions.

If $X : S \rightarrow \mathbb{R}^+$ is a distribution over a set S , we denote its support by $[X]$, that is, $[X] = \{s \in S : X(s) > 0\}$. We consider two distributions X and Y equal if $[X] = [Y]$ and $X(s) = Y(s)$ for all $s \in [X]$. We denote by $\text{rand}(S)$ the uniform distribution over S . If X is a distribution, $x \leftarrow X$ denotes the operation that assigns to variable x an element of $[X]$ selected according to X . If f, g, \dots are functions and X, Y, \dots are distributions, by $\langle f(x, y, \dots), g(x, y, \dots), \dots : x \leftarrow X; y \leftarrow Y \dots \rangle$ we denote the distribution generated by the following experiment: first select x according to X , second select y according to Y , and so on; then return the sequence $f(x, y, \dots), g(x, y, \dots), \dots$. If A is a probabilistic algorithm, $A(x)$ denotes the distribution over A 's outputs generated by A on input x . If A is a no-input algorithm, the distribution over its outputs is denoted by \mathbf{A} . A probabilistic function $f : X \rightarrow Y$ is *finite* if X and Y are both finite sets and, for every $x \in X$ and $y \in Y$, the probability that $f(x) = y$ has a finite binary representation.

B. MEDIATED AND BALLOT-BOX MECHANISMS

B.1. Standard Mediated Mechanisms and Normal-Form Mechanisms

DEFINITION 1: A *mediated normal-form mechanism* is a tuple $\mathcal{M} = (N, M, Y, g)$, where:

- N , the *set of players*, is a finite set;
- M , the *message space*, is the Cartesian product of n subsets of Σ^* , $M = M_1 \times \dots \times M_n$;
- Y , the *outcome set*, is a subset of Σ^* ; and
- g , the *outcome function*, is a probabilistic function, $g : M \rightarrow Y$.

We refer to each M_i as the *message space* (or *strategy set*) of player i , and to each $m \in M_i$ as a *message* (or *strategy*) of i . \mathcal{M} is *finite* if M and Y are finite sets and g is a finite function.

DEFINITION 2: A mediated normal-form mechanism $\mathcal{M} = (N, M, Y, g)$ is a *standard mediated mechanism* if $M_1 = \dots = M_n = Y = \{0, 1\}^L$ for some integer L ; that is, $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$.

REMARK. Without loss of generality, we can focus solely on standard mediated mechanisms. Indeed, any finite, mediated normal-form mechanism $\mathcal{M} = (N, M, Y, g)$ can be put into standard form as follows. Let \max be the cardinality of the largest set among M_1, \dots, M_n , and Y . Choose $L = \lceil \log(\max) \rceil$. 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 $\{0, 1\}^L$ and consider any string $x \in \{0, 1\}^L$ 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 $\{0, 1\}^L$. Define now $g' : (\{0, 1\}^L)^n \rightarrow \{0, 1\}^L$ 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 $g'(x'_1, \dots, x'_n) = y'$ if and only if $g(x_1, \dots, x_n) = y$. Thus, $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g')$ is a standard mediated mechanism equivalent to \mathcal{M} .

B.2. Ballot-Box Mechanisms

DEFINITION 3: An *envelope* is a triple $(j, c, 0)$, where j a positive integer, and c a symbol of Σ . A *super-envelope* is a triple (j, c, L) , where both j and L are positive integers, and $c \in \Sigma^L$. A *ballot* is either an envelope or a super-envelope. If (j, c, L) is a ballot, we refer to j as its *identifier*, to c as its *content*, and to L as its *level*. (As we shall see, L represents the number of inner envelopes contained in a ballot.)

Let B be a set of ballots, j an integer, and j_1, \dots, j_m a sequence of integers. Then, by $\text{cont}_B(j)$ we denote the symbol $c \in \Sigma$, if B has a single envelope with identifier j and content c , and the symbol \perp otherwise; and by $\text{cont}_B(j_1, \dots, j_m)$ we denote the symbol \perp if $\text{cont}_B(j_k) = \perp$ for some element j_k , and the string $\text{cont}_B(j_1) \dots \text{cont}_B(j_m)$ otherwise.

A set of ballots B is *well-defined* if distinct ballots have distinct identifiers (i.e., $c = c'$ and $l = l'$ whenever (j, c, l) and (j, c', l') belong to B). If B is a well-defined set of ballots, by B_j or the expression *ballot j* we denote the unique ballot of B whose identifier is j , and by I_B the set of identifiers of B 's ballots. To emphasize that ballot j actually is an envelope (super-envelope) we may use the expression *envelope j* (*super-envelope j*).

DEFINITION 4: A *global memory* for a set of players N consists of a triple (B, R, H) , where

- B is a well defined set of ballots;
- R is a sequence of strings in Σ^* , $R = R^1, R^2, \dots$; and
- H is a profile of sequences of bits, $H = H_1, \dots, H_n$.

We refer to B as the *ballot set*; to R as the *public record*; to each element of R as a *record*; to H as the *hidden-bit profile*; and to each H_i as the *hidden-bit sequence of player i* . The *empty global memory*, is the global memory for which the ballot set, the public record, and the hidden-bit sequence of every player are all empty. We denote the set of all possible global memories by GM .

DEFINITION 5: Ballot-box actions are functions from GM to GM . The subset of ballot-box actions available at a given global memory gm is denoted by \mathcal{A}_{gm} . The actions in \mathcal{A}_{gm} are described below, grouped in 8 classes. For each $a \in \mathcal{A}_{gm}$ we provide a formal identifier; an informal reference (to facilitate the high-level description of our constructions); and a functional specification. If $gm = (B, R, H)$, we actually specify $a(gm)$ as a program acting on variables B, R , and H , plus the auxiliary variable *maxid*, the maximum identifier in I_B .

1. (NEWENV, c) —where $c \in \Sigma$.
“Make a new envelope with public content c .”
 $R := R \circ (\text{NEWENV}, c)$; $\text{maxid} := \text{maxid} + 1$; and $B := B \cup \{(\text{maxid}, c, 0)\}$.
2. (OPENENV, j) —where j is an envelope identifier in I_B .

- “Publicly open envelope j to reveal content $cont_B(j)$.”
 $R := R \circ (\text{OPENENV}, j, cont_B(j))$ and $B := B \setminus \{B_j\}$.
3. $(\text{NEWSUPER}, j_1, \dots, j_L)$ —where j_1, \dots, j_L are distinct envelope identifiers in I_B .
 “Create a new super-envelope containing the envelopes j_1, \dots, j_L .”
 $R := R \circ (\text{NEWSUPER}, j_1, \dots, j_L); maxid := maxid + 1;$
 $B := B \cup \{(maxid, (cont_B(j_1), \dots, (cont_B(j_L)), L))\};$ and $B := B \setminus \{B_{j_1}, \dots, B_{j_L}\}$.
4. $(\text{OPENSUPER}, j)$ —where $j \in I_B$ is the identifier of a super-envelope of level L .¹⁹
 “Open super-envelope j to reveal the envelopes $maxid + 1, \dots, maxid + L$.”
 $R := R \circ (\text{OPENSUPER}, j);$ letting $cont_B(j) = (c_1, \dots, c_L),$
 $B := B \cup \{(maxid + 1, c_1, 0), \dots, (maxid + L, c_L, 0)\};$ $B := B \setminus \{B_j\};$ and $maxid := maxid + L.$
5. $(\text{PUBLICPERMUTE}, j_1, \dots, j_K, p)$ —where $p \in \text{SYM}_K$ and $j_1, \dots, j_K \in I_B$ are distinct identifiers of ballots with the same level L .
 “Publicly permute j_1, \dots, j_K according to p .”
 $R := R \circ (\text{PUBLICPERMUTE}, j_1, \dots, j_K, p);$ $B := B \cup \{(maxid + 1, cont_B(j_{p(1)}), L), \dots, (maxid + K, cont_B(j_{p(K)}), L)\};$ $B := B \setminus \{B_{j_1}, \dots, B_{j_K}\};$ and $maxid := maxid + K.$
6. (RENAME, j, k) —where j is an envelope identifier in I_B and $k \notin I_B$.
 “Rename envelope j as envelope k ”
 $R := R \circ (\text{RENAME}, j, k);$ letting $cont_B(j) = c,$ $B := B \cup \{(k, c, 0)\};$ $B := B \setminus \{B_j\};$ if $k > maxid$ then $maxid := k.$
7. $(\text{SECRETPERMUTE}, i, b, j_0, j_1)$ —where i is the active player, $b \in \{0, 1\}$, and $j_0, j_1 \in I_B$ are distinct identifiers of ballots with the same level L .
 “Let Player i secretly permute ballots j_0 and j_1 according to b .”
 $R := R \circ (\text{SECRETPERMUTE}, i, j_0, j_1);$ $H_i := H_i \circ b;$ $B := B \setminus \{B_{j_0}, B_{j_1}\};$
 $B := B \cup \{(maxid + 1, cont_B(j_b), L), (maxid + 2, cont_B(j_{1-b}), L)\};$ and $maxid := maxid + 2.$
8. $(\text{BALLOTBOX}, j_1, \dots, j_K)$ —where $j_1, \dots, j_K \in I_B$ are distinct identifiers of ballots with the same level L .
 “Ballotbox j_1, \dots, j_K ”
 $R := R \circ (\text{BALLOTBOX}, j_1, \dots, j_K);$ $p \leftarrow rand(\text{SYM}_K);$ $B := B \setminus \{B_{j_1}, \dots, B_{j_K}\};$ $B := B \cup \{(maxid + p(1), cont_B(j_1), L), \dots, (maxid + p(K), cont_B(j_K), L)\};$ and $maxid := maxid + K.$

We refer to a member of the second class as an *open-envelope action*; to a member of the first 6 classes as a *public action*; to a member of the 7th class as a *private action*; and to a member of the 8th class as an *action of Nature*. If $a = (\text{SECRETPERMUTE}, i, b, j_0, j_1)$, we refer to b as a 's *hidden bit*. We say that a and a' are *complementary private actions (for player i)* if $a = (\text{SECRETPERMUTE}, i, b, j_0, j_1)$ and $a' = (\text{SECRETPERMUTE}, i, 1 - b, j_0, j_1)$; and refer to a as the *0-action* (of the pair) if $b = 0$, and as the *1-action* otherwise.

¹⁹All the ballot-box actions involving multiple super-envelopes require as inputs and produce as outputs the ballots of the same level (see below). Thus, the level of any ballot can be deduced from the public record.

REMARK. All ballot-box actions are deterministic functions, except for the actions of Nature.

DEFINITION 6: A global memory gm is *feasible* if there exists a sequence of global memories gm^0, gm^1, \dots, gm^k , such that gm^0 is the empty global memory, $gm^i = a^i(gm^{i-1})$ for some $a^i \in \mathcal{A}_{gm^{i-1}}$ for all i , and $gm^k = gm$.

REMARK. If $gm = (B, R, H)$ is feasible, then \mathcal{A}_{gm} is easily computable from R alone. (Indeed, what ballots are in play, which ballots are envelopes and which are super-envelopes, etc., are all deducible from R .) Therefore, different feasible global memories that have the same public record also have the same set of available actions. This motivates the following definition.

DEFINITION 7: We denote by \mathcal{A}_R the set of all actions available at any global memory that has public record R .

DEFINITION 8: An ℓ -bit *ballot-box mechanism* is a tuple $\mathcal{B} = (N, K, J, PS, AF, OF)$ where

- N is a set of players;
- K , the *mechanism length*, is a positive integer;
- J , the *outcome-boundary round*, is a positive integer $\leq K$;
- PS , the *player sequence*, is a sequence of K elements from $N \cup \{\text{Nature}\}$: $PS = PS^1, \dots, PS^K$;
- AF , the *action-function sequence*, is a sequence of K functions such that each AF^k maps any public record R to a subset of \mathcal{A}_R so that: (1) $AF^k(R)$ consists of either a single action, or two complementary private actions; and (2) whenever $k > J$, $AF^k(R)$ is an open-envelope action.
- OF , the *outcome function*, is a function mapping non-empty sequences to $\{0, 1\}^\ell$.

Mechanism Play. Mechanism \mathcal{B} is played as follows. Let $gm^0 = (B^0, R^0, H^0)$ be the empty global memory. The players and Nature act one at a time, in K rounds, as specified by PS : the first one to act is PS^1 , the second is PS^2 , and so on, until PS^K acts and the mechanism terminates. At the k th round, if $j = PS^k \in N$, the information available to j consists of R^{k-1} and H_j^{k-1} —i.e., the current public record and the hidden bits of the secret actions played so far by j .²⁰ Based on this information, player j chooses an action $a^k \in AF^{k-1}(R^{k-1})$. Otherwise, if $PS^k = \text{Nature}$, then $AF^{k-1}(R^{k-1}) = \{a^k\}$, where a^k is an action of Nature. In either case, a^k maps the current global memory $gm^{k-1} = (B^{k-1}, R^{k-1}, H^{k-1})$ into a new global memory $a^k(gm^{k-1}) = gm^k = (B^k, R^k, H^k)$. After K rounds—i.e., when the general memory $gm^K = (B^K, R^K, H^K)$ is generated—the play terminates and the recommended outcome $y \in \{0, 1\}^\ell$ is obtained by evaluating function OF on the subsequence consisting of the last $K - J$ records of R^K ; that is, $y = OF(R^{(J, K]})$.

Executions. An execution e of \mathcal{B} is a sequence of global memories obtainable by playing \mathcal{B} until termination, $e = (B^0, R^0, H^0), \dots, (B^K, R^K, H^K)$. By $B^k(e), R^k(e)$, and $H^k(e)$ we denote, respectively, the ballot set, the public record, and the hidden-bit profile of e at round k . The output of e is defined to be $out(e) = OF(R^{(J, K]}(e))$.

Decision Nodes. A (round k) decision node of player i in a mechanism \mathcal{B} is a triple (k, R, H_i) —where $k < K$, and $PS^k = i$ —such that, for some execution e of \mathcal{B} , both $R = R^{k-1}(e)$ and $H_i = H_i^{k-1}(e)$. The set of all possible round k decision nodes of player i (over all possible executions of \mathcal{B}) is denoted by DN_i^k .

²⁰This implies that j knows all the actions he played in the past. In fact, all public actions are manifest in the public record, and j 's secret actions can be fully reconstructed from their hidden bits and the public record.

Strategies. A (pure) strategy of player i in \mathcal{B} is a function s_i mapping a decision node (k, R, H_i) of player i into the available action set $AF^k(R)$. We denote by $S_i^{\mathcal{B}}$ the set of all strategies of player i in \mathcal{B} , and by $S^{\mathcal{B}}$ the set of all strategy profiles in \mathcal{B} . We say that an execution e of \mathcal{B} is an execution of $s \in S^{\mathcal{B}}$ if, for all $k \leq K$, $a^k = s_i(k, R^{k-1}, H_i^{k-1})$ where $i = PS^k$. We denote by $EX(s)$ the distribution over the executions of s .²¹ We may also write $EX_{\mathcal{B}}(s)$ to highlight, if needed, the mechanism \mathcal{B} .

REMARKS.

- *Dummy Players.* In a ballot-box mechanism, each public action is performed by the player specified in PS . However, because the effect of a public action solely depends on the round at which it is performed—not on who performs it—ballot-box mechanisms might as well specify that the first player performs all public actions. Ballot-box mechanisms may even “out-source” public actions to an external entity, such as a *referee*.
- *Finiteness.* Although there are infinitely many ballot-box actions, each ballot-box mechanism \mathcal{B} is finite, because it consists of a finite number of rounds and at each round the active player has at most two actions to choose from.
- *Imperfect Information.* Ballot-box mechanisms are of imperfect-information for two reasons. First, whenever the set of available actions consists of two complementary secret actions, the active player permutes a pair of ballots, but all other players do not know which of two possible permutations was actually chosen. Second, when called into action, the ballot box permutes a sequence of ballots in a way unknown to all players. For these two reasons, the players may not know the exact content of a particular envelope. But they do know (1) which envelopes are in play—because the identifiers of the current ballots are always deducible from the public record—and (2) the exact *set* of the current envelopes’ contents—because the content of each newly created envelope is always public, and remains unaltered until the envelope is opened.
- *Extended Ballot-Box mechanisms.* One can easily envisage extending the above definition of a ballot-box mechanism to allow for richer action sets at each decision node, as well as for more general mechanism trees (than a predefined sequence of players), and more complex information structures. This might be quite desirable for achieving some other objectives not considered in this paper. We chose to have a bare-bone definition to keep the notation as simple as possible and to capture required elements for our perfect implementation exactly.

C. THE NOTION OF A PERFECT IMPLEMENTATION

C.1. Bit-by-bit Encodings

A function mapping any binary string x to a string $\bar{x} \in \Sigma^*$ is a *bit-by-bit encoding* of length ℓ if it satisfies the following 3 properties: (1) $\bar{0} \neq \bar{1}$; (2) $\bar{0}, \bar{1} \in \Sigma^\ell$; and (3) $\overline{xy} = \bar{x}\bar{y}$ (i.e., it preserves concatenation). Because a bit-by-bit encoding is fully specified by the values it takes at 0 and 1, we identify it with the pair of strings $(\bar{0}, \bar{1})$. A bit-by-bit encoding $(\bar{0}, \bar{1})$ is immediately extended to binary functions. If $f : D \subset \{0, 1\}^* \rightarrow \{0, 1\}^*$, then we define the function \bar{f} as follows: $\bar{f}(\bar{x}) = \overline{f(x)}$ for all $x \in D$.

The bit-by-bit encoding we use in our construction is an encoding of length 5, in which 0 and 1 are represented by two distinct sequences of five integers, see Section F for details.

²¹Because all strategies of s are pure and so deterministic, this distribution solely arises from the randomness of the ballot box (if used).

C.2. Ballot-Box Protocols

DEFINITION 9: A *ballot-box protocol* \mathcal{P} is a ballot-box mechanism which has its length equal to its output-boundary round —i.e., $\mathcal{P} = (N, K, K, PS, AF, OF)$ — and is executed starting with any feasible global memory. The distribution over the executions of a protocol \mathcal{P} with strategy profile s and initial global memory gm^0 is denoted by $EX_{\mathcal{P}}(s, gm^0)$.

REMARKS.

- Being always equal to the length K , the outcome-boundary round of a ballot-box protocol $\mathcal{P} = (N, K, K, PS, AF, OF)$ is redundant; and so is \mathcal{P} 's outcome function OF , because it is to be evaluated on the last $K - K = 0$ elements of the public record. Accordingly, we identify \mathcal{P} with its 1st, 2nd, 4th, and 5th components and more simply write $\mathcal{P} = (N, K, PS, AF)$.
- As “degenerate” ballot-box mechanisms, protocols retain all notions (e.g., executions, strategies, and decision nodes) and notations (e.g., $B^k(e)$, $R^k(e)$, and $H^k(e)$) of Section B.2.

DEFINITION 10: We say that ballot-box protocol (N, K, PS, AF) is the *concatenation* of protocols $P_1 = (N, K_1, PS_1, AF_1), \dots, P_m = (N, K_m, PS_m, AF_m)$ if

- $K = K_1 + \dots + K_m$;
- $PS = PS_1 \circ \dots \circ PS_m$; and
- $AF = AF_1 \circ \dots \circ AF_m$.

C.3. Ballot-Box Committers

DEFINITION 11: We say that a finite sequence x_1, \dots, x_k is an *address* if each x_i is a finite sequence of distinct positive integers and has no elements in common with any other x_j .

DEFINITION 12: Let $\mathcal{P} = (N, K, SP, AF)$ be a ballot-box protocol, $(\bar{0}, \bar{1})$ a bit-by-bit encoding, and x_1, \dots, x_n an address. We say that \mathcal{P} is a L -bit *ballot-box committer* for $(\bar{0}, \bar{1})$ with output address x_1, \dots, x_n if there exists a unique sequence U such that, for every execution e of \mathcal{P} whose initial global memory is empty, the following two properties hold:

1. *Correctness*: For every player i , $H_i^K(e) \in \{0, 1\}^L$ and $cont_{BK(e)}(x_i) = \overline{H_i^K(e)}$.
2. *Privacy*: $R^K(e) = U$.

Property Correctness requires that once \mathcal{P} is executed, the hidden-bit sequence of each player is of length L , and that the ballots corresponding to the output address x_1, \dots, x_n contain the bit-by-bit encoding of the players' hidden-bit sequences. Property Privacy guarantees that the public information available to the players at each round k of \mathcal{P} always consists of the fixed sequence U^k , no matter what strategies the players employ (and, therefore, no matter what hidden bits they choose).

REMARKS.

- Correctness implies that each element of x_i is the identifier of an envelope in $B^K(e)$ —else we would have $cont_{BK(e)}(x_i) = \perp$.
- Because each envelope of an output address contains a single symbol of Σ , each x_i consists of ℓL identifiers whenever the bit-by-bit encoding scheme $(\bar{0}, \bar{1})$ has length ℓ .

- In an execution of a ballot-box committer, player i 's final hidden-bit sequence, H_i^K , is “de facto chosen by i in an isolated room.” In fact, i 's hidden-bit sequence is initially empty, and grows by exactly one bit b each time that i is the active player and his action set consists of two complementary private actions. Whenever this is the case (and property Correctness guarantees that this is indeed the case L times), i has indeed total freedom to choose $b = 0$ by playing the available 0-action, or to choose $b = 1$ by playing the available 1-action. Besides being freely chosen, all such bits also are (1) independently chosen during \mathcal{P} and (2) totally secret at \mathcal{P} 's end. Thus, in any execution of \mathcal{P} , H_i^K depends on i 's strategy alone.

C.4. Ballot-Box Computers

DEFINITION 13: Let $f : X^a \rightarrow Y^b$ be a finite function, where $X, Y \subset \Sigma^*$; let x_1, \dots, x_a and y_1, \dots, y_b be two addresses; and let $\mathcal{P} = (N, K, PS, AF)$ be a ballot-box protocol such that AF only returns action sets of cardinality 1. We say that \mathcal{P} is a *ballot-box computer* for f (and that f has a *ballot-box computer* \mathcal{P}) with input address x_1, \dots, x_a and output address y_1, \dots, y_b if there exists a probabilistic algorithm SIM such that, for any strategy profile s and any initial global memory $gm^0 = (B^0, R^0, H^0)$ such that $cont_{B^0}(x_j) \in X$ for all j , the following two properties hold.

1. *Correctness.*

$$\left\langle cont_{B^K(e)}(y_1), \dots, cont_{B^K(e)}(y_b) : e \leftarrow EX_{\mathcal{P}}(s, gm^0) \right\rangle = f(cont_{B^0}(x_1), \dots, cont_{B^0}(x_a)).$$

2. *Privacy.*

$$\left\langle R^K(e) : e \leftarrow EX_{\mathcal{P}}(s, gm^0) \right\rangle = \left\langle F : F \leftarrow SIM(R^0) \right\rangle.$$

We refer to SIM as the *simulator* for \mathcal{P} .

REMARKS.

- Correctness implies that each element of each y_j is the identifier of an envelope in $B^K(e)$.
- The strategies s are formally there but irrelevant, because all action sets have cardinality 1.
- Simulator SIM is given R^0 as an input because (a) $gm^0 = (B^0, R^0, H^0)$ is chosen arbitrarily, and (b) R^K contains R^0 as a subsequence. Thus, without knowledge of R^0 , SIM could not produce R^K with the appropriate distribution. However, it is fair for SIM to know R^0 : if protocol \mathcal{F} starts with some non-empty global memory gm^0 , then the players have executed other protocols prior to it, and thus R^0 would be public knowledge anyway.

C.5. Ballot-Box Revealers

DEFINITION 14: Let y a sequence of integers and $\mathcal{R} = (N, K, PS, AF)$ a ballot-box protocol. We say that \mathcal{R} is a *ballot-box revealer* with address sequence y if (1) each function in AF only returns action sets consisting of a single open-envelope action; and (2) in any execution of \mathcal{R} , an envelope is opened if and only if its identifier belongs to y .

C.6. Perfect Implementation

DEFINITION 15: We say that ballot-box mechanism $\mathcal{B} = (N, K, J, PS, AF, OF)$ perfectly implements a standard mediated mechanism $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$ if there exist ballot-box protocols P_1, P_2 and P_3 , and a bit-by-bit encoding $(\bar{0}, \bar{1})$ with length ℓ such that:

1. P_1 is an L -bit committer for $(\bar{0}, \bar{1})$;
2. P_2 is a ballot-box computer, for the function \bar{g} , whose j th input address, x_j , coincides with the j th output address of P_1 for $j = 1$ to n ; and whose output addresses are y_1, \dots, y_L ;
3. P_3 is a ballot-box revealer whose address sequence is $y = y_1 \circ \dots \circ y_L$;
4. The ballot-box protocol (N, K, PS, AF) is the concatenation of P_1, P_2 , and P_3 ;
5. $J = K - \ell L$; and
6. OF is the function that, on input the records generated by opening a sequence y of ℓL envelopes, $(\text{OPENENV}, j_1, c_1), \dots, (\text{OPENENV}, j_{\ell L}, c_{\ell L})$, first computes $z = c_1, \dots, c_{\ell L} \in \Sigma^{\ell L}$, and then applies to z the decoding function of $(\bar{0}, \bar{1})$ —i.e., applies to z the function f such that $f(\bar{x}) = x$ for all $x \in \{0, 1\}^L$.

REMARKS.

- *No Signaling.* A distinctive property (and often the main advantage) of a mediated normal-form mechanism \mathcal{M} is that the players cannot signal to each other.²² This property, though not true for most extensive-form mechanisms, actually holds for any ballot-box mechanism \mathcal{B} that perfectly implements \mathcal{M} . This is quickly argued as follows. Signalling essentially arises from a player’s ability to choose among different actions. In \mathcal{B} , players can choose between different actions only during the execution of the ballot-box committer. However, all executions of a given ballot-box committer produce exactly the same public record, regardless of their chosen strategies, and thus no player can signal to any other.²³
- *Simultaneity.* One feature of the mediated mechanisms that is *essentially* captured by perfect implementation but not literally, is the “one-shot” (simultaneous) commitment of all messages and the “one-shot” revelation of the outcome. In the ballot-box mechanisms both the commitment of inputs and revelation of the output happens sequentially. Since

D. PERFECT IMPLEMENTATION IMPLIES STRATEGIC AND PRIVACY EQUIVALENCE

DEFINITION 16: Consider a ballot-box mechanism $\mathcal{B} = (N, K, J, PS, AF, Y, OF)$ and a standard mediated mechanism $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$. We say that \mathcal{B} is *strategic and privacy equivalent* to \mathcal{M} if $\forall i \in N$ there exists a bijection $\psi_i : S_i^{\mathcal{B}} \leftrightarrow \{0, 1\}^L$ and a probabilistic algorithm SIM such that, $\forall s \in S^{\mathcal{B}}$ the following properties hold:

$$\textit{Strategic Equivalence: } \langle \text{out}(e) : e \leftarrow EX_{\mathcal{B}}(s) \rangle = g(\psi_1(s_1), \dots, \psi_n(s_n)).$$

²²Of course, players may always “communicate to one another via the outcome” —at least for most outcome functions. That is, in \mathcal{M} , by sending the mediator a suitable message m_i , a player may affect the outcome so as to communicate some information to the other players. Such a communication is both “legitimate and intrinsic.” By saying that a player cannot signal we mean that he cannot communicate more information to the other players than via the outcome alone.

²³“No signalling” also holds for any ballot-box mechanism \mathcal{B} that is strategic and privacy equivalent to a normal-form mechanism \mathcal{M} , but a bit less trivially than when \mathcal{B} is a perfect implementation of \mathcal{M} .

Privacy Equivalence: $\langle R^K(e) : e \leftarrow EX_{\mathcal{B}}(s) \rangle = \langle F : e \leftarrow EX_{\mathcal{B}}(s); F \leftarrow SIM(out(e)) \rangle$.

We refer to algorithm *SIM* as \mathcal{B} 's *simulator*.

THEOREM 2: *If a ballot-box mechanism \mathcal{B} perfectly implements a standard mediated mechanism \mathcal{M} , then \mathcal{B} is strategically and privacy equivalent to \mathcal{M} .*

Proof. Let $\mathcal{B} = (N, J, K, PS, AF, OF)$ be the concatenation of a committer P_1 of length K_1 , a computer P_2 of length K_2 , and a revealer P_3 of length K_3 ; let $(\bar{0}, \bar{1})$ be the bit-by-bit encoding scheme of \mathcal{B} ; and let ℓ be the length of $(\bar{0}, \bar{1})$ —equivalently, let $\ell L = K_3$.

PROOF OF STRATEGIC EQUIVALENCE. First of all, notice that a strategy of a player i in \mathcal{B} uniquely determines a strategy of i in P_1 . (Actually, because every action set of P_2 and P_3 has cardinality 1, the viceversa is true too. Thus, i 's strategies in \mathcal{B} can be identified with his strategies in P_1 .)

Let us now define, for each player i , the function ψ_i as follows. On input $s_i \in S_i^{\mathcal{B}}$, do: choose any strategy sub-profile $s_{-i}^* \in S_{-i}^{\mathcal{B}}$; choose any execution e of \mathcal{B} with strategy profile (s_i, s_{-i}^*) ; and set $\psi_i(s_i) = H_i^{K_1}(e)$. That is, define $\psi_i(s_i)$ to be the hidden-bit sequence of i at the end of P_1 . Function ψ_i is well defined, because in the execution of a committer protocol a player's hidden-bit sequence solely depends on that player's strategy. Furthermore, because P_1 is an L -bit committer, $H_i^{K_1}(e)$ is an L -bit string. Thus, $\psi_i : S_i^{\mathcal{B}} \rightarrow \{0, 1\}^L$.

Let us now argue that ψ_i is surjective. Let $b_i = b_i^1 \dots b_i^L \in \{0, 1\}^L$, and let $s_i \in S_i^{\mathcal{B}}$ be the strategy that selects the b_k -action at the k th decision node of i where the action set consists of two complementary private actions. Then, $\psi_i(s_i) = b_i^1 \dots b_i^L$. Let us now argue that ψ_i is injective. Assume that s_i and s'_i are distinct strategies in $S_i^{\mathcal{B}}$. Then, because P_2 's and P_3 's action sets always have cardinality 1, s_i and s'_i must prescribe different actions in at least one of the L decision nodes of i in P_1 in which the action set consists of two complementary actions. If they do so at the j th such node, then $\psi_i(s_i)$ and $\psi_i(s'_i)$ would differ at the j th bit. Thus, ψ_i is a bijection: $\psi_i : S_i^{\mathcal{B}} \leftrightarrow \{0, 1\}^L$.

Finally, let $e = gm^0, \dots, gm^K$ be a random execution of \mathcal{B} with strategy profile s , where $gm^j = (B^j, R^j, H^j)$. Then, the first K_1 global memories of e constitute an execution of committer P_1 ; the next K_2 global memories constitute an execution of computer P_2 on initial global memory gm^{K_1} ; and the final K_3 global memories constitute an execution of revealer P_3 on initial global memory $gm^{K_1+K_2}$. The correctness property of P_1 implies that the content of its j th output address is $\overline{H_j^{K_1}}$. Because the input addresses of P_2 equal the output addresses of P_1 , because P_2 is a computer for \bar{g} , and because e is a random execution of P_2 on initial global memory gm^{K_1} , by the correctness property of P_2 , the contents of the output addresses of P_2 are distributed according to $g(H_1^{K_1}, \dots, H_i^{K_1})$. Since P_3 is a revealer whose input addresses equal the output addresses of P_2 , upon termination of P_3 , the output returned by *OF* is distributed according to $g(H_1^{K_1}, \dots, H_i^{K_1})$. Therefore, for any strategy profile $s \in S^{\mathcal{B}}$, $\langle out(e) : e \leftarrow EX_{\mathcal{B}}(s) \rangle$ and $g(\psi_1(s_1), \dots, \psi_n(s_n))$ are equal distributions.

PROOF OF PRIVACY EQUIVALENCE. Consider the following algorithm *SIM*, in which U is the fixed public record of P_1 guaranteed by the privacy property of Definition 12, *SIM*₂ is the simulator for P_2 guaranteed by the privacy property of Definition 13, y is the output address of P_2 , and the encoding scheme is the same bit-by-bit encoding of \mathcal{B} .

Algorithm *SIM*:

Input: $z = z_1 \cdots z_L \in \{0, 1\}^L$.

1. $F_2 \leftarrow SIM_2(U)$;
2. Compute \bar{z} (Note: $\bar{z} \in \{0, 1\}^{\ell L}$ because $(\bar{0}, \bar{1})$ has length ℓ)
3. Set $F_3 = (\text{OPENENV}, y_1, b_1), \dots, (\text{OPENENV}, y_{\ell L}, b_{\ell L})$, where $\bar{z} = b_1 \cdots b_{\ell L}$.
(F_3 is the sequence of records obtained by opening the envelope sequence y with content \bar{z} .)

Output: The sequence of records $F = U \circ F_2 \circ F_3$.

It is clear that SIM is the simulator required for \mathcal{B} 's privacy equivalence to \mathcal{M} to hold.

Q.E.D.

E. PERFECT IMPLEMENTATION IMPLIES MODULAR DESIGN

E.1. Modular Function Composition

This subsection comprises our version of basic notions and results in complexity theory —for a more standard version, see, for example, the book of Papadimitriou (1995).

DIRECTED GRAPHS. A directed graph —*digraph* for short— consists of a finite set of contiguous positive integers $V = \{1, 2, \dots, m\}$ and a subset E of $V \times V$. We refer to V as the set of *nodes* and to E as the set of *edges*. Whenever $(u, v) \in E$, we refer to u as a *parent* of v , and to v as a *child* of u . We refer to the number of parents of a node u as u 's *in-degree* and the number of children of u as u 's *out-degree*. A node of V is called a *source* if it has in-degree zero, a *sink* if it has out-degree zero, and an *internal node* otherwise.

A digraph (V, E) is *acyclic* if it has no cycles, that is, $v_k \neq v_1$ for any node sequence v_1, \dots, v_k such that $(v_i, v_{i+1}) \in E$ for $i < k$. It is immediately seen that any non-empty acyclic digraph must have at least one source and at least one sink.

A acyclic digraph (V, E) is *in standard topological order* if, (1) $u < v$ whenever $(u, v) \in E$; and (2) $a < b < c$ whenever a is a source, b is an internal node, and c is a sink. (Any acyclic digraph can be easily put in standard topological order by renaming its nodes.) In such a digraph, there is a natural order for the incoming (and outgoing) edges of each node j : namely, the k th incoming (outgoing) edge of node j is the edge (u, j) such that j has exactly $k - 1$ parents (children) x for which $x < u$.

A digraph (V, E) can be represented graphically as follows: each $u \in V$ is represented as a circle (where circles representing different nodes do not overlap) and each $(u, v) \in E$ is represented as an arrow from the circle representing u to circle representing v . (See Figure 1.)

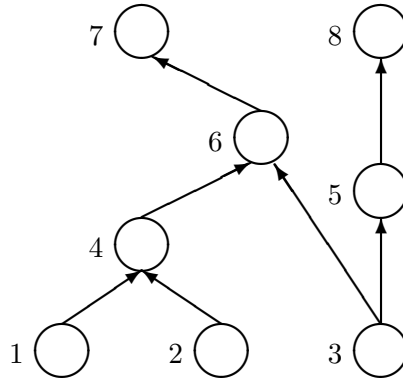


Figure 1: The graphical representation of the acyclic digraph (V, E) where $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $E = \{(1, 4), (2, 4), (3, 5), (3, 6), (4, 6), (6, 7), (5, 8)\}$. Notice that (V, E) is in standard topological order.

COMPUTATION DIGRAPHS AND THEIR FUNCTIONS. A *computation digraph* C consists of a triple $((V, E), a, func)$ such that

1. (V, E) is an acyclic digraph, in standard topological order, in which every source has out-degree one and every sink has in-degree one;
2. a is a non-negative integer less than or equal to the number of sources in (V, E) ; and
3. letting A the set of the first a sources and B the set consisting of all other sources and all internal nodes of (V, E) , $func$ is a function mapping B to finite functions such that, for each node $j \in B$ with p parents and c children, $func(j) : \{0, 1\}^p \rightarrow \{0, 1\}^c$.

We refer to the nodes of A as C 's *input nodes*, the nodes of B as C 's *computation nodes*, and to the sinks of (V, E) as C 's *output nodes*.

A computation digraph $((V, E), a, func)$ can be represented graphically by first drawing (V, E) as discussed above, and then writing $func(j)$ inside the circle corresponding to computation node j . (See Figure 2.)

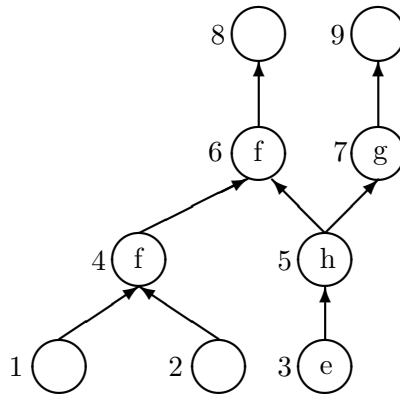


Figure 2: The graphical representation of a computation digraph where $a = 2$, $e : \emptyset \rightarrow \{0, 1\}$, $f : \{0, 1\}^2 \rightarrow \{0, 1\}$, $g : \{0, 1\}^1 \rightarrow \{0, 1\}^1$, $h : \{0, 1\} \rightarrow \{0, 1\}^2$.

A computation digraph $C = ((V, E), a, func)$ with a input nodes and b output nodes and c computation nodes can be interpreted as the finite function $\mathcal{F}_C : \{0, 1\}^a \rightarrow \{0, 1\}^b$ mapping an input $x = x_1 \dots x_a$ to an output $y = y_1 \dots y_b$ as follows:

- For $i = 1$ to a : label the outgoing edge of the i th source with bit x_i .
- For $j = a + 1$ to $a + c$ do: If computation node j has in-degree I and out-degree O then compute the labels z_1, \dots, z_O , where z_k labels the k th outgoing edge of j , as follows: (1) let y_i is the label of the i th incoming edge of j ; (2) let $f = func(j)$; and then (3) let $z_1 \dots z_O$ is a random output of $f(y_1 \dots y_I)$. (Note that because (V, E) is in standard topological order, when it comes time to label j 's outgoing edges, all of j 's incoming edges have already been labeled.)
- Set y_i be the bit labeling the incoming edge of C 's i th output node.

Note that \mathcal{F}_C is a deterministic function if the range of $func$ is a set of deterministic functions., then \mathcal{F}_C is a deterministic function.

MODULAR COMPOSITION AND MODULAR BASES. A finite function f is a *modular composition* of finite functions f_1, \dots, f_k relative to a computation digraph $C = ((V, E), r, func)$ if (1) $f = \mathcal{F}_C$ and (2) the range of $func$ is $\{f_1, \dots, f_k\}$. We say that f_1, \dots, f_k are a *modular basis* if every finite function is a modular composition of f_1, \dots, f_k .

Let AND be the deterministic Boolean function mapping bits b_1 and b_2 to $b_1 \wedge b_2$, that is, to 1 if and only if $b_1 = b_2 = 1$; NOT the deterministic Boolean function mapping a bit b to $\neg b$, that is, to $1 - b$; DUPLICATE the deterministic Boolean function mapping a bit b to the pair of bits $D(b) = (b, b)$; and COIN the uniform distribution over $\{0, 1\}$, that we interpret as the probabilistic finite function $\$: \emptyset \rightarrow \{0, 1\}$ mapping no input to a random bit. Then, it is well known in complexity theory that AND, NOT, DUPLICATE, and COIN form a modular basis.

BOOLEAN CIRCUIT REPRESENTATION OF FINITE FUNCTIONS. A Boolean circuit is a computation digraph $C = ((V, E), a, func)$ where the range of $func$ is $\{AND, NOT, DUPLICATE, COIN\}$. We say that C is deterministic if its number of sources is exactly a , and probabilistic otherwise.

A Boolean circuit representation of a finite function f is a Boolean circuit C such that $f = \mathcal{F}_C$. (See Figure 3.) It is well known in complexity theory that a Boolean-circuit representation of f can be efficiently computed from any other standard representation of f .

Note that, for every Boolean Circuit $C = ((V, E), a, func)$, $func$ necessarily associates to each computation node with no parents the function COIN, to each computation node with 1 parent and 1 child the function NOT, to each computation node with 1 parent and 2 children the function DUPLICATE, and to each computation node with 2 parents the function AND. Thus, to specify C , it suffices to specify the digraph (V, E) and the integer a . Because computational nodes with no parents are always associated with the function COIN, we refer to such nodes as the *random nodes* of C .

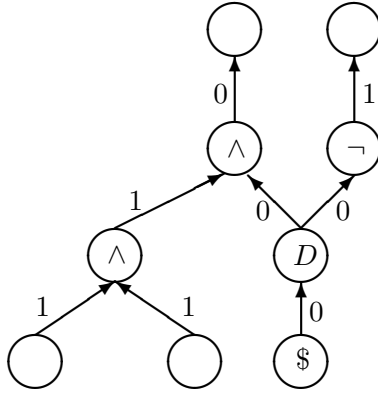


Figure 3: A probabilistic Boolean circuit C such that $\mathcal{F}_C : \{0,1\}^2 \rightarrow \{0,1\}^2$ is computed with the help of a single coin toss. The figure shows the labeling of each edge in a computation of \mathcal{F}_C on input 11, and coin-toss outcome 0, to produce output 01. (Because the functions AND, NOT and DUPLICATE are symmetric in their inputs, we need not explicitly represent the topological ordering of C .)

STANDARD BINARY ENCODING OF BOOLEAN CIRCUITS. Let $C = ((V, E), a, func)$ be a Boolean circuit with m nodes and e edges. Then, each node j of C is described by a unique l -bit string, where $l = \lceil \log(m) \rceil$: namely, the l -bit binary expansion of integer j (with leading 0s if necessary). Each edge (u, v) of E is described by a unique $2l$ -bit string: namely, the concatenation of the description of u and the description of v . We thus define the *standard binary encoding* of C to be the string $1^m 01^a 0S$, where string $S \in \{0,1\}^{2le}$ is the concatenation (in lexicographic order) of the e edges of E . Notice that this encoding is guaranteed to consist of at most $4m \lceil \log(m) \rceil$ bits.²⁴

E.2. Modular Mechanism Design

DEFINITION 17: A ballot-box computer \mathcal{P} is a *modular composition of ballot-box computers* P_1, \dots, P_k if it is the concatenation of a sequence of ballot-box protocols $\mathcal{P}_1, \dots, \mathcal{P}_K$ such that, for all j , either $\mathcal{P}_j \in \{P_1, \dots, P_k\}$ or \mathcal{P}_j 's action functions always return a single renaming action.

THEOREM 3: If g is a modular composition of functions g_1, \dots, g_k , where each g_j has a ballot-box computer G_j , then g has a ballot-box computer G that is a modular composition of G_1, \dots, G_k .

Proof. Let $g : \{0,1\}^a \rightarrow \{0,1\}^b$, let $C = ((V, E), a, func)$ be the computational digraph relative to which g is modular composition of g_1, \dots, g_k , and let c be the number of computation nodes of C . (Note that C necessarily has a input nodes and b output nodes.)

We prove the theorem by explicitly constructing a ballot-box computer G for g , whose input addresses are x_1, \dots, x_a . The idea is very simple: G will be constructed by concatenating together a sequence of c protocols chosen from $\{G^1, \dots, G^k\}$ by evaluating $func$ at each computation node of C . The only difficulty is of a syntactic nature. To meaningfully concatenate ballot-box computers, their input and output addresses must properly match. However, the protocols G^i come with

²⁴Let C have a input nodes and b output nodes. Each internal node has degree at most 3 and each source and sink has degree 1. Therefore, the sum of the degrees of all nodes is at most $3m - 2a + 2b + 2r$. Therefore $e \leq 1.5m - a - b - r$ and so the length of the encoding string is at most $m + r + 2e \lceil \log(m) \rceil \leq 4m \lceil \log(m) \rceil$.

their own input and output addresses, and these may not match at all. Furthermore, if $k < c$, some protocols must be used more than once. Therefore, if G^i occurs t times in our envisaged concatenation, we must make t copies of it ensuring that they operate on disjoint and properly chosen addresses (and thus on separate physical envelopes). To this end, we first add a suitably large “offset” to every identifier used by G^i . Then, we rename all envelopes belonging to an output address so that they match the intended input address. The details are as follows.

- For each i between 1 and k , let x_1^i, x_2^i, \dots be the input addresses of G^i .
(Recall that an address consists of a sequence of envelope identifiers: specifically a sequence of ℓ identifiers when using a bit-by-bit encoding of length ℓ .)
- For any ballot-box protocol P , we define $offset(P, i)$, the “offset of ballot-box protocol P by integer i ”, as follows. if $P = (N, K, PS, AF)$, then $offset(P, i) = (N, K, PS, \widehat{AF})$, where, for any round d and public record R , \widehat{AF}^d first runs $AF^d(R)$ to obtain an action set A , and returns the action set \widehat{A} obtained by incrementing all envelope identifiers in A by i .
- Let M be an integer such that (1) M upper bounds the number of envelopes created by any execution of any protocol G^i ; and (2) for each i between 1 and a , M is larger than any envelope identifier in x_i .
- Let $index$ be the function that, for every computational node j , returns the index in S of $func(j)$ —i.e., $index(j) = i$ whenever $func(j) = g^i$.
- For each $j = a + 1$ to $a + c$, let P_j be the protocol $offset(G^{index(j)}, jM)$.
(Comment: By construction it is clear that P_j is a ballot-box computer for $func(j)$ with input addresses $x_1^{index(j)} + jM, x_2^{index(j)} + jM, \dots$)
- For $j = a + 1$ to $a + c$, define address z_i^j as follows:
 - * If the i th incoming edge of node j is the sole outgoing edge of the e th input node of C , then $z_i^j = x^e$.
 - * Otherwise, if the i th incoming edge of node j is the e th outgoing edge of some computation node h , then z_i^j is the e th output address of protocol P_h .
- For $j = a + 1$ to $a + c$, let P'_j be the ballot-box protocol that first (1) for $l = 1, \dots, \ell$, renames the l th envelope of each address z_i^j to be the l th identifier in $x_i^{index(j)} + jM$ (i.e., the i th input address of protocol P_j) whenever —as in our case!— the latter ℓ identifiers are not in I_B ; and then (2) executes protocol P_j .
(Comment 2: In our case, P'_j is a ballot-box computer for $g^{index(j)}$ with input addresses z_1^j, z_2^j, \dots)

Let G be the concatenation of protocols $P'_{a+1}, \dots, P'_{a+c}$. As previously stated, the input addresses of G are x_1, \dots, x_a . Let us now specify the output addresses of G . If the incoming edge of C 's i th sink is the e th outgoing edge of some internal node h , then the i th output address of \mathcal{P} is the e th output address of P'_h . All that remains to show is that G is a ballot-box computer for g . First, we argue that each P'_j successfully renames all addresses, that is, that each identifier in $x_i^{index(j)} + jM$ is unused at the start of the execution of P'_j . This is so because at the start of P'_j , (1)

each identifier in $x_i^{\text{index}(j)} + jM$ is larger than jM ; (2) fewer than j protocols have been executed prior to this renaming; and (3) each previously executed protocol uses fewer than M identifiers. Therefore, each P_j' correctly computes $g^{\text{index}(j)}$ and, by construction, G correctly computes the function $g = \mathcal{F}_C$. Finally, to establish G 's privacy, consider the simulator algorithm which: (i) lets R_a be the empty public record; and (ii) for j from $a + 1$ to $a + c$, runs the simulator for P_j' on public record R_{j-1} to produce public record R_j ; and (iii) outputs public record R_{a+c} .

Q.E.D.

F. COMPUTING WITH PERMUTATIONS

The following is our rendition of a method of realizing ANDs and NOTs via permutations of 5 elements due to Barrington (1986).

NOTATION. If x is a permutation in SYM_5 , we denote by x_j the image of integer $j \in \{1, 2, 3, 4, 5\}$ under x —i.e., $x_j = x(j)$ —and identify x with the 5-symbol string $x_1x_2x_3x_4x_5$.

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

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

Proof: $a = 12453$, $b = 25341$, $[a \rightarrow b] = 34125$, $[a^{-1} \rightarrow a] = 12354$, and $[aba^{-1}b^{-1} \rightarrow a] = 13245$.

THREE OPERATORS. For any $x \in \text{SYM}_5$, the operators “ $\tilde{\cdot}$ ”, “ $'$ ” and “ $*$ ” are defined as follows:

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

BIT REPRESENTATION. 0 is represented by \mathcal{I} , and 1 by a .

REALIZING NOT & AND. If permutations x_1 and x_2 respectively represent bits b_1 and b_2 then

$$\neg b_1 = (x_1 a^{-1})^* \quad \text{and} \quad b_1 \wedge b_2 = (x_1 \tilde{x}_2 x_1^{-1} \tilde{x}_2^{-1})'$$

G. PERFECT IMPLEMENTATION OF ANY MEDIATED MECHANISM

THEOREM 1: *For every standard mediated mechanism \mathcal{M} , there exists a ballot-box mechanism \mathcal{B} that perfectly implements \mathcal{M} .*

As per Definition 15, letting $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$, to prove Theorem 1 we must exhibit: (1) a bit-by-bit encoding scheme $(\bar{0}, \bar{1})$; (2) an L -bit ballot-box committer; (3) a ballot-box computer for \bar{g} ; and (4) a ballot-box revealer.

G.1. Our Encodings

In our construction, we use the following length-5 bit-by-bit encoding $(\bar{0}, \bar{1}) = (12345, 12453)$ and interpret “12345” and “12453” as the permutations \mathcal{I} and a of Appendix F. We call this bit-by-bit encoding the *Barrington encoding*.

DEFINITION 18: Let B be a well-defined set of ballots and σ a vector mapping the index set $\{1, 2, 3, 4, 5\}$ to the identifiers of B ’s envelopes. Then σ is an *envelope encoding (of a 5-element permutation)* if $\text{cont}_B(\sigma_1) \cdots \text{cont}_B(\sigma_5) \in \text{SYM}_5$. In such a case, we shall denote by $\hat{\sigma}$ the permutation $\text{cont}_B(\sigma_1) \cdots \text{cont}_B(\sigma_5)$. If σ is an envelope encoding and $\hat{\sigma} \in \{\mathcal{I}, a\}$, then σ is an *envelope encoding of a bit*.

Therefore, whenever σ is an envelope encoding, $(\sigma_j, \hat{\sigma}_j, 0) \in B$ for all j , and σ is tautologically the envelope encoding of $\hat{\sigma}$. We reserve lower-case Greek letters to denote envelope encodings.

G.2. Protocol Conventions

We specify a protocol $\mathcal{P} = (N, K, PS, AF)$ as a list of K items, where item k fully describes round k by specifying the acting party PS^k and the available-action set A_k as follows.

- If A_k consists of a single action a , then specifying PS^k is unnecessary (either because a is an action of Nature, or because a has the same effect regardless of the player performing it). Accordingly, we completely describe round k as a party-agnostic imperative: “perform action a .”
- If A_k consists of two complementary private actions a and b , then we completely describe round k by an exhortation for the acting player i : “let player i choose between action a and action b .”
- If $a \in A_k$ is a constant, we explicitly specify a using its informal reference of Definition 5.
- If $a \in A_k$ is a function of the current public record R , we specify a as a program that returns a on input R .

SHORTCUTS. For brevity and clarity, we often contract the description of several rounds into a single, conceptual round. For instance:

- If σ is an envelope encoding, by the conceptual round “For $\ell = 1$ to 5, publicly open envelope σ_ℓ ” we mean the following 5 rounds
 - “Publicly open envelope σ_1 .
 - Publicly open envelope σ_2 .
 - Publicly open envelope σ_3 .
 - Publicly open envelope σ_4 .
 - Publicly open envelope σ_5 .”
- If p is a permutation in SYM_5 , by the conceptual round “Create an envelope encoding σ of p ” we mean the following 5 rounds
 - “Create a new envelope σ_1 with public content p_1 .
 - Create a new envelope σ_2 with public content p_2 .
 - Create a new envelope σ_3 with public content p_3 .
 - Create a new envelope σ_4 with public content p_4 .
 - Create a new envelope σ_5 with public content p_5 . Then, set $\sigma = \sigma_1, \dots, \sigma_5$.”

- If \mathcal{P} is a protocol having an envelope encoding as an input address and an envelope encoding as an output address, by the conceptual round “Set $\beta = \mathcal{P}(\alpha)$ ” we mean all of the rounds of an execution of \mathcal{P} in which \mathcal{P} ’s input address is α and \mathcal{P} ’s output address is β .

G.3. An L -bit Ballot-Box Committer For the Barrington Encoding

Protocol $Commit_L$

For player $i = 1$ to n DO: For bit $t = 1$ to L DO:

1. Create an envelope encoding $\alpha^{(i,t)}$ of the identity permutation.
2. Create an envelope encoding $\beta^{(i,t)}$ of permutation $a = 12345$.
3. Create a new super-envelope $A^{(i,t)}$ containing envelopes $\alpha_1^{(i,t)}, \dots, \alpha_5^{(i,t)}$.
4. Create a new super envelope $B^{(i,t)}$ containing envelopes $\beta_1^{(i,t)}, \dots, \beta_5^{(i,t)}$.
5. Let player i secretly permute $A^{(i,t)}$ and $B^{(i,t)}$ to obtain the super-envelope pair $(C^{(i,t)}, D^{(i,t)})$.
6. Open $C^{(i,t)}$ to expose envelopes $\gamma_1^{(i,t)}, \dots, \gamma_5^{(i,t)}$. Set $\gamma^{(i,t)} = \gamma_1^{(i,t)}, \dots, \gamma_5^{(i,t)}$.
7. Open $D^{(i,t)}$ to expose envelopes $\delta_1^{(i,t)}, \dots, \delta_5^{(i,t)}$.
8. Ballotbox $\delta_1^{(i,t)}, \dots, \delta_5^{(i,t)}$ to obtain $\eta_1^{(i,t)}, \dots, \eta_5^{(i,t)}$.

Output Addresses: For each $i \in N$, sequence $\gamma^i = \gamma^{(i,1)}, \dots, \gamma^{(i,L)}$.

LEMMA 1: *Protocol $Commit_L$ is an L -bit Ballot-Box Committer for the Barrington encoding.*

PROOF OF CORRECTNESS. At the end of Step 3, $A^{(i,t)}$ contains a sequence of 5 envelopes encoding the identity, and, at the end of Step 4, $B^{(i,t)}$ contains a sequence of 5 envelopes encoding permutation a . Thus, recalling that in the Barrington encoding $\mathcal{I} = \bar{0}$ and $a = \bar{1}$, at the end of Step 5, $C^{(i,t)}$ contains an envelope encoding of \bar{b} if player i “chooses the bit b ” (i.e., if the t th bit of H_i is b). Thus, at the end of Step 6, the content of address γ^i is \bar{H}_i , where H_i is a binary string of length L , as demanded by Definition 10.

PROOF OF PRIVACY. First notice that, at each iteration, $Commit_L$ increases *maxid*, the maximum identifier in the current ballot set, by 29. (In fact: Step 1 consists of five actions, each producing one new identifier; Step 2 similarly produces 5 new identifiers; Step 3 and Step 4 produce one new identifier each; Step 5 produces 2 new identifiers; the opening of the super-envelopes in Steps 6 and 7 exposes five envelopes, thus producing five new identifiers each; finally, Step 8 produces five new identifiers.) Now notice that, prior to iteration (i, t) , exactly $(i - 1)L + (t - 1)$ iterations have occurred. Therefore, defining *maxid* (i, t) to be the value of the maximum identifier in the ballot set at the beginning of iteration (i, t) , we have *maxid* $(i, t) = 29((i - 1)L + (t - 1))$. Thus (recalling that $a = 12453$), Privacy is established by noticing that, in any execution of $Commit_L$, the portion of the final public record generated by iteration (i, t) is the following, *fixed* sequence $U(i, t)$:

(NEWENV, 1) (NEWENV, 2) (NEWENV, 3) (NEWENV, 4) (NEWENV, 5)
 (NEWENV, 1) (NEWENV, 2) (NEWENV, 4) (NEWENV, 5) (NEWENV, 3)

(NEWSUPER, $(29((i-1)L + (t-1)) + 1, \dots, 29((i-1)L + (t-1)) + 5)$)
(NEWSUPER, $(29((i-1)L + (t-1)) + 6, \dots, 29((i-1)L + (t-1)) + 10)$)
(SECRETPERMUTE, $i, (29((i-1)L + (t-1)) + 11, 29((i-1)L + (t-1)) + 12)$)
(OPENSUPER, $29((i-1)L + (t-1)) + 13$)
(OPENSUPER, $29((i-1)L + (t-1)) + 14$)
(BALLOTBOX, $i, (29((i-1)L + (t-1)) + 20, \dots, 29((i-1)L + (t-1)) + 24)$)

Q.E.D.

REMARK. The envelopes $\delta_1^{(i,t)}, \dots, \delta_5^{(i,t)}$ produced by Steps 7 and 8 are not used by any of our subsequent protocols and “remain on the table” after the completion of our ballot-box mechanism. Since these envelopes are unused, one might consider eliminating Steps 7 and 8 and leave super-envelop $D^{(i,t)}$ instead of the five envelopes $\delta_1^{(i,t)}, \dots, \delta_5^{(i,t)}$. However, super-envelope $D^{(i,t)}$ contains information about the t th bit of player i ’s input. Therefore, to ensure player i ’s privacy, it is vital that the contents of $D^{(i,t)}$ never be made public. Theoretically, there is no problem with leaving $D^{(i,t)}$ on the table: after the execution of a mechanism is finished, there are no further actions to be taken, including envelope-opening actions. In reality, however, we would like to preserve the physical meaningfulness of ballot-box mechanisms to the maximum possible extent, and we would like to enable our players, after completing the mechanism, to leave the table and return to their normal lives without worrying about left-over envelopes whose contents could compromise their privacy in the future. To this end, we choose to include Steps 7 and 8 which produce envelopes $\delta_1^{(i,t)}, \dots, \delta_5^{(i,t)}$ containing a random permutation independent of all players’ inputs. This way the players need not worry about whether the left-over envelopes are opened at some point in the future.

As an alternative approach to “cleaning-up” our left-over envelopes, we could enrich the set of ballot-box actions with an additional “envelope-destroy” action. However, we prefer to attain the same effect by using the existing ballot-box actions. None of our subsequent ballot-box protocols produce left-over envelopes. Therefore, this type of clean-up is only needed in our ballot-box committer. Similar “clean-up” instructions appear in our other protocols.

G.4. A Ballot-Box Computer for Permutation Inverse

Permutation inverse is the function that, on input a permutation $p \in \text{SYM}_5$, returns p^{-1} .

Protocol Invert

Input address: σ —an envelope encoding of a permutation in SYM_5 .

1. Create an envelope encoding α of the identity permutation.
2. For $\ell = 1$ to 5: create a new super-envelope A_ℓ containing the pair of envelopes $(\alpha_\ell, \sigma_\ell)$.
3. Ballotbox A_1, \dots, A_5 to obtain A'_1, \dots, A'_5 .
4. For $\ell = 1$ to 5: open super-envelope A'_ℓ to expose envelope pair (μ_ℓ, ν_ℓ) .
5. For $\ell = 1$ to 5: publicly open ν_ℓ , and denote its content by $\hat{\nu}_\ell$. Set $\hat{\nu} = \hat{\nu}_1 \circ \dots \circ \hat{\nu}_5$.

6. Publicly permute μ_1, \dots, μ_5 according to $\hat{\nu}^{-1}$ to obtain ρ_1, \dots, ρ_5 . Set $\rho = \rho_1, \dots, \rho_5$.

Output address: ρ .

LEMMA 2: *Protocol Invert is a Ballot-Box Computer for permutation inverse.*

PROOF OF CORRECTNESS. To establish the correctness of *Invert* we must show that ρ is an envelope encoding of $\hat{\sigma}^{-1}$. Because Step 2 binds together, in the same super-envelope A_ℓ , the ℓ th envelope of α and σ , Step 3 applies the same, random and secret, permutation to both $\hat{\alpha}$ and $\hat{\sigma}$. Letting x be this secret permutation, Step 4 “puts on the table” the envelope encodings $\mu = \mu_1, \dots, \mu_5$ and $\nu = \nu_1, \dots, \nu_5$ where $\hat{\mu} = x\mathcal{I} = x$ and $\hat{\nu} = x\hat{\sigma}$. At the end of Step 4, both $\hat{\nu}$ and $\hat{\mu}$ are totally secret. Step 5, however, reveals $\hat{\nu}$ to all players, so that all players can compute $\hat{\nu}^{-1}$ and verify that Step 6 is correctly executed. Step 6 ensures that $\hat{\rho} = \hat{\nu}^{-1}\hat{\mu}$. Thus, $\hat{\rho} = \hat{\nu}^{-1}\hat{\mu} = \hat{\sigma}^{-1}x^{-1}x = \hat{\sigma}^{-1}$ as desired.

PROOF OF PRIVACY. Consider the following algorithm.

Algorithm *SIM-invert*:

On input a public record R , (1) compute *maxid*, the maximum identifier of a ballot set consistent with R ; (2) randomly select a permutation $r \in \text{SYM}_5$; and (3) output the following sequence of records (grouped so as to correspond to the records generated by each conceptual step of Protocol *Invert*):

(NEWENV, 1) (NEWENV, 2) (NEWENV, 3) (NEWENV, 4) (NEWENV, 5)
 (NEWSUPER, (*maxid* + 1, σ_1)) (NEWSUPER, (*maxid* + 2, σ_2)) (NEWSUPER, (*maxid* + 3, σ_3))
 (NEWSUPER, (*maxid* + 4, σ_4)) (NEWSUPER, (*maxid* + 5, σ_5))
 (BALLOTBOX, (*maxid* + 6, \dots , *maxid* + 10))
 (OPENSUPER, *maxid* + 11) (OPENSUPER, *maxid* + 12) (OPENSUPER, *maxid* + 13)
 (OPENSUPER, *maxid* + 14) (OPENSUPER, *maxid* + 15)
 (OPENENV, *maxid* + 17, r_1) (OPENENV, *maxid* + 19, r_2) (OPENENV, *maxid* + 21, r_3)
 (OPENENV, *maxid* + 23, r_4) (OPENENV, *maxid* + 25, r_5)
 (PUBLICPERMUTE, (*maxid* + 16, *maxid* + 18, *maxid* + 20, *maxid* + 22, *maxid* + 24), r^{-1})

Let us now prove that, *SIM-invert* is indeed a simulator for protocol *Invert*. For any global memory $gm^0 = (B^0, R^0, H^0)$ for which σ is an envelope encoding (of a permutation in SYM_5), consider (1) a random execution e of *Invert* with initial global memory gm^0 , and (2) a random execution E of *SIM-invert* on input R^0 . First observe that e 's records produced by Steps 1-4 are identical to the first 16 records of E . Indeed, these 16 records correspond to actions that (i) belong to the same classes; and (ii) specify the same sequence of envelope identifiers.²⁵ The remaining 6 records of the public record of e and E are, respectively,

(OPENENV, *maxid* + 17, $\hat{\nu}_1$) (OPENENV, *maxid* + 19, $\hat{\nu}_2$) (OPENENV, *maxid* + 21, $\hat{\nu}_3$)
 (OPENENV, *maxid* + 23, $\hat{\nu}_4$) (OPENENV, *maxid* + 25, $\hat{\nu}_5$)
 (PUBLICPERMUTE, (*maxid* + 16, *maxid* + 18, *maxid* + 20, *maxid* + 22, *maxid* + 24), $\hat{\nu}^{-1}$)

and

²⁵Note that the value of *maxid* at the beginning of e is derivable from R^0 , which is the same in both cases, and thus is equal to the value initially computed by *SIM-invert*.

(OPENENV, $maxid + 17, r_1$) (OPENENV, $maxid + 19, r_2$) (OPENENV, $maxid + 21, r_3$)
 (OPENENV, $maxid + 23, r_4$) (OPENENV, $maxid + 25, r_5$)
 (PUBLICPERMUTE, ($maxid + 16, maxid + 18, maxid + 20, maxid + 22, maxid + 24$), r^{-1}).

Thus, while these two record subsequences may very well be different, they are identically distributed. In fact, in the latter subsequence, $r = r_1 r_2 r_3 r_4 r_5$ is the random permutation selected by *SIM-invert*; while in the former subsequence $\hat{\nu} = \hat{\nu}_1 \hat{\nu}_2 \hat{\nu}_3 \hat{\nu}_4 \hat{\nu}_5 = x \hat{\sigma}$, where x is the realization of a random permutation (secretly selected by the ballot box). Therefore, since the product of a random permutation in SYM_5 and a fixed permutation in SYM_5 is a random permutation in SYM_5 , the two sequences are identically distributed.

Q.E.D.

G.5. A Ballot-Box Computer for Permutation Product

Permutation product is the function that, on input (p, q) where $p, q \in \text{SYM}_5$, returns pq —i.e., the permutation in SYM_5 mapping i to $p(q(i))$.

Protocol Multiply

Inputs addresses: σ and τ —each an envelope encoding of a permutation in SYM_5 .

1. Set $\alpha = \text{Invert}(\sigma)$.
2. For $\ell = 1$ to 5: create a new super-envelope A_ℓ containing the pair of envelopes (τ_ℓ, α_ℓ) .
3. Ballotbox A_1, \dots, A_5 to obtain A'_1, \dots, A'_5 .
4. For $\ell = 1$ to 5: open super-envelope A'_ℓ to expose envelope pair (μ_ℓ, ν_ℓ) .
5. For $\ell = 1$ to 5: publicly open envelope ν_ℓ , and denote its content by $\hat{\nu}_\ell$. Set $\hat{\nu} = \hat{\nu}_1 \circ \dots \circ \hat{\nu}_5$.
6. Publicly permute μ_1, \dots, μ_5 according to $\hat{\nu}^{-1}$ to obtain ρ_1, \dots, ρ_5 . Set $\rho = \rho_1, \dots, \rho_5$.

Output address: ρ .

LEMMA 3: *Protocol Multiply is a Ballot-Box Computer for permutation product.*

PROOF OF CORRECTNESS. To establish the correctness of *Multiply* we must show that ρ is an envelope encoding of $\hat{\sigma}\hat{\tau}$. The correctness of *Invert* implies that, at the completion of Step 1, α is an envelope encoding of permutation $\hat{\alpha} = \hat{\sigma}^{-1}$. Because Step 2 binds together, in the same super-envelope A_ℓ , the ℓ th envelope of τ and α , Step 3 applies the same, random and secret, permutation to both $\hat{\alpha}$ and $\hat{\sigma}$. Letting x be this secret permutation, Step 4 “puts on the table” the envelope encodings $\mu = \mu_1, \dots, \mu_5$ and $\nu = \nu_1, \dots, \nu_5$, where $\hat{\mu} = x\hat{\tau}$ and $\hat{\nu} = x\hat{\alpha} = x\hat{\sigma}^{-1}$. At the end of Step 4, both $\hat{\nu}$ and $\hat{\mu}$ are totally secret. Step 5, however, reveals $\hat{\nu}$ to all players, so that all players can compute $\hat{\nu}^{-1}$ and verify that Step 6 is correctly executed. Step 6 ensures that $\hat{\rho} = \hat{\nu}^{-1}\hat{\mu}$. Thus, $\hat{\rho} = \hat{\nu}^{-1}\hat{\mu} = \hat{\sigma}x^{-1}x\hat{\tau} = \hat{\sigma}\hat{\tau}$ as desired.

PROOF OF PRIVACY. Consider the following algorithm.

Algorithm *SIM-multiply*:

On input a public record R , (1) run *SIM-invert* on R to obtain public record R' ; (2) compute $maxid$, the maximum identifier of a ballot set consisting with R' ; (3) randomly select a permutation $r \in \text{SYM}_5$; and (4) output the public record obtained by appending to R' the following sequence of records (grouped so as to correspond to the records generated by each conceptual step of Protocol *Multiply*):

(NEWSUPER, ($maxid - 5, \tau_1$)) (NEWSUPER, ($maxid - 4, \tau_2$)) (NEWSUPER, ($maxid - 3, \tau_3$))
 (NEWSUPER, ($maxid - 2, \tau_4$)) (NEWSUPER, ($maxid - 1, \tau_5$))
 (BALLOTBOX, ($maxid + 1, \dots, maxid + 5$))
 (OPENSUPER, $maxid + 6$) (OPENSUPER, $maxid + 7$) (OPENSUPER, $maxid + 8$)
 (OPENSUPER, $maxid + 9$) (OPENSUPER, $maxid + 10$)
 (OPENENV, $maxid + 12, r_1$) (OPENENV, $maxid + 14, r_2$) (OPENENV, $maxid + 16, r_3$)
 (OPENENV, $maxid + 18, r_4$) (OPENENV, $maxid + 20, r_5$)
 (PUBLICPERMUTE, ($maxid + 11, maxid + 13, maxid + 15, maxid + 17, maxid + 19$), r^{-1})

Let us now prove that, *SIM-multiply* is indeed a simulator for protocol *Multiply*. For any global memory $gm^0 = (B^0, R^0, H^0)$ for which σ and τ are envelope encodings (of a permutation in SYM_5), consider (1) a random execution e of *Multiply* with initial global memory gm^0 , and (2) a random execution E of *SIM-multiply* on input R^0 . The privacy of protocol *Invert* implies that the 22 records produced by *Invert* in conceptual Step 1 of execution e are distributed identically to the corresponding records of R' produced by *SIM-invert* in E . Additionally, e 's records produced by Steps 2-4 of *Multiply* are identical to the first 11 records appended to R' in E . Indeed, these 11 records correspond to actions that (i) belong to the same classes; and (ii) specify the same sequence of envelope identifiers.²⁶

The remaining 6 records in the public record of e and E are, respectively,

(OPENENV, $maxid + 12, \hat{\nu}_1$) (OPENENV, $maxid + 14, \hat{\nu}_2$) (OPENENV, $maxid + 16, \hat{\nu}_3$)
 (OPENENV, $maxid + 18, \hat{\nu}_4$) (OPENENV, $maxid + 20, \hat{\nu}_5$)
 (PUBLICPERMUTE, ($maxid + 11, maxid + 13, maxid + 15, maxid + 17, maxid + 19$), $\hat{\nu}^{-1}$)

and

(OPENENV, $maxid + 12, r_1$) (OPENENV, $maxid + 14, r_2$) (OPENENV, $maxid + 16, r_3$)
 (OPENENV, $maxid + 18, r_4$) (OPENENV, $maxid + 20, r_5$)
 (PUBLICPERMUTE, ($maxid + 11, maxid + 13, maxid + 15, maxid + 17, maxid + 19$), r^{-1}).

Thus, while the two record subsequences may very well be different, they are identically distributed, for essentially the same reason as in the privacy proof of *Invert*.²⁷

Q.E.D.

²⁶Note that the value of $maxid$ at the beginning of e is derivable from R^0 , which is the same in both cases, and thus is equal to the value initially computed by *SIM-invert*.

²⁷Namely, in the latter subsequence, $r = r_1 r_2 r_3 r_4 r_5$ is the random permutation selected by *SIM-multiply*; while in the former subsequence $\hat{\nu} = \hat{\nu}_1 \hat{\nu}_2 \hat{\nu}_3 \hat{\nu}_4 \hat{\nu}_5 = x \hat{\sigma}$, where x is the realization of a random permutation (secretly selected by the ballot box). Therefore, since the product of a random permutation in SYM_5 and a fixed permutation in SYM_5 is a random permutation in SYM_5 , the two sequences are identically distributed.

G.6. A Ballot-Box Computer for Permutation Clone

Permutation clone is the function that, on input a permutation $p \in \text{SYM}_5$, returns the pair of permutations (p, p) .

Protocol Clone

Input address: σ — an envelope encoding of a permutation in SYM_5 .

1. Set $\alpha = \text{Invert}(\sigma)$.
2. Create an envelope encoding β of the identity permutation.
3. Create an envelope encoding γ of the identity permutation.
4. For $\ell = 1$ to 5: create a new super-envelope A_ℓ containing the triple of envelopes $(\beta_\ell, \gamma_\ell, \alpha_\ell)$.
5. Ballotbox A_1, \dots, A_5 to obtain A'_1, \dots, A'_5 .
6. For $\ell = 1$ to 5: open super-envelope A'_ℓ to expose envelope triple $(\mu_\ell, \nu_\ell, \eta_\ell)$.
7. For $\ell = 1$ to 5: publicly open envelope η_ℓ , and denote its content by $\hat{\eta}_\ell$. Set $\hat{\eta} = \hat{\eta}_1 \circ \dots \circ \hat{\eta}_5$.
8. Publicly permute μ_1, \dots, μ_5 according to $\hat{\eta}^{-1}$ to obtain ρ_1, \dots, ρ_5 . Set $\rho = \rho_1, \dots, \rho_5$.
9. Publicly permute ν_1, \dots, ν_5 according to $\hat{\eta}^{-1}$ to obtain ϕ_1, \dots, ϕ_5 . Set $\phi = \phi_1, \dots, \phi_5$.

Output addresses: ρ and ϕ .

LEMMA 4: *Protocol Clone is a Ballot-Box Computer for permutation clone.*

PROOF OF CORRECTNESS. To establish the correctness of *Clone* we must show that ρ and ϕ are both envelope encodings of $\hat{\sigma}$. The correctness of *Invert* implies that, at the completion of Step 1, α is an envelope encoding of permutation $\hat{\alpha} = \hat{\sigma}^{-1}$. Because Step 4 binds together, in the same super-envelope A_ℓ , the ℓ th envelope of α , β and γ , Step 5 applies the same, random and secret permutation to $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$. Letting x be this secret permutation, Step 6 “puts on the table” the envelope encodings $\mu = \mu_1, \dots, \mu_5$, $\nu = \nu_1, \dots, \nu_5$ and $\eta = \eta_1, \dots, \eta_5$ where $\hat{\mu} = x\mathcal{I}$, $\hat{\nu} = x\mathcal{I}$ and $\hat{\eta} = x\hat{\sigma}^{-1}$. Step 7, however, reveals $\hat{\eta}$ to all players, so that all players can compute $\hat{\eta}^{-1}$ and verify that Steps 8 and 9 are correctly executed. Steps 8 and 9 ensure that $\hat{\rho} = \hat{\eta}^{-1}\hat{\mu}$ and $\hat{\phi} = \hat{\eta}^{-1}\hat{\nu}$, respectively. Thus, $\hat{\rho} = \hat{\eta}^{-1}\hat{\mu} = \hat{\sigma}x^{-1}x\mathcal{I} = \hat{\sigma}$ and $\hat{\phi} = \hat{\eta}^{-1}\hat{\nu} = \hat{\sigma}x^{-1}x\mathcal{I} = \hat{\sigma}$ as desired.

PROOF OF PRIVACY. Consider the following algorithm.

Algorithm SIM-clone:

On input a public record R , (1) run *SIM-invert* on input R to obtain public record R' ; (2) compute *maxid*, the maximum identifier of a ballot set consisting with R' ; (3) randomly select a permutation $r \in \text{SYM}_5$; and (4) output the public record obtained by appending to R' the following sequence of records (grouped so as to correspond to the records generated by each conceptual step of Protocol *Clone*):

(NEWENV, 1) (NEWENV, 2) (NEWENV, 3) (NEWENV, 4) (NEWENV, 5)

(NEWENV, 1) (NEWENV, 2) (NEWENV, 3) (NEWENV, 4) (NEWENV, 5)
 (NEWSUPER, ($maxid - 5, maxid + 1, maxid + 6$)) (NEWSUPER, ($maxid - 5, maxid + 2, maxid + 7$))
 (NEWSUPER, ($maxid - 5, maxid + 3, maxid + 8$))
 (NEWSUPER, ($maxid - 5, maxid + 4, maxid + 9$)) (NEWSUPER, ($maxid - 5, maxid + 5, maxid + 10$))
 (BALLOTBOX, ($maxid + 11, \dots, maxid + 15$))
 (OPENSUPER, $maxid + 16$) (OPENSUPER, $maxid + 17$) (OPENSUPER, $maxid + 18$)
 (OPENSUPER, $maxid + 19$) (OPENSUPER, $maxid + 20$)
 (OPENENV, $maxid + 23, r_1$) (OPENENV, $maxid + 26, r_2$) (OPENENV, $maxid + 29, r_3$)
 (OPENENV, $maxid + 32, r_4$) (OPENENV, $maxid + 35, r_5$)
 (PUBLICPERMUTE, ($maxid + 21, maxid + 24, maxid + 27, maxid + 30, maxid + 33$), r^{-1})
 (PUBLICPERMUTE, ($maxid + 22, maxid + 25, maxid + 28, maxid + 31, maxid + 34$), r^{-1})

Let us now prove that, *SIM-clone* is indeed a simulator for protocol *Clone*. For any global memory $gm^0 = (B^0, R^0, H^0)$ for which σ is an envelope encoding (of a permutation in SYM_5), consider (1) a random execution e of *Clone* with initial global memory gm^0 , and (2) a random execution E of *SIM-clone* on input R^0 . The privacy of protocol *Invert* implies that the 22 records produced by *Invert* in conceptual Step 1 of execution e are distributed identically to the corresponding records of R' produced by *SIM-invert* in E . Additionally, e 's records produced by Steps 2-7 of *Clone* are identical to the first 21 records appended to R' in E . Indeed, these 21 records correspond to actions that (i) belong to the same classes; and (ii) specify the same sequence of envelope identifiers.²⁸

The remaining 7 records in the public record of e and E are, respectively,

(OPENENV, $maxid + 23, \hat{\eta}_1$) (OPENENV, $maxid + 26, \hat{\eta}_2$) (OPENENV, $maxid + 29, \hat{\eta}_3$)
 (OPENENV, $maxid + 32, \hat{\eta}_4$) (OPENENV, $maxid + 35, \hat{\eta}_5$)
 (PUBLICPERMUTE, ($maxid + 21, maxid + 24, maxid + 27, maxid + 30, maxid + 33$), $\hat{\eta}^{-1}$)
 (PUBLICPERMUTE, ($maxid + 22, maxid + 25, maxid + 28, maxid + 31, maxid + 34$), $\hat{\eta}^{-1}$)

and

(OPENENV, $maxid + 23, r_1$) (OPENENV, $maxid + 26, r_2$) (OPENENV, $maxid + 29, r_3$)
 (OPENENV, $maxid + 32, r_4$) (OPENENV, $maxid + 35, r_5$)
 (PUBLICPERMUTE, ($maxid + 21, maxid + 24, maxid + 27, maxid + 30, maxid + 33$), r^{-1})
 (PUBLICPERMUTE, ($maxid + 22, maxid + 25, maxid + 28, maxid + 31, maxid + 34$), r^{-1}).

Thus, while the two record subsequences may very well be different, they are identically distributed, for essentially the same reason as in the privacy proof of *Invert*.²⁹

Q.E.D.

G.7. A Ballot-Box Computer for a COIN

Protocol Coin

1. Create an envelope encoding α of the identity permutation.

²⁸Note that the value of $maxid$ at the beginning of e is derivable from R^0 , which is the same in both cases, and thus is equal to the value initially computed by *SIM-invert*.

²⁹Namely, in the latter subsequence, $r = r_1 r_2 r_3 r_4 r_5$ is the random permutation selected by *SIM-clone*; while in the former subsequence $\hat{\eta} = \hat{\eta}_1 \hat{\eta}_2 \hat{\eta}_3 \hat{\eta}_4 \hat{\eta}_5 = x \hat{\sigma}$, where x is the realization of a random permutation (secretly selected by the ballot box). Therefore, since the product of a random permutation in SYM_5 and a fixed permutation in SYM_5 is a random permutation in SYM_5 , the two sequences are identically distributed.

2. Create an envelope encoding β of permutation $a = 12453$.
3. Create a new super-envelope A containing envelopes $\alpha_1, \dots, \alpha_5$.
4. Create a new super envelope B containing envelopes β_1, \dots, β_5 .
5. Ballotbox A and B to obtain super-envelopes C and D .
6. Open C to expose envelopes $\gamma_1, \dots, \gamma_5$. Set $\gamma = \gamma_1, \dots, \gamma_5$.
7. Open D to expose envelopes $\delta_1, \dots, \delta_5$.
8. Ballotbox $\delta_1, \dots, \delta_5$ to obtain η_1, \dots, η_5 .
9. For $t = 1$ to 5 do: Publicly open envelope η_t to reveal content $\hat{\eta}_t$. Set $\hat{\eta} = \hat{\eta}_1 \circ \dots \circ \hat{\eta}_5$.

Output Address: Sequence γ .

LEMMA 5: *Protocol Coin is a Ballot-Box Computer for the Barrington Encoding of function COIN.*

PROOF OF CORRECTNESS. To establish correctness, we must show that γ contains the Barrington encoding of a random bit b . At the end of Step 3, A contains a sequence of 5 envelopes encoding the identity, and, at the end of Step 4, B contains a sequence of 5 envelopes encoding permutation a . Therefore, recalling that in the Barrington encoding $\mathcal{I} = \bar{0}$ and $a = \bar{1}$, at the end of Step 5, C contains an envelope encoding of \bar{b} , for a random bit b . (I.e., $b = 0$ if the ballot box, in Step 5, selects the identity permutation and $b = 1$ otherwise.) Thus, at the end of Step 6, the content of address γ is the Barrington encoding of a random bit, as desired.

PROOF OF PRIVACY. Consider the following algorithm.

Algorithm SIM-coin:

On input a public record R , (1) compute *maxid*, the maximum identifier of a ballot set consistent with R ; (2) randomly select a permutation $r \in \text{SYM}_5$; and (3) output the following sequence of records (grouped so as to correspond to the records generated by each conceptual step of Protocol *Coin*):

(NEWENV, 1) (NEWENV, 2) (NEWENV, 3) (NEWENV, 4) (NEWENV, 5)
 (NEWENV, 1) (NEWENV, 2) (NEWENV, 4) (NEWENV, 5) (NEWENV, 3)
 (NEWSUPER, (*maxid* + 1, *maxid* + 2, *maxid* + 3, *maxid* + 4, *maxid* + 5))
 (NEWSUPER, (*maxid* + 6, *maxid* + 7, *maxid* + 8, *maxid* + 9, *maxid* + 10))
 (BALLOTBOX, (*maxid* + 11, *maxid* + 12))
 (OPENSUPER, *maxid* + 13)
 (OPENSUPER, *maxid* + 14)
 (BALLOTBOX, (*maxid* + 20, *maxid* + 21, *maxid* + 22, *maxid* + 23, *maxid* + 24))
 (OPENENV, *maxid* + 25, r_1) (OPENENV, *maxid* + 26, r_2) (OPENENV, *maxid* + 27, r_3)
 (OPENENV, *maxid* + 28, r_4) (OPENENV, *maxid* + 29, r_5)

Let us now prove that, *SIM-coin* is indeed a simulator for protocol *Coin*. For any global memory $gm^0 = (B^0, R^0, H^0)$, consider (1) a random execution e of *Coin* with initial global memory gm^0 , and (2) a random execution E of *SIM-coin* on input R^0 . First observe that e 's records produced by Steps 1-8 are identical to the first 16 records of E . Indeed, these 16 records correspond to actions that (i) belong to the same classes; and (ii) specify the same sequence of envelope identifiers.³⁰ The remaining 5 records of the public record of e and E are, respectively,

$$\begin{aligned} & (\text{OPENENV}, \text{maxid} + 25, \hat{\eta}_1) (\text{OPENENV}, \text{maxid} + 26, \hat{\eta}_2) (\text{OPENENV}, \text{maxid} + 27, \hat{\eta}_3) \\ & (\text{OPENENV}, \text{maxid} + 28, \hat{\eta}_4) (\text{OPENENV}, \text{maxid} + 29, \hat{\eta}_5) \end{aligned}$$

and

$$\begin{aligned} & (\text{OPENENV}, \text{maxid} + 25, r_1) (\text{OPENENV}, \text{maxid} + 26, r_2) (\text{OPENENV}, \text{maxid} + 27, r_3) \\ & (\text{OPENENV}, \text{maxid} + 28, r_4) (\text{OPENENV}, \text{maxid} + 29, r_5) \end{aligned}$$

Thus, while these two record subsequences may very well be different, they are identically distributed. In fact, in the latter subsequence, $r = r_1 r_2 r_3 r_4 r_5$ is the random permutation selected by *SIM-coin*; while in the former subsequence $\hat{\eta} = \hat{\eta}_1 \hat{\eta}_2 \hat{\eta}_3 \hat{\eta}_4 \hat{\eta}_5 = x \hat{\delta}$, where x is the realization of a random permutation (secretly selected by the ballot box). Therefore, since the product of a random permutation in SYM_5 and a fixed permutation in SYM_5 is a random permutation in SYM_5 , the two sequences are identically distributed.

Q.E.D.

G.8. Ballot-Box Computers for AND and NOT

LEMMA 6: *There exist ballot-box computers \mathcal{P}_\neg and \mathcal{P}_\wedge for the Barrington encodings of, respectively, the Boolean functions NOT and AND.*

Proof. Notice that in Appendix F, if σ is the Barrington encoding of a bit b , then the Barrington encoding of $\neg b$ is

$$\overline{\neg b} = (\sigma a^{-1})^* = [a^{-1} \rightarrow a]^{-1} \sigma a^{-1} [a^{-1} \rightarrow a] = (12354)^{-1} \sigma (12453)^{-1} (12354) = 12354 \sigma 12534 12354.$$

Therefore, \mathcal{P}_\neg is easily constructed by creating envelope encodings of permutations 12354, 12534 and 12354 and properly executing protocol *Multiply* three times.

Similarly, if σ and τ are Barrington encodings of bits c and d , respectively, then the Barrington encoding of $c \wedge d$ is

$$\begin{aligned} \overline{c \wedge d} &= (\sigma \tilde{\tau} \sigma^{-1} \tilde{\tau}^{-1})' \\ &= [aba^{-1}b^{-1} \rightarrow a]^{-1} \sigma [a \rightarrow b]^{-1} \tau [a \rightarrow b] \sigma^{-1} [a \rightarrow b]^{-1} \tau^{-1} [a \rightarrow b] [aba^{-1}b^{-1} \rightarrow a] \\ &= 13245 \sigma 34125 \tau 34125 \sigma^{-1} 34125 \tau^{-1} 34125 13245 \end{aligned}$$

Therefore, \mathcal{P}_\wedge is easily constructed by creating two envelope encodings of 13245, creating four envelope encodings of 34125, executing *Clone* twice to create, respectively, two copies of σ and two copies of τ , *Invert* twice to create, respectively, envelope encodings of σ^{-1} and τ^{-1} , and then executing *Multiply* nine times.

Q.E.D.

³⁰Note that the value of *maxid* at the beginning of e is derivable from R^0 , which is the same in both cases, and thus is equal to the value initially computed by *SIM-coin*.

REMARK. Notice that P_{\wedge} and P_{\neg} are ballot-box computers for \overline{AND} and \overline{NOT} , the Barrington encodings of the binary functions AND and NOT . Thus, P_{\wedge} and P_{\neg} take as input envelope encodings of bits and produce envelope encodings of bits. By contrast, the component protocols *Invert*, *Multiply*, and *Clone* (that underly our construction of P_{\wedge} and P_{\neg}) are ballot-box computers operating on envelope encodings of arbitrary permutations in SYM_5 , rather than just 12345 or 12453.

G.9. A Ballot-Box Computer for \bar{g}

Recall that \bar{g} is the Barrington encoding of $g : \{0, 1\}^{nL} \rightarrow \{0, 1\}^L$. As for all finite functions, g is a modular composition of the functions AND , NOT , $DUPLICATE$ and $COIN$ (operating on bits). Accordingly, \bar{g} is a modular composition of the functions \overline{AND} , \overline{NOT} , $\overline{DUPLICATE}$ and \overline{COIN} (operating on Barrington encodings of bits). Therefore, since the protocols \mathcal{P}_{\wedge} , \mathcal{P}_{\neg} , *Clone* and *Coin* are, respectively, ballot-box computers for \overline{AND} , \overline{NOT} , $\overline{DUPLICATE}$ and \overline{COIN} , Theorem 3 implies the existence of a ballot-box computer for \bar{g} .

G.10. A Ballot-Box Revealer for the Barrington encoding

Notice that the output of any \bar{g} is a Barrington encoding of a binary string x . Thus, if x is L -bit long, its Barrington encoding \bar{x} has length $5L$ and thus the output address of a ballot-box computer for \bar{g} is a sequence of $5L$ envelope identifiers: $y = y_1, \dots, y_{5L}$. It is trivial to construct a ballot-box revealer for such an address sequence y : namely,

Protocol *Reveal_L*:

1. “For $\ell = 1$ to $5L$: publicly open envelope y_{ℓ} .”

This completes the proof of Theorem 1.

H. UNIVERSALITY AND EFFICIENCY OF OUR CONSTRUCTION

THEOREM 4: *There exists a linear-time algorithm \mathcal{C} that, on input any standard mediated mechanism \mathcal{M} , outputs a linear-time ballot-box mechanism \mathcal{B} that modularly implements \mathcal{M} .*

Prior to proving Theorem 4, let us clarify its statement by explaining:

1. What it means for \mathcal{C} to be linear-time;
2. How \mathcal{M} is represented as an input to \mathcal{C} ;
3. How \mathcal{B} is represented as an output of \mathcal{C} ; and
4. What it means for \mathcal{B} to be linear-time.

EXPLANATION 1. The precise efficiency of a concrete algorithm depends on the specific model of computation on which it is executed. The traditional model for analyzing the efficiency of concrete algorithms is the *Random-Access Memory (RAM) machine*. In essence, this is the idealization of a modern computer, where memory cells are large, plentiful and easily accessible.³¹ In our context, a RAM machine guarantees that each element of a public record can be stored in a single cell of

³¹See Cormen, Leiserson, Rivest, and Stein (2001) for a convincing explanation and justification of this model.

memory and that, after writing the address of a memory cell, the contents of the specified cell can be retrieved in unit time. As with any computer, a RAM machine can also perform such elementary computations as adding one to an integer variable in unit time.

Saying that \mathcal{C} runs in linear time means that there exists a positive constant c such that, whenever \mathcal{C} 's input is a binary string of length k , \mathcal{C} terminates within ck steps of computation.³² (As usual, the input of \mathcal{C} is assumed to be already stored in RAM when \mathcal{C} starts executing.)

EXPLANATION 2. A standard mediated mechanism $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$ is presented to \mathcal{C} as the standard binary encoding of C_g .³³ (Recall that C_g is easily computable from any other standard representation of g , and that C_g 's standard binary encoding has at most $4m \log(m)$ bits whenever C_g has m nodes.)

EXPLANATION 3. Algorithm \mathcal{C} represents the ballot-box mechanism $\mathcal{B} = (N, J, K, PS, AF, OF)$ as l -bit string consisting of:

- the cardinality n of N , written in binary,
- the integer J , written in binary,
- the integer K , written in binary,
- the sequence PS , written as $K \log(n)$ -bit integers,
- the sequence AF , written as K RAM-machine programs.
- the function OF , written as a RAM machine program.

(We also specify that \mathcal{C} includes its input as part of its output. That is, \mathcal{C} appends the standard binary encoding of C_g to the above representation of \mathcal{B} .)

EXPLANATION 4. Generally speaking, we consider a ballot-box mechanism \mathcal{B} easy to play if it is easily executable for any chosen profile of strategies. That is, we regard the time the players may invest in selecting their strategies as “orthogonal” to the efficiency of \mathcal{B} . Assume now that \mathcal{B} is described by an l -bit string, as per Explanation 3, output by compiler \mathcal{C} on input the standard mediated mechanism $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$. Then, since \mathcal{B} perfectly implements \mathcal{M} , a profile of strategies in \mathcal{B} consists of an n -tuple of L -bit strings, $m = (m_1, \dots, m_n)$.³⁴ Thus, saying that \mathcal{B} is playable in linear-time means that there exists a positive constant c such that, in any execution e with strategy profile m , the total number of steps required from each player i is upperbounded by cl , counting a ballot-box operation involving k ballots as a k steps. This is analogous to a RAM machine in which an elementary computation on k values in memory requires k steps. (Thus, because our ballot-box operations each manipulate —e.g., seal, access, or open— at

³²Representing inputs in binary prevents wasteful algorithms from being deemed efficient. For example, iterated addition is not an efficient way to multiply integers, but it would be if the input integers were represented in unary — i.e., if integer x were represented by a string of x 1s. Any alphabet with more than one symbol allows for exponentially more compact representation of inputs, which is a big jump with respect to unary representation. But adopting any alphabet with more than two symbols only shortens the input length a mere constant factor relative to the binary alphabet.

³³Indeed, C_g has nL sources and L sinks, and thus from it one immediately computes n , nL , and L , and thus a compact representation of \mathcal{M} 's player set $N = \{1, \dots, n\}$, its message space $(\{0, 1\}^L)^n$, and its outcome set $\{0, 1\}^L$.

³⁴Thus, from an efficiency point of view, it makes no difference whether the players choose their strategies given a description of \mathcal{M} or a description of \mathcal{B} , so long the outcome function g is easily derivable from \mathcal{B} 's description. Notice that this is indeed the case for the specific compiler \mathcal{C} we construct, but needs not be the case for all compilers. However, by requiring that all compilers output C_g as part of \mathcal{B} 's description, we bypass this potential difficulty of constructions different than ours.

most 5 distinct ballots, each of our ballot-box operations consists of a constant number of physical steps.)

The steps required of \mathcal{B} 's players are of two types: *computational steps* on a RAM machine (to run the program AF^k at each round k in order to compute the available actions) and *physical steps* on ballots (to carry out each chosen action on the current set of ballots). However, to show that a player's effort is linear in l , it suffices to show that his computational effort is linear in l . Notice in fact that the length of our chosen representation of \mathcal{B} upperbounds the number of rounds in an execution of \mathcal{B} and the number of physical steps is linear in the number of rounds.

The Proof of Theorem 4

We prove Theorem 4 by analyzing the proof of Theorem 1, and showing that for any standard mediated mechanism $\mathcal{M} = (N, (\{0, 1\}^L)^n, \{0, 1\}^L, g)$, the ballot-box protocol $\mathcal{B} = (N, K, PS, AF)$ that modularly implements \mathcal{M} is

1. Computable in linear time, and
2. Executable in linear time.

PROOF OF PROPERTY 1. The proof of Theorem 1 in Appendix G is quite constructive: for any standard mediated mechanism \mathcal{M} with outcome function $g : (\{0, 1\}^L)^n \rightarrow \{0, 1\}^L$, it describes a procedure—which we will now call \mathcal{C} —for constructing a ballot-box mechanism \mathcal{B} that perfectly implements \mathcal{M} . Procedure \mathcal{C} specifies \mathcal{B} as the concatenation of 3 sub-protocols: (i) an L -bit committer \mathcal{P}_1 ; (ii) a ballot-box computer \mathcal{P}_2 ; and (iii) a ballot-box revealer \mathcal{P}_3 . Therefore, we establish Property 1 by showing that the number of steps required for \mathcal{C} to output a description of each \mathcal{P}_i is linear in the length, l , of the standard binary encoding of C_g . Letting m be the number of nodes in C_g , recall that $m \log m \leq l \leq 4m \log(m)$ —see Appendix F.1. Thus to show linearity in l it suffices to show that, for each \mathcal{P}_i , the following two properties hold: (a) the number of rounds is linear in m ; and (b) there exists a constant c such that the steps required to output the action function AF^k of each round k is less than $c \log(m)$.

- Property (a) holds for \mathcal{P}_1 because this protocol consists of $nL < m$ bit commitments (i.e., one per input node of C_g), and each bit commitment consists of a 16-round protocol (simplified as 8 conceptual steps thanks to our protocol conventions).
- Property (a) holds for \mathcal{P}_2 because this protocol merely “replaces” each computation node of C_g with either protocol \mathcal{P}_\neg , protocol \mathcal{P}_\wedge , protocol *Clone* or protocol *Coin*, all four of which are fixed protocols—i.e., independent of C_g and thus consisting of a constant number of rounds.
- Property (a) holds for \mathcal{P}_3 because this protocol opens 5 envelopes for each sink of C_g .

A corollary of Property (a) is that the maximum identifier ever generated by an execution of \mathcal{B} is also linear in m , because at each round a single ballot-box action is executed, and each action generates at most 5 brand new identifiers. Therefore, every ballot identifier is describable by a $\log m$ -bit string.

To establish Property (b), we observe that each action function AF^k is of two types:

1. AF^k returns a constant action set.³⁵

In this case, the time required for \mathcal{C} to compute AF^k is just the time required to print it, which is linear in the length of the action set. This length is linear in $\log m$ because (1) the action set consists of at most 2 actions; and (2) each action can be described by a constant number of bits that identify the specific type of the action (out of the 8 available) plus $\log m$ bits for each of the (at most 5) ballot identifiers of the action.

2. AF^k returns an action set that depends on the current public record.

In this case, the action set always consists of a single `PUBLICPERMUTE` action—for example, $\{(PublicPermute, (j_1, \dots, j_5), \hat{\nu}^{-1})\}$ —where the permutation $\hat{\nu}^{-1}$ is computable from the last five elements of the public record. Thus, \mathcal{C} outputs a simple RAM-machine program that (1) accesses the last five components of the public record to compute $\hat{\nu}$; (2) computes $\hat{\nu}^{-1}$ from $\hat{\nu}$; and (3) outputs the string “ $\{(PublicPermute, (j_1, \dots, j_5), \hat{\nu}^{-1})\}$ ”. It is trivial to see that this program can be computed and printed in a number of steps linear in $\log(m)$.

A corollary of Property (b) is that, letting K be the number of rounds of \mathcal{B} , the length of \mathcal{B} 's description, \mathcal{L} , is linear in $K \log m$.

PROOF OF PROPERTY 2. Let us consider the computation required for a player i . To play \mathcal{B} , player i must (i) at each round, compute the identity of the active player; (ii) at each round, compute the set of available actions; and (iii) at the end of \mathcal{B} compute the outcome of the mechanism. Since \mathcal{L} , is linear in $K \log m$, we prove Property 2 by showing that the computation required for each of these three tasks is linear in $K \log m$.

- Computing the active player of a round k simply consists of retrieving the k th element of (the fixed sequence) PS , which can be done in unit time on a RAM machine. Thus, the total time spent for Task (i) is linear in K .
- At a round whose action set of \mathcal{B} is constant, the computation required from a player consists of reading the action(s)—which are then physically executed in constant time. Thus, the total computation associated to such rounds is upperbounded by \mathcal{L} and thus is linear in $K \log m$.

At all other rounds k , the action set is described by a program that computes the contents of 5 envelopes opened in rounds $k - 5, \dots, k - 1$.³⁶ The number of steps of each such program is upper-bounded by a fixed constant times $\log m$. (The bulk of the time goes towards writing down the $\log m$ -bit addresses of the 5 memory cells that store the previous 5 elements of the public record—which are then retrieved in constant time—and in deducing the envelope contents of each of these 5 retrieved records. Having done this, the available action is obtained by computing a finite function on the 5 envelope contents—i.e., integers between 1 and 5—

³⁵That is, the action set does not depend on the current public record. For instance, the action set $\{(NEWENV, 7)\}$. Notice that, for convenience, we have described the protocols of Appendix G via “variable identifiers”—e.g., σ_2 . But, \mathcal{C} can easily keep in RAM a table associating to such variable specific constants—e.g., “ $\sigma_2 = 7$ ”. Therefore, to determine a constant action set \mathcal{C} need only, in a unit step, to look-up the actual value of at most 5 variable identifiers. This look-up thus adds a constant to the total running time.

³⁶For instance, in each of protocols *Invert*, *Multiply* and *Clone*, there is a round k where the only available action a consists of publicly permuting envelopes j_1, \dots, j_5 according to a permutation $\hat{\nu}^{-1}$, where j_1, \dots, j_5 have already been hardwired by \mathcal{C} in the description of \mathcal{B} , while $\hat{\nu}^{-1}$ must be determined at execution time, from the contents of 5 envelopes, respectively opened at rounds $k - 5, \dots, k - 1$, which have been randomly and secretly permuted the ballot box. Therefore, the action set $\{a\}$ is obtained by executing the program consisting of (1) accessing the last five components of the public record to compute $\hat{\nu}$; (2) computing $\hat{\nu}^{-1}$ from $\hat{\nu}$; and (3) outputting the action set $\{(PublicPermute, (j_1, \dots, j_5), \hat{\nu}^{-1})\}$.

which takes constant time.) Accordingly, the total time required by Task (ii) is linear in $K \log m$.

- The outcome function of \mathcal{B} consists of performing a Barrington decoding of data found in the public record. In particular, to acquire the j th bit of the outcome, a player must retrieve 5 components of the public record from RAM and then determine if 5 integers found in these records are the Barrington encoding of zero or one. Therefore, the j th bit of the outcome can be computed in constant time. Accordingly, the total time required by Task (iii) is linear in L , where $L < K$.

Q.E.D.

I. PRIVATE OUTCOMES

J. A FINAL REMARK

In an extensive-form mechanism, whenever it is player i 's turn to act, the set of actions available to him must be fully specified. If \mathcal{M} is a concrete mechanism and an action set of i is $\{Left, Right\}$, then it must indeed be the case that *Left* and *Right* are the only two options really available to him. In most concrete settings, however, player i will have the ability of taking no action at all. For instance, if the acting player i is a driver idle at a T intersection, and his action set is $\{(turn) Left, (turn) Right\}$, he might keep on standing still rather than moving in either direction. Following computer science literature, we call such a “do-nothing action” an *abort*.³⁷ Because aborts are inherent in most concrete settings, any theory of practical implementation must deal with them.

Our ballot-box mechanisms have been carefully engineered so as to technically by-pass the issue of abort by construction. Indeed, any action set in a ballot-box mechanism consists of (1) a single public action or (2) two complementary private actions. In the first case, the action can be performed by *anyone*: all that the players need to see is that the action has been performed. In particular, public actions can be performed by an external, non trusted agent, which makes the issue of abort disappear completely. In the second case, while the other players “look away,” the acting player needs either to swap two ballots or leave them alone. Thus, performing no action is equivalent to leaving them alone.

Aborts may not so easily disappear in other concrete implementations of ideal mechanisms, but there are different ways to state perfect implementation. In one such way, the mechanism designer formally includes *abort* as an action available in each action set of a concrete mechanism; assigns special *abort outcomes* to terminal nodes reached after an abort by some player; and provides incentives that prevent players from aborting. Strategic equivalence can thus be formulated as outcome-preserving bijections between the players' strategies in the normal-form mechanism and their non-aborting strategies in its concrete implementation. In fact, all strategies that lead to aborting outcomes can be iteratively eliminated as being dominated.³⁸ This way was indeed taken in an earlier draft of our paper—that is, Izmalkov, Lepinski, and Micali (2005)—where we used a

³⁷An abort should not be confused with a non-participation option, which is an explicitly available strategic choice in many mechanisms.

³⁸In a second way, the normal-form mechanism is properly augmented *too*. That is, *abort* is formally added as a distinguished element to the message set of each player message set, distinguished abort outcomes are formally added to the outcome set, and the outcome function is properly extended so as to keep the current formulation of strategic equivalent intact.

slightly different version of ballot-box mechanisms. Specifically, in our earlier concrete mechanisms, as soon as a player played the abort action, a special terminal node was immediately reached and a sufficiently high fine imposed on the aborting player.

REFERENCES

- AUMANN, R. J., AND S. HART (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.
- BEN-OR, M., S. GOLDWASSER, AND A. WIGDERSON (1988): “Completeness theorems for fault-tolerant distributed computing,” in *Proceedings of the 20th Symposium on Theory of Computing*, pp. 1–10. ACM.
- 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.
- (2003): “Cheap talk in games with incomplete information,” *Journal of Economic Theory*, 108(1), 45–71.
- BOLTON, P., AND M. DEWATRIPONT (2005): *Contract Theory*. MIT Press, Cambridge, Massachusetts.
- BRANDT, F., AND T. SANDHOLM (2004): “(Im)possibility of unconditionally privacy-preserving auctions,” in *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 810–817. IEEE.
- CHAUM, D., C. CRÉPEAU, AND I. DAMGÅRD (1988): “Multi-party unconditionally secure protocols,” in *Proceedings of the 20th Symposium on Theory of Computing*, pp. 11–19. ACM.
- CORMEN, T. H., C. E. LEISERSON, R. L. RIVEST, AND C. STEIN (2001): *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts.
- DODIS, Y., S. HALEVI, AND T. RABIN (2000): “A cryptographic solution to a game theoretic problem,” in *In Advances in Cryptology — CRYPTO 2000, LNCS*, vol. 1880, pp. 112–130. Springer-Verlag.
- FORGES, F. (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 R. MYERSON (2005): “Sequential Equilibria in Bayesian Games with Communication,” Yale University, mimeo.
- GOLDREICH, O., S. MICALI, AND A. WIGDERSON (1987): “How to play any mental game,” in *Proceedings of the 19th Symposium on Theory of Computing*, pp. 218–229. ACM.
- GOLDWASSER, S., S. MICALI, AND C. RACKOFF (1985): “The knowledge complexity of interactive proof-systems,” in *Proceedings of the 17th Symposium on Theory of Computing*, pp. 291–304. ACM, Final version in *SIAM Journal on Computing*, 1989, 186–208.

- GORDON, S. D., AND J. KATZ (2006): “Rational Secret Sharing, Revisited,” Cryptology ePrint Archive, Report 2006/142, <http://eprint.iacr.org/>.
- GRIMM, V., F. RIEDEL, AND E. WOLFSTETTER (2003): “Low price equilibrium in multi-unit auctions: the GSM spectrum auction in Germany,” *International Journal of Industrial Organization*, 21(10), 1557–1569.
- HALPERN, J. Y., AND V. TEAGUE (2004): “Rational secret sharing and multiparty computation,” in *Proceedings of the 36th Symposium on Theory of Computing*, pp. 623–632. ACM.
- HOPPER, N., J. LANGFORD, AND L. A. VON AHN (2002): “Provably Secure Steganography,” in *Proceedings of the Crypto 2006*.
- IZMALKOV, S., M. LEPINSKI, AND S. MICALI (2005): “Rational secure computation and ideal mechanism design,” in *Proceedings of the 46th Symposium on Foundations of Computer Science*, pp. 585–594. IEEE.
- 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., S. MICALI, C. PEIKERT, AND A. SHELAT (2004): “Completely fair SFE and coalition-safe cheap talk,” in *Proceedings of the 23rd annual Symposium on Principles of distributed computing*, pp. 1–10. ACM.
- LEPINSKI, M., S. MICALI, AND A. SHELAT (2005): “Collusion-Free Protocols,” in *Proceedings of the 37th Symposium on Theory of Computing*, pp. 543–552. ACM.
- LYSYANSKAYA, A., AND N. TRIANOPOULOS (2006): “Rationality and Adversarial Behavior in Multi-Party Computation,” in *Proceedings of the Crypto 2006*.
- NAOR, M., B. PINKAS, AND R. SUMNER (1999): “Privacy Preserving Auctions and Mechanism Design,” in *Proceedings of the 1st conference on Electronic Commerce*. ACM.
- POSNER, R. A. (1981): “The Economics of Privacy,” *The American Economic Review*, 71(2), 405–409, Papers and Proceedings of the Ninety-Third Annual Meeting of the American Economic Association.
- RABIN, T., AND M. BEN-OR (1989): “Verifiable Secret Sharing and Multiparty Protocols with Honest Majority,” in *Proceedings of the 21st Symposium on Theory of Computing*, pp. 73–85. ACM.
- ROTHKOPF, M. H., T. J. TEISBERG, AND E. P. KAHN (1990): “Why are Vickrey Auctions Rare?,” *Journal of Political Economy*, 98, 94–109.
- SAIJO, T., T. N. CASON, AND T. SJÖSTRÖM (2003): “Secure Implementation Experiments: Do Strategy-proof Mechanisms Really Work?,” Discussion papers 03012, Research Institute of Economy, Trade and Industry (RIETI), available at <http://ideas.repec.org/p/eti/dpaper/03012.html>.

- SJÖSTRÖM, T., AND E. MASKIN (2002): *Handbook of Social Choice and Welfare* vol. 1, chap. Implementation Theory, pp. 237–288. North-Holland.
- STIGLER, G. J. (1980): “An Introduction to Privacy in Economics and Politics,” *The Journal of Legal Studies*, 9(4), 623–644, The Law and Economics of Privacy.
- URBANO, A., AND J. E. VILA (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 *Advances in Economic Theory, Fifth World Congress*, ed. by T. F. Bewley, pp. 33–70. Cambridge University Press, Cambridge, UK.
- YAO, A. (1986): “Protocol for Secure Two-Party Computation,” never published. The result is presented in ?.