Rational Secure Computation and Ideal Mechanism Design

Sergei Izmalkov, Silvio Micali, Matt Lepinski [IML05] FOCS'05

Arunachalaeshwaran V R

Indian Institute of Science

October 5, 2023

Part I: Motivations and Problem Statement

Part I

Motivations and Problem Statement

Motivation

- Secure Computation essentially guarantees that whatever computation n players can do with the help of a trusted party, they can also do by themselves. Fundamentally, however, this notion depends on the honesty of at least some players. The authors propose a notion that depends solely on *rationality* rather than *honesty*.
- Game theory doesn't talk about privacy.
- Secure computation doesn't take payoffs (which may exist) into account.

Motivation

Why bridge game theory and secure computation?

- Strategic form games are simultaneous and will be realized by having a trusted mediator. (Ideal game Γ)
- Secure computation can get rid of trusted mediators (when payoffs are not considered or with simple payoffs).
- Goal: Guarantee security even when all players pursue their selfish interests.

Problem Statement

Realize any (ideal) mechanism cryptographically. Albeit, in the ballot box model.

Part II: Security Notion

Part II

Security Notion

Definition and its Analysis

Rough Idea

Rational secure computation (ILM1 security) has a ideal world/real world definition. The ideal world game Γ is a finite, normal-form, *mediated* game of incomplete information with the players having their own privates types, utility functions and the game has a well-defined outcome. The the real world game (i.e the protocol) G is unmediated, extensive form game of incomplete information (having the same players, type sets, outcome set, and utility functions as Γ). We say that the protocol is ILM1 secure iff all the players find both these games equivalent.

- In Γ assume that, when player i aborts, a predetermined outcome Y_i (an unfavourable outcome for i.) is realized, and that no information about the players' reports is revealed.
- Any player i can abort G's communication prematurely, in which case the same predetermined outcome Y_i is realized as when i aborts in Γ.
- As Y_i is unfavourable outcome for i, no rational player will abort in Γ or G.

Definition and its Analysis

Informal Definition (ILM1)

G rationally and securely simulates (or realizes) Γ if the following two conditions hold:

1. For any equilibrium of Γ, of any strength (e.g., Bayesian-Nash, dominant strategy, ex post, etc.), there exists an equilibrium of G that has the same strength and induces the same payoffs for every player—and vice versa for any equilibrium of G.
2. The privacy of the players in Γ is exactly preserved in G: no group of players (whether collaborating or not) may acquire more information about other players' types or reports in G than they may in Γ.

The proposed protocol achieves a slightly more demanding (forced action) notion of security. Also, the protocol achieves perfect security.

1. Unlike Γ, G is an extensive form game so if it isn't constructed properly then it may lead to additional equilibria, but this is a problem. Why?
2. As in an extensive form game the players could transfer information from one player to another they may try to signal their type to other players. Can this be prevented by just restricting the message they can send.

The answer is that such additional equilibria may be detrimental to "the larger context." For instance, in mechanism design, the whole point is to find a game whose equilibria implement a desired function (i.e the social choice function), typically deemed to be beneficial to Society. If the players were able to introduce additional equilibria that benefit solely them, Society may suffer.

The answer is no. Preventing players in G from saying outright "I am of type X" is not enough. In order to guarantee privacy, players' communication strategies in G must be probabilistic. But even a minimal amount of entropy in a communication strategy enables steganography, that is, subliminal communication between two players that is provably undetectable by any one else.

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

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

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

- First of all, notice that, for any player i, aborting in G is fully equivalent to aborting in Γ. Though i may decide to abort at any time, Property 1 guarantees that when making this decision he has absolutely no additional information about other players's types or inputs; nor any idea of what the outcome of the game will be. Thus, i may decide to abort in G with exactly the same information as in Γ. To be sure, i can abort Γ only at the very beginning. In G, instead, he can choose the communication round at which to abort. This choice provides him with the ability to signal information about his own type, if he so desires. But this signaling does not affect any payoffs, because whenever i aborts the outcome always is the default value Y_i.
- Second of all, notice that for any player i, provided that no player aborts the game, sending a report m_i to the mediator in Γ is fully equivalent to committing m_i in the input stage of G. In either case, m_i cannot be changed any more: in Γ because there is no other action available to player i, and in G because of Property 2. Furthermore, in either case, f will be evaluated correctly: in Γ thanks to the trusted mediator, in G thanks to Property 4.

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

Part III: Communication and Computation Model

Part III

Communication and Computation Model

Communication model: Envelopes and Super Envelopes

- Players sitting around a table.
- Any player can communicate by broadcasting.
- Communicate via envelopes (a player create envelopes with the concealed value exposed publicly or private to the player). They contain a concealed message and can be read only by the owner. Transfer ownership by passing envelopes around. Envelopes guarantee privacy and integrity.
- Envelopes can be nested (these are called super-envelopes). The protocol does not nest more than two levels. The (super-)envelopes are ordered within a super-envelope.
- Whatever operation a player performs is transparent to all other players. So, no 'under the table' shenanigans can take place.
- A bunch of envelopes (respectively super-envelopes. If super-envelopes the number of envelopes within them must be equal) can be put into a ballot box and randomly shuffled (uniformly at random) and taken out. No one will know the exact permutation (guaranteed to be a permutation) unless the envelopes are read.
- (Super-)Envelopes can be discarded.

Computation Model

S_5 based realization of Boolean Circuits

- Compute based on elements of S_5. Seven constant permutations Id, a, b, $[a \to b]$, $[a^{-1} \to a]$ and $[aba^{-1}b^{-1} \to a]$ satisfying certain properties (see paper for details).
- Bit 0 is represented by Id and bit 1 by a.
- Using these constants and the group operation and group inverse operation, we can implement the NOT gate and AND gate (see paper for details). As we will see we can realize the cloning operation and random bit operation in the ballot box model, hence we can realize any computable (probabilistic) function.

Computation Model

Enveloped Permutation

We represent an element of S_5 as an enveloped permutation. That is a super-envelope containing 5 ordered envelopes where the i^{th} envelope contains the i^{th} component of the permutation.

Computation Model

Permutation Labelling

Evaluating circuit C realizing on f on specific inputs $x_1, \ldots, x_n$ yields a specific bit value for each wire w of C. A permutation-labeling of C is a function mapping each wire w of C to a label $(\vec{P}, p)$, where $p \in S_n$ is a permutation of the n players and $\vec{P}$ is a sequence of n permutations in $S_5 : P = (P_1, \ldots, P_n) \in (S_5)^n$. This is essentially used for secret sharing bit(w) hence it needs to satisfy:

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

Part IV: Protocol

Part IV

PROTOCOL

High-level Overview

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

1. Initial Step. For each input wire w of Player j, Player j computes a valid label $(\vec{P}, p)$, and gives to each player i the enveloped permutation $P_{p(i)}$.
2. Inductive Step. For each gate g, and for each output wire w of g, the players use their enveloped permutations for g's input wire(s) to jointly compute enveloped permutations $P_1, \ldots, P_n$ such that each player i owns $P_{p(i)}$, and $(\vec{P}, p)$ is a valid label for w.
3. Final Step After processing all gates as in the Inductive Step, each output wire w has a valid label (P, p) such that the product $P_1 \cdots P_n$ is Id if $\text{bit}(w) = 0$, and a otherwise. However, each enveloped permutations P_j belongs to a different player. Therefore in the final step, the players swap envelopes such that for all output wires w, if $\text{bit}(w)$ is a bit of Player i's output then Player i now owns the enveloped permutations $P_1, \ldots, P_n$. After the swap, he can open all relevant envelopes to learn $P_1, \ldots, P_n$ and thus compute the bit carried by w. [1]

[1] We will see how to implement this simultaneous swap later.

Commonly Used Routines

Cloning Routine

Player i needs to clone his input ρ (which is known to him and is an envelope permutation).

1. Player i publicly creates three envelope permutations of Id: Id_1, Id_2, Id_3.
2. Then creates 5 super-envelopes where the k^{th} one contains the k^{th} envelope of Id_j for all j.
3. He places these super-envelopes into the ballot box, shakes it and obtains three copies $\sigma_1, \sigma_2, \sigma_3$ (group them rank wise) of a random permutation (i.e σ) of them. That no one knows the permutation at this point.
4. Player i reads and reseals[2] σ_3. He computes and broadcasts a permutation π such that $\pi\sigma = \rho$ (this does not leak any information).
5. Reorder $\sigma_1, \sigma_2, \sigma_3$ according to π to obtain τ_1, τ_2, τ_3. Now, everyone knows $\tau = \tau_1 = \tau_2 = \tau_3 = \pi\sigma$. However, they have no reason to believe $\tau = \rho$. So, Player i will give a "Perfect ZK Proof" for the same.

[2] Can do without resealing but will need additional copies in step 1

Commonly Used Routines

Equality "Perfect ZK Proof" Routine

To show equality of ρ and τ_3.

1. Create a super-envelope grouping ρ and τ_3 rank wise.
2. Put it in the ballot box and obtain a random permutation.
3. Now, open each super-envelope and the two sub-envelopes within them publicly and show they are equal.

Observe that this also proves that ρ is an envelope permutation and not an invalid value like $[1, 3, 1, 2, 4]$.

Commonly Used Routines

Well-formedness of input wire "Perfect ZK Proof" Routine

This routine is used by Player i to prove that ρ is either a or Id without revealing what it is.

1. Publicly create enveloped permutations of a and Id.
2. Use the ballot box to obtain a random permutation of the enveloped permutations.
3. Player i reads both of them, picks the one that equals ρ and groups it rank-wise with ρ.
4. Again, uses the ballot box to obtain a random permutation. Now, open each super-envelope and the two sub-envelopes within them publicly and show they are equal.

Commonly Used Routines

Permutation Labelling Routine

At this point, Player i must compute from ρ, in a forced-action manner, enveloped permutations $P_1, \ldots, P_n$ such that (P, Id_n) is a valid label of his input wire w, where $P = P_1, \ldots, P_n$ and Id_n is the identity of S_n.

1. To begin with, for $j = 1$ to $n - 1$, Player i generates P_j by publicly creating an enveloped permutation Id and then using the ballot box to randomly permute these five envelopes.
2. Next, Player i privately opens, reads and reseals each envelope of $P_1, \ldots, P_{n-1}$ so as to learn $P_1, \ldots, P_{n-1}$.
3. Then, he computes P_n so that $P_1 \cdots P_n = \rho$ and generates the enveloped permutation P_n. At this point other players don't know if P_n is an enveloped permutation at all, much less $P_1 \cdots P_n = \rho$. So, player i provides a "perfect zk proof" for the same.
4. Finally, Player i transfers each enveloped permutation P_j to Player j, who privately opens, reads and reseals each of its envelope to learn P_j. Thus, $(P = (P_1, ..., P_n), \mathrm{Id}_n)$ is a valid permutation labeling of w.

1. First, for each j, he uses the same cloning procedure of Input Commitment to produce a clone $P_j^{'}$ of P_j. This also proves P_j is an enveloped permutation.
2. Next, Player i uses $n-1$ times the group operation routine to produce P from $P_1^{'}, \ldots, P_n^{'}$.
3. After Player i finishes using group operation routine $n-1$ times, all players are guaranteed that $P = P_1^{'} \cdots P_n^{'}$, and yet gain no additional information about any $P_j^{'}$. Because each $P_j^{'}$ is a clone of P_j, they are also guaranteed that indeed $P = P_1 \cdots P_n$.

Commonly Used Routines

Group Operation Routine

Let α and β be two enveloped permutations owned by and known to the same player i. To produce an enveloped permutation γ and prove in perfect zero knowledge that $\gamma = \alpha\beta$, Player i acts as follows.

1. First, he publicly creates an enveloped permutation Id.
2. Next he creates five new super envelopes, $E_1, \ldots, E_5$. He packs into each E_k a pair of envelopes: the first is the k^{th} envelope of Id, and the second is the k^{th} envelope of β.
3. He then uses the ballot box to obtain $E_1^{'}, \ldots, E_5^{'}$, a random reordering of $E_1, \ldots, E_5$.
4. Next, he publicly opens $E_1^{'}, \ldots, E_5^{'}$, and organizes the "exposed" envelopes into two new enveloped permutations, μ and ν: the k^{th} envelope of μ (respectively, ν) is the first (respectively, second) envelope from $E_k^{'}$.
5. Therefore, everyone is guaranteed that μ is a random element of S_5 (unknown to everyone), and that $\nu = \mu\beta$.
6. At this point, Player i privately opens, reads and reseals each envelope of μ, thus becoming the only player to know μ.
7. Then, he computes and broadcasts π (leaks no information), a permutation in S_5 such that $\pi\mu = \alpha$.
8. Finally, Player i publicly reorders (other players monitor) the envelopes of μ according to π so as to obtain a new enveloped permutation ϕ. Similarly, he reorders the envelopes of ν according to π to obtain γ. Player i gives "Perfect zk proof" of $\phi = \alpha$ to complete this routine.

This also gives a simple way perform group inversion operation.

In this step we compute the intermediate parts of circuit C^3. This is essentially done using the group operation and group inversion operation. It is slightly more complicated than just using the implementation of AND gate (or NOT gate) because one player may not have access to the inputs directly but they maybe secret shared among all the players. We can re-randomize the secret sharing using the group inversion operation if it is needed.

[3] Not in the original paper

The paper lists three different ways of performing simultaneous exchange (everyone needs to observe this enforce forced action like the other steps) in a footnote but each of them requires some kind of new assumption or compromise:

1. Using a "revolving-door" like physical device. (Requires an additional assumption of a physical device)
2. Have a referee who is responsible for transferring who can be held accountable if he fails to transfer the (super-)envelopes properly. Although, this is a weaker assumption than a mediator this doesn't come for free.
3. Lastly, a slightly weaker version of simultaneous in which we punish player(s) who fail to transfer the envelopes properly. Something like this might be needed to preventing peeking. Punishing in both these cases can be seen as an abort.

Part V: Motivations and Problem Statement

Part V

Implications for Mechanism Design

Main Theorem (Informal)

Any mediated game of incomplete information Γ can be rationally and securely simulated by a ballot-box game G.
So, once we have a ideal mechanism we can get a perfectly secure protocol essentially for free in the ballot box model (if we don't worry too much about concrete efficiency and multiple equilibria (not an issue when truth telling is a dominant strategy equilibrium)).

Modular Mechanism Design (and UC security)

This protocol is universally composable (information theoretically) and enjoys nice properties like synchronous reducibility. Modularity achieved in follow up paper which deals with a sequence of mediated (correlated) games.

Privacy and Mechanism Transparency

The protocol dispenses with the need to trade-off between the amount of information that is publicly revealed and the amount of trust bestowed on the mechanism. More generally, the protocol enables one to implement social-choice functions with "private outcomes." In particular, assuming that actual transactions can be kept private, we can implement second-price auctions where the losers learn only that they have lost, the winner learns that he has won and at what price, and the seller learns only the winner and the price.

Part VI: Motivations and Problem Statement

Part VI

Discussion

Follow up papers

Searching google scholar for these three authors brings up this series of papers.

1. Verifiably secure devices [ILM08] (ILM1 and ILM2 security)
2. Perfect implementation [ILM11]
3. Perfect implementation of normal-form mechanisms [ILM05a]
4. Universal mechanism design [ILM05b]
5. Perfect concrete implementation of arbitrary mechanisms: A quick summary of joint work with Sergei Izmalkov and Matt Lepinski [Mic10]
6. The Security Power of the Ballot Box [LI05]

Limitations

- Inefficient, practical implementations will likely relax security to be computational (rather than perfect) for efficiency.
- Strong assumption of transparent computation (i.e the ballot box model).
- Signalling using other channels. (Prior communication not a problem as G is a mediated game of incomplete information and these things can considered to be part of the prior.)
- How to go about hiding types or prove ignorance of types of other players? (why proofs of ignorance don't work)
- Mesh network only practical for small n.
- What is synchronous reducibility?

References I

[ILM05a] Sergei Izmalkov, Matt Lepinski, and Silvio Micali. **"Perfect implementation of normal-form mechanisms".** In: (2005).

[ILM05b] Sergei Izmalkov, Matt Lepinski, and Silvio Micali. **"Universal mechanism design".** In: *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*. 2005.

[ILM08] Sergei Izmalkov, Matt Lepinski, and Silvio Micali. **"Verifiably secure devices".** In: *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5*. Springer. 2008, pp. 273–301.

[ILM11] Sergei Izmalkov, Matt Lepinski, and Silvio Micali. **"Perfect implementation".** In: *Games and Economic Behavior* 71.1 (2011), pp. 121–140.

[IML05] Sergei Izmalkov, Silvio Micali, and Matt Lepinski. **"Rational secure computation and ideal mechanism design".** In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE. 2005, pp. 585–594.

[LI05] Matt Lepinski and Sergei Izmalkov. **"The Security Power of the Ballot Box".** In: (2005).

[Mic10] Silvio Micali. **"Perfect concrete implementation of arbitrary mechanisms: A quick summary of joint work with Sergei Izmalkov and Matt Lepinski".** In: *Proceedings of the Behavioral and Quantitative Game Theory: Conference on Future Directions*. 2010, pp. 1–5.