Subcontracted Rational SFE

Matt Lepinski and Silvio Micali

November 1, 2005

Abstract

In their paper, "Rational Secure Computation and Ideal Mechanism Design," Izmalkov, Lepinski and Micali show that any one-shot mediated game can be simulated by the players themselves, without the help of a trusted mediator, using physical envelopes and a ballot-box. We show that communication between the players is not essential to the ILM protocol. That is, we provide a protocol for rational secure function evaluation (Rational SFE) where the players just send a set of envelopes to a referee who simply performs a sequence of publicly verifiable actions. That is, the players can "subcontract" all of the computation to an untrusted referee. In addition to providing a communication structure that more closely matches the ideal game, our protocol also enables us to better simulate mediated games in which abort is not a dominated action.

1 Prior Work

The general notion of secure multi-party computation was put forward and exemplified by Goldreich, Micali and Wigderson [3], based on the two-party results of Yao [6]. In particular, the notion encompasses *secure function evaluation (SFE)*. Assume that n parties, where party i has a secret input x_i wish to evaluate a a vector valued function f on their secret inputs, without revealing any more about their original x_i's than strictly necessary. In essence, an SFE protocol enables the n parties, without any external help, to communicate so that each player i computes y_i, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ with the same privacy and correctness as when each player i confides his x_i to a trusted party, who then computes f and privately hands out y_i to each player i. The notion of an SFE has found much favor and has been formalized and implemented for a variety of communication models. In particular, the original paper envisaged that all players communicate via broadcasting (and use encryption to enforce the privacy constraints). As for another example, [1, 2] assume that each pair of players is endowed with a dedicated and perfect communication channel (i.e. no one can tap the conversation of two players i and j and no one else can inject messages over their channel).

Izmalkov, Lepinski and Micali [4] point that all previous SFE protocols rely on the honesty of some players, and hence they can be inappropriate in setting where different outputs of f yield different rewards. Thus, they introduce and exemplify the new notion of a *rational*

SFE, whose security depends on the rationality of the players and not on their honesty. Let us assume for simplicity only, that f actually returns the same value Y to all players, that is, that the players want to compute a n-input, single-output function: they want to compute $f(x_1, \ldots, x_n) = Y$). Then a Rational SFE for such an f and for any utilities associated to f's outputs guarantees that n rational players can communicate so as to perfectly simulate (i.e., enjoy the same privacy and incentives) of the following mediated game: each player i hands an input x_i to a trusted mediator, who, if all players provide him with a valid input in f's domain, computes f and announces the result, or else, if a player does not provide him with a valid input, announces the identity of the player and imposes a sufficiently large monetary fine on him, so that no rational player would choose not to provide a valid input to the mediator. To provide such a simulation of a mediated game, ILM use the ballot-box communication model. (In essence, the players may generate and exchange envelopes, or secretly randomize the order of some envelopes. At the end, all players hand their yet unopened envelopes to a referee who publicly opens all of them, so as to enable all player to compute the output Y.)

A crucial feature of a Rational SFE is that players should be prevented from signaling to one another (else no perfect simulation of a mediated game may exist, because such signaling is by definition impossible when a player can only send a single message—their input— to the trusted mediator). The need for imposing a monetary fine in the mediated game stems from the fact that in the ballot-box protocol that perfectly simulates it a player may abort at any time. In which case, there is no way of properly continuing the protocol so as to produce the proper Y. This is so because each player i owns a sequence of envelopes at each point in time whose content is secret to all other players, but *known* to i. Thus, if i aborts destroying or taking away his envelopes, the protocol cannot be completed without his help. Thus no simulation may be perfect unless we allow players to "abort" also in the mediated game (i.e., by not sending a proper input to the trusted mediator) but dissuade this from happening.

2 Results

We put forward a Rational SFE (indeed, even the first SFE!) satisfying the following property in the ballot-box model: Each player takes only a single initial action, which consists of handing a sequence of envelopes to the referee. The contents of the envelopes provided by a player i encode i's input. After all players have done so, the referee performs a deterministic sequence of ballot-box operations such that all players can observe that the right operation has been performed. The last operation of the referee is to publicly open all envelopes so that all players learn the right output Y.

In essence, we succeed in implementing the ILM protocol without the requirement that a player acting on a set of envelopes actually knows their contents. In a sense, after the initial input stage, all envelopes are "owned" by the referee who performs all necessary operations without ever learning their content (except of course for the last public opening).

Our result enables us to match any mediated game without relying on monetary fines. Whatever the mediator could have done when a player i aborts in the mediated game, we could do in our ballot-box protocol when player i does not provide envelopes to the referee.

(As in the ILM case, a player not providing envelopes that encode a valid input is quickly exposed and treated as not having provided any envelopes altogether.)

(Notice that if the players can provide the referee with their initial envelopes without noting whether or not the other players do the same, then we can even simulate any mediated game in which abort is interpreted by the mediator as a player confiding to him a default input.)

Our result, even in the standard SFE setting provides the first polynomial-time solution to a problem raised by Feige, Kilian and Naor.[5] That is, the problem of performing SFE when each player sends only a single message to an untrusted external party. (We note that our model is slightly different from —and incomparable with— the original FKN model.)

3 Techniques

The simplest way to obtain our result is to allow for a probability of error that is negligible in an auxiliary security parameter k that is independent of the function f, and to disregard efficiency (though remaining polynomial time in k and the time to compute f). Still, even our simplest protocol is a "Las Vegas" type algorithm. That is, the error is of the best possible type: When the protocol produces an output, the result is both correct and private, and with negligible probability (in k) it produces no output at all.

As in ILM, we follow the same basic data encoding as in GMW. Each bit is encoded as one of two permutations in S_5. The identity permutation corresponds to the bit 0 and a special permutation which we denote a corresponds to the bit 1. In ILM, each permutation is encoded using a sequence of five envelopes. Instead, we encode a permutation α using 120k identical sequences (each consisting of five envelopes whose contents are the standard representation of the permutation α).

At the beginning of the protocol, each player i for each bit of his input sends to the referee 120k copies of a five envelope sequence encoding this bit of his input. For now let us assume that each player publicly creates these 120k sequences using the following subprotocol. First, the player publicly creates $120k + 1$ sequences encoding the identity and $120k + 1$ sequences encoding the permutation a. The player then privately creates one sequence of five envelopes, which we denote by ϕ, that encodes his chosen bit. The player then puts the $120k+1$ sequences encoding the identity into a super envelope and the $120k+1$ sequences encoding a into a second super envelope. The player then uses the ballot-box to randomize the order of the two super-envelopes. He then opens the two super envelopes to reveal a stack of $120k + 1$ sequences on the right and a stack of $120k + 1$ sequences on the left. He privately reads (and reseals) one sequence from each stack to determine which is the identity and which is a. Next then discards the stack of envelopes that does not encode his chosen bit, leaving him with one stack of $120k + 1$ sequences in a stack and the single sequence ϕ which he privately created to encode his bit. We denote by ψ the top sequence of five envelopes from the stack of un-discarded sequences.

The player needs to prove that ϕ and ψ encode the same permutation. To do this, the player creates five super envelopes, such that the jth super envelope contains the jth envelope from sequence ϕ and the jth envelope from sequence ψ. The player then uses the ballot box to randomizes the order of the five super-envelopes. He then opens the super envelopes to reveal five pairs of envelopes. Finally, he opens publicly opens these five pairs of envelopes and reveals their contents. Notice that if $\phi = \psi$ then the two values contained in each pair of envelopes will be the same. Furthermore, in such a case, the revealed values provide no information about ϕ and ψ other than that they are equal. If any pair of revealed values are different then this is treated as an abort by the active player. This concludes the proof that $\phi = \psi$. At the conclusion of this proof the active player has $120k$ sequences of envelopes which can be given to the referee as the valid encoding of the player's input bit.

We have now explained how the initial inputs are encoded as $120k$ five-envelope sequences. In essence, the ILM protocol reduces the secure computation any function to the following task: A player i has a proper encoding of permutation α and a proper encoding of permutation β and must produce a proper encoding of the permutation $\alpha\beta$. In the ILM paper, they present a procedure for this task which requires that player i knows α. Here we present a new procedure, making essential use of our redundant data encoding, that allows the referee (who has a proper encoding of α and a proper encoding of β) to produce a proper encoding of $\alpha\beta$ *without knowing anything about α or β*. Note that our procedure destroys both the encoding of α and the encoding of β and therefore (in order to use the values α and β again later in our Rational SFE) we must also have a procedure which given a proper encoding of a permutation ϕ produces two proper encodings of the same permutation ϕ.

We first present our procedure which given an encoding of α and an encoding of β produces an encoding of $\alpha\beta$. First, the referee creates $120k$ sequences that encode the identity permutation. He then creates five super envelopes where the jth super envelope contains the jth envelope from the $120k$ identities followed by the jth envelope from the $120k$ sequences for β. Next, the referee uses the ballot box to randomly permute these five super envelopes. If ρ is the secret permutation chosen by the ballot-box, then the when the referee opens the five super envelopes, he obtains $120k$ sequences containing ρ and $120k$ sequences containing $\rho\beta$. Note that if $\rho = \alpha$ then the $120k$ sequences containing $\rho\beta$ are a proper encoding of $\alpha\beta$ and we are done.

In order to determine whether this is the case, the referee takes one sequence containing ρ and one sequence containing α and pairs the envelopes of ρ and the envelopes of α into five super envelopes. The ballot-box is then used to permute these five super envelopes and opens them to obtain two five-envelope sequences. Notice that if $\rho = \alpha$ then these two five-envelope sequences both contain the same random permutation (and otherwise they contain a random pair of permutations whose "difference" is $\rho\alpha^{-1}$). The referee thus opens both five-envelope sequences and everyone learns whether $\rho = \alpha$.

As stated previously, if the test reveals that $\rho = \alpha$ then the referee has successfully created a proper encoding of $\alpha\beta$ as desired. Otherwise, the referee is left with $120k - 1$ sequences containing ρ and $120k$ sequences containing $\rho\beta$. As before, these sequences can be placed in five super envelopes and permuted using the ballot box to obtain (for a random and independent ρ') $120k - 1$ sequences containing ρ' and $120k$ sequences containing $\rho'\beta$. As before the referee can test whether $\rho' = \alpha$ and if so he has successfully created a proper encoding of $\alpha\beta$. Otherwise he repeats the process until either (1) he successfully creates a proper encoding of $\alpha\beta$ or (2) he has failed $120k$ times in a row an run out of five-envelope sequences containing α. (Notice that event (2) happens with probability negligible in k).

As previously discussed, we also need a procedure which takes in a proper encoding of permutation ϕ and produces two proper encoding of permutation ϕ. To do this, the referee first publicly creates $360k$ five-envelope sequences containing the identity permutation. He then creates five super envelopes where the jth super envelope contains the jth envelope from each of the $360k$ identities. Next, the referee uses the ballot box to randomly permute these five super envelopes. If ρ is the secret permutation chosen by the ballot-box, then the when the referee opens the five super envelopes, he obtains $360k$ sequences containing ρ. Note that if $\rho = \phi$ then he has $360k$ five-envelope sequences containing ϕ. In such a case the referee has succeeded since he can easily use $240k$ of these $360k$ sequences to create two proper encodings of ϕ. To test whether $\rho = \phi$ he uses the same equality testing subprocedure from the multiplication procedure above.

If the test reveals that $\rho \neq \phi$ then the referee is left with $360k - 1$ sequences containing ρ. As before, these sequences can be placed in five super envelopes and permuted using the ballot box to obtain (for a random and independent ρ') $360k - 1$ sequences containing ρ'. As before the referee can test whether $\rho' = \phi$ and if so he can easily create two proper encodings of ϕ. Otherwise the referee repeats the process until either (1) he succeeds and is able to create two proper encodings of ϕ or (2) he has failed $120k$ times in a row an run out of five-envelope sequences from his original encoding of ϕ. (Notice that event (2) happens with probability negligible in k).

In summary, our protocol follows the same structure as the ILM protocol, except we use redundant data encoding scheme along with the above multiplication procedure and the above duplication procedure in place the ILM multiplication and duplication procedures which require the player performing the operations to know the value α (for multiplication) and ϕ for duplication.

Note that above we required each player i to execute a public input commitment protocol in order to create the $120k$ five-envelope sequence that encodes his input bit. Instead, we could player i privately create the $120k$ five-envelope sequences and hand them to the referee. The difficulty in this that the player may choose to give the referee $120k$ five-envelope sequences which do not properly encode any bit. (In particular, the $120k$ sequences might not all have identical contents). The solution is to have the player instead submit $240k$ (supposedly identical) five-envelope sequences to the referee. The referee could then pick $60k$ random pairs of sequences and test that they are equal. Such testing provides great confidence that a super majority of the remaining $120k$ sequences have identical contents. This naturally, requires some changes to the above procedures to account for the fact that a "proper encoding" now guarantees only that a super majority of the sequences are identical (as opposed to all of the sequences being identical as above). A full discussion of these modifications is beyond the scope of this technical memo and will be presented in a later version of this work.

References

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for fault-tolerant distributed computing. In *Proc. of STOC '88*, pages 1–10, 1988.

[2] D. Chaum, C. Crépeau, and I. Damgård. Multi-party unconditionally secure protocols. In *STOC '88*, 1988.

[3] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87*, pages 218–229. ACM, 1987.

[4] Matt Lepinski Sergei Izmalkov and Silvio Micali. Rational secure computation and ideal mechanism design. In *Proc of FOCS '05*, 2005.

[5] Joe Kilian Uri Feige and Moni Naor. A minimal model for secure computation. In *Proc of STOC '94*, 1994.

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