Enforcing Input Correctness via Certification in Garbled Circuit Evaluation

Yihua Zhang[1], Marina Blanton[2], and Fattaneh Bayatbabolghani[1]

[1] Computer Science and Engineering, University of Notre Dame
{yzhang16,fbayatba}@nd.edu
[2] Computer Science and Engineering, State University of New York at Buffalo
mblanton@buffalo.edu

Abstract. Secure multi-party computation allows a number of participants to securely evaluate a function on their private inputs and has a growing number of applications. Two standard adversarial models that treat the participants as semi-honest or malicious, respectively, are normally considered for showing security of constructions in this framework. In this work, we go beyond the standard security model in the presence of malicious participants and treat the problem of enforcing correct inputs to be entered into the computation. We achieve this by having a certification authority certify user's information, which is consequently used in secure two-party computation based on garbled circuit evaluation. The focus of this work on enforcing correctness of garbler's inputs via certification, as prior work already allows one to achieve this goal for circuit evaluator's input. Thus, in this work, we put forward a novel approach for certifying user's input and tying certification to garbler's input used during secure function evaluation based on garbled circuits. Our construction achieves notable performance of adding only one (standard) signature verification and $O(n\rho)$ symmetric key/hash operations to the cost of garbled circuit evaluation in the malicious model via cut-and-choose, in which ρ circuits are garbled and n is the length of the garbler's input in bits. Security of our construction is rigorously proved in the standard model.

Keywords: Garbled circuits, Input certification, Input verification, Secure function evaluation

1 Introduction

Secure multi-party computation (SMC) is a mature research area of computer science that has experienced dramatic advances in recent years. A new secure multi-party construction or protocol is expected to be shown secure against a formal security definition specifying the adversarial model. The two most fundamental and now standard security models correspond to modeling the computation participants as semi-honest (or honest-but-curious or passive) or malicious (or active). With semi-honest adversaries, the participants are trusted to follow the prescribed computation, while a malicious adversary can instruct the participants under its control to arbitrarily deviate from the computation in the attempt to learn authorized information about the honest parties' inputs. While these definitions are strong and do not tolerate unintended information leakage, the largest limitation of these standard definitions is that they provide no guarantees with respect to inputs of the computation.[3] Thus, a dishonest participant is able to modify its input into the computation, which results in other participants receiving incorrect results, while the dishonest party itself might be able to compute true output based on its knowledge of the original data transformation. This also allows a dishonest participant to set its inputs into the computation in such a way as to learn the maximum amount of information about input of other participant(s) from the output it receives.

The issue with inability of honest participants to control inputs of malicious participants under the current security definitions has been recognized in the literature and various techniques were proposed to mitigate the problem. Examples include employing game-theoretic techniques to incentivize providing truthful inputs into secure multi-party computation (see, e.g., [14, 15, 28]) and input certification in the context of specific applications such as private set intersection [9, 11] and anonymous credentials [8]. More recently, input certification or input validity verification in the form of any function has been added to general secure computation techniques. In particular, Blanton and Bayatbabolghani [3] put forward an efficient construction for server-aided secure two-party computation based on garbled circuit evaluation, where the inputs of the two users are certified and equality of the signed data and inputs into the computation is verified by utilizing signatures with protocols. Katz et al. [16] design an efficient mechanism for adding input verification in the form of an arbitrary function to two-party computation based on garbled circuits so that the computation takes place only if the inputs pass verification (and the computation takes place on the same inputs that were verified). We continue this line of work in this paper.

The focus of this article is on enforcing input correctness in secure computation via input certification. Because signature-based input certification is more amenable to integration with existing malicious-adversary techniques based on homomorphic encryption or secret sharing than garbled circuit evaluation, we would like to tackle the more interesting case of garbled circuit evaluation. To that extent, our starting point was the work of Blanton and Bayatbabolghani [3] for the server-aided two-party setting. We aim to enforce input correctness of both circuit garbler and evaluator in the standard two-party setting using garbled circuit evaluation techniques. Toward this goal, we notice that the mechanism for enforcing input correctness of the user evaluating the circuit in [3] will also work with the regular two-party setup with no changes. In particular, that mechanism requires the circuit evaluator to prove via zero-knowledge proofs of knowledge that the inputs provided into the oblivious transfer (at the time of retrieving garbled labels corresponding to the evaluator's inputs) are consistent with the

[3] Note that this does not refer to ill-formed inputs which can be detected in the beginning of the computation, but rather to well-formed incorrect or deceptive inputs.

valued signed by a certification authority. The mechanism for enforcing input correctness of the other user in [3], however, cannot be used for enforcing input correctness of the circuit garbler in the conventional setup. Thus, the focus of this work is on designing an efficient mechanism for enforcing input correctness of the garbler with the help of a certification authority.

Our setup assumes the presence of a certification authority that can certify users' data. For example, in the case of genomic or medical data, the facility performing genome sequencing or running a medical test will issue certification that can later be used with secure two-party computation. Obviously, the use of certification should not reveal any information about the certified values. For example, [3] used signatures with protocols that allows the signature owner to prove statements about signed values in zero-knowledge. Once the certification step is complete, the user will be able to use that information with garbled circuit evaluation to prove correctness of the supplied inputs.

In this work we put forward a novel, non-standard certification construction for use with garbled circuit evaluation that favorably compares with existing signature schemes and other prior work in terms of its performance. The certification authority's work includes only one public-key operation (producing a regular signature) for user's input of any size and the number of symmetric key or hash function operations is $O(n\rho)$, where n is the bitlength of the user's input to be certified and ρ is a statistical security parameter (that determines the number of circuits being garbled). The size of a certificate is $O(n + \rho)$, and the cost of secure function evaluation is only insignificantly higher than that of regular garbled circuit evaluation in the malicious model with no enforcement of input correctness. In particular, the work associated with using garbler's input certification and verification in secure function evaluation based on garbled circuits is only one signature verification and $O(n\rho)$ symmetric key/hash operations when the garbler's input is n bits long. The downside of the approach is that the certification authority's work is linear in the number of times the certificate is to be used in secure computation. That is, in order to use the same input in up to k independent secure function evaluations (with possibly different functions), the user will need to obtain a certificate of size $O(n + \rho k)$ and the certification authority's work increases to $O(n\rho k)$.

2 Related Work

Literature on secure two-party and multi-party computation is extensive and provides different mechanisms for securely evaluating a function on private inputs. Following the seminal work of Yao [30], it has been known that any computable function can be securely evaluated. Consequent work focused on providing stronger security guarantees such as security in the presence of malicious or covert adversaries (see, e.g., [21, 10] among many others) or improving performance of existing techniques (see, e.g., [5, 2] among others). There is also a large variety of custom constructions optimized for evaluating specific functions with the goal of providing more efficient solutions than their generic counterparts (see, e.g., [23, 4] among others).

The original Yao's construction is secure against semi-honest participants, and several solutions for making it resilient to malicious behavior exist. Early examples of constructions secure in the malicious model include [13, 12] that utilize zero knowledge proofs of knowledge. An alternative approach to making secure function evaluation based on garbled circuits resilient to malicious behavior is to use cut-and-choose techniques [20, 24, 19], which we further detail in Section 3.

Moving closer to the focus of this work, several publications treat the issue of input consistency or correctness in different settings and using different mechanisms. For example, the issue of input consistency arises in the context of garbled circuit evaluation based on cut-and-choose techniques when multiple circuits need to be evaluated on the same (consistent) inputs and was treated in several publications [22, 20, 29]. Another work by Kolesnikov et al. [17] proposed a solution for input consistency across multiple user interactions with the help of a semi-honest server at low cost. That is, possibly malicious users engage in executions of secure two-party function evaluation with each other, but the user's input must remain the same for multiple instances of secure computation.

With respect to enforcing input correctness, one line of research applies game theory to function design to incentivize participants to provide their truthful inputs into the computation (see, e.g., [14, 26, 15, 27, 28] among others). These publications typically treat the parties as rational. For instance, Shoham and Tennenholtz [26] studied the question of which boolean functions can be computed by rational agents in a distributed setting and more recent work of Wallrabenstein and Clifton [27, 28] revisits game specifications using the specifics of secure multi-party computation setups. In addition, input certification was used with certain types of functionalities to guarantee that computation is run on the same inputs as the inputs previously signed by some certification authority. For example, [9, 11] provide solutions for performing private set intersection on certified sets and [8] incorporates certification into anonymous credentials. These techniques, however, are not applicable to general functionalities.

More recently, Blanton and Bayatbabolghani [3] incorporated input certification into general secure function evaluation based on garbled circuits. In particular, the computation takes place between two possibly malicious users who use the help of an untrusted semi-honest server and ensures that the users can only enter inputs that were previously certified into the computation. The construction uses signatures with protocols [6] for this purpose and ties them to the way input is provided into the garbled circuit using zero-knowledge proofs. This paper is used as the starting point for this work. As we mentioned earlier, their mechanism for garbler's input certification applies to regular two-party computation based on garbled circuits, but it is not the case for the circuit garbler and this is why we focus on this task in this work.

Katz et al. [16] proposed a solution to enforce input validity of both garbler and evaluator in secure function evaluation based on garbled circuits. It formulates input validation in the form of two predicates $f_1(\cdot)$ and $f_2(\cdot)$ applied to the input of the first and second participant, respectively. The construction then ensures that the main function $f(\cdot,\cdot)$ is evaluated on the the same inputs as those provided during input validation. The underlying techniques include ElGamal-based commitments and a special form of oblivious transfer (OT). Note that the predicates f_1 and f_2 are specified in the form of Boolean circuits and are evaluated using garbled circuits themselves, which means that they could be used to privately verify a private signature on a private input, but the resulting circuit is large. Concurrently with [16], Baum [1] treats a similar problem, but the solution is based on universal hash functions and committed OT. In particular, the predicates $f_1(\cdot)$ and $f_2(\cdot)$ are evaluated by the respective party locally, but the join computation includes evaluation of a hash function on the output of the predicate to verify consistency of the input into the computation. More generally, the techniques of [1] improve performance of secure function evaluation based on garbled circuits when portions of the computation (or sub-circuits) depend only on one party's input.

We note that it is possible to combine the techniques of [16, 1] with input certification using signatures with protocols. For example, instead of evaluating predicate f_1 on private input x of the first party P_1, P_1 could prove in zero-knowledge that it possesses a signature on input x and connect it to the evaluation of $f(x, y)$, where y is the input of the second party P_2, in the same way as in [16]. This will serve as an alternative construction to the techniques presented in this paper. Suppose that P_1 is the circuit garbler G and we only consider G's input certification, similar to the construction presented in this work. Then if we take the approach of [16] and combine it with signatures with protocols (such as [6, 7]), its performance compared to our approach can be evaluated as follows. Besides the common components such as garbling and checking/evaluation of $O(\rho)$ circuits and executing OT for circuit evaluator's input, the approach based on the techniques of [16] requires $O((n_1 + n_3)\rho)$ public-key operations (modulo exponentiations), where n_1 is the bitlength of G's input and n_3 is the bitlength of the output. In our construction, however, only one signature is verified and the parties perform $O(n_1\rho)$ symmetric key operations (PRF or hash function evaluations) with small constants hidden behind the big-O notation.

If instead of using signatures with protocols we directly compare the approach of [16] to our work, then the number of public key operation in the solution of [16] is still $O((n_1 + n_3)\rho)$ and there is a need to securely verify a private signature on private input using a garbled circuit. This circuit is expected to be large; for example, based on the circuit sizes reported in [18] we can estimate that verification of an RSA signature using a 1024-bit modulus uses over 40 million gates, and the number is obviously several times higher for stronger RSA with a 2048- or 3072-bit modulus (a circuit for producing a signature with RSA-1024 is over 40 billion gates in size).

The techniques of [1] are conceptually similar to those used in [16], but there is no special output handling and public-key operations are used only in the form of commitments and OT. Compared to our construction, the solution of [1] requires invocation of $O((n_1 + \rho)\rho)$ OTs (no OTs or similar operations are used for the garbler's input in our construction), the size of the circuit that needs to be evaluated in the malicious setting is larger due to the need to enforce input consistency using a hash function evaluation, and the predicate in the form of a signature verification needs to be evaluated once using a garbled circuit.

3 Garbled Circuit Evaluation

The use of garbled circuits allows two parties P_1 and P_2 to securely evaluate a Boolean circuit of their choice. Given an arbitrary function $f(x, y)$ that depends on private inputs x and y of P_1 and P_2, respectively, the parties first represent it as a Boolean circuit. One party acts as a circuit generator (or garbler) G and creates a garbled representation of the circuit by associating both values of each binary wire with random labels. The other party acts as a circuit evaluator E and evaluates the circuit in its garbled representation without knowing the meaning of the labels that it handles during the evaluation. The output labels can be mapped to their meaning and revealed to either or both parties.

Once the circuit is garbled, the garbler communicates it (in the form of garbled Boolean gates) to the evaluator together with the labels of the input wires corresponding to the garbler's input bits. The labels of the input wires corresponding to the evaluator's input bits are communicated to the evaluator by means of 1-out-of-2 Oblivious Transfer (OT).

The standard construction used for the semi-honest setting also provides security in the presence of a malicious evaluator (i.e., if the evaluator deviates from the prescribed computation, the evaluation might abort, but the evaluator will not be able to learn unauthorized information). However, to guarantee security in the malicious setting (with either malicious garbler or evaluator) additional techniques need to be employed. A widely used approach to detecting incorrectly garbled circuits is based on cut-and-choose. Given a security parameter ρ, G garbles $O(\rho)$ circuits and sends them to E. The evaluator chooses a number of them to be opened and checks them for correctness. The remaining circuits are evaluated by E, with the algorithm specifying how possible differences in the circuit outputs are to be reconciled (e.g., by using the majority) without disclosing that information to G. This approach allows for the probability of E accepting the output of incorrect circuit evaluation to be at most negligible in ρ.

There are, however, additional attacks that can be mounted at the time of input specification in the attempt to learn additional information about the other party's input and the corresponding countermeasures. First, the utilized approach must enforce that the OT protocol is resilient to malicious behavior. Second, it must enforce that both G and E input consistent bits into all evaluation circuits. This is typically easy to achieve for the evaluator (i.e., the evaluator specifies a single input bit and learns the corresponding labels for all circuits being evaluated), while additional techniques need to be used to enforce G's input consistency. Third, the protocol must be resilient to a selective failure attack, in which G attempts to learn a bit of E's input by providing incorrect information during OT. For example, G could enter an incorrect wire label corresponding to 0 for E's input bit into OT. Then if E's input is 0, it is unable to proceed and aborts and otherwise it succeeds, allowing G to learn one bit of E's input. A solution to this problem is to use ρ input bits into the circuit for each bit of E's input, where all ρ bits are XORed together to result in E's original input bit. Now if G launches a selective-failure attack on a single bit, the leaked bit reveals no information about E's original input. This attack is only successful and results in revealing a bit of E's input when G recovers all ρ corresponding bits of the input, which only has a negligible (in security parameter ρ) probability of success.

Lastly, we need to ensure that authentic output is delivered to both parties after the evaluation. (Note that in general, fairness cannot be achieved in two-party computation, preventing one of the parties from learning the output, but at least we can require that only authentic outputs are accepted.) Various mechanisms for achieving this goal exist and for the purposes of this work any such mechanism will suffice. However, for concreteness of our security analysis, we will assume the following: the meaning of the output wire labels is opened to E. If G is also to learn the (same or different) output, the function is modified as in [20] to compute G's output in a protected form together with the corresponding message authentication tag, which G can verify and recover its output if verification was successful.

We now proceed with defining security of a two-party secure computation protocol in the presence of malicious participants using the standard real-ideal model setting and simulation paradigm. The execution of protocol Π takes place between parties P_1 and P_2 and an adversary $\mathcal{A}$ who can corrupt one of them. Each participant receives its input and security parameters 1^κ and 1^ρ, where κ denotes the computation security parameter and ρ the statistical security parameter. $\mathcal{A}$ receives all information that the party it corrupted has and can also instruct the corresponding corrupted party to behave in a certain way. Let $\mathrm{VIEW}_{\Pi,\mathcal{A}}$ denote a tuple consisting of the view of $\mathcal{A}$ at the end of an execution of Π and the output of the honest party.

In the ideal model, all parties interact with a trusted party TP who evaluates function f. The execution begins with each party receiving its inputs and security parameters. Each honest party sends to TP its true input and a malicious party can send an arbitrary value. If TP does not receive input from both parties or if it receives an abort message, it outputs $\perp$ (empty) to the participants. Otherwise, the participants receive f evaluated on submitted inputs. Let $\mathrm{VIEW}_{f,S}$ denote a tuple consisting of the output that simulator S with access to the corrupted party's information produces based on its view and the output of the honest party after ideal execution of function f. We obtain the following security definition:

Definition 1. *Let Π be a protocol that computes function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ with party P_1 contributing input x and party P_2 contributing input y. We say that Π securely evaluates f if for each probabilistic polynomial-time in κ adversary $\mathcal{A}$ in the real model and all $x, y \in \{0,1\}^*$, there exists* *probabilistic S in the ideal model that run in time polynomial in $\mathcal{A}$'s runtime and* $\mathrm{IDEAL}_{f,S}(x,y) \cong \mathrm{REAL}_{\Pi,A}(x,y)$*. Here $\cong$ denotes computational indistinguishability (in the security parameter κ).*

When inputs are certified, the participants have access to certification in the real protocol execution, while in the ideal model the simulator will need to simulate input certification of honest parties (without access to their inputs). Furthermore, if we want to guarantee that the computation takes place on the input equal to the input encoded in the certification, we need to show that a malicious participant is unable to provide an input that differs from the certified input and results in successful completed protocol with more than a negligible probability.

4 Proposed Construction

In our construction we use three independent hash functions h_1, h_2, and h_3 with properties defined below. For the purposes of our security analysis, we need to assume that h_1 and h_2 are universal hash functions, while collision resistance is sufficient for h_3. In addition, we require h_1 to support homomorphic XOR as in $h_1(x \oplus y) = h_1(x) \oplus h_1(y)$. A collection of hash functions $\mathcal{H} = \{h : A \to B\}$ is called universal if for any distinct $x, y \in A$, the probability that a uniformly chosen $h \in \mathcal{H}$ satisfies that $h(x) = h(y)$ is at most $1/|B|$, i.e., its output is uniformly distributed over the function's range. An efficient implementation of a universal hash function with homomorphic XOR can be found in [25]. Collision resilience of a hash function h is defined as inability of an adversary to find two distinct x and y such that $h(x) = h(y)$ with more than a negligible probability in the output size of h. For the purpose of this work, we let the output size of hash functions be governed by the security parameter κ so that the probability of finding collisions is negligible in κ. Collision resilience follows for our universal hash functions h_1 and h_2.

Our construction also relies on a pseudo-random function (PRF) $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, security of which is defined in a standard way. That is, for a randomly chosen key $k \in \{0,1\}^\kappa$, an adversary who can query F_k on different inputs a polynomial number of times is unable to distinguish F_k from a random function f with more than a negligible probability.

The intuition behind our construction is to let the certification authority (CA) associate input bits with random values, which are to be used in circuit generation and evaluation, and generate some "fingerprint" information for enforcing correctness. In this scheme, the CA does not need to perform signing of input bits as in traditional signature schemes, but only performs symmetric key operations and computes one signature independent of the input or circuit size. The evaluator consequently will use the "fingerprint" information to check circuits, and both the garbler and evaluator use information provided by the CA to generate input labels.

In more detail, for each input wire i corresponding to the garbler's input, the CA chooses two random strings s_i^0 and s_i^1 that mean 0 and 1, respectively, and similarly two (pseudo-)random values for each input wire i of each garbled circuit j (denoted by t_{2nj+2i} and $t_{2nj+2i+1}$ in the protocol description where n is the input size). The CA also encodes information that binds all of these random values by their meaning (i.e., all s and t values that correspond to 0 across all input wires and the equivalent values that correspond to 1) and by input wire (i.e., both t values that correspond to a given input wire in a given circuit). These bindings are stored in an encrypted form and are opened to the evaluator for circuits that are to be checked. The CA also releases information related to G's input x as $s_i^{x_i}, s_i^{1-x_i}$ (without revealing the meaning of these strings). These values are used both during circuit checking (for opened circuits to verify their correctness) and circuit evaluation (to compute labels corresponding to garbled input wires using $s_i^{x_i}$'s). In the following, we describe the proposed protocol in detail.

In what follows, x is the input to be signed and sk is the private signing key of the certification authority S. We assume that ρ circuits need to be garbled, where a fraction of them is being checked and the remaining circuits are evaluated.

Input certification. The input consists of user's input $x = x_0 \ldots x_{n-1}$ to be certified and the signer S holds its private signing key sk (with the corresponding public key pk available to all users).

1. For each input bit $i = 0, \ldots, n-1$, S chooses two random strings $s_i^0, s_i^1 \leftarrow \{0,1\}^\kappa$ representing bits 0 and 1, respectively.
2. S computes $H^0 = h_1(s_0^0 \oplus \cdots \oplus s_{n-1}^0)$ and $H^1 = h_1(s_0^1 \oplus \cdots \oplus s_{n-1}^1)$.
3. S chooses a secret key $k' \leftarrow \{0,1\}^\kappa$ for a PRF F and for $i = 0, \ldots, 2n\rho - 1$ computes a pseudo-random string $t_i = F_{k'}(i)$.
4. For $j = 0, \ldots, \rho - 1$, S computes $P_j^0 = H^0 \oplus h_1(h_2(t_{2nj}) \oplus h_2(t_{2nj+2}) \oplus \cdots \oplus h_2(t_{2nj+2n-2}))$, and $P_j^1 = H^1 \oplus h_1(h_2(t_{2nj+1}) \oplus h_2(t_{2nj+3}) \oplus \cdots \oplus h_2(t_{2nj+2n-1}))$.
5. For each $j = 0, \ldots, \rho - 1$, S computes $V_{0,j} = h_3(h_1(h_2(t_{2nj}) \oplus h_2(t_{2nj+1})))$, $V_{1,j} = h_3(V_{0,j} || h_1(h_2(t_{2nj+2}) \oplus h_2(t_{2nj+3}))$, $\ldots$, and $V_{n-1,j} = h_3(V_{n-2,j} || h_1(h_2(t_{2nj+2n-2}) \oplus h_2(t_{2nj+2n-1})))$ and sets $Q_j = V_{n-1,j}$.
6. For each $j = 0, \ldots, \rho - 1$, S chooses a random encryption key $ck_j \leftarrow \{0,1\}^\kappa$ and computes $\mathsf{Enc}_{ck_j}(P_j^0 || P_j^1 || Q_j)$.
7. S concatenates $(s_i^{x_i} || s_i^{1-x_i})_{i=0}^{n-1}$, $(\mathsf{Enc}_{ck_j}(P_j^0 || P_j^1 || Q_j))_{j=0}^{\rho-1}$, and stores the resulting string as c. Here each $s_i^{x_i}$ represents input bit x_i and $s_i^{1-x_i}$ its complement. S signs c as $\mathsf{Sign}_{sk}(c)$.
8. S returns $\langle c, \mathsf{Sign}_{sk}(c), (ck_j)_{j=0}^{\rho-1}, k' \rangle$ to the user. Note that information included in c does not reveal whether x_i was 0 or 1.

The above information will be used for producing n garbled label pairs for input wires corresponding to input x into ρ circuits. As described below, each garbled pair for input wire i in circuit j will be formed as $\ell_{i,j}^0 = h_1(s_i^0 \oplus h_2(t_{2n_1j+2i}))$ and $\ell_{i,j}^1 = h_1(s_i^1 \oplus h_2(t_{2n_1j+2i+1}))$. Then the values P_j^0, P_j^1, and Q_j are used to verify correctness and consistency of the garbled circuit j. For each garbled circuit j, these values are opened during the circuit checking phase and are used by E to verify correctness of input labels. The intuition behind the checks is that P_j^0 encodes information about all inputs labels corresponding to input bits equal to 0, while P_j^1 encodes information about all labels corresponding to input bits equal to 1. We refer to checks that use P_j^0 and P_j^1 as "horizontal checks." Additionally, information encoded in Q_j encodes information about each pair of labels $\ell_{i,j}^0, \ell_{i,j}^1$ for each input wire i of the circuit. We refer to this check as the "vertical check."

Secure function evaluation with certified inputs. Garbler G holds private input $x = x_0 \ldots x_{n_1-1}$, $(ck_j)_{j=0}^{\rho-1}$, k' and supplies public $\langle c, \mathsf{Sign}_{sk}(c)\rangle$, where $c = (s_i^{x_i}||s_i^{1-x_i})_{i=0}^{n_1-1}||(\mathsf{Enc}_{ck_j}(P_j^0||P_j^1||Q_j))_{j=0}^{\rho-1}$. Evaluator E holds private input $y = y_0 \cdots y_{n_2-1}$.

0. Initial check: E verifies signature on c using certification authority's public key pk and aborts if verification fails. E stores all components of c.
1. Circuit garbling:
   (a) For each circuit $j = 0, \ldots, \rho$, G generates labels for its own input wires $i = 0, \ldots, n_1 - 1$ as $\ell_{i,j}^0 = h_1(s_i^0 \oplus h_2(t_{2n_1 j+2i}))$ and $\ell_{i,j}^1 = h_1(s_i^1 \oplus h_2(t_{2n_1 j+2i+1}))$, where each t_i is computed as $F_{k'}(i)$.
   (b) G generates the rest of the circuits in the same way as traditional garbled circuits and sends all circuits to E.
2. Circuit checking:
   (a) E select a predefined number of circuits out of ρ of them at random as checking circuits and asks G to open the selected circuits.
   (b) G reveals each label pair for each wire of each checking circuit and E verified them. If any circuit is malformed, E aborts the computation.
   (c) For each checking circuit with its original index j, G sends to E the key ck_j. E decrypts P_j^0, P_j^1, and Q_j and checks whether P_j^0 is equal to $\ell_{0,j}^0 \oplus \cdots \oplus \ell_{n_1-1,j}^0$ and whether P_j^1 is equal to $\ell_{0,j}^1 \oplus \cdots \oplus \ell_{n_1-1,j}^1$. If any check fails, E aborts the computation.
   (d) For each checking circuit with its original index j, E computes $S_{0,j} = h_3(\ell_{0,j}^0 \oplus \ell_{0,j}^1 \oplus h_1(s_0^{x_0}) \oplus h_1(s_0^{1-x_0}))$, $S_{1,j} = h_3(S_{0,j}||\ell_{1,j}^0 \oplus \ell_{1,j}^1 \oplus h_1(s_1^{x_1}) \oplus h_1(s_1^{1-x_1}))$, $\ldots$, $S_{n_1-1,j} = h_3(S_{n_1-1,j}||\ell_{n_1-1,j}^0 \oplus \ell_{n_1-1,j}^1 \oplus h_1(s_{n_1-1}^{x_{n_1-1}}) \oplus h_1(s_{n_1-1}^{1-x_{n_1-1}}))$. If $S_{n_1-1,j}$ is not equal to Q_j, E aborts.
3. Circuit evaluation:
   (a) The circuits that have not been opened in step 2 are evaluation circuits. For each evaluation circuit with the original index j, G sends to E values $t_{2n_1 j+x_0}, \ldots, t_{2n_1 j+2n_1-2+x_{n_1-1}}$.
   (b) For each evaluation circuit with its original index j, E computes labels for G's input wires $i = 0, \ldots, n_1 - 1$ as $\ell_{i,j}^{x_i} = h_1(s_i^{x_i} \oplus h_2(t_{2n_1 j+2i+x_i}))$.
   (c) G and E engage in OT for E to learn garbled labels corresponding to its input and E evaluates each evaluation circuit and determines the output in the same was as in traditional garbled circuits.

We can see that all circuits garbled by an honest G pass the checks of steps 2(c-d). In particular, for each circuit j, each label $\ell_{i,j}^0$ is constructed as $\ell_{i,j}^0 = h_1(s_i^0 \oplus$

$h_2(t_{2n_1j+2i})$), so that $\bigoplus_{i=0}^{n_1-1} \ell_{i,j}^0 = h_1\left(\left(\bigoplus_{i=0}^{n_1-1} s_i^0\right) \oplus \left(\bigoplus_{i=0}^{n_1-1} h_2(t_{2n_1j+2i})\right)\right)$. This is exactly how the value of $P_j^0 = h_1\left(\bigoplus_{i=0}^{n_1-1} s_i^0\right) \oplus h_1\left(\bigoplus_{i=0}^{n_1-1} h_2(t_{2n_1j+2i})\right) = h_1\left(\left(\bigoplus_{i=0}^{n_1-1} s_i^0\right) \oplus \left(\bigoplus_{i=0}^{n_1-1} h_2(t_{2n_1j+2i})\right)\right)$ was constructed by the certification authority. The same applies to checking whether P_j^1 is equal to $\bigoplus_{i=0}^{n_1-1} \ell_{i,j}^1$. As far as the check in step 2(d) does, we see that it will verify if the value of $S_{i,j}$ computed by E for each i is identical to the value $V_{i,j}$ computed by the certification authority in step 5 of input certification. Note that $S_{0,j} = h_3(\ell_{0,j}^0 \oplus \ell_{0,j}^1 \oplus h_1(s_0^{x_0}) \oplus h_1(s_0^{1-x_0})) = h_3(h_1(s_0^0 \oplus h_2(t_{2n_1j})) \oplus h_1(s_0^1 \oplus h_2(t_{2n_1j+1})) \oplus h_1(s_0^{x_0}) \oplus h_1(s_0^{1-x_0})) = h_3(h_1(h_2(t_{2n_1j}) \oplus h_2(t_{2n_1j+1}))) = V_{0,j}$. Similar derivations will apply to other values of i as well, giving us correctness.

5 Security Analysis

We first demonstrate security according to the simulation paradigm in Definition 1 and then proceed with showing that the proposed construction enforces input correctness. We use notation negl to denote a function negligible in its input.

To prove that no information leakage takes place during protocol execution, we rely on the following result:

Lemma 1. *Information $(s_i^{x_i}, s_i^{1-x_i})_{i=0}^{n-1}, P_j^0, P_j^1, Q_j$ computed as part of certification for n-bit x for some fixed j together with pairs $(\ell_{i,j}^0 = h_1(s_i^0 \oplus h_2(t_{2nj+2i}))$, $\ell_{i,j}^1 = h_1(s_i^0 \oplus h_2(t_{2nj+2i+1})))_{i=0}^{n-1}$ is indistinguishable for any PPT adversary from the same information computed for any n-bit $x' \neq x$.*

Proof. We show that the above values reveal no information about x, from which the claim of the lemma will follow. First, note that the values $s_i^{x_i}, s_i^{1-x_i}$ were chosen uniformly at random and in the absence of other information about x_i, s_i^0's or s_1^1, the ordering reveals no information about x. In other words, the pairs $s_i^{x_i}, s_i^{1-x_i}$ are distributed identically to uniformly chosen random values in the absence of additional information. In what follows, we construct a number of hybrid views and demonstrate that the views are indistinguishable from each other.

Hybrid 0: The same as the original set of values consisting of $s_i^{x_i}$'s, $s_i^{1-x_i}$'s, P_j^0, P_j^1, Q_j, $\ell_{i,j}^0$'s, and $\ell_{i,j}^1$'s for a fixed j and $i = 0, \ldots, n-1$.

Hybrid 1: In this view, we replace each $h_2(t_{2nj+2i})$ and $h_2(t_{2nj+2i+1})$ used in the creation of P_j^0, P_j^1, Q_j, and each $\ell_{i,j}^0$ and $\ell_{i,j}^1$ with strings $r_{i,j,0}$ and $r_{i,j,1}$, respectively, chosen uniformly at random over h_2' range. Because we are modifying all values constituently, any relationships between them (such as $P_j^0 = \bigoplus_{i=0}^{n-1} \ell_{i,j}^0$, etc.) will still hold. These values are indistinguishable from the values in Hybrid 0 because h_2 is a universal hash function. In particular, inputs t_{2nj+2i} and $t_{2nj+2i+1}$ into the hash function are pseudo-random and satisfy the min-entropy requirements for the output of h_2 to be treated as pseudo-random [25].

Hybrid 2: In this view, we replace $s_i^0 \oplus r_{i,j,0}$ and $s_i^1 \oplus r_{i,j,1}$ with uniformly chosen random strings $r'_{i,j,0}$ and $r'_{i,j,1}$, respectively, of the same length (or, equivalently, we replace $h_1(s_i^b \oplus r_{i,j,b}) = h_1(s_i^b) \oplus h_1(r_{i,j,b})$ with $h_1(r'_{i,j,b})$ for $b = \{0, 1\}$) in the creation of P_j^0, P_j^1, and each $\ell_{i,j}^0$ and $\ell_{i,j}^1$. This view is distributed identically to Hybrid 1 because XOR of a uniformly chosen string with any value produces a uniformly chosen string.

Now we arrive at a view where all s_i^0's and s_i^1's have been completely eliminated. The remaining information is the pairs $s_i^{x_i}, s_i^{1-x_i}$ and values derived from random $r_{i,j,b}$ and $r'_{i,j,b}$. It is clear that no information about x is revealed to a computationally bounded adversary and the claim of this lemma follows. □

Now we are ready to proceed with showing that the construction complies with the security requirements for secure two-party function evaluation in the presence of malicious adversaries. In our result and its proof below, we assume a secure realization of the OT protocol secure in the presence of malicious parties is available to the participants, which they can call as a sub-protocol. Similarly, we rely on a secure realization of garbled circuit evaluation (in the presence of a semi-honest garbler).

Theorem 1. *Given an OT protocol secure against malicious participants and a circuit garbling and evaluation scheme secure against a semi-honest garbler and malicious evaluator, the proposed construction for secure function evaluation with certified inputs is secure according to Definition 1.*

Proof. First, suppose G is malicious and we denote the corresponding adversary as A_G. In the ideal world, simulator S_G resides between A_G and the trusted party and simulates G's view during the protocol execution as follows:

1. S_G invokes A_G on its input.
2. At the initial check step, S_G acts as honest E and verifies the signature on c. If verification fails, S_G aborts the execution.
3. S_G continues to act as an honest E would in steps 1 and 2, i.e., it receives the circuits, asks a predefined fraction of them at randomly chosen indices to be opened, and checks the opened circuits. If any of the checks fail, S_G aborts the execution.
4. In step 3(a), S_G simulates the OT with A_G. If any malicious behavior is detected, S_G aborts the execution and otherwise receives $t_{2n_1j+x_0}, \ldots, t_{2n_1j+2n_1-2+x_{n_1-1}}$ from A_G for each evaluation circuit j.
5. S_G obtains $f(x, y)$ from TP and communicates G's output according to the protocol to A_G.

We now need to analyze the differences in the real and simulated views. The first difference comes from the fact that S_G simulates the OT in step 4 as it does not possess E's real input. A_G, however, has a negligible chance in observing any differences because of the security of the underlying OT protocol (i.e., its simulation is indistinguishable from real execution). The second difference comes from the fact that in real execution E might fail to output the correct output because the circuits that it evaluated were incorrect. This, however, can happen only with probability $\mathsf{negl}(\rho)$. We therefore obtain that A_G is unable to distinguish the views with more than a negligible probability and they are indistinguishable.

Now suppose E is malicious. In the ideal world, simulator S_E resides between E and TP and simulates E's view during the protocol execution as follows:

1. S_E obtains certification $\langle c, \mathsf{Sign}_{sk}(c), (ck_j)_{j=0}^{\rho-1}, k' \rangle$ from the certification authority for some input $x' = x'_0 \ldots x'_{n_1-1}$ of its choice.
2. S_E invokes A_E on its input.
3. S_E extracts A_E's choices j for which circuits are to be opened during circuit checking and constructs those circuits correctly as an honest G would using information from c and the corresponding strings t_i's derived using k'.
4. To construct evaluation circuits, S_E creates G's input labels as an honest G would using certification information obtained for input x'. S_E then receives $f(x, y)$ from TP and construct the remaining portion of the garbled circuits so that they always output $f(x, y)$ (i.e., the circuits are input-insensitive).
5. S_E sends all circuits to A_E and opens the checking circuits upon A_E's request as an honest G would.
6. During circuit evaluation, S_E executes the OT with A_E as an honest G would. If any malicious behavior is detected, S_E aborts the execution and otherwise it sends $t_{2n_1j+x'_0}, \ldots, t_{2n_1j+2n_1-2+x'_{n_1-1}}$ generated using k' to A_E for each evaluation circuit j.
7. S_E continues as an honest G would.

With this simulation, we have two sources of potential differences between the real and simulated views. The first comes from modifying the way the evaluation circuits are constructed in step 4 to output a fixed value. This change has been shown in [20] to be indistinguishable to an adversary who has only a single set of labels corresponding to the inputs. That is, evaluation of modified circuits on arbitrary inputs is indistinguishable from evaluation of correct circuits on inputs x and y.

The second difference is that the certification corresponds to a randomly chosen x' instead of actual G's input x, which we need to analyze in more detail. In particular, c that A_E observes includes the pair $s_i^{x'_i}, s_i^{1-x'_i}$ for each input bit x'_i of x'. Because the certification authority chooses these strings uniformly at random for any possible input, the pairs $s_i^{x_i}, s_i^{1-x_i}$ and $s_i^{x'_i}, s_i^{1-x'_i}$ are distributed identically in the real and simulated views. For each opened (checking) circuit j, A_E also learns the values of P_j^0, P_j^1, Q_j, as well as G's input wire label pairs $\ell_{i,j}^0, \ell_{i,j}^1$ for each input wire i. From Lemma 1 we learn that these values are indistinguishable for inputs x and x' as well.

Lastly, for each evaluation circuit j, A_E only sees values $s_i^{x'_i}$, $s_i^{1-x'_i}$, $t_{2n_1j+2i+x'_i}$, $\mathsf{Enc}_{ck_j}(P_j^0||P_j^1||Q_j)$. Assuming CPA-security of the underlying encryption scheme, there is only a negligible chance that ciphertexts $\mathsf{Enc}_{ck_j}(P_j^0||P_j^1||Q_j)$ reveal any information to A_E. Lastly, in the absence of other relevant information, triples

$s_i^{x'_i}, s_i^{1-x'_i}, t_{2n_1j+2i+x'_i}$ and $s_i^{x_i}, s_i^{1-x_i}$, $t_{2n_1j+2i+x_i}$ are identically distributed, giving A_E no information about the input bit to which the triple corresponds. Thus, the views in ideal and real executions are indistinguishable as well.

We obtain that the real and simulated views are indistinguishable when either G or E is corrupt, which concludes the proof. □

Recall that we rely on current countermeasures for defeating G's incorrect behavior such as cut-and-choose that instructs garbling of $O(\rho)$ circuits, checking a fraction of them, and evaluating the rest. Then according to the previously showed results, we obtain that any property verified at the circuit checking time will hold in the computed result with probability $1 - \mathsf{negl}(\rho)$. Our input correctness result for the garbler is as follows:

Theorem 2. *If all checking circuits pass all tests at the checking phase, the function is evaluated using garbler's inputs certified by the certification authority with probability* $1 - \mathsf{negl}(\kappa) - \mathsf{negl}(\rho)$.

In order to show this, we first prove a supplementary result.

Lemma 2. *If the checks of steps 2(c) and 2(d) hold for a garbled circuit and E can successfully evaluate the circuit (i.e., evaluation does not abort), the circuit used correct label pairs* (ℓ_i^0, ℓ_i^1) *generated by the certification authority for each of G's input wire* i *with probability* $1 - \mathsf{negl}(\kappa)$.

Proof. For simplicity of presentation, in this proof we omit notation j corresponding to the jth garbled circuit. Recall that we refer to the tests of step 2(c) as the "horizontal check," i.e., a consistency check performed over all inputs with the same value (such as zero bits and one bits) across all input bits, and the tests of step 2(d) as the "vertical check," i.e., a consistency check performed over both inputs of each input bit.

Recall that E evaluates a garbled circuit on labels $\ell_i^{x_i}$ computed as $h_1(s_i^{x_i} \oplus h_2(t_{2i+x_i}))$, where each $s_i^{x_i}$ comes from the certification authority. Thus, the goal of a corrupt G is to create a garbled circuit in such a way that E will compute a valid label for bit $1 - x_i$ using $s_i^{x_i}$ supplied by the certification authority. This could involve providing E with an incorrect value of t_{2i+x_i} or simply using $\ell_i^{x_i}$ to represent bit $1 - x_i$ for the ith input wire in the circuit. As we show below, the probability that the adversary is successful in carrying out either of this attacks is negligible in the security parameter κ.

Suppose that malicious G selects one specific input wire i for which it wants to corrupt labels. In what follows, we analyze two cases: (1) the adversary supplies the correct t_{2i+x_i} value to E, sets $\hat{\ell}_i^{1-x_i} = h_1(s_i^{x_i} \oplus h_2(t_{2i+x_i}))$ in the circuit, and adjusts other values accordingly and (2) the adversary supplies an incorrect $\hat{t}_{2i+x_i}$ to E, sets $\hat{\ell}_i^{1-x_i} = h_1(s_i^{x_i} \oplus h_2(\hat{t}_{2i+x_i}))$ in the circuit, and adjusts other values as needed. Recall that it is given that the circuit passes the checks of steps 2(c) and 2(d) and that the circuit could be successfully evaluated (i.e., the gates are correctly formed and use the labels that E computes). In what follows, we refer to the adversary and G interchangeably.

1. In this case, G communicates t_{2i+x_i} to E who will compute the value $h_1(s_i^{x_i} \oplus h_2(t_{2i+x_i}))$ at circuit evaluation time, while G's intent is to set $\hat{\ell}_i^{1-x_i} = \ell_i^{x_i} = h_1(s_i^{x_i} \oplus h_2(t_{2i+x_i}))$ instead of the original $\ell_i^{1-x_i}$ during circuit garbling. Then if G sets $\hat{\ell}_i^{1-x_i} = \ell_i^{x_i}$, it will need to set $\hat{\ell}_i^{x_i} = \ell_i^{1-x_i}$ to be able to pass the vertical check (or break collision resistance of h_3 which only has a negligible probability of success). Now because the labels for both input bits of the ith input wire are modified from their expected values, the adversary needs to use other input labels to compensate for the difference to be able to pass the horizontal check. In particular, P^0 and P^1 values computed in step 2(c) of the protocol will be different by $\ell_i^{x_i} \oplus \ell_i^{1-x_i}$ from its expected value and G might use labels from $k \geq 1$ other input wires to compensate for the difference.

   Now note that because h_1 is a universal hash function and its output is distributed uniformly over the entire space, $\ell_i^{x_i} \oplus \ell_i^{1-x_i}$ will also be uniformly distributed over the output space. Thus, if the adversary attempts to swap the input labels for any number of other input wires, it has only a negligible chance of matching the difference exactly. That is, for every new combination of input wires with swapped labels there is only a negligible probability of eliminating the difference and the adversary is limited to trying a polynomial number of such combinations.

   If the adversary employs some strategy other than swapping input labels to compensate for the difference, it will be equivalent to solving case 2 below.
2. In this case, G sets $\hat{\ell}_i^{1-x_i} = h_1(s_i^{x_i} \oplus h_2(\hat{t}_{2i+x_i}))$ for some $\hat{t}_{2i+x_i} \neq t_{2i+x_i}$ instead of the original $\ell_i^{1-x_i}$ during circuit garbling. We can further subdivide this into two cases:
   (a) The adversary was able to produce $\hat{\ell}_i^{1-x_i} = h_1(s_i^{x_i} \oplus h_2(\hat{t}_{2i+x_i}))$ to have the same value as the expected label $\ell_i^{1-x_i} = h_1(s_i^{1-x_i} \oplus h_2(t_{2i+x_i}))$. Then if $s_i^{x_i} \oplus h_2(\hat{t}_{2i+x_i}) = s_i^{1-x_i} \oplus h_2(t_{2i+x_i})$ and consequently $h_2(\hat{t}_{2i+x_i}) = s_i^{x_i} \oplus s_i^{1-x_i} \oplus h_2(t_{2i+x_i})$, the adversary has to break the one-way property of hash function h_2 to succeed in determining $\hat{t}_{2i+x_i}$, which it has to provide to E. This has only a negligible probability of success. Otherwise, $s_i^{x_i} \oplus h_2(\hat{t}_{2i+x_i}) \neq s_i^{1-x_i} \oplus h_2(t_{2i+x_i})$ and the adversary has to break the collision resistance property of h_2 to succeed, which also has only a negligible probability of success.
   (b) The adversary produces label $\hat{\ell}_i^{1-x_i} = h_1(s_i^{x_i} \oplus h_2(\hat{t}_{2i+x_i}))$ that differs from the expected label $\ell_i^{1-x_i}$. Then in order to pass the vertical check, the adversary will need to compute $\hat{\ell}_i^{x_i}$ such that $\hat{\ell}_i^{x_i} \oplus \hat{\ell}_i^{1-x_i} = \ell_i^{x_i} \oplus \ell_i^{1-x_i}$ or break the collision resilience property of hash function h_3. The probability of success in the latter case is negligible in the security parameter, while the former case can be analyzed as follows.

      Suppose G computes $\hat{\ell}_i^{x_i} = \hat{\ell}_i^{1-x_i} \oplus \ell_i^{x_i} \oplus \ell_i^{1-x_i} \neq \ell_i^{x_i}$. Then the use of $\hat{\ell}_i^{x_i}$ will fail the horizontal check if other labels are not modified. Consequently, the adversary might attempt to modify other labels to pass the check. Suppose the adversary chooses other $k \geq 1$ input wires $i_1, \ldots, i_k$ for which it will modify labels $\ell_{i_j}^{x_i}$ in the attempt to pass the horizontal

check. That is, G is to compute $\hat{\ell}_{i_1}^{x_i}, \ldots, \hat{\ell}_{i_k}^{x_i}$, where $\bigoplus_{j=1}^{k} \hat{\ell}_{i_j}^{x_i}$ equals a specific value. First, suppose that $k = 1$. This means that G is to compute $\hat{\ell}_{i_1}^{x_i}$ that simultaneously satisfies $\hat{\ell}_{i}^{x_i} \oplus \hat{\ell}_{i_1}^{x_i} = t_1$ and $\hat{\ell}_{i_1}^{x_i} \oplus \ell_{i_1}^{1-x_i} = t_2$ for some fixed values t_1 and t_2 to pass the horizontal check using P^{x_i} and the vertical check for input wire i_1. Because $\hat{\ell}_{i}^{x_i}$ and $\ell_{i_1}^{1-x_i}$ are produced using hash function h_1, their values are uniformly distributed over the hash function's output space. This means that $\hat{\ell}_{i_1}^{x_i}$ can meet both of these requirements only with a negligible probability if $\ell_{i_1}^{1-x_i}$ remains unchanged. Thus, G could also change both labels for input wire i_1 (i.e., it sets $\hat{\ell}_{i_1}^{x_1} = t_1 \oplus \hat{\ell}_{i}^{x_i}$ and $\hat{\ell}_{i_1}^{1-x_1} = t_2 \oplus \hat{\ell}_{i_1}^{x_i}$). This, however, will require G to supply a consistent $\hat{t}_{2i_1+x_{i_1}}$ to E (for bit x_{i_1} which is equal to either x_i or $1 - x_i$) to produce $\hat{\ell}_{i_1}^{x_{i_1}}$ using authentic $s_{i_1}^{x_{i_1}}$, which in turn can be done only if G inverts h_2.
Now if we generalize the analysis to $k > 1$, we will still run into the case that there are more contradicting constraints on the values of $\hat{\ell}_{i_j}^{x_i}$ than the number of labels that G needs to set resulting in inability of the adversary to meet all of the constraints simultaneously or G will be required to invert h_2 at least for one input wire i_j.

Thus, we obtain that the adversary can succeed in modifying its input with probability at most negligible in κ in every possible case, which concludes the proof. □

We can now return to proving our main input correctness result.

Proof (Theorem 2). Recall that based on prior results, all properties verified during the circuit checking phase will hold for the computed result with probability $1 - \mathsf{negl}(\rho)$. This means that the conditions of Lemma 2 will also be satisfied with probability $p_1 = 1 - \mathsf{negl}(\rho)$. Furthermore, the result of Theorem 2 guarantees garbler's input correctness with probability $p_2 = 1 - \mathsf{negl}(\kappa)$, and input correctness will hold when both conditions are true. This happens with probability $p_1 \cdot p_2$ giving us probability of input correctness $1 - \mathsf{negl}(\kappa) - \mathsf{negl}(\rho)$ as desired. □

6 Conclusions

In this work, we treat the problem of strengthening the security guarantees of traditional formulations of secure multi-party computation. This is achieved by enforcing input correctness through input certification and tying certification to instances of secure function evaluation. The focus of this work is specifically on enforcing correctness of the garbler's input in secure two-party computation based on garbled circuit evaluation in the malicious model. We put forward a new certification mechanism specifically designed to be used with garbled circuit evaluation and show how to integrate certificates into secure computation to guarantee correctness of garbler's input. Our construction incurs minimal overhead and adds only one signature verification and $O(n\rho)$ symmetric key/hash operations to conventional garbled circuit evaluation using cut-and-choose in the malicious model that consist of garbling ρ circuits and where n is the size of the garbler's input in bits. We formally prove security of our construction.

Acknowledgments

This work was supported in part by grants 1223699 and 1319090 from the National Science Foundation and FA9550-13-1-0066 from the Air Force Office of Scientific Research. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. C. Baum. On garbling schemes with and without privacy. In *International Conference on Security and Cryptography for Networks (SCN)*, pages 468–485, 2016.
2. M. Bellare, V. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE Symposium on Security and Privacy (SP)*, pages 478–492, 2013.
3. M. Blanton and F. Bayatbabolghani. Efficient server-aided secure two-party function evaluation with applications to genomic computation. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 4:1–22, 2016.
4. M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *European Symposium on Research in Computer Security (ESORICS)*, pages 190–209, 2011.
5. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security (ESORICS)*, pages 192–206, 2008.
6. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks (SCN)*, pages 268–289, 2002.
7. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO*, pages 56–72, 2004.
8. J. Camenisch, D. Sommer, and R. Zimmermann. A general certification framework with applications to privacy-enhancing certificate infrastructures. In *IFIP International Information Security Conference (SEC): Security and Privacy in Dynamic Environments*, pages 25–37, 2006.
9. J. Camenisch and G. Zaverucha. Private intersection of certified sets. In *Financial Cryptography and Data Security (FC)*, pages 108–127, 2009.
10. I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO*, pages 643–662, 2012.
11. E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security (FC)*, pages 143–159, 2010.
12. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.

13. S. Goldwasser, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with an honest majority. In *ACM Symposium on the Theory of Computing (STOC)*, pages 218–229, 1987.
14. J. Halpern and V. Teague. Rational secret sharing and multiparty computation. In *ACM Symposium on Theory of Computing (STOC)*, pages 623–632, 2004.
15. M. Kantarcioglu and R. Nix. Incentive compatible distributed data mining. In *IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT)*, 2010.
16. J. Katz, A. J. Malozemoff, and X. Wang. Efficiently enforcing input validity in secure two-party computation. IACR Cryptology ePrint Archive Report 2016/184, 2016.
17. V. Kolesnikov, R. Kumaresan, and A. Shikfa. Efficient verification of input consistency in server-assisted secure function evaluation. In *International Conference on Cryptology and Network Security (CANS)*, pages 201–217, 2012.
18. B. Kreuter, a. shelat, B. Mood, and K. Butler. PCF: A portable circuit format for scalable two-party secure computation. In *USENIX Security Symposium*, 2013.
19. Y. Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *Advances in Cryptology – CRYPTO*, pages 1–17, 2013.
20. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology – EUROCRYPT*, pages 52–78, 2007.
21. Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.
22. P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. In *International Workshop on Public Key Cryptography (PKC)*, pages 458–473, 2006.
23. A. Sadehgi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology (ICISC)*, pages 229–244, 2009.
24. A. Shelat and C. Shen. Two-output secure computation with malicious adversaries. In *Advances in Cryptology – EUROCRYPT*, pages 386–405, 2011.
25. A. Shelat and C. H. Shen. Fast two-party secure computation with minimal assumptions. In *ACM Conference on Computer and Communications Security (CCS)*, pages 523–534, 2013.
26. Y. Shoham and M. Tennenholtz. Non-cooperative computation: Boolean functions with correctness and exclusivity. *Theoretical Computer Science (TCS)*, 343(1):97–113, 2005.
27. J. Wallrabenstein and C. Clifton. Equilibrium concepts for rational multiparty computation. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 226–245, 2013.
28. J. Wallrabenstein and C. Clifton. Realizable rational multiparty cryptographic protocols. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 134–154, 2014.
29. D. Woodruff. Revisiting the efficiency of malicious two-party computation. In *Advances in Cryptology – EUROCRYPT*, pages 79–96, 2007.
30. A. C. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.