58th Annual IEEE Symposium on Foundations of Computer Science

Garbled Protocols and Two-Round MPC from Bilinear Maps

Sanjam Garg
Dept. of Computer Science
University of California, Berkeley
Berkeley, USA
Email: sanjamg@berkeley.edu

Akshayaram Srinivasan
Dept. of Computer Science
University of California, Berkeley
Berkeley, USA
Email: akshayaram@berkeley.edu

***Abstract*—In this paper, we initiate the study of *garbled protocols* — a generalization of Yao's garbled circuits construction to distributed protocols. More specifically, in a garbled protocol construction, each party can independently generate a garbled protocol component along with pairs of input labels. Additionally, it generates an encoding of its input. The evaluation procedure takes as input the set of all garbled protocol components and the labels corresponding to the input encodings of all parties and outputs the entire transcript of the distributed protocol.**

We provide constructions for garbling arbitrary protocols based on standard computational assumptions on bilinear maps (in the common random string model). Next, using garbled protocols we obtain a general compiler that compresses any arbitrary round multiparty secure computation protocol into a two-round UC secure protocol. Previously, two-round multiparty secure computation protocols were only known assuming witness encryption or learning-with errors. Benefiting from our generic approach we also obtain protocols (i) for the setting of random access machines (RAM programs) while keeping communication and computational costs proportional to running times, while (ii) making only a black-box use of the underlying group, eliminating the need for any expensive non-black-box group operations. Our results are obtained by a simple but powerful extension of the non-interactive zero-knowledge proof system of Groth, Ostrovsky and Sahai [Journal of ACM, 2012].

***Keywords*-Circuit Garbling, Bilinear Maps, Universal Composability**

I. Introduction

Yao's garbled circuits [60] (also see [3], [47], [9]) are enormously useful in cryptography. In a nutshell, Yao's construction on input a circuit C generates a garbled circuit $\widetilde{C}$ along with input labels $\{\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}\}$ such that $\widetilde{C}$ and $\{\mathsf{lab}_{i,x_i}\}$ can be used to compute $C(x)$ and nothing more. Over the years, Yao's construction has found numerous applications (to name a few [1], [7], [21], [43], [32]) and several extensions [29], [4], [49] have been investigated. Furthermore, in light of their usefulness, substantial research has been invested to improve the practical efficiency of these constructions [7], [46], [55], [8], [45], [61], [39].

Garbled circuits, while tremendously useful in the two-party setting, when used in the multiparty setting lead to comparatively inferior solutions. For example, Yao's garbled circuits along with a two-round 1-out-of-2 oblivious transfer (OT) protocol [57], [2], [51], [40] gives an easy solution to the problem of (semi-honest) two-round secure computation in the two-party setting. However, the same problem for the multiparty setting turns out to be much harder. Beaver, Micali and Rogaway [7] show that garbled circuits can be used to realize a constant round multi-party computation protocol. However, unlike the two-party case, this protocol is not two rounds.

A. Garbled Protocols

In this paper, we introduce a generalization of Yao's construction from circuits to distributed protocols. We next elaborate on (i) what it means to garble a protocol, (ii) why this notion is interesting, and (iii) if we can realize this notion.

What does it mean to garble a protocol? Consider an arbitrary protocol Φ over n-parties $P_1, \ldots, P_n$ with inputs $x_1, \ldots, x_n$, respectively. Just as in garbled circuits, a garbled protocol construction allows each party P_i to independently generate a garbled protocol component $\widetilde{\Phi}_i$ along with input labels $\{\mathsf{lab}^i_{j,0}, \mathsf{lab}^i_{j,1}\}$. However, now the party P_i additionally generates an input encoding $\widetilde{x}_i$. Correctness requires that the set of all garbled protocol components $\{\widetilde{\Phi}_i\}_{i\in[n]}$ and the set of labels corresponding to the input encodings of all parties $\{\mathsf{lab}^i_{j,z_j}\}_{i\in[n],j\in[|z|]}$ where $z := \widetilde{x}_1 \| \cdots \| \widetilde{x}_n$ can be used to generate the entire transcript of the protocol Φ. Detailing the security guarantee, we require the existence of an efficient simulator Sim such that for any set $H \subseteq [n]$ of honest parties and inputs $\{x_i\}_{i\in[n]}$ of the parties we have that

$$\{\tilde{\Phi}_i, \mathsf{lab}^i_{\widetilde{x}_1\|\ldots\|\widetilde{x}_n}, \widetilde{x}_i\}_{i\in[n]} \overset{c}{\approx} \mathsf{Sim}(H, \Phi(x_1, \ldots x_n), \{x_i\}_{i\notin H})$$

where $\overset{c}{\approx}$ denotes computational indistinguishability and $\Phi(x_1, \ldots, x_n)$ denotes the transcript of Φ.

Why consider Garbled Protocols? We illustrate the power of garbled protocols by showing how they can be used to realize a two-round (semi-honest) multiparty secure computation protocol. Looking ahead, our protocol is analogous to the construction of two-round, two party secure computation protocol using garbled circuits.

Take any n-party secure computation protocol Φ and let $x_1, \ldots, x_n$ be the respective inputs of the parties. Each party

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.60

starts by independently generating $\left\{\widetilde{\Phi}_i, \{\mathsf{lab}^i_{j,0}, \mathsf{lab}^i_{j,1}\}, \widetilde{x}_i\right\}$. In the first round, each party distributes the generated values $\widetilde{x}_i$ to every other party. On receiving the first messages of all other parties, each party sends its second round message $\left(\widetilde{\Phi}_i, \{\mathsf{lab}^i_{j,z_j}\}\right)$ (with $z := \widetilde{x}_1 \| \cdots \| \widetilde{x}_n$) to every other party. Finally, by correctness of garbled protocols we have that each party can locally execute the garbled protocol to obtain the output from the transcript $\Phi(x_1, \ldots x_n)$. On the other hand, the security of the garbled protocols and Φ ensure that nothing else beyond the output is leaked.

Can we garble protocols? Our main result is a garbled protocols construction based on standard computational assumptions on bilinear maps [12], [42]. A bit more precisely:

Informal Theorem. *Assuming the subgroup decision assumption or the decision linear assumption on groups with bilinear maps there exists a garbled protocol construction (in the common reference string model).*

We also show a modification of this construction such that it makes only black-box use of the underlying group and avoids any expensive non-black-box group operations.

B. Applications to Two-Round Multiparty Secure Computation

Using the above primitive, we obtain a general compiler that converts an arbitrary (polynomial) round (semi-honest) multi-party secure computation protocol into a two-round UC secure [16] protocol against static adversaries. Previously, such compilers [22], [34] were known under stronger computational assumptions such as indistinguishability obfuscation [6], [23] or witness encryption [24].[1]

Furthermore, instantiating this compiler with any multiparty secure computation protocol (e.g., the one by Goldreich, Micali, and Wigderson [30]) we obtain the first two-round multiparty computation protocol based on bilinear maps. Prior to this work, constructions of two-round multiparty computation protocols [50], [54], [15] were only known based on lattice assumptions such as the learning-with-errors [58].[2] We also obtain the following extensions:

- *Black-Box Use of the Group:* With the goal of obtaining a two-round multiparty computation protocol that makes black-box use of the underlying cryptographic primitives, we modify our compiler from above. More specifically, building on the non-interactive OT protocol of Bellare and Micali [10] (based on the CDH assumption [20]), we obtain a compiler that converts any arbitrary round (malicious secure) protocol Φ^{OT} in the OT-hybrid model into a two-round UC secure protocol against static adversaries while only making black box use of the underlying group.
  Instantiating, this new compiler with an information theoretic protocol in the OT-hybrid model [44], [41] yields a two-round multiparty computation protocol based on bilinear maps while avoiding expensive non-black-box use of the underlying group.[3]
- *Extension to RAM programs:* Instantiating the above compilers with appropriate multi-party secure computation protocols for RAM programs [53], [33], we also obtain the first two-round multiparty secure RAM computation protocol (and its black-box version) without first converting the RAM program to a circuit based on standard techniques [19], [56].
  We note that the multi-key fully-homomorphic encryption [5], [48], [18], [50], [54], [15] based two-round secure computation techniques do not work for the setting of RAM programs. This is because fully-homomorphic encryption techniques need interaction for disclosing what locations are accessed by the oblivious RAM programs.[4] On the other hand, our use of garbled protocols does not suffer from this limitation.[5]

II. Technical Overview

At the heart of our garbled protocols construction is a simple but powerful extension of homomorphic proof commitments scheme. This primitive was first considered by Groth, Ostrovsky and Sahai [36] who used it to realize a non-interactive zero-knowledge proof system based on bilinear maps. Below we start by (i) recalling GOS construction of homomorphic proof commitments, (ii) how we augment them, and (iii) use them to realize garbled protocols. Finally we give details on how to obtain two round, secure multiparty computation protocol making black-box use of the underlying group.

A. Starting Point: Homomorphic Proof Commitments

A homomorphic proof commitment scheme is a (non-interactive) commitment scheme com that supports homomorphic operations and provides some additional proof properties. In particular, it is additively homomorphic, i.e., $\mathrm{com}(b_0 + b_1; r_0 + r_1) = \mathrm{com}(b_0; r_0) \cdot \mathrm{com}(b_1; r_1)$ where the message space is over $\mathbb{Z}_p$. Furthermore, given a commitment

[1] We note that the recent constructions of lockable obfuscation [35], [59] based on standard assumptions such as learning with errors is insufficient to obtain such a compiler since these works assume that the lock value has some min-entropy.

[2] In two recent works, Boyle et al. [13], [14] also obtain constructions of two-round multiparty computation based on DDH. However, their results are applicable only for the setting of constant number of parties — a special case of our result. Also, they assume the need for public-key infrastructure while we just assume a common reference string.

[3] However, unlike our non black-box protocol, the length of the common reference string of our black-box construction grows linearly with the number of parties.

[4] An oblivious RAM program is a RAM program compiled with an oblivious RAM scheme [52], [31].

[5] Another approach would be to use garbled RAM [49], [28], [27], [26] However, those constructions suffer from the same limitation as Yao's garbled circuits in terms of supporting multiparty protocols. Specifically, garbled RAM can be used to construct two-round two-party secure computation protocol, but the multiparty protocol is only (larger than two) constant rounds [49], [25].

$c = \mathrm{com}(b; r)$, the corresponding committed value b and randomness r, a prover can generate a NIZK proof proving that $b \in \{0, 1\}$ without leaking anything else about the value b.

GOS show that homomorophic proof commitments can be used to generate NIZK proofs for arbitrary NP-statements. This is done in two steps:

1) First, GOS show that given three commitments $c_0 = \mathrm{com}(b_0; r_0), c_1 = \mathrm{com}(b_1; r_1)$, and $c_2 = \mathrm{com}(b_2; r_2)$ a prover given b_0, b_1, b_2 and r_0, r_1, r_2 can generate a NIZK proof proving that $b_2 = \mathsf{NAND}(b_0, b_1)$. This, in fact, can be done very simply by just proving that each one of b_0, b_1, b_2 and $b_0 + b_1 + 2b_2 - 2$ is in $\{0, 1\}$. In other words, the prover generates a proof showing that each one c_0, c_1, c_2 and $c_0 \cdot c_1 \cdot {c_2}^2 \cdot \mathrm{com}(-2; 0)$ is commitments to a value in $\{0, 1\}$. Looking at the table of a NAND gate (as GOS prove), it is not too hard to prove that these conditions are simultaneously satisfied if and only if values $b_2 = \mathsf{NAND}(b_0, b_1)$.
2) Using the above trick, Groth et al. provide NIZK proofs for arbitrary NP-statements by converting them to a circuit SAT instance. More specifically, given a circuit C composed entirely of NAND gates, a prover can prove that $\exists \mathsf{wit}$ such that $C(\mathsf{wit}) = 1$. The prover achieves this as follows: it commits to the value assigned to every wire of the circuit C on input wit and proves that (i) each of the committed values is in $\{0, 1\}$, (ii) each NAND gate in C has been computed correctly, and (iii) the output of the circuit is 1.

Now, we very briefly describe how the GOS construction works in the setting of composite order groups with bilinear maps. GOS commitments are generated with respect to a commitment key which can either be in the binding mode or in the hiding mode and keys generated in the two modes are computationally indistinguishable.[6] The commitment key ck consists of a description of a source group $\mathbb{G}$ (of order $n = pq$), a target group $\mathbb{G}_T$, a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a group element h. In the binding mode, h is chosen randomly from the subgroup[7] $\mathbb{G}_q$ and in the hiding mode h is chosen randomly from $\mathbb{G}$. The commitment keys in the two modes are indistinguishable from the sub-group decision assumption. The commitment c to a message $m \in \mathbb{Z}_p$ using randomness r is given by $g^m h^r$. When h is chosen randomly from $\mathbb{G}$, c information theoretically hides m and when h is chosen from the sub-group $\mathbb{G}_q$ there exists unique $(m, r) \in \mathbb{Z}_p \times \mathbb{Z}_n$ such that $c = g^m h^r$. The homomorphic property is easy to observe. The proof π certifying that c is a commitment to 0 or 1 is given by $(g^{2m-1} h^r)^r$. The verification procedure relies on fact that if c is of the form h^r or gh^r then either c or cg^{-1} have order 1 or q (when h is chosen in the binding mode). This is ensured by checking if $e(h, \pi) = e(c, cg^{-1})$.

[6]In particular, the commitments generated using the binding key are perfectly binding whereas the ones generated using the hiding key are perfectly hiding.

[7]Recall that $\mathbb{G}_q$ is a sub-group of $\mathbb{G}$ with order q

B. New Technical Tool: Homomorphic Proof Commitments with Encryption

Armed with the above understanding of homomorphic proof commitments, we now explain how to augment them to support an encryption, decryption functionality. Specifically, an encryptor given a commitment c and a message msg can generate a ciphertext that can be efficiently decrypted using a proof π certifying the fact that c is a commitment to 0 or 1. Our security requirement is that if c is not a commitment to 0 or 1 then semantic security holds, i.e., for all $\mathsf{msg}, \mathsf{msg}'$ encryptions of msg are indistinguishable from encryptions of msg'. Note that if c is not a commitment to 0 or 1 then the prover cannot generate a proof certifying this fact. We call this primitive a homomorphic proof commitment with encryption. A careful reader might have noticed that the security provided by a homomorphic proof commitment with encryption is very similar to the security guarantee of a witness encryption [24]. Indeed, homomorphic proof commitment with encryption is a witness encryption scheme for a special language.

Next, we describe how the above abstract notion can be realized. An elegant aspect of our work is that this augmentation to the homomorphic proof commitments of GOS can be done without changing their construction. The encryption procedure on input a commitment $c = g^m h^r$ and a message msg essentially outputs the ciphertext $(h^s, e(c^s, cg^{-1}) \cdot \mathsf{msg})$ for a randomly chosen $s \leftarrow \mathbb{Z}_n$.[8] To decrypt this ciphertext using a proof $\pi = (g^{2m-1} h^r)^r$, compute $e(h^s, \pi)$ and use it to unmask the message msg. The key idea while proving security is that when h is chosen in the binding mode, h^s "loses" some information about s — specifically, $s \mod p$ is uniformly distributed even given h^s. Furthermore, this entropy in s is transferred to the masking factor $e(c^s, cg^{-1}) = e(h^s, \pi) e(g, g)^{sm(m-1)}$ when m is not 0 or 1. This allows us to argue that the message msg remains hidden.

C. Realizing Garbled Protocols

In this subsection we highlight the key challenge in constructing garbled protocols for the multiparty setting and how homomorphic proof commitments with encryption can be used to overcome this barrier.

The key challenge. With the goal of explaining the challenge involved, we start by considering garbled protocols in the *easy* case of two parties. We will focus only on how P_1 generates its garbled protocol components as the components generated by P_2 will be analogous. For the case of two parties, P_1 can just garble the next message

[8]The actual construction uses a strong randomness extractor and we avoid this in the informal overview.

functions of the protocol Φ (using Yao's garbled circuits) and send them over to the P_2. The only issue with this approach is how does P_1's garbled next message functions read the messages generated by P_2 in the execution of Φ. A natural idea is to have P_2 commit to its input x_2 (and also its randomness in case Φ is a randomized protocol) in its input encoding $\tilde{x}_2$ which will then be hard-coded inside the garbled next-message functions. Next, P_1 can generate garblings of next message functions in a manner so that P_2 would be able to evaluate those garblings as long as it can prove to P_1's garbled circuit that it has been generating its own messages consistent with the committed input x_2. At a very high level this can be achieved by letting P_1's garbled next message functions output ciphertexts containing encryptions of certain labels that P_2 can decrypt only if it has been generating its own messages correctly.

However, the techniques from the literature for doing this based on standard assumptions involve P_2's secret state in the decryption step. Consequently, these techniques fail even for the three party setting because the third party, say, P_3 does not have access to P_2's secret state. Gordan et al. [34] (building on Garg et al. [22]) observe that witness encryption [24] for NP can be used to solve this problem. The idea is: (i) P_1 outputs a witness encryption which allows decryption given just a NIZK proof certifying the correctness of computation, and (ii) P_2 outputs a proof for certifying this very fact. Next, using the proof, P_3 can decrypt P_1's ciphertext while secrecy of P_2's state is also maintained.

In this work, we show that the same intuition can be realized using homomorphic proof commitments with encryption. However, recall that homomorphic proof commitments with encryption are very weak. The encryption process cannot in "one-shot" verify that P_2 generated its messages correctly. Instead, our idea for this is that P_1 keeps P_2 on a "very tight leash," making sure that P_2 computes *every* NAND gate in the execution of Φ correctly.

The rest of this subsection is organized as follows. (1) We start by making some assumptions on the structure of distributed protocol Φ. We note that these assumptions can be made without loss of generality. (2) Next, we give a garbling scheme for such structured protocols.

Structure of Φ. Let Φ be a n-party protocol. For the purposes of this informal overview, we will assume that Φ is deterministic. Let T be the round complexity of the protocol. We assume that each party P_i maintains a local state that is updated at the end of every round. The local state is a function of the input and the set of messages received from other parties.

At the beginning of the t^{th} round, every party P_i runs a program Φ_i on input t to obtain an output (i^*, f, g). [9] Here, i^* denotes the *active party* in round t. The active party P_{i^*} computes *one* NAND gate on a pair of bits of its state and writes the computed bit to its state. The inputs to the NAND gate are given by the bits in the indices f and g of the local state of P_{i^*}. Additionally, for a (pre-determined) subset of rounds $B_{i^*} \subseteq \{t \in [T] : (i^*, \cdot, \cdot) = \Phi_i(t)\}$, P_{i^*} outputs the computed bit to other parties. In this case, all the parties copy this bit to their state.

We note that any protocol can be compiled to follow this format at an additional cost of increasing the round complexity by a polynomial factor.

Garbling Scheme for Protocols. The garbled protocol component $\widetilde{\Phi}_i$ generated by P_i consists of a sequence of T garbled circuits and a set of labels for evaluating the first garbled circuit in the sequence. These garbled circuits have a special structure, namely, the t^{th} garbled circuit in the sequence outputs the labels for evaluating the $(t+1)^{th}$ garbled circuit and thus starting from the first garbled circuit we can execute every garbled circuit in the sequence. At a high level, the t^{th} garbled circuit corresponds to the computation done by party P_i in the t^{th} round of the protocol Φ. In a bit more details, the t^{th} garbled circuit takes as input the local state obtained after the first $t-1$ rounds, updates the local state and outputs the labels corresponding to the updated state for evaluating the next garbled circuit. This ensures that at the end of the T^{th} evaluation, we can obtain the transcript of the protocol from the final local state of party P_i. The encoding of an input x_i is given by a set of homomorphic commitments $\{c_{i,k}\}$ to each individual bit of the input x_i.

To look a bit more closely into the working of the t^{th} garbled circuit, let us assume that P_i is the active party in the t^{th} round. Our assumption on the structure of Φ implies that in the t^{th} round, P_i has to update its local state by computing a NAND of two bits in its current state and write the output to a specific location. Further, if $t \in B_i$, P_i has to communicate this bit to the other parties and the other parties have to copy this bit to their state. In particular, this means that the labels output by the t^{th} garbled circuit in every other protocol component $\widetilde{\Phi}_j$ for $j \neq i$ must reflect this communicated bit. The main technical challenge we solve is in designing a non-interactive method to realize this communication and also ensure at the same time that P_i computes *each* NAND gate correctly. This is done using homomorphic proof commitment with encryption. Let us start with a method to realize the communication.

Recall that by our assumption on Φ, the updated state of every party can only be one of two choices. This choice is determined by the output of the NAND computation done by the active party. Let the NAND computation done in round t take as input the bits in positions f and g of the local state of party P_i. For simplicity, let us assume that f, g correspond to indices where the input of P_i is written. Let d be a commitment to 0 using some fixed randomness (known to all parties) and let $\overline{d}$ be a commitment to 1 (again using some fixed randomness). Applying the GOS trick, we

[9] We assume that $\Phi_1(t) = \Phi_2(t) = \ldots = \Phi_n(t)$ for every $t \in [T]$.

deduce that if the output of the NAND computation is 0 then $e_0 = c_{i,f} \cdot c_{i,g} \cdot d^2 \cdot \mathrm{com}(-2;0)$ is a commitment to $\{0,1\}$; else $e_1 = c_{i,f} \cdot c_{i,g} \cdot \overline{d}^2 \cdot \mathrm{com}(-2;0)$ is a commitment to $\{0,1\}$. Now, we let every other garbled protocol component $\widetilde{\Phi}_j$ for $j \neq i$ output two zero-one encryptions: one under the commitment e_0 containing the set of labels of the updated state assuming that the communicated bit is 0; and the other under the commitment e_1 assuming that the communicated bit is 1. The active party outputs a zero-one proof that either e_0 or e_1 is a commitment to a message in $\{0,1\}$. Using this proof, every party can recover the correct set of labels corresponding to the updated state.

Note that the above described solution reveals the output of the NAND gate in the clear to the other parties. This is necessary for the case where the bit is communicated to other parties but is undesirable if the NAND is an internal computation as it might reveal some information about the secret state of party P_i. On the contrary, every other party must somehow ensure that P_i computes this NAND gate correctly. We solve this problem by augmenting the input encoding with a commitment to a string of random bits i.e., the input encoding will be a homomorphic commitment to every bit of $x_i \| r_i$ where r_i is a random string. To prove that an internal NAND computation is done correctly, the active party P_i generates a zero-one proof that either $e_0 = c_{i,f} \cdot c_{i,g} \cdot d^2 \cdot \mathrm{com}(-2;0)$ or $e_1 = c_{i,f} \cdot c_{i,g} \cdot \overline{d} \cdot \mathrm{com}(-2;0)$ is a commitment to $\{0,1\}$ where d is now a commitment to a random bit generated as a part of the input encoding. $\overline{d}$ denotes the commitment to the flipped bit. Now, a proof that either e_0 or e_1 contains a commitment to $\{0,1\}$ reveals the output of the NAND computation masked with the random bit committed in d and hence completely hides the output. Note that the homomorphic property of the commitment scheme enables every party to efficiently generate $\overline{d}$. A downside of this approach is that the size of the input encoding grows with the round complexity of Φ. But using techniques from the recent work of Cho et al. [17], we can make the size of the input encoding succinct i.e., grow only with the size of the input. We won't delve into the details.

D. Black-Box Two-Round MPC

Instantiating the above garbled protocols construction with a semi-honest secure Φ, we obtain a two-round multi-party computation protocol based on bilinear maps.[10] However, the protocol makes non-black box use of the underlying homomorphic proof commitment with encryption as well as cryptographic operations that Φ might invoke. In this subsection, we explain how to obtain a two-round MPC protocol by making black-box use of a homomorphic proof commitment with encryption as well as a DDH hard group.

Designing a protocol that makes black-box use of a homomorphic proof commitment with encryption is somewhat straightforward. We observe that the proofs and the ciphertexts computed within the garbled circuit can in fact be precomputed and hardwired in its description. Later, the garbled circuit chooses the appropriate pre-computed values based on its inputs. We note that this pre-computation is possible because the output of each garbled circuit depends only on a constant number of bits in its input.

We now explain how to obtain a protocol that makes black-box use of cryptographic operations invoked by Φ.

Suppose Φ was an information theoretic secure MPC then the compiled protocol already makes black-box use of the underlying cryptographic primitives. But information theoretic secure MPC protocols can exist only if a majority of the parties are honest [11] and secure channels are present between every pair of parties. However, the situation in the OT hybrid model is different. There exist constructions of information theoretic protocols tolerating dishonest majority and malicious behavior [44], [41] in the OT hybrid model. We will be using such a protocol to design our black-box two round MPC.

Let Φ be an information theoretic secure protocol in the OT hybrid model tolerating malicious behavior. At a high level, our black-box two round MPC protocol generates OT correlations [11] in the first round and later hardwires these correlations in the garbled circuits to enable Φ perform information theoretic OTs. We now explain how to generate such OT correlations building on the non-interactive oblivious transfer by Bellare and Micali [10].

Let us first recall the OT protocol of Bellare and Micali in the common random string model. The crs consists of a random group element X. The sender samples a random exponent a and computes $A := g^a$ and sends it over to the receiver. The receiver samples a random exponent b and computes $B := g^b$. It then samples a random bit c and computes $C_0 := (1-c)B + c(\frac{X}{B})$ and $C_1 := cB + (1-c)(\frac{X}{B})$ and sends them over to A. Notice that by construction of C_0 and C_1, B knows the discrete log of C_c. The sender on receiving C_0 and C_1 sets the two random strings (s_0, s_1) to be (C_0^a, C_1^a) and the receiver sets (c, s_c) to be (c, A^b). Note that assuming the DDH assumption, the other string s_{1-c} is indistinguishable to a randomly distributed string. Building on this protocol and additionally using Groth-Sahai [38] proofs to obtain malicious security, we obtain a two round MPC protocol making black-box use a homomorphic proof commitment with encryption and a DDH hard group.

[10] For technical reasons, we need the protocol Φ to be semi-malicious [5]. The semi-malicious security is a generalization of semi-honest security where the adversary is still restricted to follow the protocol but can choose its random coins arbitrarily. Note that the protocol described in [30] is semi-maliciously secure.

[11] Recall that OT correlations consists of a random pair of strings (s_0, s_1) provided to the sender and a pair (c, s_c) where c is a random bit provided to the receiver.

III. Organization

In Section IV we formally define homomorphic proof commitment with encryption and give a construction based on sub-group decision assumption. The construction from decision linear assumption appears in the full version. In Section V, we give the definition of garbling scheme for protocols and in Section V-B we give a construction based on any homomorphic proof commitment with encryption. The results on extending garbled protocols to two-round UC secure MPC and two-round MPC making black-box use of the underlying group appears in the full version.

IV. Homomorphic Proof Commitments with Encryption

In this section we provide definitions of homomorphic proof commitments with encryption – namely, a homomorphic proof commitment scheme with some additional encryption and decryption functionality. We then give constructions of this primitive based on the sub-group decision. The construction from decision linear assumption appears in the full version.

We recall the definition of homomorphic proof commitments from Groth et al. [37] for realizing non-interactive zero-knowledge proofs. Much of the description below has been taken verbatim from Groth et al. [37]. We keep the notation identical to Groth et al. [37, Section 3] for the sake of a reader familiar with Groth et al. [37].

A homomorphic proof commitment scheme is a non-interactive commitment scheme with some special properties that we define below. Recall first that in a non-interactive commitment scheme there is a key generator, which generates a public commitment key ck. The commitment key ck defines a message space $\mathcal{M}_{ck}$, a randomizer space $\mathcal{R}_{ck}$ and a commitment space $\mathcal{C}_{ck}$. We will require that the key generation algorithm is probabilistic polynomial time and outputs keys of length $\theta(\lambda)$. It will in general be obvious which key we are using, so we will sometimes omit it in our notation. There is an efficient commitment algorithm com that takes as input the commitment key, a message and a randomizer and outputs a commitment, $c = \mathrm{com}(m; r)$. We call (m, r) an opening of c.

The commitment scheme must be binding and hiding. Binding means that it is infeasible to find two openings with different messages of the same commitment. Hiding means that given a commitment it is infeasible to guess which message is inside the commitment. We want a commitment scheme that has two different flavors of keys. The commitment key can be perfectly binding, in which case a valid commitment uniquely defines one possible message. Alternatively, the commitment key can be perfectly hiding, in which case the commitment reveals no information whatsoever about the message. We require that these two kinds of keys are computationally indistinguishable.

We will consider commitments, where both the message space $(\mathcal{M}, +, 0)$, the randomizer space $(\mathcal{R}, +, 0)$ and the commitment space $(\mathcal{C}, \cdot, 1)$ are finite abelian groups. The commitment scheme should be homomorphic, *i.e.*, for all messages and randomizers we have

$$\mathrm{com}(m_1 + m_2; r_1 + r_2) = \mathrm{com}(m_1; r_1)\mathrm{com}(m_2; r_2).$$

We will require that the message space has a generator 1, and also that it has at least order 4. The property that sets homomorphic proof commitments apart from other homomorphic commitments, is that there is a way to prove that a commitment contains a message belonging $\{0, 1\}$. More precisely, if the key is of the perfect binding type, then it is possible to prove that there exists an opening $(m, r) \in \{0, 1\} \times \mathcal{R}$. On the other hand, if it is a perfect hiding key, then the proof will be perfectly witness-indistinguishable, *i.e.*, it is impossible to tell whether the message is 0 or 1.

In the sections that follow, we will use $(K_{\mathrm{binding}}, K_{\mathrm{hiding}}, \mathrm{com}, P_{01}, V_{01})$ to denote a homomorphic proof commitment scheme. We refer the readers to [37] for the formal definition.

A. The Definition

$(K_{\mathrm{binding}}, K_{\mathrm{hiding}}, \mathrm{com}, P_{01}, V_{01}, E_{01}, D_{01})$ is a homomorphic proof commitments with encryption if $(K_{\mathrm{binding}}, K_{\mathrm{hiding}}, \mathrm{com}, \mathrm{Topen}, P_{01}, V_{01})$ is homomorphic proof commitment and E_{01}, D_{01} are PPT algorithms such that E_{01} on input a commitment key ck, a commitment $c = \mathrm{com}(ck, m; r)$ and a message msg outputs a ciphertext ct and D_{01} given ck, the commitment c, the ciphertext ct and a proof π such that $V_{01}(ck, c, \pi) = 1$ outputs the encrypted message msg. In other words, given a proof π such that c is a commitment to a message in $\{0, 1\}$, we can decrypt the ciphertext ct. Formally, we require that E_{01} and D_{01} satisfy the following correctness and security properties.

- **Perfect Correctness.** For any ck (in the support of $K_{\mathrm{binding}}, K_{\mathrm{hiding}}$), $m \in \{0, 1\}$, randomness r, and proof π generated by $P_{01}(ck, m, r)$ and message msg,

$$\Pr\left[\mathsf{ct} \leftarrow E_{01}(ck, \mathrm{com}(ck, m; r), \mathsf{msg}) \wedge D_{01}(ck, \mathsf{ct}, \pi) = \mathsf{msg}\right] = 1$$

- **Statistical Semantic-Security.** For all (possibly unbounded) adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr\Big[(ck, \cdot) \leftarrow K_{\mathrm{binding}}(1^\lambda); (c, \mathsf{msg}_0, \mathsf{msg}_1, \mathsf{st}) \leftarrow \mathcal{A}_1(ck); b \leftarrow \{0, 1\}; \mathsf{ct} \leftarrow E_{01}(ck, c, \mathsf{msg}_b) : \mathcal{A}_2(ck, \mathsf{ct}, \mathsf{st}) = b \wedge \exists\, m \notin \{0, 1\}, r \in \mathcal{R} \text{ s.t. } c = \mathrm{com}(ck, m; r)\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

[12] We have been a little imprecise in this overview. In order to use Groth-Sahai proofs we cannot rely on DDH assumption as GS proofs assume the existence of an efficiently computable bilinear map. In the actual construction we assume CDH is hard.

Perfectly binding key generation $K_{\text{binding}}(1^k)$:

1) $(p, q, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}_{\text{BGN}}(1^k)$. Let $n = pq$.
2) Sample $x \leftarrow \mathbb{Z}_q^*$ and compute $h = g^{px}$.
3) Let $ck = (n, \mathbb{G}, \mathbb{G}_T, e, g, h)$.
4) Let $xk = (ck, q)$.
5) Return (ck, xk).

Perfectly hiding key generation $K_{\text{binding}}(1^k)$:

1) $(p, q, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}_{\text{BGN}}(1^k)$. Let $n = pq$.
2) $x \leftarrow \mathbb{Z}_n^*$ and compute $h = g^x$.
3) Let $ck = (n, \mathbb{G}, \mathbb{G}_T, e, g, h)$.
4) Let $tk = (ck, x)$
5) Return (ck, tk)

Commitment $\text{com}_{ck}(m)$:

The key ck defines message space $\mathbb{Z}_p$, randomizer space $\mathbb{Z}_n$ and commitment space $\mathbb{G}$. To commit to message $m \in \mathbb{Z}_p$ do

1) $r \leftarrow \mathbb{Z}_n$
2) Return $\text{com}_{ck}(m; r) = g^m h^r$

WI proof $P_{01}(ck, m, r)$:

Given $(m, r) \in \{0,1\} \times \mathbb{Z}_n$ we make the WI proof for commitment to 0 or 1 as $\pi = (g^{2m-1} h^r)^r$.

Verification $V_{01}(ck, c, \pi)$:

To verify a WI proof π of commitment c containing 0 or 1, check $e(c, cg^{-1}) = e(h, \pi)$.

Figure 1. Homomorphic Proof Commitment from sub-group decision taken verbatim from [37]

We say that scheme has *computational* semantic-security if the above requirement holds only against PPT $\mathcal{A}$.

B. Construction from Sub-group Decision Assumption

In this subsection we give a construction of homomorphic proof commitment with encryption from the sub-group decision assumption.

At a high level, our construction $(K_{\text{binding}}, K_{\text{hiding}}, \text{com}, P_{01}, V_{01}, E_{01}, D_{01})$ is obtained by supplementing the homomorphic proof commitment scheme of Groth et al. [37], namely $(K_{\text{binding}}, K_{\text{hiding}}, \text{com}, P_{01}, V_{01})$ (given in Figure 1) with encryption E_{01} and decryption D_{01} operations (given in Figure 2). The supplemental encryption/decryption algorithms make use of a $(\log p, \mathsf{negl}(\lambda))$-strong randomness extractor $\text{RandExt} : \mathbb{G} \times \{0,1\}^* \to \{0,1\}^{\frac{\log p}{2}}$.

We now show correctness and security of the above construction.

Lemma 4.1: Assuming the subgroup decision assumption, the construction described in Figures 1 and 2 is a homomorphic proof commitment with encryption.

Proof: We note that $(K_{\text{binding}}, K_{\text{hiding}}, \text{com}, P_{01}, V_{01}, \text{Ext})$ is a homomorphic proof commitment scheme as argued by Groth et al. [37]. We now prove that (E_{01}, D_{01}) satisfy perfect correctness and statistical semantic-security.

Encrypt $E_{01}(ck, c, \mathsf{msg})$:

To encrypt $\mathsf{msg} \in \{0,1\}^{\frac{\log p}{2}}$,

1) Choose $s \leftarrow \mathbb{Z}_n$.
2) Choose $v \leftarrow \{0,1\}^*$ as the seed of RandExt.
3) Output $(v, h^s, \text{RandExt}(v, e(c^s, cg^{-1})) \oplus \mathsf{msg})$.

Decrypt $D_{01}(ck, c, \pi, \mathsf{ct})$:

1) Parse ct as $(v, \mathsf{ct}_1, \mathsf{ct}_2)$.
2) Output $\text{RandExt}(v, e(\mathsf{ct}_1, \pi)) \oplus \mathsf{ct}_2$.

Figure 2. Supplemental Encryption and Decryption.

Perfect Correctness. Let $c = \text{com}(ck, m; r)$ where $m \in \{0,1\}$. Let $\mathsf{ct} = (v, h^s, \text{RandExt}(v, e(c^s, cg^{-1})) \oplus \mathsf{msg})$ and $\pi = (g^{2m-1} h^r)^r$. To prove correctness it is sufficient to show that $e(h^s, \pi) = e(c^s, cg^{-1})$.

$$\begin{aligned} e(c^s, cg^{-1}) &= e(g^m h^r, g^{m-1} h^r)^s \\ &= e(g,g)^{sm(m-1)} e(h,g)^{sr(m-1)} e(g,h)^{smr} e(h,h)^{sr^2} \\ &= e(h,g)^{sr(m-1)} e(g,h)^{smr} e(h,h)^{sr^2} \\ &= e(h,g)^{sr(2m-1)} e(h,h)^{sr^2} \\ &= e(h^s, (g^{2m-1} h^r)^r) \\ &= e(h^s, \pi) \end{aligned}$$

Statistical Semantic Security. We first prove the following claim.

Claim 4.2: Let $(ck, \cdot) \leftarrow K_{\text{binding}}(1^\lambda)$. Let S denote the random variable uniformly distributed in $\mathbb{Z}_n$. Then

$$H_\infty(e(g,g)^S | (ck, h^S)) \geq \log p$$

Proof: Let $q_1 \equiv q^{-1} \mod p$ and $p_1 = p^{-1} \mod q$. By Chinese remainder theorem, any $s \in \mathbb{Z}_n$ can be expressed as $s_q p p_1 + s_p q q_1$ where $s_p \equiv s \mod p$ and $s_q \equiv s \mod q$. As $(ck, \cdot) \leftarrow K_{\text{binding}}(1^\lambda)$, therefore we have that $h = g^{px}$ for some $x \in \mathbb{Z}_q^*$. Thus, for any $s \in \mathbb{Z}_n$, $h^s = g^{x(s_q p^2 p_1) \mod n}$.

Let S be uniformly distributed in $\mathbb{Z}_n$. By Chinese remainder theorem, $S_p \equiv S \mod p$ and $S_q \equiv S \mod q$ are uniform and independent random variables in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ respectively. Also, $h^S = g^{x S_q p^2 p_1 \mod n}$. Therefore, conditioned on fixing h^S (which fixes S_q) and ck, g^S is still uniformly distributed over a set of size p since S_p is randomly distributed in $\mathbb{Z}_p$. Thus, $H_\infty(e(g,g)^S | (ck, h^S)) \geq \log p$. ∎

Consider a commitment $c = \text{com}(m; r)$ such that $m \notin \{0,1\}$. Let S be a random variable uniformly distributed in $\mathbb{Z}_n$. Then, we have that

$$e(c^S, cg^{-1}) = e(g,g)^{Sm(m-1)} e(h^S, g)^{r(m-1)} e(g, h^S)^{mr} e(h, h^S)^{r^2}$$

Since $m \notin \{0,1\}$, conditioned on fixing (h^S, ck), we infer from Claim 4.2 that $H_\infty(e(c^S, cg^{-1})) \geq \log p$. Now,

relying on the fact that the output of randomness extractor is statistically close to uniform we conclude statistical semantic security for the scheme. ■

V. Garbling Protocols

In this section we give the definition of garbling scheme for protocols and give an instantiation based on a homomorphic proof commitment with encryption.

A. Definition

Let Φ be a n-party protocol.[13] Let x_i be the input of party i and let Φ_i be the next-message function for party i. We define the transcript of Φ to be the set of all messages exchanged between parties. The transcript is denoted by $\Phi(x_1, \ldots, x_n)$ when Φ is run with inputs $x_1, \ldots, x_n$. The transcript is also assumed to be the output of the protocol.

Definition 5.1: A Garbling scheme for protocols is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Garble}, \mathsf{Eval})$ with the following syntax, correctness and security properties.

- $\mathsf{Setup}(1^\lambda)$: It is a PPT algorithm that takes as input the security parameter (encoded in unary) and outputs a reference string σ.
- $\mathsf{Garble}(\sigma, i, \Phi_i, x_i)$: It is a PPT algorithm that takes as input a reference string σ, the index i of a party, the next message function Φ_i and the input x_i and outputs
  - A garbled protocol component $\widetilde{\Phi}_i$ of the next message function Φ_i.
  - An encoding $\widetilde{x}_i$ (of length ℓ_e) of the input x_i.
  - A set of encoding labels $\{\mathsf{lab}^i_{j,0}, \mathsf{lab}^i_{j,1}\}_{j \in [n \cdot \ell_e]}$ for the input encodings of all parties.
- $\mathsf{Eval}(\{\widetilde{\Phi}_i\}, \{\widetilde{x}_i\}, \{\mathsf{lab}^i_{\widetilde{x}_1 \| \ldots \| \widetilde{x}_n}\})$: It is a deterministic algorithm that takes as input the set of garbled protocol components $\{\widetilde{\Phi}_i\}$, a set of input encodings $\{\widetilde{x}_i\}$ and the encoding labels $\{\mathsf{lab}^i_{\widetilde{x}_1 \| \ldots \| \widetilde{x}_n}\}$ corresponding to the input encodings $\widetilde{x}_1 \| \ldots \| \widetilde{x}_n$ and outputs a string y or the symbol $\perp$.
- **Correctness:** For every protocol Φ and every set of inputs $\{x_i\}$,

$$\Pr\Big[\sigma \leftarrow \mathsf{Setup}(1^\lambda); (\widetilde{\Phi}_i, \widetilde{x}_i, \{\mathsf{lab}^i_{j,0}, \mathsf{lab}^i_{j,1}\}) \leftarrow \mathsf{Garble}(\sigma, i, \Phi_i, x_i)\ \forall\ i \in [n] : \Phi(x_1, \ldots, x_n) = \mathsf{Eval}(\{\widetilde{\Phi}_i\}, \{\widetilde{x}_i\}, \{\mathsf{lab}^i_{\widetilde{x}_1 \| \ldots \| \widetilde{x}_n}\})\Big] = 1$$

- **Semi-Honest Security:** There exists a PPT algorithm Sim such that for every protocol Φ, every subset $H \subseteq [n]$ of honest parties and every choice of inputs $\{x_i\}_{i \in [n]}$ of the parties we have that:

$$\Big\{\sigma, \{\widetilde{\Phi}_i, \widetilde{x}_i, \mathsf{lab}^i_{\widetilde{x}_1 \| \ldots \| \widetilde{x}_n}\}_{i \in [n]}\Big\} \overset{c}{\approx} \Big\{\mathsf{Sim}(1^\lambda, \Phi, H, \{x_i\}_{i \notin H}, \Phi(x_1, \ldots, x_n))\Big\}$$

where $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ and for each $i \in [n]$ we have that $(\widetilde{\Phi}_i, \widetilde{x}_i, \{\mathsf{lab}^i_{j,0}, \mathsf{lab}^i_{j,1}\}) \leftarrow \mathsf{Garble}(\sigma, i, \Phi_i, x_i)$.

B. Construction

In this subsection we give a construction of a garbling scheme for protocols from a homomorphic proof commitment with encryption and a garbling scheme for circuits (which is implied by the existence of a commitment scheme). The main theorem that we prove in this section is:

Theorem 5.2: Assuming the existence of a homomorphic proof commitment with encryption there exists a construction of garbling scheme for protocols satisfying Definition 5.1.

Before describing the construction, we give some notation to describe the n-party protocol Φ and make additional assumptions on the structure of Φ. These assumptions can be made without loss of generality.

Notation for Φ. Recall that x_i denotes the input of party i and Φ_i denotes its next message function. We assume that the length of the input of each party is m. Let T be the round complexity of Φ.

Structure of Φ. We assume that each party P_i maintains a local state that is updated at the end of every round. The local state is a function of the input, the random tape and the set of messages received from other parties.

At the beginning of the t^{th} round, every party P_i runs the program Φ_i on input t to obtain an output (i^*, f, g). [14] Here, i^* denotes the *active party* in round t. The active party P_{i^*} computes *one* NAND gate on a pair of bits of its state and writes the computed bit to its state. The inputs to the NAND gate are given by the bits in the indices f and g of the local state of P_{i^*}. Additionally, for a (pre-determined) subset of rounds $B_{i^*} \subseteq \{t \in [T] : (i^*, \cdot, \cdot) = \Phi_i(t)\}$, P_{i^*} outputs the computed bit to other parties. In this case, all the parties copy this bit to their state. For the rest of the rounds where P_{i^*} is active, it outputs the computed bit masked with a random bit. In those rounds, every other party ignores this message.

To describe this structure more formally, let $B := \cup_i B_i$. Let the initial state of the party P_i be $r_i \| (x_i, s_i)$ where $x_i \in \{0,1\}^m$ is the input, $s_i \in \{0,1\}^s$ be the random tape used in the computation of Φ and $r_i \in \{0,1\}^T$ are the masking bits. We will let r_i have the form

$$r_{i,k} := \begin{cases} 0 & \text{if } k \in [T] \cap B \\ \text{uniform in } \{0,1\} & \text{if } k \in [T] \setminus B \end{cases}$$

We consider $r_i \| (x_i, s_i)$ as the actual input of party P_i.

For every $i \in [n]$, let y_i be the state of party P_i before the beginning of round t. Let $(i^*, f, g) := \Phi_i(t)$. The parties compute their updated state y'_i at the end of round t as

[13] For simplicity, we assume that Φ is deterministic. For the case where Φ is randomized, we extend the input string of each party to include its random coins so that Φ is a deterministic protocol in the inputs of the parties.

[14] We assume that $\Phi_1(t) = \Phi_2(t) = \ldots = \Phi_n(t)$ for every $t \in [T]$.

$$y'_{i^*,k} := \begin{cases} y_{i^*,k} & k \neq t \\ \mathsf{NAND}(y_{i^*,f}, y_{i^*,g}) & k = t \end{cases}$$

$$\text{for } i \neq i^* \quad y'_i := \begin{cases} y_i & t \notin B_{i^*} \vee \mathsf{NAND}(y_{i^*,f}, y_{i^*,g}) = 0 \\ y_i \oplus e_t & t \in B_{i^*} \wedge \mathsf{NAND}(y_{i^*,f}, y_{i^*,g}) = 1 \end{cases}$$

where e_k is the k-th unit vector. Finally, we let $\ell = T+m+r$ to denote the length of the local state of every party.

Remark 5.3: We observe that any protocol Φ can be re-written to follow the above format at an additional cost of increasing the round complexity by a polynomial (in the computational complexity of Φ) factor.

Construction. We make use of the following fact from [37].

Fact 5.4 ([37]): Let $\mathcal{M}$ be the message space of a homomorphic proof commitment with encryption. Further, $\mathcal{M}$ is a finite cyclic group with neutral element 0 and generator 1. Let $b_0, b_1, b_2 \in \{0,1\}$. If the order of the group is at least 4, then $b_2 = \neg(b_0 \wedge b_1)$ if and only if $b_0 + b_1 + 2b_2 - 2 \in \{0,1\}$

We give the formal description of our construction below. The construction uses a homomorphic proof commitment with encryption $(K_{\text{binding}}, K_{\text{hiding}}, P_{01}, V_{01}, E_{01}, D_{01})$ and a garbling scheme for circuits $(\mathsf{GarbleCkt}, \mathsf{EvalCkt})$.

- $\mathsf{Setup}(1^\lambda)$**:**
  1) Sample $(ck, \cdot) \leftarrow K_{\text{binding}}(1^\lambda)$ and output $\sigma := ck$ as the reference string.
- $\mathsf{Garble}(\sigma, i, \Phi_i, x_i)$**:**
  1) Compute $(\widetilde{x}_i, y_i, sk_i) \leftarrow \mathsf{Encode}(\sigma, i, x_i)$ where the function Encode is described below.
  2) Set $\mathsf{label}^{i,T+1} := \big((0,0), \ldots, (0,0)\big)$ where $(0,0)$ is repeated $\ell + n\ell_e + n\ell$ times and $\ell_e := |\widetilde{x}_i|$.
  3) **for** each t from T down to 1, $\big(\widetilde{\mathsf{P}}^{i,t}, \mathsf{label}^{i,t}\big) \leftarrow \mathsf{GarbleCkt}(1^\lambda, \mathsf{P}_\Phi[i, t, sk_i, ck, \mathsf{label}^{i,t+1}])$ where P_Φ is described below.
  4) Parse $\mathsf{label}^{i,1}$ as $\{\mathsf{st}^i_{k,0}, \mathsf{st}^i_{k,1}\}_{k\in[\ell]}$, $\{\mathsf{en}^i_{k,0}, \mathsf{en}^i_{k,1}\}_{k\in[n\ell_e]}$, $\{\mathsf{tr}^i_{k,0}, \mathsf{tr}^i_{k,1}\}_{k\in[n\ell]}$.
  5) Set $\mathsf{st}^i := \{\mathsf{st}^i_{k,y_{i,k}}\}_{k\in[\ell]}$ and $\mathsf{tr}^i := \{\mathsf{tr}^i_{k,0}\}_{k\in[n\ell]}$.
  6) Set the garbled protocol component $\widetilde{\Phi}_i := \big(\{\widetilde{\mathsf{P}}^{i,t}\}_{t\in[T]}, \mathsf{st}^i, \mathsf{tr}^i\big)$, the input encoding to $\widetilde{x}_i$ and the encoding labels to be $\{\mathsf{en}^i_{k,0}, \mathsf{en}^i_{k,1}\}_{k\in[n\ell_e]}$.
- $\mathsf{Eval}(\{\widetilde{\Phi}_i\}, \{\overline{x}_i\}, \{\mathsf{en}^i_{\widetilde{x}_1\|\ldots\|\widetilde{x}_n}\})$**:**
  1) For every $i \in [n]$, parse $\widetilde{x}_i$ as $\{c_{i,k}\}_{k\in[\ell]}$. For every $k \in B$, check if $c_{i,k} := \text{com}(ck, 0; 0^\lambda)$. If not, output $\perp$.
  2) Parse $\widetilde{\Phi}_i$ as $\big(\{\widetilde{\mathsf{P}}^{i,t}\}_{t\in[T]}, \mathsf{st}^i, \mathsf{tr}^i\big)$.
  3) Set $\widetilde{\mathsf{label}}^i := \big(\mathsf{st}^i, \mathsf{en}^i_{\widetilde{x}_1\|\ldots\|\widetilde{x}_n}, \mathsf{tr}^i\big)$ and the initial tracking strings $u_i := 0^\ell$ for every $i \in [n]$.
  4) **for** every round t from 1 to $T-1$ **do**:
     a) Let $(i^*, f, g) := \Phi_1(t)$.
     b) Compute $\big(\overline{\mathsf{label}}^{i^*}, \beta, \pi_{i^*,t}\big) \leftarrow \mathsf{EvalCkt}(\widetilde{\mathsf{P}}^{i^*,t}, \widetilde{\mathsf{label}}^{i^*})$ and $\overline{\mathsf{label}}^i \leftarrow \mathsf{EvalCkt}(\widetilde{\mathsf{P}}^{i,t}, \widetilde{\mathsf{label}}^i)$ for every $i \neq i^*$.
     c) **for** every $i \in [n]$ **do**,
        i) Parse $\overline{\mathsf{label}}^i$ as $(\overline{\mathsf{st}}^i, \overline{\mathsf{en}}^i, \overline{\mathsf{tr}}^i)$.
        ii) Compute d_f, d_g, e_0, e_1 exactly as in P_Φ using the tracking string u_{i^*}.
        iii) Parse $\overline{\mathsf{st}}^i$ as $\big(\{\widehat{\mathsf{st}}^i_k\}_{k\neq t}, \mathsf{stct}^i_0, \mathsf{stct}^i_1\big)$ and compute $\widehat{\mathsf{st}}^i_t := D_{01}(ck, e_\beta, \mathsf{stct}^i_\beta, \pi_{i^*,t})$. Update $\mathsf{st}^i := \{\widehat{\mathsf{st}}^i_k\}_{k\in[\ell]}$.
        iv) Parse $\overline{\mathsf{tr}}^i$ as $\left\{\{\widehat{\mathsf{tr}}^i_{(j-1)\ell+k}\}_{k\in[\ell]\setminus\{t\}}, \mathsf{trct}^i_{j,0}, \mathsf{trct}^i_{j,1}\right\}_{j\in[n]}$. For every $j \in [n]$, compute $\widehat{\mathsf{tr}}^i_{(j-1)\ell+t} := D_{01}(ck, e_\beta, \mathsf{trct}^i_{j,\beta}, \pi_{i^*,t})$. Update $\mathsf{tr}^i := \{\widehat{\mathsf{tr}}^i_k\}_{k\in[n\ell]}$.
        v) Update $\widetilde{\mathsf{label}}^i := \big(\mathsf{st}^i, \mathsf{en}^i_{\widetilde{x}_1\|\ldots\|\widetilde{x}_n}, \mathsf{tr}^i\big)$.
        vi) Update $u_{i^*,t}$ to β. If $t \in B_{i^*}$, update every $u_{j,t}$ to β for all $j \in [n]$.
  5) Compute $y := \mathsf{EvalCkt}(\widetilde{\mathsf{P}}^{i,T}, \widetilde{\mathsf{label}}^i)$ and output y.
- $\mathsf{Encode}(\sigma, i, x_i)$**:**
  1) Choose $s_i \leftarrow \{0,1\}^s$ as the random tape of party P_i in the protocol Φ.
  2) Let $B := \cup_i B_i$. Choose randomness $\{\omega_{i,k}\}_{k\in[\ell]}$ and the initial state $y_i := r_i \| (x_i, s_i)$ (with length ℓ) as:

$$r_{i,k} := \begin{cases} 0 & \text{if } k \in [T] \cap B \\ \text{uniform in } \{0,1\} & \text{if } k \in [T] \setminus B \end{cases}$$

$$\omega_{i,k} := \begin{cases} 0^\lambda & \text{if } k \in B \\ \text{uniform in } \{0,1\}^\lambda & \text{otherwise} \end{cases}$$

  3) For each $k \in [\ell]$, compute $c_{i,k} := \text{com}(ck, y_{i,k}; \omega_{i,k})$.
  4) Output $\widetilde{x}_i := \{c_{i,k}\}_{k\in[\ell]}$, the initial state y_i and the secret randomness $sk_i := \{\omega_{i,k}\}_{k\in[\ell]}$.
- $\mathsf{P}_\Phi[i, t, sk_i, \{ck_i\}, \mathsf{label}]$**:**
  Input. The state y_i of party P_i, the set of encodings $\{\widetilde{x}_j\}$ and the set of tracking strings $\{u_j\}$
  Hardcoded. The index i of the party, the round number t, the secret randomness sk_i, the commitment key ck and a set of labels $\mathsf{label} := \big\{\{\mathsf{st}_{k,0}, \mathsf{st}_{k,1}\}_{k\in[\ell]}, \{\mathsf{en}_{k,0}, \mathsf{en}_{k,1}\}_{k\in[n\ell_{enc}]}, \{\mathsf{tr}_{k,0}, \mathsf{tr}_{k,1}\}_{k\in[n\ell]}\big\}$.
  1) Let $(i^*, f, g) := \Phi_i(t)$.
  2) Parse $\widetilde{x}_{i^*}$ as $\{c_{i^*,k}\}_{k\in[\ell]}$.
  3) Let d_f and d_g be the commitments to the bits $y_{i^*,f}$ and $y_{i^*,g}$ where y_{i^*} is the current state of the active party. These commitments are computed as follows: for $h \in \{f, g\}$, $d_h := c_{i^*,h}$ if $u_{i^*,h} = 0$; else, $d_h := \frac{\text{com}(ck,1;0^\lambda)}{c_{i^*,h}}$.

4) Compute $e_0 := d_f d_g c_{i^*,t}^2 \mathrm{com}(ck, -2; 0^\lambda)$ and $e_1 := d_f d_g \left(\frac{\mathrm{com}(ck,1;0^\lambda)}{c_{i^*,t}}\right)^2 \mathrm{com}(ck, -2; 0^\lambda)$.
Set $\alpha := \mathsf{NAND}(y_{i,f}, y_{i,g})$.
5) For $b \in \{0,1\}$, compute $\mathsf{stct}_b := \begin{cases} E_{01}(ck, e_b, \mathsf{st}_{t,b}) & \text{if } t \in B_{i^*} \\ E_{01}(ck, e_b, \mathsf{st}_{t,y_{i,t}}) & \text{if } t \notin B_{i^*} \wedge i \neq i^* \\ E_{01}(ck, e_b, \mathsf{st}_{t,\alpha}) & \text{if } t \notin B_{i^*} \wedge i = i^* \end{cases}$.
Set $\overline{\mathsf{st}} := \{\mathsf{st}_{k,y_{i,k}}\}_{k \neq t}, \mathsf{stct}_0, \mathsf{stct}_1$.
6) Set $\overline{\mathsf{en}} := \{\mathsf{en}_{k,z_k}\}_{k \in [|z|]}$ where $z = \widetilde{x}_1 \| \ldots \| \widetilde{x}_n$.
7) For $b \in \{0,1\}$ and $j \in [n]$, compute $\mathsf{trct}_{j,b} := \begin{cases} E_{01}(ck, e_b, \mathsf{tr}_{(j-1)\ell+t,b}) & \text{if } (t \in B_{i^*}) \vee (j = i^* \\ E_{01}(ck, e_b, \mathsf{tr}_{(j-1)\ell+t,u_{j,t}}) & \text{otherwise} \end{cases}$
Set $\overline{\mathsf{tr}} := \left\{\{\mathsf{tr}_{(j-1)\ell+k,u_{j,k}}\}_{k \in [\ell] \setminus \{t\}}, \mathsf{trct}_{j,0}, \mathsf{trct}_{j,1}\right\}_{j \in [n]}$.
8) If $i = i^*$ then parse sk_i as $\{\omega_{i,k}\}_{k \in [\ell]}$. For $h \in \{f, g\}$, set $\omega'_{i,h} := \begin{cases} \omega_{i,h} & \text{if } u_{i,h} = 0 \\ -\omega_{i,h} & \text{otherwise} \end{cases}$
Compute $\pi_{i,t} := P_{01}(ck, e_\beta, \rho_\beta)$ where $\beta := y_{i,t} \oplus \alpha, \rho_0 = \omega'_{i,f} + \omega'_{i,g} + 2\omega_{i,t}, \rho_1 = \omega'_{i,f} + \omega'_{i,g} - 2\omega_{i,t}$.
9) If $t \neq T$ then output $\overline{\mathsf{label}} := (\overline{\mathsf{st}}, \overline{\mathsf{en}}, \overline{\mathsf{tr}})$ and additionally output $(\beta, \pi_{i,t})$ if $i = i^*$.
If $t = T$ then output the transcript of the protocol from the state $\{y_{i,k}\}_{k \in B}$.

Correctness. To argue correctness, it is sufficient to show that the local state of each party is updated correctly at the end of every round number t. We show this by induction on the number of rounds. The base case is clear. Let us assume that the hypothesis is true for the first t rounds. Let y_i be the local state of party P_i at the end of round t. Let $(i^*, f, g) := \Phi_1(t+1)$. We consider two cases:

- **Case-1:** $t + 1 \notin B_{i^*}$. In this case, the local state of parties $i \neq i^*$ does not change i.e., $y'_i = y_i$. The local state of party P_{i^*} is updated as $y'_{i^*,t+1} := \mathsf{NAND}(y_{i^*,f}, y_{i^*,g})$ and $y'_{i^*,k} = y_{i^*,k}$ for $k \neq t+1$. Notice that for the case where $t + 1 \notin B_{i^*}$, program P_Φ outputs the labels corresponding to the string $\{y'_{i,k}\}_{k \neq t+1}$ in the clear and outputs two zero-one encryptions of the same label corresponding to $y'_{i,t+1}$ under the commitments e_0 and e_1 respectively. Thus, decrypting stct^i_β using the proof $\pi_{i^*,t+1}$ yields the label corresponding to $y'_{i,t+1}$ for every $i \in [n]$. Thus, the updated states of every party is correct as per the computation of Φ.
- **Case-2:** $t + 1 \in B_{i^*}$. In this case, $y'_{i,t+1} = \mathsf{NAND}(y_{i^*,f}, y_{i^*,g})$ and $y'_{i^*,k} = y_{i^*,k}$ for $k \neq t+1$ for every party $i \in [n]$. The program P_Φ outputs the labels corresponding to the string $\{y'_{i,k}\}_{k \neq t+1}$ in the clear and outputs a zero-one encryption of the label $y'_{i,t+1}$ under the commitment $e_{y'_{i,t+1}}$ for every $i \in [n]$. Notice that by our construction $y_{i,t+1} = 0$ and thus $\beta := y'_{i,t+1}$. Thus, decrypting stct^i_β using the proof $\pi_{i^*,t+1}$ yields the label corresponding to $y'_{i,t+1}$ for every $i \in [n]$. Thus, even in this case the updated state of every party is correct as per the computation of Φ.

Security The security of the above construction is argued in the full version of the paper.

ACKNOWLEDGMENT

Research supported in part from 2017 AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, NSF CRII Award 1464397, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

REFERENCES

[1] Martín Abadi and Joan Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):1–12, 1990.

[2] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, Heidelberg, May 2001.

[3] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

[4] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011.

[5] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[6] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[7] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[8] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy*, pages 478–492. IEEE Computer Society Press, May 2013.

[9] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012.

[10] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and spplications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557. Springer, Heidelberg, August 1990.

[11] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[12] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[13] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

[14] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT 2017*, LNCS. Springer, Heidelberg, 2017. (to appear).

[15] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 190–213. Springer, Heidelberg, August 2016.

[16] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic receiver oblivious transfer and applications. *Manuscript*, 2017.

[18] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015.

[19] Stephen A. Cook and Robert A. Reckhow. Time bounded random access machines. *J. Comput. Syst. Sci.*, 7(4):354–375, 1973.

[20] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[21] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th ACM STOC*, pages 554–563. ACM Press, May 1994.

[22] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, February 2014.

[23] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[24] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[25] Sanjam Garg, Divya Gupta, Peihan Miao, and Omkant Pandey. Secure multiparty RAM computation in constant rounds. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 491–520. Springer, Heidelberg, October / November 2016.

[26] Sanjam Garg, Steve Lu, and Rafail Ostrovsky. Black-box garbled RAM. In Venkatesan Guruswami, editor, *56th FOCS*, pages 210–229. IEEE Computer Society Press, October 2015.

[27] Sanjam Garg, Steve Lu, Rafail Ostrovsky, and Alessandra Scafuro. Garbled RAM from one-way functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 449–458. ACM Press, June 2015.

[28] Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled RAM revisited. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 405–422. Springer, Heidelberg, May 2014.

[29] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, Heidelberg, August 2010.

[30] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[31] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[32] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2008.

[33] S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 513–524. ACM Press, October 2012.

[34] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015.

[35] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. Cryptology ePrint Archive, Report 2017/274, 2017. http://eprint.iacr.org/2017/274.

[36] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. In *Proceedings of Eurocrypt 2006, volume 4004 of LNCS*, pages 339–358. Springer, 2006.

[37] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[38] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[39] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 567–578. ACM Press, October 2015.

[40] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012.

[41] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[42] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.

[43] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004.

[44] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[45] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457. Springer, Heidelberg, August 2014.

[46] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008.

[47] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[48] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[49] Steve Lu and Rafail Ostrovsky. How to garble RAM programs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 719–734. Springer, Heidelberg, May 2013.

[50] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[51] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[52] Rafail Ostrovsky. Efficient computation on oblivious RAMs. In *22nd ACM STOC*, pages 514–523. ACM Press, May 1990.

[53] Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In *29th ACM STOC*, pages 294–303. ACM Press, May 1997.

[54] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 217–238. Springer, Heidelberg, October / November 2016.

[55] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, Heidelberg, December 2009.

[56] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[57] Michael O. Rabin. How to exchange secrets with oblivious transfer, 1981.

[58] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[59] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. Cryptology ePrint Archive, Report 2017/276, 2017. http://eprint.iacr.org/2017/276.

[60] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[61] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, April 2015.