Upper Bounds on the Communication Complexity of Optimally Resilient Cryptographic Multiparty Computation (Full Version)*

Martin Hirt[1] and Jesper Buus Nielsen[2]

[1] ETH Zurich, Deptartment of Computer Science
hirt@inf.ethz.ch
[2] University of Aarhus, Department of Computer Science
buus@daimi.au.dk**

Abstract. We give improved upper bounds on the communication complexity of optimally-resilient secure multiparty computation in the cryptographic model. We consider evaluating an n-party randomized function and show that if f can be computed by a circuit of size c, then $\mathcal{O}(cn^2\kappa)$ is an upper bound for active security with optimal resilience $t < n/2$ and security parameter κ. This improves on the communication complexity of previous protocols by a factor of at least n. This improvement comes from the fact that in the new protocol, only $\mathcal{O}(n)$ messages (of size $\mathcal{O}(\kappa)$ each) are broadcast during the *whole* protocol execution, in contrast to previous protocols which require at least $\mathcal{O}(n)$ broadcasts *per gate*. Furthermore, we improve the upper bound on the communication complexity of passive secure multiparty computation with resilience $t < n$ from $\mathcal{O}(cn^2\kappa)$ to $\mathcal{O}(cn\kappa)$. This improvement is mainly due to a simple observation.

1 Introduction

1.1 Secure multiparty computation

Secure multiparty computation (MPC) allows a set of n players to compute an arbitrary function of their inputs in a secure way. More generally, we consider reactive computations, which are specified as a circuit with input gates, evaluation gates (e.g., AND and OR gates), random gates, and output gates.

Security is specified with respect to an adversary corrupting up to t of the players for a defined threshold t. A *passive adversary* can inspect the internal state of corrupted players, an *active adversary* can take full control over them. A protocol is *t-secure* if an adversary attacking the protocol with t corruptions can only obtain inevitable goals w.r.t. gathering information and influencing

* The proceedings version of this paper appeared at Asiacrypt 2005 [HN05].
** Supported by FICS, Foundations in Cryptography and Security, Center of the Danish Research Council for Natural Sciences.

the output of the protocol. I.e. it can only learn the inputs and outputs of the corrupted players, and, if it is active, only influence the inputs of the corrupted players.

1.2 Brief history of MPC

The MPC problem dates back to Yao [Yao82]. Independently Goldreich, Micali and Wigderson and Chaum, Damgård and van de Graaf [GMW87,CDG87] presented solutions to the MPC problem. Their protocols provide cryptographic security against a computationally bounded active adversary corrupting up to $t < n/2$ of the players. Later, unconditionally secure MPC protocols were proposed by Ben-Or, Goldwasser and Wigderson [BGW88] and Chaum, Crépeau and Damgård [CCD88] for the *secure-channels model*, where perfectly secure channels are assumed between every pair of parties. These protocols have resilience $t < n/3$. Later Rabin and Ben-Or [RB89] and independently Beaver [Bea91b] presented protocols with resilience $t < n/2$ for the secure-channels model with broadcast channels.

1.3 Previous work on the complexity of secure MPC

There has been substantial research on the complexity of secure MPC, both the round complexity and the communication complexity in messages and bits.

As for the round complexity of secure MPC, it is now known that in a network without any setup any functionality can be computed securely in three rounds and that there exists functionalities which cannot be computed in two rounds without setup [GIKR02]. Furthermore, it is known that after an initial setup phase, any functionality can be computed in two rounds [GIKR02,CDI05] and that there exist functionalities which cannot be computed in one round even after a setup phase. Even though the resulta in [GIKR02,CDI05] only applies to a setting where the number of parties is relatively small, the above results go a long way in resolving the exact round complexity of secure MPC.

As for the communication complexity, the picture is much more open, and we are far from knowing the exact communication complexity of secure MPC. The *communication complexity* of a protocol is measured as the total number of bits sent by all uncorrupted parties during the protocol execution.

Very few results are known about the lower bound on the communication complexity, except those which follow trivially from known lower bounds on the communication complexity of Byzantine agreement — since the model of secure MPC requires agreement on the output, Byzantine agreement is a special case of secure MPC. For the upper bound on the communication complexity, much more is known.

The seminal protocols with passive security tend to be very communication-efficient, in contrast to their active-secure counterparts, that require high communication complexities. The high communication complexities of active-secure protocols is mainly due to their intensive use of a Byzantine agreement primitive, which is to be simulated by communication-intensive broadcast protocols.

The most efficient broadcast protocols for $t < n$ communicate $\Omega(n^2\ell)$ bits for broadcasting an ℓ-bit message [BGP92,CW92]. We denote the communication complexity for broadcasting an ℓ-bit message by $\mathcal{B}(\ell)$.

Over the years, several protocols have been proposed which improve the efficiency of active-secure MPC. In the cryptographic model (with $t < n/2$), all protocols presented so far [GV87,BB89,BMR90,BFKR90,Bea91a,GRR98,CDM00,CDD00] require every player to broadcast one message for each multiplication gate. For a circuit with c gates, this results in a total communication complexity of $\Omega(cn\mathcal{B}(\kappa)) = \Omega(cn^3\kappa)$, where κ denotes the security parameter of the protocol. In the secure-channels model with broadcast with $t < n/2$, things are even worse: The most efficient protocol in this model [CDD+99] requires $\Omega(n^4)$ κ-bit messages to be broadcast for every multiplication gate.

In the secure-channels model with $t < n/3$, recently more efficient protocols were proposed [HMP00,HM01]: The latter protocol requires only $\mathcal{O}(n^2)$ broadcasts in total (independently of the size of the circuit), and communicates an additional $\mathcal{O}(cn^2)$ bits in total. This result is based on the so-called *player-elimination framework*, where subsets of players with faulty majority are eliminated. This prevents corrupted players from repetitively disturbing and slowing down the computation. Unfortunately, the player-elimination framework cannot capture models with $t < n/2$: In order to reconstruct an intermediate value (a wire), at least $t + 1$ players are required. After eliminating a group of players with faulty majority, the remaining set of players does not necessarily contain $t + 1$ honest players (it might even contain only one single player), hence the remaining players cannot reconstruct intermediate results — and would have to restart the whole computation.

1.4 Contributions

We consider upper bounds on the communication complexity of active-secure MPC protocol in the cryptographic model with $t < n/2$ and passive-secure MPC protocols in the cryptographic model with $t < n$. The most efficient active-secure protocol for this model is the protocol by Cramer, Damgård and Nielsen [CDN01]. This protocol requires every player to broadcast $\mathcal{O}(1)$ κ-bit values for each multiplication gate in the circuit. When replacing the broadcast primitive by the most efficient broadcast protocol with resilience $t < n/2$ known today (but unknown at the time when [CDN01] was published), this results in an overall communication complexity of $\mathcal{O}(cn^3\kappa)$ for evaluating a circuit with c gates. The same upper bound for active security was proved by Jakobsson and Juels [JJ00] using similar techniques.

We improve the upper bound for active security by constructing a new MPC protocol for the cryptographic model with resilience $t < n/2$: The new protocol requires every player to broadcast $\mathcal{O}(1)$ κ-bit values *in total*, i.e., during the whole protocol execution. Additionally, the players communicate $\mathcal{O}(n^2\kappa)$ bits per multiplication over the normal channels. This results in a total communication

complexity of $\mathcal{O}(cn^2\kappa + n\mathcal{B}(\kappa)) = \mathcal{O}(cn^2\kappa + n^3\kappa)$. If every party has just one input to the circuit, then $c \geq n$ and $\mathcal{O}(cn^2\kappa + n^3\kappa) = \mathcal{O}(cn^2\kappa)$.[3]

The new protocol follows the basic paradigm of [CDN01], enhanced with ideas of [Bea91a] and [HMP00] and several novel technical contributions. Our protocol essentially improves over the best known upper bound for active security by a factor n.

Using a simple observation about threshold homomorphic encryption-based MPC protocols we also present a passive secure protocol with resilience $t < n$, communicating only $\mathcal{O}(cn\kappa)$ bits. This improves the best known upper bound for passive security, as given by the protocol of Franklin and Haber [FH96], by a factor n.

2 Preliminaries

In this section we discuss our model of security of protocols and we sketch the technical setting for threshold homomorphic encryption based MPC. The reader familiar with these issues can safely skip this section.

2.1 Model

We consider n players that are pairwise connected with authenticated open channels and we assume synchronous communication. The adversary may corrupt any t of the players. All parties and the adversary are restricted to probabilistic polynomial time. We consider a *static* adversary, which corrupts all parties before the protocol execution.

Specifying a multiparty functionality. We assume that the task to be realized is given by an arithmetic circuit with input, addition, multiplication, randomizing and output gates, all over some ring $\mathbb{M}$. We consider reactive circuits where some input gates might appear after output gates. We assume that the circuit is divided into layers being either input layers, consisting solely of input gates, evaluation layers consisting of addition, multiplication, and randomizing gates, and output layers, consisting solely of output gates. An input gate G specifies its layer and the party that is to supply the value for the gate. A negation gate specifies its layer and a gate in a previous layer, from which it takes its input. An addition gate as well as a multiplication gate specifies its layer and two gates in a previous layer, from which it takes its input. An output gate specifies its layer and a gate in a previous layer, which is to be revealed.

The ideal evaluation. To explain the multiparty functionality specified by a reactive circuit, it is convenient to image an ideal process, where the parties are connected to a fully trusted party with secure channels. The ideal evaluation of

[3] For simplicity we specify all bounds in the following for circuits with $c = \Theta(n)$. Bounds for $c \leq n$ are obtained by letting $c = n$.

the circuit takes place in a layer by layer manner. For each input layer, for every gate specifying P_i as the party to contribute the input, P_i sends to the trusted party an input value $v \in \mathbb{M}$ over a secure line. If no value is sent, the trusted party sets v to be 0. For each evaluation layer, the trusted party computes values of all evaluation gates according to the circuit; Randomizing gates are set to be uniformly random values $v \in_R \mathbb{M}$ and addition gates and multiplication gates are evaluated in the expected manner. For each output layer, the trusted party sends the value of all output gates in the layer to all parties.

Notice that in the ideal evaluation an adversary controlling some set of corrupted parties can only achieve inevitable goals: Of information it only learns the output and the corrupted parties' inputs and, if it is active, the only influence it can exert on the evaluation is changing the corrupted parties' inputs to the function.

The goal of a protocol for a circuit is to realize the same functionality in a real-life network.

The real-life model. We assume that the network has a *setup phase*. In the setup phase a *setup function* $s : \{0,1\}^* \to (\{0,1\}^*)^{n+1}, r \mapsto (p, s_1, \ldots, s_n)$ is evaluated on a random input, and the value p is made public. The value s_i is only given to the party P_i. The reason for having a setup phase is that we will be interested in MPC protocols with active resilience $t < n/2$, and without a setup phase not even the Byzantine agreement problem [LSP82], which is a special case of the general MPC problem, can be solved with active resilience $t < n/2$. The function s is specified as part of the general protocol. In particular, s is not allowed to depend on the circuit.

Defining security. There are many proposals on how to model the security of an n-party protocol, i.e. for what it means for a protocol to realize the ideal evaluation of a circuit. Common to most is that the real-life adversary can only obtain goals comparable to those of an ideal-model adversary, i.e. inevitable goals.

The comparison of the protocol execution to the ideal evaluation is made by requiring that the complete view of an adversary attacking the protocol execution can be simulated given only the view of an adversary attacking the ideal evaluation with the same corrupted parties. This captures exactly the idea that the information gathering and the influencing capabilities of the adversary include nothing extra to that of which the adversary is entitled. This so-called simulation approach to comparing the protocol execution to the ideal evaluation originates in the definition of zero-knowledge proof in [GMR85] by Goldwasser, Micali and Rackoff. For the MPC setting the simulation approach is introduced by Goldreich, Micali and Wigderson [GMW87] and elaborated on in a large body of later work [GL90,MR91,Bea91b,BCG93,HM00,Can00,Can01]. Of these models, the universally composable (UC) security framework of Canetti [Can01] gives the strongest security guarantees. When proving an upper bound it makes sense to consider the strongest security notion. The core model in [Can01] is asynchronous, but contains hints on how to apply it to a synchronous setting as

we consider here. This was e.g. done in [DN03]. It is straight-forward to formally cast our reactive circuit model in the model of [DN03], and we can prove all our protocols secure in this model.

For the detail of proofs permitted in this extended abstract we will not need any formal details about this particular simulation model. The informal proof sketches given in subsequent sections can easily be extended to fully formal simulation proofs using by now standard proof techniques for threshold homomorphic encryption based MPC, see e.g. [CDN01,DN03].

2.2 Homomorphic encryption scheme

In our protocols we assume the existence of a semantically secure (in the sense of IND-CPA [BDPR98]) probabilistic public-key encryption function $E_Z : \mathbb{M} \times \mathbb{R} \to \mathbb{E}, (m, \alpha) \mapsto M$, where Z denotes the public key, $\mathbb{M}$ denotes a set of messages, $\mathbb{R}$ denotes the set of random strings, and $\mathbb{E}$ denotes the set of encryptions. We write E instead of E_Z for shorthand. The decryption function is $D_z : \mathbb{E} \to \mathbb{M}, M \mapsto m$, where z denotes the secret key. Again, we write D instead of D_z.

We require that E is a group homomorphism, i.e., $E(m_1, \alpha_1) \oplus E(m_2, \alpha_2) = E(m_1 + m_2, \alpha_1 \boxtimes \alpha_2)$ for the corresponding group operations $+$ in $\mathbb{M}$, $\boxtimes$ in $\mathbb{R}$, and $\oplus$ in $\mathbb{E}$. We require that $\mathbb{M}$ is a ring $\mathbb{Z}_M$ for $M > 1$. The other groups can be arbitrary.

In general we use capital letters to denote the encryption of the corresponding lowercase letters. For $a \in \mathbb{N}$ and $B \in \mathbb{E}$ and $\alpha \in \mathbb{R}$ we write aB as a shorthand for $B \oplus \cdots \oplus B$ with $a-1$ additions and we use α^a as a shorthand for $\alpha \boxtimes \cdots \boxtimes \alpha$ with $a-1$ multiplications. We use $A \ominus B$ to denote $A \oplus (-B)$, where $-B$ denotes the inverse of B in $\mathbb{E}$.

We define a ciphertext-randomization function $\mathbb{R} : \mathbb{E} \times \mathbb{R} \to \mathbb{E}, (M, \gamma) \mapsto (M \oplus E(0, \gamma))$. If $M = E(m, \alpha)$, then $\mathbb{R}(M, \gamma) = E(m, \alpha \boxtimes \gamma)$. If γ is uniformly random in $\mathbb{R}$ and independent of α, then $\alpha \boxtimes \gamma$ is uniformly random in $\mathbb{R}$ and independent of α, so $\mathbb{R}(M, \gamma)$ will be a new independent, uniformly random encryption of m. We say that $M' = \mathbb{R}(M, \gamma)$ is a randomization of M.

We also require that there exists a passive secure threshold function sharing of D_z between n parties. I.e. for a given threshold t we split the decryption key z in n shares $z_1, \ldots, z_n$ and there exists a share-decryption function $SD_{z_i} : \mathbb{E} \to \mathbb{S}, M \mapsto m_i$, where $\mathbb{S}$ denotes the set of message shares. And there exists a combining function $C : \mathbb{S}^{t+1} \to \mathbb{M}, (m^{(1)}, \ldots, m^{(t+1)}) \mapsto m$, with the property that if $M = E_Z(m)$ and $m^{(j)} = SD_{z_{i_j}}(M)$ for $i = 1, \ldots, t+1$ and $t+1$ distinct key shares z_{i_j}, then $m = C(m^{(1)}, \ldots, m^{(t+1)})$. We require that the semantic security holds even when the distinguisher is given any t decryption key shares prior to the distinguishing game. Furthermore, for all $M = E_Z(m)$, given M, m and any t key shares one can efficiently compute *all* decryption shares $m_i = D_{z_i}(M)$ for $i \in \{1, \ldots, n\}$. This requirement is made to guarantee that no subset of the parties of size at most t learns anything from the other parties' decryption shares, which they could not have computed themselves from the result of the decryption.

Realizations. The probabilistic encryption function of Paillier [Pai99], enhanced by threshold decryption [FPS00,DJ01], satisfies all required properties. This scheme has $\mathbb{M} = \mathbb{Z}_N$ for an RSA modulus N. A scheme satisfying the requirements can also be build based on the QR assumption [CDN01,KY02]. For this scheme $\mathbb{M} = \mathbb{Z}_2$.

2.3 Non-malleable zero-knowledge proofs

In this section we give a brief overview of the approach used in [CDN01] to efficiently realize non-malleable zero-knowledge proofs. We first recall the notion of Σ-protocol, then describe a general transformation of a Σ-protocol into a non-malleable zero-knowledge proof and then discuss some of the proofs used in the main text of the paper.

Σ-protocols. A binary relation is a subset $R \subseteq \{0,1\}^* \times \{0,1\}^*$. We use $L(R)$ to denote the set $\{x \in \{0,1\}^* | \exists w((x,w) \in R)\}$. For two binary relations R_1 and R_2 we define binary relations $R_1 \wedge R_2$ and $R_1 \vee R_2$ by $((x_1,x_2),(w_1,w_2)) \in R_1 \wedge R_2$ if and only if $(x_1,w_1) \in R_1$ and $(x_2,w_2) \in R_2$ and $((x_1,x_2),w) \in R_1 \vee R_2$ iff $(x_1,w) \in R_1$ or $(x_2,w) \in R_2$.

A Σ-protocol for proof of membership (knowledge) for R is a three-move special honest-verifier zero-knowledge proof of membership (knowledge) for R [Cra96]. The input to the prover is $(x,w) \in R$ and the input to the verifier is x. Two Σ-protocols Σ_1 and Σ_2 can easily be combined to a new Σ-protocols $\Sigma_1 \wedge \Sigma_2$ and $\Sigma_1 \vee \Sigma_2$ for $R_1 \wedge R_2$ respectively $R_1 \vee R_2$, using two proofs in parallel respectively the so-called OR-construction.

From Σ-protocols to non-malleable zero-knowledge proofs. If the prover and verifier share a trapdoor commitment scheme, then Σ-protocols can be turned into zero-knowledge proofs by having the prover commit to the first message and reveal it after seeing the challenge [Dam00]. If the trapdoor commitment scheme is non-malleable [DCIO98] with respect to opening [FF00], then the resulting zero-knowledge proof will be non-malleable. If the expansion factor of the commitment scheme is constant, then the transformation yields a zero-knowledge protocol with communication complexity in the order of the communication complexity of the Σ-protocol; The expansion factor is the number of bits used to commit to κ bits, divided by κ.

Realizations. For the non-malleable trapdoor commitment scheme used for turning Σ-protocols into non-malleable zero-knowledge protocols several possibilities exist. One can construct a non-malleable trapdoor commitment scheme with constant expansion factor from any trapdoor commitment scheme with constant expansion factor by simply giving each party an independent, random commitment key [CDN01]. Trapdoor commitment schemes with constant expansion factor can be based on either the DCR assumption or the strong RSA assumption. We could also use the commitment scheme from [DG02], which is based on

the strong RSA assumption. This scheme only needs one short global key to be set up.

Robust threshold homomorphic encryption. For the active-secure protocol we need several non-malleable zero-knowledge proofs associated to the threshold homomorphic encryption scheme. To help specify our requirements we define three binary relations R_Z^{PK}, R_Z^{CM} and R_Z^{CSD} associated with the encryption key.

The relation R_Z^{PK} is used to prove Plaintext Knowledge of encryptions; Here the verifier has an encryption M as input, where the prover claims to know the plaintext, and the prover has the witness (m, α), where $M = E(m, \alpha)$. The relation R_Z^{CM} is used to prove Correctness of the Multiplication of a b_i factor onto A; Here the verifier has three encryptions A, B and C as input, where the prover claims that $D(C) = D(A)D(B)$, and the prover has the witness (b, β, γ), where $B = \mathbb{E}(b, \beta)$ and $C = \mathbb{R}(bA, \gamma)$. Finally, the relation R_Z^{CSD} is used to prove Correctness of the Shares Decryption m_i of an encryption M. Here the verifier has input M, i and m_i, where the prover claims that m_i is the i'th message share for M, and the prover has input z_i, where z_i is the i'th key share and $m_i = SD_{z_i}(M)$. [4]

We require that there exists a Σ-protocol Σ_Z^{PK} for proof of knowledge for the relation R^{PK} and we require that there exist Σ-protocols Σ_Z^{CM} and Σ_Z^{CSD} for proof of membership for the relations R_Z^{CM} respectively R_Z^{CSD}. Using the Σ-protocol Σ_Z^{PK} and the $\wedge$ operation on Σ-protocols can ensure that Σ_Z^{CM} is also a proof of plaintext knowledge for B. We furthermore require that the communication complexity of each of these Σ-protocols is $\mathcal{O}(\kappa)$. We will use R and Σ as shorthands for R_Z respectively Σ_Z.

Given such Σ-proofs they can be transformed into non-malleable zero-knowledge proofs using the technique discussed in Section 2.3.

Realizations. The threshold homomorphic encryption schemes discussed in Section 2.2 have the necessary Σ-protocols. The scheme based on the QR assumption in addition needs the strong RSA assumption for the existence of the Σ-protocols.

[4] More formally, the relation R_Z^{PK} is given by $(x, w) \in R_Z^{\mathrm{PK}}$ iff $x \in \mathbb{E}$, $w = (m, r) \in \mathbb{M} \times \mathbb{R}$ and $x = E_Z(m, r)$. The relation R_Z^{CM} is given by $(x, w) \in R_Z^{\mathrm{CM}}$ iff $x = (A, B, C) \in \mathbb{E}^3$, $w = (b, \beta, \gamma) \in \mathbb{M} \times \mathbb{R} \times \mathbb{R}$, $B = E_Z(b, \beta)$ and $C = (bA) \oplus E_Z(0, \gamma)$. For convenience in defining the third relation we require that z is uniquely given by Z, that the i'th key-share z_i of z is uniquely given by z, that z_i contains a witness to the fact that it is the i'th key-share corresponding to Z and that SD_{z_i} is deterministic. The relation R_Z^{CSD} is given by $(x, w) \in R_Z^{\mathrm{CSD}}$ iff $x = (i, M, m_i) \in \{1, \ldots, n\} \times \mathbb{M} \times \mathbb{S}$, $w = z_i \in \{0, 1\}^*$, $m_i = SD_{z_i}(M)$ and z_i is the i'th key share of the decryption key z corresponding to Z.

3 Active-secure MPC protocol for $t < n/2$

In this section we present our upper bound on the communication complexity of an active-secure MPC protocol. The upper bound is given by a protocol. We first give an overview on this protocol, then present the required sub-protocols, and finally analyze the security and the communication complexity.

3.1 Overview

In the protocol description we use $\mathcal{P} = \{P_1, \ldots, P_n\}$ to denote the set of parties. We assume that the parties agree on the circuit before the protocol is run. The circuit is specified over the ring $\mathbb{M}$ of the encryption scheme with input gates, addition gates, multiplication gates, randomizing gates, and output gates. The proposed protocol can easily be modified to evaluate Boolean circuits, see Section 3.7 for details. In the simplest case, when the parties wish to evaluate a deterministic function, the circuit will consist of a layer of inputs gates, then the arithmetic gates necessary to evaluate the function, and finally the output gates. However, we also consider randomized gates, set to an unknown random values, and reactive circuits, where some players may receive output *before* some (other) players provide inputs.

The proposed protocol follows Beaver's circuit randomization approach [Bea91a]: In a preparation phase, a pool of random triples (a, b, c), with $c = ab$, are generated, encrypted and distributed to all players. In the evaluation phase, for each multiplication one prepared triple is used. This approach brings two advantages: First, it might be simpler to generate random products (instead of multiplying two given values). Second, the load of the multiplication protocol is shifted to the preparation phase, where all triples are generated *in parallel*, and costs can be amortized.

More formally, the protocol proceeds in three phases:

Setup Phase: In the *setup phase* a random key pair (Z, z) is generated and the decryption key z is shared among the parties with threshold t, where $t < n/2$.

Preparation Phase: In a *preparation phase*, c_M random triples $\left(a^{(i)}, b^{(i)}, c^{(i)}\right) \in \mathbb{M}^3$ (for $i = 1, \ldots, c_M$) with $c^{(i)} = a^{(i)}b^{(i)}$ are generated, encrypted, and given to every player in $\mathcal{P}$, where c_M denotes the number of multiplication gates in the circuit. Furthermore, c_R random values $r^{(i)} \in \mathbb{M}$ (for $i = 1, \ldots, c_R$) are generated and encrypted, where c_R denotes the number of random gates in the circuit.

Evaluation Phase: In an *evaluation phase*, the gates of the circuit are processed level by level, associating to each gate a random ciphertext encrypting the (output) value of the gate. The various gates are handled as follows: For each *input gate*, the designated input party broadcasts an encryption of its input for that gate. *Addition gates* are handled non-interactively using the homomorphic properties of the encryption scheme. For each *multiplication gate* one prepared triple from the preparation phase is used as described

in [Bea91a]. For each *randomizing gate*, an encryption of a prepared random value $r^{(i)}$ is used. For the *output gates*, the ciphertexts are decrypted using the threshold function sharing of D_z.

In the subsequent sections we describe the phases of the protocol in detail, and finally analyze the overall complexity of the protocol.

3.2 Setup phase

The setup function generates $((Z, pk, H), z_1, \ldots, z_n)$, where (Z, z) is a random key pair with z split into $(z_1, \ldots, z_n)$ with threshold t, pk is a random key for a non-malleable trapdoor commitment scheme,[5] and H is a random hash function chosen from a class of collision-resistant hash functions, which is used by a protocol described in the following section. The setup function also sets up digital signatures to allow to do Byzantine Agreement (BA) for resilience $t < n/2$, as discussed in Section 2.1.

One could consider a simpler setup function which only sets up digital signature keys. This allows to realize BA for resilience $t < n/2$, which in turn allows to run a secure protocol to compute the setup function for the remaining values. Either a specialized protocol or one of the general MPC protocols. In all cases this would add a term $p = \mathcal{O}(\text{poly}(n+\kappa))$ to our bounds, where p is independent of the circuit to be evaluated, giving a bound $\mathcal{O}(cn^2\kappa + \text{poly}(n+\kappa))$.

3.3 Preparation phase

The goal of this phase is to securely generate c_M encrypted triples $\left(A^{(i)}, B^{(i)}, C^{(i)}\right)$ $(i = 1, \ldots, c_M)$, where $a^{(i)}$ and $b^{(i)}$ are uniformly random values from $\mathbb{M}$ unknown by all parties and $c^{(i)} = a^{(i)}b^{(i)}$, and furthermore, to generate c_R encrypted random values $R^{(i)}$ $(i = 1, \ldots, c_R)$.

The preparation phase proceeds in three stages: First, c_M random factors $A^{(1)}, \ldots, A^{(c_M)}$ are generated. Second, the factors $B^{(1)}, \ldots, B^{(c_M)}$ and the products $C^{(1)}, \ldots, C^{(c_M)}$ are computed in parallel. Third, the random values $R^{(1)}, \ldots, R^{(c_R)}$ for the randomizing gates are prepared.

In each stage, every player in $\mathcal{P}$ contributes to the generation of the values. However, not all these contributions will be considered. Instead, the players in $\mathcal{P}$ agree on a subset $\mathcal{P}_{\text{ok}} \subseteq \mathcal{P}$ with the following two properties: (1) Every player in $\mathcal{P}_{\text{ok}}$ successfully verified the contribution of every other player in $\mathcal{P}_{\text{ok}}$, and (2) the majority of the players in $\mathcal{P}_{\text{ok}}$ is honest. Given both properties are satisfied, the output of the stage (so far known only to $\mathcal{P}_{\text{ok}}$) can easily be made known to the players in $\mathcal{P} \setminus \mathcal{P}_{\text{ok}}$. This interim reduction of the player set is similar to the player elimination framework of [HMP00], but opposed to this, can also be applied to settings with $t < n/2$.

For the sake of easier presentation, we use a vector notation: We denote the triples by $(\vec{A}, \vec{B}, \vec{C})$ and the random values by $\vec{R}$. Furthermore, we extend

[5] To be used in the non-malleable zero-knowledge proofs (see [CDN01]).

all operators on group elements also to vectors of group elements, where the semantics is component-wise application of the operator.

Prepare c_M random ciphertexts $\vec{A}$. We first present a protocol to generate a single random encryption A, and will then extend it to generate c_M random ciphertexts $\vec{A}$ at once. The protocol proceeds as follows:

1. Every player $P_i \in \mathcal{P}$ selects at random $a_i \in \mathbb{M}$ and computes an encryption $A_i = \mathbb{E}(a_i)$.
2. Every player $P_i \in \mathcal{P}$ sends A_i to every player $P_j \in \mathcal{P}$, and proves to P_j interactively that he knows the plaintext of A_i.
3. Every player P_i broadcasts the hash value $h_i = H(A_i)$ among all players in $\mathcal{P}$, where H denotes the collision-resistant hash function defined in the setup phase.
4. Initially we set the set of mutually agreeing players to $\mathcal{P}_{\text{ok}} = \mathcal{P}$. Then, in sequence, every player $P_j \in \mathcal{P}_{\text{ok}}$ verifies for every player $P_i \in \mathcal{P}_{\text{ok}}$ whether
   - the broadcast hash value h_i matches the received encryption A_i, i.e., $h_i \stackrel{?}{=} H(A_i)$, and
   - the bilateral interactive proof by P_i is accepting for P_j.

   If P_j's verifications succeed for all players $P_i \in \mathcal{P}_{\text{ok}}$, then P_j broadcasts $\perp$ to confirm so. Otherwise, P_j picks the index i of some player $P_i \in \mathcal{P}_{\text{ok}}$ that failed in P_j's verification, and broadcasts i. In the latter case, both players P_i and P_j are removed from the set $\mathcal{P}_{\text{ok}}$ of agreeing players, i.e., all players set $\mathcal{P}_{\text{ok}} \leftarrow \mathcal{P}_{\text{ok}} \setminus \{P_i, P_j\}$.
5. Every player $P_j \in \mathcal{P}_{\text{ok}}$ sets $A = \bigoplus_{P_i \in \mathcal{P}_{\text{ok}}} A_i$ and sends it to every $P_i \in \mathcal{P} \setminus \mathcal{P}_{\text{ok}}$.
6. Every player $P_i \in \mathcal{P} \setminus \mathcal{P}_{\text{ok}}$ sets A as the majority of received values by players in $\mathcal{P}_{\text{ok}}$.

We first argue that at the end of the protocol, all players in $\mathcal{P}$ hold the same encryption A, and then, that the plaintext of A is unknown to the adversary. One can easily verify that all honest players in $\mathcal{P}_{\text{ok}}$ compute the same value A (otherwise they hold a collision of H). Furthermore, the majority of players in $\mathcal{P}_{\text{ok}}$ is honest (at least half of the removed players $\mathcal{P} \setminus \mathcal{P}_{\text{ok}}$ is corrupted), hence in Step 5, the majority of players $P_j \in \mathcal{P}_{\text{ok}}$ distributes the correct value A, and all players in $\mathcal{P}$ will decide for the same value A. In order to argue about the secrecy of the plaintext of A, observe that at least one player in $\mathcal{P}_{\text{ok}}$ is honest and chooses a_i uniformly at random. Since the encryption scheme is semantically secure[6] and the proof of plaintext knowledge for a_i is zero-knowledge, the protocol reveals zero knowledge about a_i to the corrupted parties.[7] Since all (corrupted) parties

[6] Notice that the fact that the decryption key is shared between the parties is no problem for the semantic security as the adversary can inspect at most t parties; Since the decryption key is shared with threshold t, the t shares known by the adversary gives zero knowledge about the decryption key.

[7] Here we colloquially distinguish between information and knowledge. Since A_i determines a_i clearly the adversary has full information about a_i. However, by the

$P_j \in \mathcal{P}_{\text{ok}}$ gave a non-malleable proof of plaintext knowledge of *their* contribution a_j, and this proof was accepted by all parties in $\mathcal{P}_{\text{ok}}$ (at least one of them being honest), their shares a_j are independent of the share a_i. It follows that A is an encryption of a uniformly random value $a = \sum_{i \in \mathcal{P}_{\text{ok}}} a_i$ of which the adversary has zero knowledge. This informal sketch of the security can be turned into a formal simulation proof using known proof techniques, see e.g. [CDN01,DN03].

In order to generate c_M random ciphertexts $\vec{A}$, the above protocol is slightly modified:

1. Every player $P_i \in \mathcal{P}$ selects at random $\vec{a_i} \in \mathbb{M}^{c_M}$ and computes its component-wise ciphertexts $\vec{A}_i$.
2. Every player $P_i \in \mathcal{P}$ sends $\vec{A}_i$ to every player $P_j \in \mathcal{P}$, and proves to P_j interactively that he knows the plaintext of each component of $\vec{A}_i$.
3. Every player P_i broadcasts the hash value $h_i = H(\vec{A}_i)$ among all players in $\mathcal{P}$.
4. Set $\mathcal{P}_{\text{ok}} = \mathcal{P}$ and, in sequence, every player $P_j \in \mathcal{P}_{\text{ok}}$ verifies for every player $P_i \in \mathcal{P}_{\text{ok}}$ whether
   - the broadcast hash value h_i matches the received ciphertexts $\vec{A}_i$, i.e., $h_i \stackrel{?}{=} H(\vec{A}_i)$, and
   - all the bilateral interactive proofs by P_i are accepting for P_j.

   If P_j's verifications succeed for all players $P_i \in \mathcal{P}_{\text{ok}}$, then P_j broadcasts $\perp$ to confirm so. Otherwise, P_j picks the index i of some player $P_i \in \mathcal{P}_{\text{ok}}$ that failed in P_j's verification, and broadcasts i. In the latter case, both players P_i and P_j are removed from the set of agreeing players, i.e., all players set $\mathcal{P}_{\text{ok}} \leftarrow \mathcal{P}_{\text{ok}} \setminus \{P_i, P_j\}$.
5. Every player $P_j \in \mathcal{P}_{\text{ok}}$ sets $\vec{A} = \bigoplus_{P_i \in \mathcal{P}_{\text{ok}}} \vec{A}_i$ and sends it to every $P_i \in \mathcal{P} \setminus \mathcal{P}_{\text{ok}}$.
6. Every player $P_i \in \mathcal{P} \setminus \mathcal{P}_{\text{ok}}$ sets $\vec{A}$ as the majority of received vectors by players in $\mathcal{P}_{\text{ok}}$.

The security of this protocol follows immediately from the security of the previous protocol. The communication complexity of the protocol is $\mathcal{O}(c_M n^2 \kappa + n\mathcal{B}(\kappa))$ bits.

Prepare random ciphertexts $\vec{B}$ and products $\vec{C}$. The B and C values of the triples are generated similarly to the A values. For the sake of simplicity, we present solely the protocol for generating a single triple. The generalization to vectors of triples is straight-forward along the lines of the protocol for generating $\vec{A}$.

1. Every player $P_i \in \mathcal{P}$ selects at random $b_i \in \mathbb{M}$, computes $B_i = \mathbb{E}(b_i)$ and $C_i = \mathbb{R}(b_i A)$.

---

semantic security and the fact that the adversary is polynomial time bounded, it has zero *knowledge* about a_i.

2. Every player $P_i \in \mathcal{P}$ sends B_i and C_i to every player $P_j \in \mathcal{P}$, and proves to P_j interactively that he knows the plaintext b_i of B_i, and that C_i is a randomization of $b_i A$.
3. Every player P_i broadcasts the hash value $h_i = H(B_i, C_i)$ among all players in $\mathcal{P}$.
4. Set $\mathcal{P}_{\text{ok}} = \mathcal{P}$ and, in sequence, every player $P_j \in \mathcal{P}_{\text{ok}}$ verifies for every player $P_i \in \mathcal{P}_{\text{ok}}$ whether
   - the broadcast hash value h_i matches the received ciphertexts (B_i, C_i), i.e., $h_i \stackrel{?}{=} H(B_i, C_i)$, and
   - all the bilateral interactive proofs by P_i are accepting for P_j.

   If P_j's verifications succeed for all players $P_i \in \mathcal{P}_{\text{ok}}$, then P_j broadcasts $\perp$ to confirm so. Otherwise, P_j picks the index i of some player $P_i \in \mathcal{P}_{\text{ok}}$ that failed in P_j's verification, and broadcasts i. In the latter case, both players P_i and P_j are removed from the set of agreeing player, i.e., all players set $\mathcal{P}_{\text{ok}} \leftarrow \mathcal{P}_{\text{ok}} \setminus \{P_i, P_j\}$.
5. Every player $P_j \in \mathcal{P}_{\text{ok}}$ sets $B = \bigoplus_{P_i \in \mathcal{P}_{\text{ok}}} B_i$, and $C = \bigoplus_{P_i \in \mathcal{P}_{\text{ok}}} C_i$, and sends them to every $P_i \in \mathcal{P} \setminus \mathcal{P}_{\text{ok}}$.
6. Every player $P_i \in \mathcal{P} \setminus \mathcal{P}_{\text{ok}}$ sets B and C to be the majority of received values from players in $\mathcal{P}_{\text{ok}}$.

The correctness of the resulting triple (A, B, C) follows directly from the distributive law in groups. The security of the protocol can be argued along the lines of the proof of the previous protocol.

The above protocol can be extended to vector-values in a straight-forward manner. The communication complexity of the extended protocol is $\mathcal{O}(c_M n^2 \kappa + n\mathcal{B}(\kappa))$ bits.

Prepare c_R random values $\vec{R}$. The random $\vec{R}$ vector is prepared exactly as the random $\vec{A}$ vector, only the corresponding $\vec{B}$ and $\vec{C}$ vectors are not generated.

3.4 Evaluation phase

In the evaluation phase, the circuit is evaluated layer by layer. In the following, we give the protocols for evaluating the different types of gates.

Input gates. When a party P_i is to provide an input for some gate G, the parties proceed as follows:

1. P_i computes $V_i = \mathbb{E}(v_i)$ broadcasts V_i.
2. P_i bilaterally proves (in zero-knowledge) knowledge of plaintext v_i to every player $P_j \in \mathcal{P}$.
3. Each $P_j \in \mathcal{P}$, lets $b_j = 1$ if the proof from P_i was accepted and lets $b_j = 0$ otherwise.
4. The parties in $\mathcal{P}$ run a BA with input b_j from P_j. Let the output be $b \in \{0, 1\}$.

5. If $b = 1$, then each $P_j \in \mathcal{P}$ sets the encryption for gate G to be the broadcast value V_i; Otherwise, P_j sets the encryption for gate G to be $E(0, e)$, where 0 and e denotes the neutral elements from $\mathbb{M}$ respectively $\mathbb{R}$.

After this protocol the input gate is defined to the same value by all parties. The proof of knowledge given by P_i serves the purpose of guaranteeing independence of inputs. The privacy of the protocol follows from the semantic security of the encryption scheme, using that the proofs are zero-knowledge.

Using that the communication complexity of one zero-knowledge proof is $\mathcal{O}(\kappa)$, the communication complexity for giving one input is seen to be $\mathcal{O}(\mathcal{B}(\kappa) + n\kappa + \mathcal{B}(1))$. Assuming that $\mathcal{B}(\kappa) \geq n\kappa$, this is $\mathcal{O}(\mathcal{B}(\kappa))$.

Output gates. When the value of some gate G (with associated ciphertext M) is to be revealed towards a party P_j, the parties proceed as follows:

1. Every player $P_i \in \mathcal{P}$ computes $m_i = SD_{z_i}(M)$ and sends it to P_j.
2. Every player $P_i \in \mathcal{P}$ gives a zero-knowledge proof to every other party P_j that m_i is a correct i'th decryption share.
3. P_j collects $t + 1$ decryption shares for which the proof of correct decryption share succeeded and combine them to obtain $m = D(M)$.

Since at least $t + 1$ parties are honest, P_j will be able to collect $t + 1$ shares where the proof succeeded. By the soundness of the zero-knowledge proof all collected shares will be correct, except with negligible probability. By the way the values $(z_1, \ldots, z_n)$ were set up and the requirements on the share combining algorithm have that indeed $m = D_z(M)$.

The privacy of the protocol follows from the requirements on the threshold decryption protocol: from the result of the protocol and the key shares of the t corrupted parties, the adversary could compute the key shares of the honest parties on its own. Therefore the protocol leaks zero knowledge about the key shares of the honest parties.

The communication complexity is seen to be $\mathcal{O}(n\kappa)$ per output gate and party to learn the output. If all parties are to learn the output, the communication complexity is $\mathcal{O}(n^2\kappa)$ per output gate.

If only one party is to learn the output and the output should be private, the decryption shares sent to P_j should be sent over private channels. This does not affect the order of the communication complexity.

Addition gates. For an addition gate G where the input gates of G has associated ciphertexts M_1 and M_2, the associated ciphertext of G is set to be $M_G = M_1 \oplus M_2$. As the $\oplus$-operator is deterministic, all parties agree on the encryption M_G, and by the homomorphic properties of $\oplus$ it holds that $D(M_G) = D(M_1) + D(M_2)$.

Multiplication gates. For a multiplication gate G where the two input gates have associated ciphertexts M_1 and M_2, the associated ciphertext M_G of G is computed as follows:

1. Every party $P_i \in \mathcal{P}$ picks the prepared triple (A, B, C) that is associated with the gate.
2. Every party $P_i \in \mathcal{P}$ computes $D = A \oplus M_1$ and $E = B \oplus M_2$.
3. Every party $P_i \in \mathcal{P}$ invokes the decryption protocol from Section 3.4 on D and E. Denote the results by d respectively e.
4. Every party sets $M_G = (eM_1) \ominus (dB) \oplus C$.

The above way to use a prepared triple is from [Bea91a].

We argue that the protocol maintains agreement on the associated ciphertexts. Assume that the parties agree on M_1 and M_2. By the fact that $\oplus$ is a function, the parties will agree on D and E. Therefore the decryption protocol will return correct and consistent d and e values to the parties. Using that $\ominus$ and $\oplus$ are functions it then follows that the parties will agree on M_G.

We then argue the correctness of the protocol. By the correctness of the decryption protocol and the homomorphic properties of $\oplus$ and $\ominus$ we have that $D(M_G) = em_1 - db + c = (b+m_2)m_1 - (a+m_1)b + ab = m_1 m_2$, where $m_1 = D(M_1)$ and $m_2 = D(m_2)$.

For the privacy, the only values that are revealed are d and e. However, since a and b are independent, uniformly random elements from $\mathbb{M}$ unknown to any adversary which inspects at most t parties, it follows that d and e are uniformly random and independent of m_1 and m_2 in the view of the adversary. Therefore the protocol leaks zero knowledge about m_1 and m_2.

The communication complexity per gate is that of two invocations of the decryption protocol, i.e. $\mathcal{O}(n^2\kappa)$.

Randomizing gates. When the circuit is evaluated, the randomizing gates should be initialized by uniformly random values. To reflect the ideal evaluation the random values used for initialization should be unknown to all parties. Therefore, to every random gate, one random encrypted value $R^{(i)}$ is associated.

3.5 Complexity analysis

In this section we consider the complexity of the active-secure protocol. Summing the complexities stated in the presentation of the protocol gives us a total complexity of $\mathcal{O}(((c_M + c_R)n^2\kappa + n\mathcal{B}(\kappa)) + c_I\mathcal{B}(\kappa) + c_O n^2\kappa + c_M n^2\kappa)$, where c_M denotes the number of multiplication gates, c_R denotes the number of randomizing input gates, c_I denotes the number of input gates, and c_O denotes the number of output gates. This is seen to be $\mathcal{O}((c_M + c_R + c_O)n^2\kappa + n\mathcal{B}(\kappa) + c_I\mathcal{B}(\kappa))$.

In the synchronous model with $t < n/2$, broadcasting (and/or doing BA on) a total of ℓ bits can be done with complexity $\mathcal{O}(n^2\ell + n^3\kappa)$ under the strong RSA assumption and the assumption the RSA signatures are secure (c.f. [Nie03]). We have $n + c_I$ broadcasts of κ-bit messages, giving $\ell = (n + c_I)\kappa$ and (a bit informally) $n\mathcal{B}(\kappa) + c_I\mathcal{B}(\kappa) = \mathcal{O}(n^2(n + c_I)\kappa + n^3\kappa) = \mathcal{O}(c_I n^2\kappa + n^3\kappa)$. This immediately gives us the bound $\mathcal{O}((c_M + c_R + c_O + c_I)n^2\kappa + n^3\kappa)$ on the communication complexity of the overall protocol.

Theorem 1. *Under the QR assumption (or the DCR assumption), the strong RSA assumption and the assumption that RSA signatures are secure, $\mathcal{O}(cn^2\kappa)$ is an upper bound on the communication complexity of an active-secure protocol with resilience $t < n/2$ for evaluating an n-party function with arithmetic circuit complexity $c \geq n$.*

3.6 Ongoing computations

The result for active security assumes that the size of the circuit is known before the computation starts, to allow for a preparation phase. For an on-going reactive computation, even the circuit might be specified as the computation unfolds and in particular the length of the computation might not be specified on beforehand. Our result can be extended to such a setting. We simply hold a pool of prepared triples, and each time it dries out we prepare at least twice as many triples as last time. After polynomially many activations, this gives a maximum of $\mathcal{O}(\log(\kappa))$ runs of the preparation phase and prepares at most twice as many triples as needed. This gives the bound $\mathcal{O}(cn^2\kappa + n^3\kappa\log(\kappa))$.

3.7 Boolean circuits

The proposed protocol evaluates a circuit of arithmetic gates, where the underlying ring is the message space of the encryption scheme. We can extend the protocol to evaluate a Boolean circuit, even when the message space of the encryption scheme is larger (e.g., when using Paillier encryption). In the sequel, we present the necessary modifications for Boolean circuits over AND and NOT gates. The protocol for Boolean circuits has the same communication complexity as the protocol for arithmetic circuits.

Input gates. In the input protocol, the player providing input must prove that the input is in $\{0, 1\}$. Therefore, the zero-knowledge proof for proving plaintext knowledge is augmented by a zero-knowledge proof for proving that the plaintext is either 0 or 1.

AND-gates. As it is guaranteed that all wires are encryptions of either 0 or 1, AND-gates can be realized as multiplication gates.

NOT-gates. A NOT-gates can be computed by using the homomorphism of the encryption scheme. Given an encrypted bit B, its negation can be computed as $\mathbb{E}(1) \ominus B$. Every player can compute the encrypted value of a negation gate locally, without communicating with other players.

Randomizing gates. It must also be ensured that the output of randomizing gates are in $\{0, 1\}$. If $M > 2$ (as is the case for Paillier's cryptosystem), and we want to stay within the new upper bound, a new protocol is needed for this.

0. Let $\vec{R}^{(0)} = \mathbb{E}(\vec{0}, \vec{e})$ be a constant vector of length c_R, where each element is the constant encryption $\mathbb{E}(0, e)$. Let $\mathcal{P}_{\text{ok}} = \mathcal{P}$, let $\mathcal{P}_{\text{done}} = \emptyset$, let $i_{\text{prev}} = 0$, let $i_{\text{next}} = 1$ and let Prev be an empty stack.
1. $P_{i_{\text{next}}}$ computes $\vec{R}^{(i_{\text{next}})}$ from $\vec{R}^{(i_{\text{prev}})}$ as follows: For each element $R^{(i_{\text{prev}})}$ in $\vec{R}^{(i_{\text{prev}})}$, pick $\alpha \in_R \mathbb{R}$ and $b \in_R \{0,1\}$ and, if $b = 0$, let $R^{(i_{\text{next}})} = \mathbb{E}(0, \alpha) \oplus R^{(i_{\text{prev}})}$, and if $b = 1$, let $\vec{R}^{(i_{\text{next}})} = \mathbb{E}(1, \alpha) \ominus R^{(i_{\text{prev}})}$.
2. $P_{i_{\text{next}}}$ broadcasts the hash value $h_i = H(\vec{R}^{(i_{\text{next}})})$ among all players in $\mathcal{P}$.
3. $P_{i_{\text{next}}}$ sends $\vec{R}^{(i_{\text{next}})}$ to every player $P_j \in \mathcal{P}$, and gives to P_j (for each element $R^{(i_{\text{prev}})}$) a non-malleable zero-knowledge proof of knowledge of α for which *either* $R^{(i_{\text{next}})} = \mathbb{E}(0, \alpha) \oplus R^{(i_{\text{prev}})}$ *or* $R^{(i_{\text{next}})} = \mathbb{E}(1, \alpha) \ominus R^{(i_{\text{prev}})}$.
4. The parties $\mathcal{P}$ enter a BA on whether to accept the proofs given by $P_{i_{\text{next}}}$: Each party $P_j \in \mathcal{P}$ enters with $b_j = 1$ iff in the above step it received $\vec{R}^{(i_{\text{next}})}$ such that $h_i = H(\vec{R}^{(i_{\text{next}})})$ and the bilateral proof from $P_{i_{\text{next}}}$ to P_j was accepted.
5. – If the outcome of the BA is $b = 0$, then all parties in $\mathcal{P}$ set $\mathcal{P}_{\text{ok}} = \mathcal{P}_{\text{ok}} \setminus \{i_{\text{next}}\}$ and set i_{next} to be the smallest $i \in \mathcal{P}_{\text{ok}} \setminus \mathcal{P}_{\text{done}}$.
   – If the outcome of the BA is $b = 1$, then all parties in $\mathcal{P}$ set $\mathcal{P}_{\text{done}} = \mathcal{P}_{\text{done}} \cup \{i_{\text{next}}\}$, push i_{prev} on Prev, let $i_{\text{prev}} = i_{\text{next}}$ and set i_{next} to be the smallest $i \in \mathcal{P}_{\text{ok}} \setminus \mathcal{P}_{\text{done}}$.

   In both cases, if $\mathcal{P}_{\text{ok}} \setminus \mathcal{P}_{\text{done}} = \emptyset$, then go to Step 8.
6. The party $P_{i_{\text{next}}}$ broadcasts a bit $b \in \{0,1\}$, where $b = 0$ iff $i_{\text{prev}} \neq 0$ and $P_{i_{\text{next}}}$ never received $\vec{R}^{(i_{\text{prev}})}$ such that $h_{i_{\text{prev}}} = H(\vec{R}^{(i_{\text{prev}})})$ (in Step 3).
7. – If $i_{\text{prev}} = 0$ or $P_{i_{\text{next}}}$ broadcast 1, then all parties in $\mathcal{P}$ go to Step 1.
   – If $i_{\text{prev}} \neq 0$ and $P_{i_{\text{next}}}$ broadcast 0, then all parties set $\mathcal{P}_{\text{ok}} = \mathcal{P}_{\text{ok}} \setminus \{i_{\text{prev}}, i_{\text{next}}\}$. Then i_{prev} is set to be the top of Prev (which is then popped) and i_{next} is set to be the smallest $i \in \mathcal{P}_{\text{ok}} \setminus \mathcal{P}_{\text{done}}$ (if $\mathcal{P}_{\text{ok}} \setminus \mathcal{P}_{\text{done}} = \emptyset$, then go to Step 8.) Then all parties in $\mathcal{P}$ go to Step 6.
8. All parties in $\mathcal{P}$ which knows $\vec{R}^{(i_{\text{prev}})}$ such that $h_{i_{\text{prev}}} = H(\vec{R}^{(i_{\text{prev}})})$ sends $\vec{R}^{(i_{\text{prev}})}$ to all parties.
9. All parties in $\mathcal{P}$ waits for a value $\vec{R}^{(i_{\text{prev}})}$ for which $h_{i_{\text{prev}}} = H(\vec{R}^{(i_{\text{prev}})})$ to arrive and outputs $\vec{R}^{(i_{\text{prev}})}$.

We first argue termination and agreement: It is straight-forward to verify that the procedure reaches Step 8. Since at this point $P_{i_{\text{prev}}}$ at some point broadcast $h_{i_{\text{prev}}}$ and had its proof accepted by a majority of the parties in $\mathcal{P}$, at least one honest party must have received $\vec{R}^{(i_{\text{prev}})}$ such that $h_{i_{\text{prev}}} = H(\vec{R}^{(i_{\text{prev}})})$. At least that party will echo $\vec{R}^{(i_{\text{prev}})}$ in Step 8 and thus all parties will terminate in Step 9. Since $h_{i_{\text{prev}}}$ is a broadcast value, all parties will output the same value $\vec{R}^{(i_{\text{prev}})}$ unless a collision under H is found.

We then argue that $\vec{R}^{(i_{\text{prev}})}$ is a vector of encryptions of random bits of which the adversary has zero knowledge. At termination we clearly have that $\mathcal{P}_{\text{ok}} \subseteq \mathcal{P}_{\text{done}}$. Furthermore, at termination $\mathcal{P}_{\text{ok}}$ will contain a majority of honest

parties and there exists a sequence $i_0 = 0 < i_1 < \cdots < i_{l-1} < i_l \leq n$ such that $\mathcal{P}_{\text{ok}} = \{i_1, \ldots, i_l\}$ and for $m = 1, \ldots, l$, the vector $\vec{R}^{(i_m)}$ was computed by P_{i_m} from $\vec{R}^{(i_{m-1})}$ as specified in Step 1. Since the proof of knowledge ensures that each party "flips" the encryptions independently and at least one party in $\mathcal{P}_{\text{ok}}$ is honest it follows that $\vec{R}^{(i_l)}$ is a vector of encryptions of independent random bits unknown to the adversary.

Each party broadcasts (at most) κ bits in Step 2 and one bit in Step 6. Besides this n BAs are executed and each party $P_{i_{\text{next}}}$ sends the vector $\vec{R}^{(i_{\text{next}})}$ to all parties and gives the non-malleable zero-knowledge proofs of knowledge in Step 3. Assuming that $\mathcal{B}(k)$ dominates the cost of one Byzantine agreement, the total communication complexity of this is $\mathcal{O}(c_R n^2 \kappa + n\mathcal{B}(\kappa))$, as desired.

The above protocol can be seen as a strengthening of the protocol used in the original preparation phase to deal with large values being build sequentially from large contributions from all parties. Similar protocols can be used to prepare c gates for the Mix-and-Match protocol in [JJ00] with complexity $\mathcal{O}(cn^2\kappa + n\mathcal{B}(\kappa))$ and for mixing c ciphertext in anonymizing networks and voting (with n servers) with complexity $\mathcal{O}(cn^2\kappa + n\mathcal{B}(\kappa))$. In both cases an optimization over $\Theta(cn\mathcal{B}(\kappa)) = \Theta(cn^3\kappa)$.

4 Passive-secure MPC protocol for $t < n$

In this section we present an upper bound on the communication complexity of a passive secure MPC protocol. Again the upper bound is given by a protocol. As opposed to the active secure protocol, the passive protocol is not based on novel technical contributions but rather a neat observation.

The essential observation is that from the threshold homomorphic encryption based MPC protocol of [CDN01] each gate has a short publicly known representation, namely the associated encryption. This is opposed to e.g. secret sharing based protocols, where the representation is exactly *shared* among the parties and therefore inherently large ($\Theta(n\kappa)$). This observation allows to designate some party P_{king} which drives the protocol and evaluates the circuit gate by gate, with help of the other parties.

The protocol proceeds along the lines of the active protocol, though no preparation phase is needed anymore. The details are given below.

Setup phase. In the setup phase the setup function s generates a random key pair (Z, z), splits z into $(z_1, \ldots, z_n)$ with threshold $t = n - 1$, sets $p = Z$ and sets $s_i = z_i$ for $i = 1, \ldots, n$. Furthermore one designated party P_{king} is chosen, called the king, e.g. $P_{\text{king}} = P_1$.

Input gates. When a party P_i is to provide the input $v_i \in \mathbb{M}$, the parties proceed as follows:

1. P_i selects $\alpha_i \in_R \mathbb{R}$, computes and sends $V_i = E(v_i, \alpha_i)$ to P_{king}.
2. P_{king} sends V_i to all parties.

The privacy of the protocol follows from the semantic security of the encryption scheme.

Output gates. The value of some gate G with associated ciphertext M is revealed as follows:

1. Every party P_i computes and sends $m_i = SD_{z_i}(M)$ to P_{king}.
2. P_{king} computes $m = C(m_1, \ldots, m_n)$ and sends it to all parties.

The security of this protocol is argued along the lines of the active-secure protocol. The communication complexity is $\mathcal{O}(n\kappa)$.

If the value is to be revealed privately to only one party P_j, then the parties send their decryption shares m_i privately to P_j, who computes $m = C(m_1, \ldots, m_n)$.

Addition gates. The king computes the value of addition gates using the homomorphism of the encryption scheme.

Multiplication gates. For a multiplication gate G where the two input gates have associated ciphertexts M_1 and M_2, the associated ciphertext M_G of G is computed as follows:

1. Every party $P_i \in \mathcal{P}$ selects $a_i \in_R \mathbb{M}$, $\alpha_i, \beta_i \in_R \mathbb{R}$, computes $A_i = \mathbb{E}(a_i, \alpha_i)$ and $C_i = \mathbb{R}(a_i M_2, \beta_i)$, and sends A_i and C_i to P_{king}.
2. P_{king} computes $A = M_1 \bigoplus_{P_i \in \mathcal{P}} A_i$ and $C = \bigoplus_{P_i \in \mathcal{P}} C_i$ and sends A and C to all parties,
3. Every party $P_i \in \mathcal{P}$ computes its decryption share $a_i = SD_{z_i}(A)$ and sends it to P_{king}.
4. P_{king} decrypts $a = C(a_1, \ldots, a_n)$, computes $G_M = aM_2 \ominus C$ and send it to all parties.

The security is argued as for the active-secure protocol. The communication complexity is $\mathcal{O}(n\kappa)$.

Randomizing gates. An encryption of a random value m, unknown to the adversary, is computed as follows:

1. Every party $P_i \in \mathcal{P}$ selects $a_i \in \mathbb{M}$, $\alpha_i \in \mathbb{R}$, computes $A_i = \mathbb{E}(a_i, \alpha_i)$ and sends it to P_{king}.
2. P_{king} computes $A = \bigoplus_{P_i \in \mathcal{P}} A_i$ and sends it to all parties.

Complexity analysis. It is straight forward to verify that the total number of bits sent by the parties is $\mathcal{O}((c_I + c_M + c_O + c_R)n\kappa)$.

Theorem 2. *Under the QR assumption (or the DCR assumption), $\mathcal{O}(cn\kappa)$ is an upper bound on the communication complexity of a passive secure protocol with resilience $n-1$ for evaluating an n-party randomized function with arithmetic circuit complexity c.*

5 Conclusions and open problems

We presented new upper bounds on the communication complexity of optimally resilient active-secure MPC and optimally resilient passive-secure MPC. In both cases we improved the previously best bounds by a factor n. The improvement of the bound for active security was based on a combination of previous techniques for efficient MPC along with several novel technical contributions, as opposed to the improvement of the bound for passive security, which was based on a simple observation.

Our bounds were based either on the DCR assumption or on the QR assumption (in both cases requiring, additionally the strong RSA assumption and the assumption that RSA signatures are secure for active security). Even though these assumptions are standard assumptions, they are very specific. It is an interesting open problem to achieve the same bounds under general assumptions, as e.g. the existence of one-way functions. One approach would be to investigate the efficiency of active-secure information-theoretic MPC with $t < n/2$. It is known that the player elimination framework does not apply to this threshold [HMP00,HM01]. The ideas presented here might however allow to obtain similar results in this model. The new upper bound for passive security however seems very challenging to obtain under general assumptions.

It is an interesting open problem to obtain the new bound for also adaptive security. In [DN03] an adaptively secure version of the protocol from [CDN01] was presented. However, the techniques from [DN03] do not allow to make our protocol here adaptive secure while staying within the bound $\mathcal{O}(cn^2\kappa + n^3\kappa)$. We stress that although our protocol cannot be proven adaptively secure (we cannot construct a simulator), there is no obvious way for an adaptive adversary to violate the correctness or the security of the computation. This is in contrast to some folklore trick for improving efficiency, namely to have the players agree on a small random subset of players, who then perform the whole protocol.[8] In this approach, an adaptive adversary can trivially violate both privacy and correctness of the protocol, simply by corrupting the majority (or even all) of the players in the subset, once this is randomly chosen.

Another interesting open problem is to prove non-trivial lower bounds on the communication complexity of secure MPC.

6 Acknowledgments

We would like to thank Ivan Damgård, Serge Fehr and Matthias Fitzi for many fruitful discussions, and the anonymous referees for their helpful comments.

[8] Note that is is even unclear how this subset is to be chosen such that it contains an honest majority, given that the original set of players satisfies the optimal bound $t < n/2$. Furthermore, the trick only works if n is large.

References

[BB89] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *PODC'89*, pp. 201–209, 1989.

[BCG93] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation (extended abstract). In *25th STOC*, pp. 52–61, 1993.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for publickey encryption schemes. In H. Krawczyk, ed., *Crypto '98*, pp. 26–45, 1998. LNCS 1462.

[Bea91a] D. Beaver. Efficient multiparty protocols using circuit randomization. In J. Feigenbaum, ed., *Crypto '91*, pp. 420–432, 1991. LNCS 576.

[Bea91b] D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[BFKR90] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead (extended abstract). In A. J. Menezes and S. A. Vanstone, ed., *Crypto '90*, pp. 62–76, 1990. LNCS 537.

[BGP92] P. Berman, J. A. Garay, and K. J. Perry. Optimal early stopping in distributed consensus. In *Proceedings of the sixth International Workshop on Distributed Algorithms*, pp. 221–237, 1992.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th STOC*, pp. 1–10, 1988.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *22nd STOC*, pp. 503–513, 1990.

[Can00] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, winter 2000.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, 2001.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th STOC*, pp. 11–19, 1988.

[CDD+99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In J. Stern, ed., *EuroCrypt '99*, pp. 311–326, 1999. LNCS 1592.

[CDD00] R. Cramer, I. Damgård, and S. Dziembowski. On the complexity of verifiable secret sharing and multiparty computation. In *22nd STOC*, pp. 325–334, 2000.

[CDG87] D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In C. Pomerance, ed., *Crypto '87*, pp. 87–119, 1987. LNCS 293.

[CDM00] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In B. Preneel, ed., *EuroCrypt 2000*, pp. 316–334, 2000. LNCS 1807.

[CDN01] R. Cramer, I. Damgaard, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EuroCrypt 2001*, pp. 280–300, 2001. LNCS 2045.

[Cra96] R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols.* PhD thesis, CWI and University of Amsterdam, 1996.

[CW92] B. A. Coan and J. L. Welch. Modular construction of a byzantine agreement protocol with optimal message complexity. *Information and Computation*, 97(1):61–85, March 1992.

[Dam00] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In B. Preneel, ed., *EuroCrypt 2000*, pp. 418–430, 2000. LNCS 1807.

[DCIO98] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *30th STOC*, pp. 141–150, 1998.

[DG02] I. Damgaard and J. Groth. Non-interactive and reusable non-maleable commitment schemes. In *34th STOC*, 2002.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, ed., *4th Public Key Cryptography*, pp. 110–136, 2001. LNCS 1992.

[DN03] I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In D. Boneh, ed., *Crypto 2003*, 2003. LNCS 2729.

[FF00] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In M. Bellare, ed., *Crypto 2000*, pp. 414–432, 2000. LNCS 1880.

[FH96] M. Franklin and S. Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, Autumn 1996.

[FPS00] P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Proceedings of Financial Crypto 2000*, 2000.

[GIKR02] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-round secure multiparty computation. In M. Yung, ed., *Crypto 2002*, pp. 178–193, 2002. LNCS 2442.

[CDI05] R. Cramer, I. Damgård, and Y. Ishai. Local conversion of secret-sharing schemes with applications to threshold cryptography. In *TCC 2005*, p.342–362, 2005. LNCS 3378.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In A. J. Menezes and S. A. Vanstone, ed., *Crypto '90*, pp. 77–93, 1990. LNCS 537.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th STOC*, pp. 291–304, 1985.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *19th STOC*, pp. 218–229, 1987.

[GRR98] R. Gennaro, M. Rabin, and T. Rabin. Simplified VSS and fast-track multi-party computations with applications to threshold cryptography. In *PODC'98*, 1998.

[GV87] O. Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In C. Pomerance, ed., *Crypto '87*, pp. 73–86, 1987. LNCS 293.

[HM00] M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, winter 2000.

[HM01] M. Hirt and U. Maurer. Robustness for free in unconditional multi-party computation. In J. Kilian, ed., *Crypto 2001*, pp. 101–118, 2001. LNCS 2139.

[HMP00] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In T. Okamoto, ed., *ASIACRYPT 2000*, pp. 143–161, 2000. LNCS 1976.

[HN05] M. Hirt and J. B. Nielsen. Upper bounds on the communication complexity of optimally resilient cryptographic multiparty computation. In B. Roy, ed., *ASIACRYPT 2005*, pp. 79–99, 2005. LNCS 3788.

[JJ00] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, ed., *ASIACRYPT 2000*, pp. 162–177, 2000. LNCS 1976.

[KY02] J. Katz and M. Yung. Threshold cryptosystems based on factoring. In Y. Zheng, ed., *ASIACRYPT 2002*, pp. 192–205, 2002. LNCS 2501.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):381–401, July 1982.

[MR91] S. Micali and P. Rogaway. Secure computation. In J. Feigenbaum, ed., *Crypto '91*, pp. 392–404, 1991. LNCS 576.

[Nie03] J. B. Nielsen. On protocol security in the cryptographic model. Dissertation Series DS-03-8, BRICS, Department of Computer Science, University of Aarhus, August 2003.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In J. Stern, ed., *EuroCrypt '99*, pp. 223–238, 1999. LNCS 1592.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *21th STOC*, pp. 73–85, 1989.

[Yao82] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pp. 160–164, 1982.