Multiparty Computation from Threshold Homomorphic Encryption

Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen

BRICS[⋆] Department of Computer Science
University of Århus
Ny Munkegade
DK-8000 Arhus C, Denmark
{cramer,ivan,buus}@brics.dk

Abstract. We introduce a new approach to multiparty computation (MPC) basing it on homomorphic threshold crypto-systems. We show that given keys for any sufficiently efficient system of this type, general MPC protocols for n parties can be devised which are secure against an active adversary that corrupts any minority of the parties. The total number of bits broadcast is $O(nk|C|)$, where k is the security parameter and $|C|$ is the size of a (Boolean) circuit computing the function to be securely evaluated. An earlier proposal by Franklin and Haber with the same complexity was only secure for passive adversaries, while all earlier protocols with active security had complexity at least quadratic in n. We give two examples of threshold cryptosystems that can support our construction and lead to the claimed complexities.

1 Introduction

The problem of multiparty computation (MPC) dates back to the papers by Yao [Yao82] and Goldreich et al. [GMW87]. What was proved there was basically that a collection of n parties can efficiently compute the value of an n-input function, such that everyone learns the correct result, but no other new information. More precisely, these protocols can be proved secure against a polynomial time bounded adversary who can *corrupt* a set of less than $n/2$ parties initially, and then make them behave as he likes, we say that the adversary is *active*. Even so, the adversary should not be able to prevent the correct result from being computed and should learn nothing more than the result and the inputs of corrupted parties. Because the set of corrupted parties is fixed from the start, such an adversary is called *static* or non-adaptive.

There are several proposals on how to define formally the security of such protocols [MR91,Bea91,Can00], but common to them all is the idea that security means that the adversary's view can be *simulated* efficiently by a machine that has access to only those data that the adversary is *entitled* to know.

⋆ Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

B. Pfitzmann (Ed.): EUROCRYPT 2001, LNCS 2045, pp. 280–300, 2001.

Proving correctness of a simulation in the case of [GMW87] requires a complexity assumption, such as existence of trapdoor one-way permutations. This is because the model of communication considered there is such that the adversary may see every message sent between parties, this is known as the *cryptographic model*. Later, *unconditionally* secure MPC protocols were proposed by Ben-Or et al. and Chaum et al.[BGW88,CCD88], in the model where *private* channels are assumed between every pair of parties. In this paper, however, we are only interested in the cryptographic model with an active and static adversary.

Over the years, several protocols have been proposed which, under specific computational assumptions, improve the efficiency of general MPC, see for instance [CDM00,GRR98]. Virtually all proposals have been based on some form of *verifiable secret sharing* (VSS), i.e., a protocol allowing a dealer to securely distribute a secret value s among the parties, where the dealer and/or some of the parties may be cheating. The basic paradigm is to ensure that all inputs and intermediate values in the computation are VSS'ed; this prevents the adversary from causing the protocol to terminate early or with incorrect results. In all these earlier protocols, the number of bits sent was $\Omega(n^2k|C|)$, where n is the number of parties, k is a security parameter, and $|C|$ is the size of a circuit computing the function. Here, C may be a Boolean circuit, or an arithmetic circuit over a finite field, depending on the protocol.

In [FH96] Franklin and Haber propose a protocol for *passive* adversaries which achieves complexity $O(nk|C|)$. This protocol is not based on VSS (there is no need since the adversary is passive) but instead on a so called joint encryption scheme, where a ciphertext can only be decrypted with the help of all parties, but still the length of an encryption is independent of the number of parties.

2 Our Results

In this paper, we present a new approach to building multiparty computation protocols with active security, namely we start from any secure threshold encryption scheme with certain extra homomorphic properties. This allows us to avoid the need to VSS all values handled in the computation, and therefore leads to more efficient protocols, as detailed below.

The MPC protocols we construct here can be proved secure against an active and static adversary who corrupts any minority of the parties. Like the protocol of [FH96], our construction requires an initial phase where keys for the threshold cryptosystem are set up. This can be done by a trusted party, or by any suitable MPC. In particular, the techniques of Damgård and Koprowski [DK01] could be used to make this phase reasonably efficient for the example cryptosystems we present here (see below). We stress that unlike some earlier proposals for preprocessing in MPC, the complexity of this phase does not depend on the number or the size of computations to be done later. In the following we therefore focus on the complexity of the actual computation. In our protocol the computation can be done only by broadcasting a number of messages, no encryption is needed to set up private channels. The complexities we state are therefore simply the

number of bits broadcast. This does not invalidate comparison with earlier protocols because first, the same measure was used in [FH96] and second, the earlier protocols with active security have complexity quadratic in n even if one only counts the bits broadcast. Our protocol has complexity $O(nk|C|)$ bits and requires $O(d)$ rounds, where d is the depth of C. To the best of our knowledge, this is the most efficient general MPC protocol proposed to date for active adversaries. Note that all complexities stated here and in the previous section are for computing deterministic functions. Probabilistic functions can be handled using standard techniques, see Section 8.1 for details.

Here, C is an arithmetic circuit over a ring R determined by the cryptosystem used, e.g., $R = \boldsymbol{Z}_N$ for an RSA modulus N, or $R = GF(2^k)$. While such circuits can simulate any Boolean circuit with a small constant factor overhead, this also opens the possibility of building an ad-hoc circuit over R for the desired function, possibly exploiting the fact that with a large R, we can manipulate many bits in one arithmetic operation.

The complexities given here assume existence of sufficiently efficient threshold cryptosystems. We give two examples of such systems with the right properties. One is based on Paillier's cryptosystem [Pai99] and Damgård and Jurik's generalisation thereof[DJ01], the other one is a variant of Franklin and Haber's cryptosystem [FH96], which is secure assuming that both the QR assumption and the DDH assumption are true (this is essentially the same assumption as the one made in [FH96]). While the first example is known (from [DJ01,FPS00]), the second is new and may be of independent interest.

Franklin and Haber in [FH96] left as an open problem to study the communication requirements for active adversaries. We can now say that under the same assumption as theirs, active security comes essentially for free.

2.1 Concurrent Related Work

In concurrent independent work, Jacobson and Juels[MJ00] present an idea for MPC somewhat related to ours, the *mix-and-match approach*. It too is based on threshold encryption (with extra algebraic properties, similar to, but different from the ones we use). Beyond this, the techniques are completely different. For Boolean circuits and in the random oracle model, they get the same message complexity as we obtain (without using random oracles). The round complexity is larger than ours (namely $O(n+d)$). Another difference is that mix-and-match is inherently limited to circuits where gates can be specified by constant size truth-tables, thus excluding arithmetic circuits over large rings. On the other hand, while mix-and-match can be based on the DDH assumption alone, it is not known if this is possible for our notion of threshold *homomorphic* encryption.

In [MH00], Hirt, Maurer and Przydatek show an MPC protocol designed for the private channels model. It can be transformed to our setting by implementing the channels using secure public-key encryption. This results in protocol that can be based on any secure public-key encryption scheme, with essentially the same communication complexity as ours, but with lower resilience, i.e. tolerating only less than $n/3$ active cheaters.

3 An Informal Description

In this section, we give an informal introduction to some main ideas. All the concepts introduced here will be treated more formally later in the paper. We will assume that from the start, the following scenario has been established: we have a semantically secure threshold public-key system given, i.e., there is a public encryption key pk known by all parties, while the matching private decryption key has been shared among the parties, such that each party holds a share of it.

The message space of the cryptosystem is assumed to be a ring R. In practice R might be $\boldsymbol{Z}_N$ for some RSA modulus N. For a plaintext $a \in R$, we let $\overline{a}$ denote an encryption of a. We then require certain homomorphic properties: from encryptions $\overline{a}, \overline{b}$, anyone can easily compute an encryption of $a + b$, which we denote $\overline{a} \boxplus \overline{b}$. We also require that from an encryption $\overline{a}$ and a constant $\alpha \in R$, it is easy to compute a random encryption of αa.

Finally we assume that three secure and efficient protocols are available:

Proving you know a plaintext If P_i has created an encryption $\overline{a}$, he can give a zero-knowledge proof of knowledge that he knows a.

Proving multiplications correct Assume that P_i is given an encryption $\overline{a}$, chooses a constant α, computes a random encryption $\overline{\alpha a}$ and broadcasts $\overline{\alpha}, \overline{\alpha a}$. He can then give a zero-knowledge proof that indeed $\overline{\alpha a}$ contains the product of the values contained in $\overline{\alpha}$ and $\overline{a}$.

Threshold decryption For the third protocol, we have common input pk and an encryption $\overline{a}$, in addition every party also uses his share of the private key as input. The protocol computes securely a as output for everyone.

We can then sketch how to perform securely a computation specified as a circuit doing additions and multiplications in R.

The MPC protocol would simply start by having each party publish encryptions of his input values and give zero-knowledge proofs that he knows these values and also, if we are simulating a Boolean circuit, that the values are 0 or 1. Then any operation involving addition or multiplication by constants can be performed with no interaction. This leaves only the following problem: Given encryptions $\overline{a}, \overline{b}$ (where it may be the case that no parties knows a nor b), compute securely an encryption of $c = ab$. This can be done by (a slightly more elaborate version of) the following protocol:

1. The parties generate an additive secret sharing of a:
   (a) Each party P_i chooses at random a value $d_i \in R$, broadcasts an encryption $\overline{d_i}$, and proves he knows d_i. Let d denote $\sum_{i=1}^{n} d_i$.
   (b) The parties use the third protocol to decrypt $\overline{a} \boxplus \overline{d_1} \boxplus \ldots \boxplus \overline{d_n}$.
   (c) Party P_1 sets $a_1 = (a + d) - d_1$, all other parties P_i set $a_i = -d_i$. Note that $a = \sum_{i=1}^{n} a_i$.
2. Each P_i broadcasts an encryption $\overline{a_i b}$, and invoke the second protocol with inputs $\overline{b}, \overline{a_i}$ and $\overline{a_i b}$.

3. Let H be the set of parties for which the previous step succeeded, and let C be the complement of H. The parties decrypt $\boxplus_{i \in C} \overline{a_i}$, learn $a_C = \sum_{i \in C} a_i$, and compute $\overline{a_C b}$. From this, and $\{\overline{a_i b} |\ i \in H\}$, all parties can compute an encryption $(\boxplus_{i \in H} \overline{a_i b}) \ \boxplus \overline{a_C b}$, which is indeed an encryption of ab.

At the final stage we know encryptions of the output values, which we can just decrypt. Intuitively this is secure if the encryption is secure because, other than the outputs, only random values and values already known to the adversary are ever decrypted. We will give proofs of this intuition in the following.

The above multiplication protocol is a more efficient version of a related idea from [FH96], where we have exploited the homomorphic properties to add protection against faults without loosing efficiency. Other papers have exploited homomorphic properties to construct efficient protocols for multiplication. In [GV87] Goldreich and Vainish used the homomorphic properties of the QR problem to construct a two-party protocol for multiplication over $GF(2)$ and in [KK91] Kurosawa and Kotera generalised it to $GF(L)$ for small L using the homomorphic properties of the Lth residuosity problem. Using also the Lth residuosity problem [Kur91] constructs an efficient ZKIP for multiplication over $GF(L)$.

4 Preliminaries and Notation

Let A be a probabilistic polynomial time (PPT) algorithm, which on input $x \in \{0,1\}^*$ and random bits $r \in \{0,1\}^*$ outputs a value $y \in \{0,1\}^*$. We write $y \leftarrow A(x)[r]$ to denote that y should be computed by running A on x and r. By $y \leftarrow A(x)$ we mean that y should be computed using uniformly random r and by $y \in A(x)$ we mean that y is among the values, that $A(x)$ outputs.

4.1 The MPC Model

We prove security in the MPC model from [Can00], with an open authenticated broadcast channel, against active non-adaptive adversaries corrupting any minority of the parties. We index the parties by $N = \{1, \ldots, n\}$ and let $C \subset N$ denote the subset of corrupted parties, k is the security parameter, and x_i is the secret input of party P_i. We study functions giving a common output $y = f(x_1, \ldots, x_n)$. We extend the model to handle this by saying that oracles for such functions broadcast y on the broadcast channel. This to assure that the ideal-model adversary learns the public output even though no party is corrupted. The technical report [CDN00] contains a more detailed description of the model. In the following we let secure mean secure against minority adversaries.

4.2 Σ-protocols

In this section, we look at two-party zero-knowledge protocols of a particular form. Assume we have a binary relation R consisting of pairs (x, w), where we think of x as a (public) instance of a problem and w as a witness, a solution

to the instance. Assume also that we have a 3-move proof of knowledge for R: this protocol gets a string x as common input for prover and verifier, whereas the prover gets as private input w such that $(x, w) \in R$. Conversations in the protocol are of form (a, e, z), where the prover sends a, the verifier chooses e at random, the prover sends z, and the verifier accepts or rejects. There is a security parameter k, such that the length of both x and e are linear in k. We will only look at protocols where also the length of a and z are linear in k. Such a protocol is said to be a *Σ-protocol* if we have the following:

- The protocol is *complete*: if the prover gets as private input w such that $(x, w) \in R$, the verifier always accepts.
- The protocol is *special honest verifier zero-knowledge*: from a challenge value e, one can efficiently generate a conversation (a, e, z), with probability distribution equal to that of conversation between the honest prover and verifier where e occurs as challenge.
- A cheating prover can answer only one of the possible challenges: more precisely, from the common input x and any pair of accepting conversations $(a, e, z), (a, e', z')$ where $e \neq e'$, one can compute efficiently w such that $(x, w) \in R$.

It is easy to see that the definition of Σ-protocols is closed under parallel composition.

5 Threshold Homomorphic Encryption

In this section we formalise the notion of threshold homomorphic encryption.

Definition 1 (Threshold Encryption Scheme). *We call the tuple* $(K, \mathrm{KD}, R, E, \mathrm{Decrypt})$ *a* threshold encryption scheme *if the following holds.*

Key space *The* key space $K = \{K_k\}_{k \in \boldsymbol{N}}$ *is a family of finite sets of keys of the form* $(pk, sk_1, \ldots, sk_n)$. *We call* pk *the* public key *and call* sk_i *the* private key share *of party* i. *There exists a PPT* key-generator K *which given* k *generates a random key* $(pk, sk_1, \ldots, sk_n) \leftarrow K(k)$ *from* K_k. *By* sk_C *for* $C \subset N$ *we denote the family* $\{sk_i\}_{i \in C}$.

Key-generation *There exists a* n*-party protocol* KD *securely evaluating the key-generator* K.

Message Sampling *There exists a PPT algorithm* R, *which on input* pk *outputs a uniformly random element from a set* R_{pk}. *We write* $m \leftarrow R_{pk}$.

Encryption *There exists a PPT algorithm* E, *which on input* pk *and* $m \in R_{pk}$ *outputs an encryption* $\overline{m} \leftarrow E_{pk}(m)$ *of* m. *By* C_{pk} *we denote the set of possible encryptions for the public key* pk.

Decryption *There exists a secure protocol* Decrypt *which on common input* $(\overline{M}, pk)$, *and secret input* sk_i *for the honest party* P_i, *where* sk_i *is the secret*

key share of the public key pk and $\overline{M}$ *is a set of encryptions of the messages* $M \subset R_{pk}$*, returns* M *as common output.*[1]

Threshold semantic security *Let* A *be any PPT algorithm, which on input* 1^k*,* C *such that* $|C| < n/2$*, public key pk, and corresponding private keys* sk_C *outputs two messages* $m_0, m_1 \in R_{pk}$ *and some arbitrary value* $s \in \{0,1\}^*$*. Let* $X_i(k, C)$ *denote the distribution of* (s, c_i)*, where* $(pk, sk_1, \ldots, sk_n) \leftarrow K(k)$*,* $(m_0, m_1, s) \leftarrow A(1^k, C, pk, sk_C)$*, and* $c_i \leftarrow E_{pk}(m_i)$*. Then* $X_i = \{X_i(k, C)\}_{k \in \boldsymbol{N}, C:|C|<n/2}$ *for* $i = 0, 1$ *are distribution ensembles over the index set* $\{C||C| < n/2\}$ *and we require that* $X_0 \stackrel{c}{\approx} X_1$*.*

A threshold homomorphic encryption scheme in addition has these properties:

Message ring For all public keys pk, the message space R_{pk} is a ring in which we can compute efficiently using the public key only. We denote the ring $(R_{pk}, \cdot_{pk}, +_{pk}, 0_{pk}, 1_{pk})$.

$+_{pk}$-homomorphic There exists a PPT algorithm, which given public key pk and encryptions $\overline{m}_1 \in E_{pk}(m_1)$ and $\overline{m}_2 \in E_{pk}(m_2)$ outputs a uniquely determined encryption $\overline{m} \in E_{pk}(m_1 +_{pk} m_2)$. We write $\overline{m} \leftarrow \overline{m}_1 \boxplus_{pk} \overline{m}_2$. Further more there exists a similar algorithm, $\boxminus_{pk}$, for subtraction.

Multiplication by constant There exists a PPT algorithm, which on input pk, $m_1 \in R_{pk}$ and $\overline{m}_2 \in E_{pk}(m_2)$ outputs a random encryption $\overline{m} \leftarrow E_{pk}(m_1 \cdot_{pk} m_2)$. We write $\overline{m} \leftarrow m_1 \boxdot_{pk} \overline{m}_2 \in E_{pk}(m_1 \cdot_{pk} m_2)$. We assume that we can also multiply a constant from the right.

Blindable There exists a PPT algorithm Blind, which on input $pk, \overline{m} \in E_{pk}(m)$ outputs an encryption $\overline{m}' \in E_{pk}(m)$ such that $\overline{m}'$ is distributed identically to $E_{pk}(m)[r]$, where r is chosen uniformly random.

Check of ciphertextness Given $y \in \{0,1\}^*$ and pk, where pk is a public key, it is easy to check whether $y \in C_{pk}$.[2]

Proof of plaintext knowledge Let $L_1 = \{(pk, y) | pk \text{ a public key} \wedge y \in C_{pk}\}$. There exists a Σ-protocol for the relation over $L_1 \times (\{0,1\}^*)^2$ given by $(pk, y) \sim (x, r) \Leftrightarrow x \in R_{pk} \wedge y = E_{pk}(x)[r]$.

Proof of correct multiplication Let $L_2 = \{(pk, x, y, z) | pk \text{ is a public key} \wedge x, y, z \in C_{pk}\}$. There exists a Σ-protocol for the relation over $L_2 \times (\{0,1\}^*)^3$ given by $(pk, x, y, z) \sim (d, r_1, r_2) \Leftrightarrow y = E_{pk}(d)[r_1] \wedge z = (d \boxdot_{pk} x)[r_2]$.

[1] We need that the Decrypt protocol is secure when executed in parallel. The MPC-model in [Can00] is however not security preserving under parallel composition, so we have to state this required property of the Decrypt protocol by simply letting the input be sets of ciphertexts.

[2] This check can be either directly or using a Σ-protocol: we will always use the test in a context, where a party publishes an encryption and then the recipients either check locally that $y \in C_{pk}$ or the publisher proves it using a Σ-protocol. In the following sections we adopt the terminology to the case, where the recipients can perform the test locally. Details for the case where a Σ-protocol is used are easy extractable.

6 Multiparty Σ-Protocols

In Section 7 we describe how to implement general multiparty computation from a threshold homomorphic encryption scheme, but as the first step towards this we show how one can generally and efficiently extend two-party Σ-protocols, as those for proof of plaintext knowledge and proof of correct multiplication in a threshold homomorphic encryption scheme, into secure multiparty protocols. We will need two essential tools in this section: the notion of *trapdoor commitments* and a multiparty protocol for generating a sufficiently random bit string. Our underlying purpose here is to allow a party to prove a claim using a Σ-protocol such that all other parties will be convinced and to do it much more efficiently than doing the original Σ-protocol independently with each of the other parties.

6.1 Generating (Almost) Random Common Challenges

First of all we want to be able to generate a common challenge for the Σ-protocols. Suppose first that $n \leq 16k$. Then we create a challenge by letting every party choose at random a $\lceil 2k/n \rceil$-bit string, and concatenate all these strings. This produces an m-bit challenge, where $2k \leq m \leq 16k$. We can assume without loss of generality that the basic Σ-protocol allows challenges of length m bits (if not, just repeat it in parallel a number of times). It is easy to see that with this construction, at least k bits of a challenge are chosen by honest parties and are therefore random, since a majority of parties are assumed to be honest. This is equivalent to doing a Σ-protocol where the challenge length is the number of bits chosen by honest parties. The cost of doing such a proof is $O(k)$ bits. If $n > 16k$, we will assume, as detailed later, that an initial preprocessing phase returns as public output a description of a random subset A of the parties, of size $4k$. It is easy to see that, except with probability exponentially small in k, A will contain at least k honest parties. We then generate a challenge by letting each party in A choose one bit at random, and then continue as above.

6.2 Trapdoor Commitments

A trapdoor commitment scheme can be described as follows: first a public key pk is chosen based on a security parameter k, by running a PPT *generator* G.

There is a fixed function *commit* that the committer C can use to compute a commitment c to s by choosing some random input r, computing $c = commit(s, r, pk)$, and broadcasting c. Opening takes place by broadcasting s, r; it can then be checked that $commit(s, r, pk)$ is the value C broadcasted originally.

We require the following:

(Perfect) Hiding For a pk correctly generated by G, uniform r, r' and any s, s', the distributions of $commit(s, r, pk)$ and $commit(s', r', pk)$ are identical.

(Computational) Binding For any C running in expected polynomial time (in k) the probability that C on input pk computes s, r, s', r' such that $commit(s, r, pk) = commit(s', r', pk)$ and $s \neq s'$ is negligible.

Trapdoor Property The algorithm for generating pk also outputs a string t, the trapdoor. There is an efficient algorithm which on input t, pk outputs a commitment c, and then on input any s produces r such that $c = commit(s, r, pk)$. The distribution of c is identical to that of commitments computed in the usual way.

In other words, the commitment scheme is binding if you know only pk, but given the trapdoor, you can cheat arbitrarily. Finally, we also assume that the length of a commitment to s is linear in the length of s.[3] Existence of commitments with all these properties follow in general merely from existence of Σ-protocols for hard relations, and this assumption in turn follows from the properties we already assume for the threshold cryptosystems. For concrete examples that would fit with the examples of threshold encryption we use, see [CD98].

6.3 Putting Things Together

In our global protocol, we assume that the initial preprocessing phase independently generates for each party P_i a public key k_i for the trapdoor commitment scheme and distributes it to all participating parties. We may assume in the following that the simulator for our global protocol knows the trapdoors t_i for (some of) these public keys. This is because it is sufficient to simulate in the hybrid model where parties have access to a trusted party that will output the k_i's on request. Since this trusted party gets no input from the parties, the simulator can imitate it by running G itself a number of times, learning the trapdoors, and showing the resulting k_i's to the adversary.

In our global protocol there are a number of *proof phases*. In each such phase, each party in some subset N'[4] is supposed to give a proof of knowledge: each P_i in the subset has broadcast an x_i and claims he knows w_i such that (x_i, w_i) is in some relation R_i which has an associated Σ-protocol. We then do the following:

1. Each P_i in N' computes the first message a_i in his proof and broadcasts $c_i = commit(a_i, r_i, k_i)$.[5]

[3] In principle any commitment scheme can be transformed to fulfil this. Assume that a scheme C has commitments of length k^c and consider the modified scheme C' which on security parameter k runs C on security parameter $k' = k^{1/c}$. This scheme is still a commitment scheme as $\delta(k')$ is still negligible and now the commitments has length k. However in the new scheme the basic cryptographic primitives providing the security is instantiated at a much lower key-size, and indeed such a reduction is only *weakly security preserving*[pac96]. The remaining reductions in this paper are all *polynomially security preserving* and for the security of the protocol to be polynomially preserved relative to the underlying computational assumptions the above reduction should be avoided.

[4] The subset N' is the subset of the parties that still participate, i.e. have not been excluded due to deviation from the protocol.

[5] The intuition behind the use of (independently generated instances) of perfectly hiding trapdoor commitments in the proofs of knowledge of e.g. a plaintext is to avoid malleability issues, and to ensure "independence of inputs" where necessary.

2. Make random challenge e according to the method described earlier.
3. Each P_i in N' computes the answer z_i to challenge e, and broadcasts a_i, r_i, z_i
4. Every party can check every proof given by verifying $c_i = commit(a_i, r_i, k_i)$ and that (a_i, e, z_i) is an accepting conversation.

It is clear that such a proof phase has communication complexity no larger than n times the complexity of a single Σ-protocol, i.e. $O(nk)$ bits. We denote the execution of the protocol by $(\mathcal{A}', N'') \leftarrow \Sigma(\mathcal{A}, x_{N'}, w_{H\cap N'}, k_N)$, where $\mathcal{A}$ is the state of the adversary before the execution, $x_{N'} = \{x_i\}_{i\in N'}$ are the instances that the parties N' are to prove that they know a witness to, $w_{H\cap N'} = \{w_i\}_{i\in H\cap N'}$ are witnesses for the instances x_i corresponding to honest P_i, $k_N = \{k_i\}_{i\in N}$ is the commitment keys for all the parties, $\mathcal{A}'$ is the state of the adversary after the execution, and $N'' \subset N'$ is the subset of the parties completing the proof correctly. The reason why the execution only depends on the witnesses $w_{H\cap N'}$ is that the corrupted parties are controlled by the adversary and their witnesses, if even well-defined, are included in the start-state $\mathcal{A}$ of the adversary.

Now let $t_H = \{t_i\}_{i\in H}$ be the commitment trapdoors for the honest parties. We describe a procedure $(\mathcal{A}', N'', w_{N''\cap C}) \leftarrow S_\Sigma(\mathcal{A}, x_{N'}, t_H, k_N)$ that will be used as subroutine in the simulation of our global protocol. $S_\Sigma(\mathcal{A}, x_{N'}, k_N, t_H)$ will have the following properties:

- $S_\Sigma(\mathcal{A}, x_{N'}, k_N, t_H)$ runs in expected polynomial time and the part $(\mathcal{A}', N'')$ of the output is perfectly indistinguishable from the output of a real execution $\Sigma(\mathcal{A}, x_{N'}, w_{H\cap N'}, k_N)$ given the start state $\mathcal{A}$ of the adversary (which we assume includes $x_{N'}$ and k_N).
- Except with negligible probability $w_{N''\cap C} = \{w_i\}_{i\in N''\cap C}$ are valid witnesses to the instances x_i corresponding the corrupted parties completing the proofs correctly.

The algorithm of S_Σ is as follows:

1. For each P_i: if P_i is honest, use the trapdoor t_i for k_i to compute a commitment c_i that can be opened arbitrarily and show c_i to the adversary. If P_i is corrupt, receive c_i from the adversary.
2. Run the procedure for choosing the challenge, choosing random contributions on behalf of honest parties. Let e_0 be the challenge produced.
3. For each P_i do (where the adversary may choose the order in which parties are handled): *If P_i is honest*, run the honest verifier simulator to get an accepting conversation (a_i, e_0, z_i). Use the commitment trapdoor to compute r_i such that $c_i = commit(a_i, r_i)$ and show (a_i, r_i, z_i) to the adversary. *If P_i is corrupt*, receive (a_i, r_i, z_i) from the adversary.
   The current state $\mathcal{A}'$ of the adversary and the subset N'' of parties correctly completing the proof is copied to the output from this simulation subroutine. In addition, we now need to find witnesses for x_i from those corrupt P_i that sent a correct proof in the simulation. This is done as follows:
4. For each corrupt P_i that sent a correct proof in the view just produced, execute the following loop:

(a) Rewind the adversary to its state just before the challenge is produced.
(b) Run the procedure for generating the challenge using fresh random bits on behalf of the honest parties. This results in a new value e_1.
(c) Receive from the adversary proofs on behalf of corrupted parties and generate proofs on behalf of honest parties, w.r.t. e_1, using the same method as in Step 3. If the adversary has made a correct proof a_i', r_i', e', z_i' on behalf of P_i, exit the loop. Else go to Step 4a.

If $e_0 \neq e_1$ and $a_i = a_i'$ compute and output a witness for x_i, from the conversations $(a_i, e_0, z_i), (a_i', e_1, z_i')$. Else output $c_i, a_i, r_i, a_i', r_i'$ (this will be a break of the commitment scheme). Go on to next corrupt P_i.

It is clear by inspection and assumptions on the commitments and Σ-protocols that the part $(\mathcal{A}', N'')$ of the output is distributed correctly. For the running time, assume P_i is corrupt and let ϵ be the probability that the adversary outputs a correct a_i, r_i, z_i given some fixed but arbitrary value $View$ of the adversary's view up to the point just before e is generated. Observe that the contribution from the loop to the running time is ϵ times the expected number of times the loop is executed before terminating, which is $1/\epsilon$, so that to the total contribution is $O(1)$ times the time to do one iteration, which is certainly polynomial. As for the probability of computing correct witnesses, observe that we do not have to worry about cases where ϵ is negligible, say $\epsilon < 2^{-k/2}$, since in these cases $P_i \notin N''$ with overwhelming probability. On the other hand, assume $\epsilon \geq 2^{-k/2}$, let $\bar{e}$ denote the part of the challenge e chosen by honest parties, and let $pr()$ be the probability distribution on $\bar{e}$ given the view $View$ and given that the choice of $\bar{e}$ leads to the adversary generating a correct answer on behalf of P_i. Clearly, both $\bar{e_0}$ and $\bar{e_1}$ are distributed according to $pr()$. Now, the a priori distribution of $\bar{e}$ is uniform over at least 2^k values. This, and $\epsilon \geq 2^{-k/2}$ implies by elementary probability theory that $pr(\bar{e}) \leq 2^{-k/2}$ for any e, and so the probability that $\bar{e_0} = \bar{e_1}$ is $\leq 2^{-k/2}$. We conclude that except with negligible probability, we will output either the required witnesses, or a commitment with two different valid openings. However, the latter case occurs with negligible probability. Indeed, if this was not the case, observe that since the simulator never uses the trapdoors of k_i for corrupt P_i, the simulator together with the adversary could break the binding property of the commitments. Formulating a reduction proving this formally is straightforward and is left to the reader.

7 General MPC from Threshold Homomorphic Encryption

Assume that we have a threshold homomorphic encryption scheme as described in Section 5. In this section we describe the FuncEval$_f$ protocol which securely computes any polynomial time computable n-party function $y \leftarrow f(x_1, \ldots, x_n)$ using a uniform polynomially sized family of arithmetic circuits over R_{pk}.

Our approach works for any reasonable encoding of f as an arithmetic circuit. This can allow for efficient encodings of arithmetic function if one can exploiting

knowledge about the rings R_{pk} over which the function is evaluated. For simplicity we will however here assume that f is encoded using a circuit taking inputs from $\{0_{pk}, 1_{pk}\}$, using $+$, $-$, and $\cdot$ gates, and using the same circuit for a fixed security parameter. Since our encryption scheme is only $+$ and $-$-homomorphic we will be needing a protocol Mult for securely computing an encryption of $m_1 \cdot_{pk} m_2$ given encryptions of m_1 and m_2.

We assume that the parties has access to a trusted party Preprocess, which at the beginning of the protocol outputs a public value $(k_1, \ldots, k_n)$, where k_i is a random public commitment key for a trapdoor commitment scheme as described in Section 6.2. If $n > 16k$ then further more the trusted party returns a public description of a random $4k$-subset of the parties as described in Section 6.1[6]. As described in Section 6.3, we can then from the Σ-protocols of the threshold homomorphic encryption scheme for proof of plaintext knowledge and correct multiplication construct n-party versions, which we call POPK resp. POCM. The corresponding versions of our general simulation routine S_Σ for these protocols will be called S_{POPK} resp. S_{POCM}.

7.1 The Mult Protocol

Description. All honest parties P_i know public values $k_N = \{k_i\}_{i \in N}$, pk, and encryptions $\overline{a}, \overline{b} \in E_{pk}(a)$, for some possible unknown $a, b \in R_{pk}$, and private values sk_i. Further more a set N' of participating parties, those that have not been caught deviating from the protocol, is known to all parties. The corrupted parties are controlled by an adversary and the parties want to compute a common value $\overline{c} \in E_{pk}(ab)$ without anyone learning anything new about a, b, or $a \cdot b$.

Implementation.

1. First all participating parties additively secret share the value of a.
   (a) P_i, for $i \in N'$ chooses a value d_i uniformly at random in R_{pk}, computes an encryption $\overline{d}_i \leftarrow E(d_i)$, broadcasts it, and participates in POPK to check that each P_i knows r_i and d_i such that $\overline{d}_i = E_{pk}(d_i)[r_i]$.
   (b) Let N'' denote parties completing the proof in (a) and let $d = \sum_{i \in N''} d_i$. All parties compute $\overline{d} = \boxplus_{i \in N''} \overline{d}_i$ and $\overline{e} = \overline{a} \boxplus \overline{d}$.
   (c) The parties in N'' call Decrypt to compute the value $a + d$ from $\overline{e}$.
   (d) The party in N'' with smallest index sets $\overline{a_i} \leftarrow \overline{e} \boxminus \overline{d_i}$ and $a_i \leftarrow a + d - d_i$. The other parties in N'' set $\overline{a_i} \leftarrow \boxminus \overline{d_i}$ and $a_i \leftarrow -d_i$.
2. Each party P_i for $i \in N''$ computes $\overline{f}_i \leftarrow a_i \boxdot \overline{b}$, broadcasts $\overline{f}_i$, and participates in POCM to check that all $\overline{f}_i$ was computed correctly. Let X be the subset failing the proof and let $N''' = N'' \setminus X$.
3. The parties compute $\overline{a}_X = \boxplus_{i \in X} \overline{a}_i$ and decrypt it using Decrypt to obtain $a_X = \sum_{i \in X} a_i$.
4. All parties compute $\overline{c} \leftarrow (\boxplus_{i \in N'''} \overline{f}_i) \boxplus (a_X \boxdot \overline{b}) \in E_{pk}(ab)$.

[6] In the following we present the case where $n \leq 16k$.

Theorem 1. *There exists a simulator for the* Mult *protocol, which produces a view in the ideal-world that is computationally indistinguishable from the view produced by an execution of the* Mult *protocol in the real-world.*

Proof outline: We give an outline of the main ideas used in the simulation. The main obstacle is Step 1c, where $a + d$ should be handed to the adversary to simulate an oracle call to the Decrypt oracle. By choosing $\overline{d}_i$ correctly for honest parties and using S_{POPK} the simulator can learn d, but it cannot learn a. We handle this by letting one of the honest party P_s choose $\overline{d}_s$ as $\text{Blind}(E_{pk}(d'_s) \boxminus \overline{a})$ for uniformly random $d'_s \in R_{pk}$. Then all values are still correctly distributed, but now the simulator can compute $a + d$ as $(\sum_{i \in N'' \setminus \{s\}} d_i) + d'_s$. Observe that the simulator now cannot compute the value a_s which is necessary later. Doing the same computation on d'_s it can however compute $a'_s = a_s + a$. Now assume that the simulator has access to an oracle returning an encryption $\overline{c}'$ of $a \cdot b$. Then it simulates Step 2 by computing $a_s \boxdot \overline{b}$ as $\text{Blind}((a'_s \boxdot \overline{b}) \boxminus \overline{c}')$ and running S_{POCM}. Step 3 is simulated by giving $\sum_{i \in X} a_i$ to the adversary (this value the simulator can compute as $s \notin X$) and Step 4 is simulated by doing the correct computation as the necessary values are available.

By the properties of the knowledge extractors and Blind it follows that the view produced as above is statistically indistinguishable from the view of a real-world execution. Now instead of the oracle value $\overline{c}'$ use a random encryption of 0_{pk}. If this changes the view produced by the simulator except computationally negligible, then *as the simulator does not use the secret keys of any honest party* the simulator would be a distinguisher of encryptions of $a \cdot b$ and encryptions of 0_{pk} contradicting the semantic security of the encryption scheme. The technical report [CDN00] contains a detailed proof. □

As the Decrypt protocol is assumed to be secure under parallel composition and our multiparty zero-knowledge proofs have been proven to be secure, so are then trivially the Mult protocol.

7.2 The FuncEval$_f$ Protocol

Now assume that the description of a arithmetic circuit for evaluating f is given. The parties then evaluate f in the following manner. First the parties run the Preprocess and the KD oracles and obtain the keys $(k_1, \ldots, k_n)$ for the trapdoor commitment scheme and $(pk, sk_1, \ldots, sk_n)$ for the encryption scheme. The key sk_i is private to P_i. Each party P_i then does a bitwise encryption $\overline{x}_{i,j} \leftarrow E_{pk}(x_{i,j})$ of its input x_i, broadcasts the encryptions, and proves in zero-knowledge, that $\overline{x}_{i,j}$ does in fact contain either 0 or 1.[7] For those P_i failing the above proof the parties exclude them, take x_i to be $000\ldots00$, and compute $\overline{x}_{i,j} \leftarrow E_{pk}(x_{i,j})[r]$ for some fixed agreed upon string $r \in \{0,1\}^{p(k)}$. In this way all parties get to know common legal encrypted circuit inputs for all parties. Then the circuit is evaluated. In each round all gates that are ready to be evaluated are evaluated

[7] We will be needing a Σ-protocol for doing this, but such protocol is easy to implement for the examples of threshold encryption that we give in Section 8.

in parallel. Addition and subtraction gates are evaluated locally using $\boxplus$ and $\boxminus$; and multiplication gates are evaluated using the Mult protocol. Finally the parties decrypt the output gates and output the reviled values.

Theorem 2. *The* FuncEval_f *protocol as described above securely evaluates* f *in the presence of active non-adaptive adversary corrupting at most a minority of the parties.*[8] *The round complexity is in the order of the depth of the circuit for* f *and the communication complexity of the protocol is* $O((nk+d)|f|)$ *bits, where* $|f|$ *denotes the size of the circuit for evaluating* f *and* d *denotes the communication complexity of a decryption.*

Proof outline: Given a real-world adversary $\mathcal{A}$ we construct a simulator $\mathcal{S}(\mathcal{A})$ running in the ideal-world. The initial oracle calls are simulated by running the generators locally and giving the appropriate values to $\mathcal{A}$. The simulator saves $(k_1, \ldots, k_n, pk, \{sk_i\}_{i \in C})$ for later use, *but discards* sk_i *for all honest parties.* Assume for now that $\mathcal{S}$ has access to an oracle giving it the values $\overline{x}_{i,j}$ for all honest parties. The simulator then gives the $\overline{x}_{i,j}$ values to $\mathcal{A}$ and receive $\overline{x}_{i,j}$ for all corrupted parties from $\mathcal{A}$. Using the knowledge extractor the simulator then learns all $x_{i,j}$ for corrupted parties that completed the proof that $\overline{x}_{i,j}$ contains 0 or 1. It uses these values as input to the ideal evaluation of f and learns the output y. Now the gate evaluation is simulated using the simulator for the Mult protocol. The decryption (by oracle call) of output gates are simulated by just handing the correct value to $\mathcal{A}$. These values are known as the correct output y of the computation is known to the simulator. This simulation is by the properties of the knowledge extractors and the Mult simulator computationally indistinguishable from that of a real-world execution. We get rid of the oracle values $\overline{x}_{i,j}$ as we did in the proof of Theorem 1.

The round complexity follows by inspection. The gates that give rise to communication are the input, multiplication, and output gates. The communication used to handle these gates is in the order of n encryptions ($O(nk)$ bits), n zero-knowledge proofs ($O(nk)$ bits as we have assumed that the Σ-protocols have communication complexity $O(k)$) and 1 decryption ($O(d)$ bits by definition). The total communication complexity therefore is $O((nk+d)|f|)$ as claimed. The technical report [CDN00] contains a detailed proof. □

The threshold homomorphic encryption schemes we present in Section 8 both have $d = O(kn)$. It follows that for deterministic f the FuncEval_f protocol based on any of these schemes has communication complexity $O(nk|f|)$ bits.

8 Examples of Threshold Homomorphic Cryptosystems

In this section, we describe some concrete examples of threshold systems meeting our requirements, including Σ-protocols for proving knowledge of plaintexts, correctness of multiplications and validity of decryptions.

[8] Generally we can make the protocol secure against any corruption structure for which the threshold cryptosystem is secure and for which one can generate short random challenges containing at least k random bits as in Section 6.1.

Both our examples involve choosing as part of the public key a k-bit RSA modulus $N = pq$, where p, q are chosen such that $p = 2p' + 1, q = 2q' + 1$ for primes p', q' and both p and q have $k/2$ bits. For convenience in the proofs to follow, we will assume that the length of the challenges in all the proofs is $k/2-1$.

8.1 Basing It on Paillier's Cryptosystem

In [Pai99], Paillier proposes a probabilistic public-key cryptosystem where the public key is a k-bit RSA modulus N and an element $g \in \boldsymbol{Z}^*_{N^2}$ of order divisible by N. The plaintext space for this system is $\boldsymbol{Z}_N$. In [DJ01] the crypto-system is generalised to have plaintext space $\boldsymbol{Z}_{N^s}$ for any s smaller than the factors of N and there g has order divisible by N^s. To encrypt $a \in \boldsymbol{Z}_{N^s}$, one chooses $r \in \boldsymbol{Z}^*_{N^{s+1}}$ at random and computes the ciphertext as $\overline{a} = g^a r^{N^s} \bmod N^{s+1}$. The private key is the factorisation of N, i.e., $\phi(N)$ or equivalent information. Under an appropriate complexity assumption given in [Pai99], this system is semantically secure, and it is trivially homomorphic over $\boldsymbol{Z}_{N^s}$ as we require here: we can set $\overline{a} \boxplus \overline{b} = \overline{a} \cdot \overline{b} \bmod N^{s+1}$. Furthermore, from α and an encryption $\overline{a}$, a *random* encryption of αa can be obtained by multiplying $\overline{a}^\alpha \bmod N^{s+1}$ by a random encryption of 0. In [DJ01] a threshold version of this system has been proposed, based on a variant of Shoup's [Sho00] technique for threshold RSA. A multiplication protocol was also given in [DJ01], though for a slightly different setting. We will not go into further details here, but note that using known techniques the multiplication protocol can be modified to meet our definition of a threshold homomorphic encryption scheme. The technical report [CDN00] contains more details.

Generalisations of FuncEval. Using standard techniques, the FuncEval-protocol can be extended to handle probabilistic functions. In this section we describe how this works when we instantiate using the Paillier cryptosystem. We show how random $\boldsymbol{Z}_{N^s}$-gates (outputting a uniformly random element from $\boldsymbol{Z}_{N^s}$ unknown to all parties) and random 0/1-gates (outputting a uniformly random element from $\{0,1\}$ unknown to all parties) can be implemented securely in a constant number of rounds.

As a step-stone we also recall how to implement inversion gates and do unbounded fan-in multiplication of invertible elements in a constant number of rounds (see [BB89]). Due to lack of space only the protocols are given, but they can all be proven secure using the techniques of the previous sections.

In the following, if $n \geq 16k$, let the random group denote the $4k$-subset A given in the preprocessing and if $n < 16k$ let the random group be all the parties. Assume that the parties in the random group are indexed $1, \ldots, r(n,k)$. Observe that $r(n,k) \in O(\min(n,k))$ and that except with negligible probability the random group contains a honest party.

Random $\boldsymbol{Z}_{N^s}$-gates. All the parties in the random group pick a uniformly random element $a_i \in \boldsymbol{Z}_{N^s}$, broadcast an encryption $\overline{a_i}$, and prove knowledge of a_i.

Then all the parties compute $\overline{a} = \boxplus_{i=1}^{r(n,k)} \overline{a_i}$ and a is the secret uniformly random $\boldsymbol{Z}_{N^s}$-element. The communication complexity is $O(r(n,k)k)$.

Inversion gates. Given $\overline{a}$, where a is invertible, the parties generates $\overline{b}$ using a random $\boldsymbol{Z}_{N^s}$-gate. Since b is invertible except with negligible probability, we assume in the following that it is. The parties compute $\overline{ab}$ and reveals ab (this is secure since ab is a uniformly random invertible element). The parties computes $(ab)^{-1}$ and $\overline{a^{-1}} = (ab)^{-1} \boxdot \overline{b}$. The communication complexity is $O(nk)$.

Constant-round unbounded fan-in multiplication of invertible elements. Given encryptions $\overline{x_1}, \ldots, \overline{x_l}$ of invertible elements the parties generate secret random $\boldsymbol{Z}_{N^s}$-elements $\overline{y_0}, \ldots, \overline{y_l}$, compute $\overline{y_0^{-1}}, \ldots, \overline{y_l^{-1}}$, and compute and reveal $z_i = y_{i-1} x_i y_i^{-1}, i = 1, \ldots, l$. Then they compute $\overline{\prod_{i=1}^{l} x_i} = \overline{y_0^{-1}}(\prod_{i=1}^{l} z_i)\overline{y_l}$. The communication complexity of this is $O(lnk)$.

Random 0/1*-gates.* Each party in the random group generates a random bit b_i, publishes $\overline{b_i}$, proves knowledge of $b_i \in \{0,1\}$, and all the parties compute $\overline{b} = \left[\left(\boxdot_{i=1}^{r(n,k)}(1 \boxminus 2 \boxdot \overline{b_i})\right) \boxplus 1\right] \boxdot 2^{-1}$. The communication complexity of this is $O(r(n,k)nk)$.

8.2 Basing It on QRA and DDH

In this section, we describe a cryptosystem which is a simplified variant of Franklin and Haber's system [FH96], a somewhat similar (but non-threshold) variant was suggested by one the authors of the present paper and appears in [FH96].

For this system, we choose an RSA modulus $N = pq$, where p, q are chosen such that $p = 2p' + 1, q = 2q' + 1$ for primes p', q'. We also choose a random generator g of $SQ(N)$, the subgroup of quadratic residues modulo N (which here has order $p'q'$). We finally choose x at random modulo $p'q'$ and let $h = g^x \bmod N$. The public key is now N, g, h while x is the secret key.

The plaintext space of this system is $\boldsymbol{Z}_2$. We set $\Delta = n!$ (recall that n is the number of parties). Then to encrypt a bit b, one chooses at random r modulo N^2 and a bit c and computes the ciphertext

$$((-1)^c g^r \bmod N,\ (-1)^b h^{4\Delta^2 r} \bmod N)$$

The purpose of choosing r modulo N^2 is to make sure that g^r will be close to uniform in the group generated by g even though the order of g is not public. It is clear that a ciphertext can be decrypted if one knows x. The purpose of having $h^{4\Delta^2 r}$ (and not h^r) in the ciphertext will be explained below.

The system clearly has the required homomorphic properties, we can set:

$$(\alpha, \beta) \boxplus (\gamma, \delta) = (\alpha\gamma \bmod N, \beta\delta \bmod N)$$

Finally, from an encryption (α, β) of a value a and a known b, one can obtain a *random* encryption of value $ba \bmod 2$ by first setting (γ, δ) to be a random encryption of 0 and then outputting $(\alpha^b \gamma \bmod N, \beta^b \delta \bmod N)$.

We now argue that under the Quadratic Residuosity Assumption (QRA) and the Decisional Diffie Hellman Assumption (DDH), the system is semantically secure. Recall that DDH says that the distributions $(g, h, g^r \bmod p, h^r \bmod p)$ and $(g, h, g^r \bmod p, h^s \bmod p)$ are indistinguishable, where g, h both generate the subgroup of order p' in $\boldsymbol{Z}_p^*$ and r, s are independent and random in $\boldsymbol{Z}_{p'}$. By the Chinese remainder theorem, this is easily seen to imply that also the distributions $(g, h, g^r \bmod N, h^r \bmod N)$ and $(g, h, g^r \bmod N, h^s \bmod N)$ are indistinguishable, where g, h both generate $SQ(N)$ and r, s are independent and random in $\boldsymbol{Z}_{p'q'}$. Omitting some tedious details, we can then conclude that the distributions

$$
\begin{aligned}
(g, h, (-1)^c g^r \bmod N, h^{4\Delta^2 r} \bmod N) \\
(g, h, (-1)^c g^r \bmod N, h^{4\Delta^2 s} \bmod N) \\
(g, h, (-1)^c g^r \bmod N, -h^{4\Delta^2 s} \bmod N) \\
(g, h, (-1)^c g^r \bmod N, -h^{4\Delta^2 r} \bmod N)
\end{aligned}
$$

are indistinguishable, using (in that order) DDH, QRA and DDH.

Threshold Decryption. Shoup's method for threshold RSA [Sho00] can be directly applied here: he shows that if one secret-shares x among the parties using a polynomial computed modulo $p'q'$ and publishes some extra verification information, then the parties can jointly and securely raise an input number to the power $4\Delta^2 x$. This is clearly sufficient to decrypt a ciphertext as defined here: to decrypt the pair (a, b), compute $ba^{-4\Delta^2 x} \bmod N$. We do not describe the details here, as the protocol from [Sho00] can be used directly. We only note that decryption can be done by having each party broadcast a single message and prove by a Σ-protocol that it is correct. The communication complexity of this is $O(nk)$ bits. In the original protocol the random oracle model is used when parties prove that they behave correctly. However, the proofs can instead be done according to our method for multiparty Σ-protocols without loss of efficiency (Section 6). This also immediately implies a protocol that will decrypt several ciphertexts in parallel.

Proving You Know a Plaintext. We will need an efficient way for a party to prove in zero-knowledge that a pair (α, β) he created is a legal ciphertext, and that he knows the corresponding plaintext. A pair is valid if and only if α, β both have Jacobi symbol 1 (which can be checked easily) and if for some r we have $(g^2)^r = \alpha^2 \bmod N$ and $(h^{8\Delta^2})^r = \beta^2 \bmod N$. This last pair of statements can be proved non-interactively and efficiently by a standard equality of discrete log proof appearing in [Sho00]. Note that the squarings of α, β ensure that we are working in $SQ(N)$, which is necessary to ensure soundness.

This protocol has the standard 3-move form of a Σ-protocol. It proves that an r fitting with α, β *exists*. But it does not prove that the prover *knows* such an r (and hence knows the plaintext), unless we are willing to also assume the strong RSA assumption[9]. With this assumption, on the other hand, the equality of discrete log proof is indeed a proof of knowledge.

However, it is possible to do without this extra assumption: observe that if β was correctly constructed, then the prover knows a square root of β (namely $h^{2\Delta^2 r} \bmod N$) iff $b = 0$ and he knows a root of $-\beta$ otherwise. One way to exploit this observation is if we have a commitment scheme available that allows committing to elements in $\boldsymbol{Z}_N$. Then P_i can commit to his root α, and prove in zero-knowledge that he knows α and that $\alpha^4 = \beta^2 \bmod N$. This would be sufficient since it then follows that α^2 is β or $-\beta$.

Here is a commitment scheme (already well known) for which this can be done efficiently: choose a prime P, such that N divides $P - 1$ and choose elements G, H of order N modulo P, but where no party knows the discrete logarithm of H base G. This can all be set up initially (recall that we already assume that keys are set up once and for all). Then a commitment to α has form $(G^r \bmod P, G^\alpha H^r \bmod P)$, and is opened by revealing α, r. It is easy to see that this scheme is unconditionally binding, and is hiding under the DDH assumption (which we already assumed). Let $[\alpha]$ denote a commitment to α and let $[\alpha][\beta] \bmod P$ be the commitment you obtain in the natural way by componentwise multiplication modulo P. It is then clear that $[\alpha][\beta] \bmod P$ is a commitment to $\alpha + \beta \bmod N$.

It will be sufficient for our purposes to make a Σ-protocol that takes as input commitments $[\alpha], [\beta], [\gamma]$, shows that the prover knows α and shows that $\alpha\beta = \gamma \bmod N$. Here follows such a protocol:

1. Inputs are commitments $[\alpha], [\beta], [\gamma]$ where P_i claims that $\alpha\beta = \gamma \bmod N$. P_i chooses a random δ and makes commitments $[\delta], [\delta\beta]$.
2. The verifier send a random e.
3. P_i opens the commitment $[\alpha]^e[\delta] \bmod P$ to reveal a value e_1. P_i opens the commitment $[\beta]^{e_1}[\delta\beta]^{-1}[\gamma]^{-e} \bmod P$ to reveal 0.
4. The verifier accepts iff the commitments are correctly opened as required.

Using standard techniques, it is straightforward to show that this protocol is a Σ-protocol. The technical report [CDN00] contains more details.

Proving Multiplications Correct. Finally, we need to consider the scenario where party P_i has been given an encryption C_a of a, has chosen a constant b, and has published encryptions C_b, D, of values b, ba, and where D has been constructed by P_i as we described above. It follows from this construction that if $b = 1$, then $D = C_a \boxplus E$ where E is a random encryption of 0. Assuming $b = 1$, E can be easily reconstructed from D and C_a.

[9] That is, assume that it is hard to invert the RSA encryption function, even if the adversary is allowed to choose the public exponent

Now we want a Σ-protocol that P_i can use to prove that D contains the correct value. Observe that this is equivalent to the statement

$$((C_b \text{ encrypts } 0)\ AND\ (D \text{ encrypts } 0))\ OR$$
$$((C_b \text{ encrypts } 1)\ AND\ (E \text{ encrypts } 0))$$

We have already seen how to prove by a Σ-protocol that an encryption (α, β) contains a value b, by proving that you know a square root of $(-1)^b\beta$. Now, standard techniques from [CDS94] can be applied to building a new Σ-protocol proving a monotone logical combination of statements such as we have here.

9 An Optimisation of the FuncEval Protocol

The following optimisation of the FuncEval-protocol was brought to our attention by an anonymous referee. The optimisation applies to the situation where at most $(\frac{1}{2} - c)n$ parties, for some $c > 0$, can be corrupted and n is larger than k. In that case we can use the random group for doing the entire computation. The decryption keys for the threshold cryptosystem are distributed only to the random group and all parties are given the public key. All parties then broadcast encryptions of their inputs as before. Then the parties in the random group do the actual computation and broadcast the result. The communication complexity of this is $O(k^2|C|)$ as the initial broadcast of inputs are dominated by the computation. This is better than $O(kn|C|)$ if $n > k$. The same optimisation applies to any MPC protocol by letting the parties secret share their input among the random group initially. This typically reduces a complexity of $O(k^d n^e|C|)$ to $O(k^{d+e}|C|)$. Finally k^{d+e} can be replaced by k to obtain a communication complexity of $O(k|C|)$ using the *weakly security preserving* reduction of Footnote 3. Note that the last part of the transformation has no practical value, it is a property of the security model allowing to sell security for cuts in complexity.

References

ACM88. *Proceedings of the Twentieth Annual ACM STOC*, Chicago, Illinois, 2–4 May 1988.

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

Bea91. D. Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *Advances in Cryptology - Crypto '91*, pages 377–391, Berlin, 1991. Springer-Verlag. LNCS Vol. 576.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In ACM [ACM88], pages 1–10.

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

CCD88. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In ACM [ACM88], pages 11–19.

CD98. Ronald Cramer and Ivan Damgaard. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free. In Hugo Krawczyk, editor, *Advances in Cryptology - Crypto '98*, pages 424–441, Berlin, 1998. Springer-Verlag. LNCS Vol. 1462.

CDM00. Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multiparty computation from any linear secret-sharing scheme. In Bart Preneel, editor, *Advances in Cryptology - EuroCrypt 2000*, pages 316–334, Berlin, 2000. Springer-Verlag. LNCS Vol. 1807.

CDN00. Ronald Cramer, Ivan B. Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. Research Series RS-00-14, BRICS, Department of Computer Science, University of Aarhus, June 2000. Updated version available at Cryptology ePrint Archive, record 2000/064, `http://eprint.iacr.org/`.

CDS94. R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - Crypto '94*, pages 174–187, Berlin, 1994. Springer-Verlag. LNCS Vol. 839.

DJ01. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography, Fourth International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Proceedings*, 2001. LNCS. Obtainable from `http://www.daimi.au.dk/~ivan`.

DK01. Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In *these proceedings*.

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

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

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM STOC*, pages 218–229, New York City, 25–27 May 1987.

GRR98. R. Gennaro, M. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. ACM PODC'98*, 1998.

GV87. O. Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In Carl Pomerance, editor, *Advances in Cryptology - Crypto '87*, pages 73–86, Berlin, 1987. Springer-Verlag. LNCS Vol. 293.

KK91. Kaoru Kurosawa and Motoo Kotera. A multiparty protocol for modulo operations. Technical Report SCIS91-3B, 1991.

Kur91. Kaoru Kurosawa. Zero knowledge interactive proof system for modulo operations. In *IEICE Trans.*, volume E74, pages 2124–2128, 1991.

MH00. Bartosz Przydatek, Martin Hirt, and Ueli M. Maurer. Efficient secure multiparty computation. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, pages 143–161, Berlin, 2000. Springer. LNCS Vol. 1976.

MJ00. Ari Juels and Markus Jakobsson. Mix and match: Secure function evaluation via ciphertexts. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, pages 162–177, Berlin, 2000. Springer. LNCS Vol. 1976.

MR91. S. Micali and P. Rogaway. Secure computation. In Joan Feigenbaum, editor, *Advances in Cryptology - Crypto '91*, pages 392–404, Berlin, 1991. Springer-Verlag. LNCS Vol. 576.

pac96. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

Pai99. P. Paillier. Public-key cryptosystems based on composite degree residue classes. In Jacques Stern, editor, *Advances in Cryptology - EuroCrypt '99*, pages 223–238, Berlin, 1999. Springer-Verlag. LNCS Vol. 1592.

Sho00. Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology - EuroCrypt 2000*, pages 207–220, Berlin, 2000. Springer-Verlag. LNCS Vol. 1807.

Yao82. Andrew C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, 3–5 November 1982. IEEE.