On Secure Two-party Integer Division

Morten Dahl[1], Chao Ning[2,3]*, and Tomas Toft[1]

[1] Aarhus University, Denmark ** *** † {mdahl,ttoft}@cs.au.dk
[2] IIIS, Tsinghua University, Beijing, China ncnfl@mail.tsinghua.edu.cn
[3] School of Computer Science and Technology, Shandong University, Jinan, China

Abstract. We consider the problem of *secure integer division*: given two Paillier encryptions of ℓ-bit values n and d, determine an encryption of $\lfloor \frac{n}{d} \rfloor$ without leaking any information about n or d. We propose two new protocols solving this problem.
The first requires $\mathcal{O}(\ell)$ arithmetic operations on encrypted values (secure addition and multiplication) in $\mathcal{O}(1)$ rounds. This is the most efficient constant-rounds solution to date. The second protocol requires only $\mathcal{O}\left((\log^2 \ell)(\kappa + \operatorname{loglog} \ell)\right)$ arithmetic operations in $\mathcal{O}(\log^2 \ell)$ rounds, where κ is a correctness parameter. Theoretically, this is the most efficient solution to date as all previous solutions have required $\Omega(\ell)$ operations. Indeed, the fact that an $o(\ell)$ solution is possible at all is highly surprising.

Key words: Secure two-party computation, Secure integer division

1 Introduction

Secure multiparty computation (MPC) allows two or more mutually mistrusting parties to evaluate a function on private data without revealing additional information. Classic results show that any function can be computed with polynomial overhead but specialised protocols are often used to improve efficiency: integer arithmetic can for instance be *simulated* using $\mathbb{Z}_M$ arithmetic. On the other hand, this makes non-arithmetic operations difficult, including determining which of two sums is the larger (as needed in the double auction of Bogetoft et al. [BCD+09]), or performing an integer division of sums (essentially the *computation of the mean* problem of Kiltz et al. [KLM05]).

[*] This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174 and 61173139.

[**] The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed.

[***] The authors acknowledge support from the Center for research in the Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council.

[†] The authors acknowledge support from Confidential Benchmarking (COBE), supported by The Danish Research Council for Technology and Production.

In this paper we consider the problem of secure integer division – computing $\lfloor n/d \rfloor$ given n and d – in the two-party setting. Immediate applications include statistics on data from companies in the same line of business, as well as data-mining tasks, e.g., the k-means clustering protocol of Jagannathan and Wright [JW05]. Further, since the problem of secure integer division is equivalent to that of secure modulo reduction – $n \bmod m = n - m \cdot \lfloor n/m \rfloor$ – any such protocol may be utilized in joint key-generation, e.g., as done by Algesheimer et al. [ACS02].

Related work. Algesheimer et al. introduced the problem of secure integer division in the context of passively secure RSA-modulus generation with honest majority [ACS02]; active security is achievable using standard techinques. Their solution was based on Newton iteration and required $\mathcal{O}(\ell)$ work and communication (using the notation of this paper) in $\mathcal{O}(\log \ell)$ rounds, where ℓ is the bit-length of the inputs. The protocols were implemented by From and Jakobsen in the passively secure three-party setting [FJ05]. Recently, Catrina and Dragulin have used similar ideas to construct secure fixed-point arithmetic [CD09].

Regarding constant-rounds solutions, Kiltz et al. proposed specialised protocols based on Taylor series for the related, but simpler, problem of computing the means in a two-party setting [KLM05]. Damgård et al. [DFK$^+$06] observed that combining the ideas of [ACS02,KLM05] and bit-decomposition (BD) implied constant-rounds modulo reduction and hence integer division. No details were presented, though naturally complexity was at least that of BD, $\mathcal{O}(\ell \log \ell)$.

The simpler problem where d is known to all parties (a single party) has been studied by Guajardo et al. [GMS10] and Ning and Xu [NX10] (Veugen [Veu10]).

Finally, we remark that it is possible to "switch technique" mid-protocol and use homomorphic encryption for arithmetic and (small) Yao circuits for primitives such as integer division as done by Henecka et al. [HKS$^+$10]. However, achieving active security in this setting typically requires the use of cut-and-choose techniques. Moreover, while it is possible to use generic non-interactive zero-knowledge proofs to demonstrate correct protocol execution to independent observers – e.g. clients which have supplied the inputs as in the Danish "sugar beet auction" [BCD$^+$09] – this will be much more expensive than using non-generic zero-knowledge proofs as our solution allows.

Contribution. We present two two-party protocols for the problem of secure integer division: given Paillier encryptions of ℓ-bit values n and d, compute an encryption of $\lfloor n/d \rfloor$ without leaking any information. Both are based on Taylor series. The first protocol requires $\mathcal{O}(\ell)$ encryptions to be exchanged between the parties in a constant number of rounds; this is quite practical for small inputs, e.g., up to 40 bits. The second protocol communicates $\mathcal{O}\left((\log^2 \ell)(\kappa + \operatorname{loglog} \ell)\right)$ encryptions in $\mathcal{O}(\log^2 \ell)$ rounds. Moreover, we are able to avoid bit-decomposition; indeed, as the latter complexity is *sub-linear* in the bit-length, it precludes the use of bit-decomposition. That a sub-linear solution is possible at all is quite surprising, but the construction is of theoretical rather than practical interest.

Though our protocols are presented in the two-party Pailier-based setting, they are applicable in other settings providing secure arithetmic, e.g. the protocols of Ben-Or et al. [BGW88]. However, the sub-linear solution requires the presence of two mutually incorruptible parties, at least with current knowledge.

Acknowledgements. The authors would like to thank Troels Sørensen, Jesper Buus Nielsen, and the anonymous reviewers for their comments and suggestions.

2 Preliminaries

After presenting Paillier encryption and secure two-party computations we introduce a set of protocols used in our constructions. All sub-protocols are secure against malicious (i.e., potentially deviating) attackers. Regarding complexity, we shall use R_π and C_π to denote respectively the number of rounds used and the number of ring elements communicated during a single run of protocol π.

Paillier encryption. Paillier's encryption scheme [Pai99] is an additively homomorphic, sematically secure public key encryption scheme based on the decisional composite residuosity assumption of RSA-moduli. Suppressing the randomness used for encryption, we write $[m]$ to denote an encryption of m.

Secure computation. Secure multi-party computation can be based on Paillier encryption with a threshold key using the protocols of Cramer et al. [CDN01]. The threshold sharing can be constructed using the ideas of Damgård and Jurik [DJ01]. Though not explicitly stated, apart from guaranteed termination, the protocols of [CDN01] are still valid even if all but a single party are corrupt. In particular this allows the two-party setting. We assume the following setting:

- Alice and Bob know a public Paillier key and share the decryption key.
- Inputs and intermediary values are held in encrypted form by *both* parties.

Paillier encryption is additively homomorphic, hence given $[m]$ and $[m']$ both parties may compute an encryption $[m + m']$. We will use infix operations in the plaintext space and write $[m + m'] \leftarrow [m] + [m']$ for this operation. To perform a multiplication, the parties need to run a protocol; see [CDN01] for details.

Zero-knowledge proof of boundedness. In addition to secure arithmetic in $\mathbb{Z}_M$ we require a zero-knowledge proof of boundedness, i.e. that Alice and Bob may demonstrate to each other that the plaintext of an encryption $[m]$ sent to the other party (where the sender knows m) is smaller than some public bound B. For Paillier encryption this can be achieved with $\mathcal{O}(1)$ communication (of ring elements) using integer commitments and the fact that any non-negative integer can be written as a sum of four squares. See [Bou00,Lip03] for further discussion.

Computing the greater-than relation. Given encryptions $[m]$ and $[m']$ of ℓ-bit values, obtain an encryption $[b]$ of a bit b such that $b = 1$ iff $m > m'$. A constant-rounds protocol $\pi^{c}_{>?}$ for this can be based off of the comparison protocol of Nishide and Ohta [NO07]; communication complexity is $C_{\pi^c_{>?}} = \mathcal{O}(\ell)$ ring elements. We use $\pi^c_{\leq ?}$ as syntactic sugar for running $\pi^c_{>?}$ with inputs swapped.

A sub-linear protocol, denoted $\pi^s_{>?}$ and $\pi^s_{\leq ?}$, is possible due to Toft [Tof11]. Its complexity is $C_{\pi^s_{>?}} = \mathcal{O}\left((\log \ell)(\kappa + \text{loglog}\, \ell)\right)$ ring elements in $R_{\pi^s_{>?}} = \mathcal{O}(\log \ell)$ rounds, where κ is a correctness parameter.

Computing the inverse of an element. Given an encryption $[x]$ of $x \in \mathbb{Z}^*_M$, compute an encryption $\left[x^{-1}\right]$ of its inverse. We use the protocol from [BB89] which performs this task in a constant number of rounds and communicating a constant number of field elements. We shall use this protocol in both the constant-rounds and the sub-linear protocol and hence simply denote it by π_{inv}.

Bit-decompositon (BD). Decomposing an encrypted ℓ-bit value $[m]$ into binary form – i.e. computing bits $[m_{\ell-1}], \ldots, [m_0]$ such that $m = \sum_{i=0}^{\ell-1} 2^i \cdot m_i$ – is not strictly required (details appear in the full version) but we use it here for clarity. We denote by π^c_{BD} the BD protocol of Reistad and Toft [RT10]; this uses $C_{\pi^c_{\text{BD}}} = \mathcal{O}(\ell)$ communication.

Prefix-or of a sequence of bits. Given encrypted bits $[x_{\ell-1}], \ldots, [x_0]$, compute encrypted bits $[y_{\ell-1}], \ldots, [y_0]$ such that $y_i = \bigvee_{j=i}^{\ell-1} x_j$. An $\mathcal{O}(1)$-rounds protocol communicating $C_{\pi^c_{\text{pre-}\vee}} = \mathcal{O}(\ell)$ elements, $\pi^c_{\text{pre-}\vee}$, is provided in [DFK$^+$06].

Powers of a number. Given an encrypted number $[x]$ and public $\omega \in \mathbb{Z}$, compute $\left[x^1\right], \left[x^2\right], \ldots, [x^\omega]$. $\pi^c_{\text{pre-}\Pi}$ achieves this using $C_{\pi^c_{\text{pre-}\Pi}} = \mathcal{O}(\omega)$ communication in $\mathcal{O}(1)$ rounds using a prefix-product computation, [BB89,DFK$^+$06].

3 The Intuition Behind the Constructions

In this section we take a high-level view and present the ideas behind the desired computation. The following sections then explain how to do this securely in the stated complexity. Assume in the following that n and d are ℓ-bit integers, and let k be a suitable large, public integer. Our solutions then consist of two steps:

I. Compute an encrypted approximation $[\tilde{a}]$ of $a = \lfloor 2^k/d \rfloor$
II. Compute $[\lfloor n/d \rfloor]$ as $\left\lfloor ([\tilde{a}] \cdot [n])/2^k \right\rfloor$

Step I is explained over the reals in Section 3.1. This is then converted to integer computation in Section 3.2 and finally realised using $\mathbb{Z}_M$ arithmetic in Section 3.3. Note that the integer division in step II is simpler as 2^k is public.

3.1 The Taylor Series

As in [KLM05] or the constant depth division circuit of Hesse et al. [HAB02], we start with a geometric series to compute a "k-shifted" approximation of $1/d$:

$$\frac{1}{\alpha} = \sum_{i=0}^{\infty} (1-\alpha)^i = \sum_{i=0}^{\omega} (1-\alpha)^i + \epsilon_\omega \tag{1}$$

where $\epsilon_\omega = \sum_{i=\omega+1}^{\infty}(1-\alpha)^i$. This is easily verified for any real $0 < \alpha < 1$. Further, approximating $1/\alpha$ by keeping only the first $\omega + 1$ terms of the summation introduces an additive error of ϵ_ω. If $0 < 1 - \alpha \leq 1/2$ then this error is at most

$$\epsilon_\omega = \sum_{i=\omega+1}^{\infty} (1-\alpha)^i \; = \; (1-\alpha)^{\omega+1} \cdot \sum_{i=0}^{\infty} (1-\alpha)^i \; \leq \; 2^{-\omega-1} \cdot \frac{1}{\alpha} \; \leq \; 2^{-\omega}. \qquad (2)$$

By picking ω sufficiently large this ensures an appropriately small error below.

3.2 Converting the Taylor Series to an Integer Computation

Multiplying $1/\alpha$ by a power of two "shifts" the value; this ensures that each of the $\omega + 1$ terms of the finite sum of Eq. (1) are integer. The non-integer part of the shifted value is entirely contained in ϵ_ω, which will be discarded.

Let $\ell_d = \lfloor \log_2(d) + 1 \rfloor$ be the bit-length of d, i.e. $2^{\ell_d - 1} \leq d < 2^{\ell_d}$; define ℓ_n similarly. Any $\omega \geq max\{\ell_n - \ell_d, 0\}$ provides sufficient accuracy, however, the public ω cannot depend on the secret ℓ_n and ℓ_d. Thus, we let $\omega = \ell \geq \ell_n - \ell_d$. For $\alpha = d/2^{\ell_d}$ and $k = \ell^2 + \ell$ the following provides $1/d$ shifted up by k bits:

$$\frac{2^k}{d} \quad = \quad 2^{k-\ell_d} \cdot \frac{1}{d/2^{\ell_d}} \quad = \quad \left(2^{k-\ell_d(\omega+1)} \sum_{i=0}^{\omega} \left(2^{\ell_d} - d\right)^i \cdot 2^{\ell_d(\omega-i)} \right) + 2^{k-\ell_d} \cdot \epsilon_\omega.$$

We define the desired approximation of $2^k/d$ as

$$\tilde{a} = 2^{k-\ell_d(\omega+1)} \cdot \sum_{i=0}^{\omega} \left(2^{\ell_d} - d\right)^i \cdot 2^{\ell_d(\omega-i)}. \qquad (3)$$

Note that not only is this an integer since $k \geq \ell_d(\omega + 1)$ and $2^{\ell_d} > d$, it may also be computed as the product of $2^{k-\ell_d(\omega+1)}$ and the evaluation of the integer polynomial with coefficients $2^{\ell_d(\omega-i)}$ for $0 \leq i \leq \omega$ at point $2^{\ell_d} - d$. Furthermore, since $0 < 1 - d/2^{\ell_d} \leq 1/2$ we have a bound on the additive error by Eq. (2):

$$2^{k-\ell_d} \cdot \epsilon_\omega \leq 2^{k-\ell_d-\omega}.$$

This ensures that the result computed in step II is off by at most 1; we have:

$$\left\lfloor \frac{n}{d} \right\rfloor = \left\lfloor \frac{n \cdot \left(\tilde{a} + 2^{k-\ell_d} \cdot \epsilon_\omega\right)}{2^k} \right\rfloor = \left\lfloor \frac{n \cdot \tilde{a}}{2^k} + \frac{n \cdot 2^{k-\ell_d} \cdot \epsilon_\omega}{2^k} \right\rfloor \qquad (4)$$

and see that the second summand is bound by

$$\frac{n \cdot 2^{k-\ell_d} \cdot \epsilon_\omega}{2^k} \leq \frac{n \cdot 2^{k-\ell_d-\omega}}{2^k} < \frac{2^k}{2^k} = 1$$

since $\ell_n \leq \omega$. $\lfloor \frac{n \cdot \tilde{a}}{2^k} \rfloor$ is the desired result *except* that the sum of the error, $n \cdot 2^{k-\ell_d} \cdot \epsilon_\omega$, and the discarded bits of the approximation, $n \cdot \tilde{a} \bmod 2^k$, may be greater than 2^k; i.e. there may be an additive error of -1 due to a lost carry.

To recap: Given integers $2^{k-\ell_d(\omega+1)}$, $2^{\ell_d} - d$ and $2^{\ell_d(\omega-i)}$ for $0 \leq i \leq \ell$, performing step I yields an approximation $\tilde{a}$ of $2^k/d$ using Eq. (3). Down-shifting this almost gives the desired result, namely $\tilde{q} \in \{q, q-1\}$, where $q = \lfloor n/d \rfloor$.

3.3 Performing the Integer Computation Using $\mathbb{Z}_M$ Arithmetic

The underlying primitives provide secure $\mathbb{Z}_M$ arithmetic, with $M = p \cdot q$ being the Paillier key whose secret key is held jointly by the parties. We assume[4] that

$$M \gg 2^{\ell^2+\ell+\kappa_s},$$

where κ_s is a statistical security parameter, e.g. $\kappa_s = 100$. This implies that no "overflow" modulo M occurs in Eq. (3), hence it can be seen as occurring in $\mathbb{Z}_M$. However, for efficiency reasons we rephrase the expression as

$$\tilde{a} = 2^{k-\ell_d(\omega+1)} \cdot \sum_{i=0}^{\omega} \left(2^{\ell_d} - d\right)^i \cdot 2^{\ell_d(\omega-i)} = 2^{k-\ell_d} \cdot \sum_{i=0}^{\omega} \left(\left(2^{\ell_d} - d\right) \cdot 2^{-\ell_d}\right)^i \quad (5)$$

where addition and multiplication occur in $\mathbb{Z}_M$. Although this should no longer be seen as an integer computation, the key observation is that it is irrelevant *how* the encryption $[\tilde{a}]$ is obtained; what matters is that the plaintext is correct. Essentially this altered calculation can be viewed as using the encoding of rational values suggested in [FSW02]. Note that this simplifies the desired calculation: we now only need the values $2^{k-\ell_d}$, $2^{\ell_d} - d$, and $2^{-\ell_d}$ as well as the evaluation of a $\mathbb{Z}_M$-polynomial with known coefficients (all equal to 1).

4 The Overall Division Protocol

Having presented the desired $\mathbb{Z}_M$-expression for computing the approximation $\tilde{a} \approx 2^k/d$ in Section 3.3 above, the goal now is to give a high-level view of the actual protocol. We first formalise the required sub-tasks, and then present the overall protocol based on assumed protocols for these. Instantiating these protocols with either the constant-rounds (Section 5) or the sub-linear (Section 6) versions of the sub-protocols we obtain our two division protocols.

4.1 Sub-tasks and Sub-protocols

In addition to the basic primitives of Section 2 we require the following sub-protocols:

- π_{BL}: Given an encryption $[d]$ of an ℓ-bit value d, determine an encryption $\left[2^{\ell_d}\right]$ for $\ell_d = \lfloor log_2(d) + 1 \rfloor$
- π_{poly}: Given an encryption $[p]$ of $p \in \mathbb{Z}_M^*$, evaluate the known polynomial $A(x) = \sum_{i=0}^{\omega} x^i$ over $\mathbb{Z}_M$ securely at point p, i.e. compute encryption $[A(p)]$
- π_{trunc}: Given an encryption $[\hat{q}]$ of an $(\ell + k)$-bit value $\hat{q} \in \mathbb{Z}_M$, compute an encryption $[\tilde{q}]$ of an approximation of $\lfloor \hat{q}/2^k \rfloor$ s.t. $\tilde{q} = \lfloor \hat{q}/2^k \rfloor + \epsilon$ for $\epsilon \in \{0, 1\}$.

[4] M needs to be at least a thousand bits long to ensure security of the Paillier scheme and hence this assumption is not as bad as it may appear at first glance.

4.2 The High-level View

The full division protocol is seen in Figure 1 and proceeds by the following steps:

I. Compute an encryption $[\tilde{a}]$ of the approximation
   (a) Determine $\left[2^{\ell_d}\right]$ and in turn compute $\left[2^{k-\ell_d}\right]$ and $[p] = \left[(2^{\ell_d} - d) \cdot 2^{-\ell_d}\right]$
   (b) Evaluate the polynomial of Eq. (5) in $[p]$ and securely multiply by $\left[2^{k-\ell_d}\right]$
II. Compute $[\lfloor n/d \rfloor]$
   (a) Obtain encryption $[\tilde{q}]$ of $\tilde{q} \approx \lfloor n/d \rfloor$ by computing and truncating $[n \cdot \tilde{a}]$
   (b) Eliminate errors introduced by approximations, i.e., compute $[q]$ from $[\tilde{q}]$

where the elimination of errors are performed by two secure comparions.

A: sk_A $\qquad$ $pk = M, [n], [d]$ $\qquad$ B: sk_B

$$\left[2^{\ell_d}\right] \leftarrow \pi_{\text{BL}}\left([d]\right)$$
$$\longleftrightarrow$$
$$\left[2^{-\ell_d}\right] \leftarrow \pi_{\text{inv}}\left(\left[2^{\ell_d}\right]\right)$$
$$\longleftrightarrow$$
$$[p] \leftarrow \left(\left[2^{\ell_d}\right] - [d]\right) \cdot \left[2^{-\ell_d}\right]$$
$$\longleftrightarrow$$
$$[\tilde{a}] \leftarrow 2^k \cdot \left[2^{-\ell_d}\right] \cdot \pi_{\text{poly}}([p])$$
$$\longleftrightarrow$$

$$[\hat{q}] \leftarrow [n] \cdot [\tilde{a}]$$
$$\longleftrightarrow$$
$$[\tilde{q}] \leftarrow \pi_{\text{trunc}}([\hat{q}], k)$$
$$\longleftrightarrow$$

$$[r] \leftarrow [n] - [d] \cdot [\tilde{q}]$$
$$\longleftrightarrow$$
$$[\epsilon_+] \leftarrow \pi_{\leq ?}\left([d] + [d], [r] + [d]\right)$$
$$\longleftrightarrow$$
$$[\epsilon_-] \leftarrow \pi_{> ?}\left([d], [r] + [d]\right)$$
$$\longleftrightarrow$$
$$[q] \leftarrow [\tilde{q}] + [\epsilon_+] - [\epsilon_-]$$

Fig. 1. The full division protocol, $\pi_{\text{div}}([n], [d]) \mapsto [\lfloor n/d \rfloor]$

Correctness. Correctness follows almost entirely from the previous section. For the plaintext of $[\hat{q}]$, the most significant bits are off by at most 1:

$$\lfloor \hat{q}/2^k \rfloor \in \{\lfloor n/d \rfloor, \lfloor n/d \rfloor - 1\}.$$

The execution of π_{trunc} may introduce an additional additive error, i.e. we have

$$\tilde{q} \in \{\lfloor n/d \rfloor - 1, \lfloor n/d \rfloor, \lfloor n/d \rfloor + 1\}.$$

Using $r = n - d \cdot \tilde{q} \in [-d; 2d[$ we can securely determine which case we are in. Namely, $\tilde{q} + 1 = \lfloor n/d \rfloor$ when $d \leq r$ and $\tilde{q} - 1 = \lfloor n/d \rfloor$ when $0 > r$. In order to deal only with positive integers we scale these tests to respectively $2d \leq r + d$ and $d > r + d$. Letting ϵ_+ and ϵ_- denote the Boolean outcome of these tests, it follows that $q = \tilde{q} + \epsilon_+ - \epsilon_- = \lfloor n/d \rfloor$.

Privacy. The protocol reveals no information about the inputs (other than the desired encryption of the result). This follows from the fact that no value is ever decrypted and that we only invoke secure sub-protocols which do not leak information. We note that π_{inv} and π_{poly} require the input to be invertible – this is indeed the case as M is the product of two odd primes, $p, q \approx \sqrt{M}$, while $2^{\ell_d}, 2^{\ell_d} - d \leq 2^{\ell} \ll \sqrt{M}$. Further, the input $[n \cdot \tilde{a}]$ for the truncation is $\ell + k$-bit long as $n < 2^{\ell}$ and $\tilde{a} \leq 2^k/d \leq 2^k$, and hence the input is of the correct size.

A formal security proof using the real/ideal paradigm requires the construction of a simulator for each party. These are straightforward to construct from the simulators of the sub-protocols; as our protocol consists of the sequential evaluation of sub-protocols, the overall simulator simply consists of the sequential execution of the simulators of these.

Complexity. The complexity depends on the details of the sub-protocols π_{BL}, π_{poly}, π_{trunc}, and $\pi_{>?}$. Formally we have

$$\begin{aligned} R_{\pi_{\mathrm{div}}} &= R_{\pi_{\mathrm{BL}}} + R_{\pi_{\mathrm{inv}}} + R_{\pi_{\mathrm{poly}}} + R_{\pi_{\mathrm{trunc}}} + 2 \cdot R_{\pi_{>?}} + 3 \cdot R_{\pi_{\mathrm{mult}}} \\ &= R_{\pi_{\mathrm{BL}}} + R_{\pi_{\mathrm{poly}}} + R_{\pi_{\mathrm{trunc}}} + \mathcal{O}(R_{\pi_{>?}}) + \mathcal{O}(1) \\ C_{\pi_{\mathrm{div}}} &= C_{\pi_{\mathrm{BL}}} + C_{\pi_{\mathrm{inv}}} + C_{\pi_{\mathrm{poly}}} + C_{\pi_{\mathrm{trunc}}} + 2 \cdot C_{\pi_{>?}} + 3 \cdot C_{\pi_{\mathrm{mult}}} \\ &= C_{\pi_{\mathrm{BL}}} + C_{\pi_{\mathrm{poly}}} + C_{\pi_{\mathrm{trunc}}} + \mathcal{O}(C_{\pi_{>?}}) + \mathcal{O}(1) \end{aligned} \tag{6}$$

such that for the constant-rounds instantiation we get $R_{\pi^c_{\mathrm{div}}} = R_{\pi^c_{\mathrm{BL}}} + R_{\pi^c_{\mathrm{poly}}} + R_{\pi^c_{\mathrm{trunc}}} + \mathcal{O}(1)$ and $C_{\pi^c_{\mathrm{div}}} = C_{\pi^c_{\mathrm{BL}}} + C_{\pi^c_{\mathrm{poly}}} + C_{\pi^c_{\mathrm{trunc}}} + \mathcal{O}(\ell)$. Likewise, for the sublinear instantiation we get $R_{\pi^s_{\mathrm{div}}} = R_{\pi^s_{\mathrm{BL}}} + R_{\pi^s_{\mathrm{poly}}} + R_{\pi^s_{\mathrm{trunc}}} + \mathcal{O}(log\ \ell)$ and $C_{\pi^s_{\mathrm{div}}} = C_{\pi^s_{\mathrm{BL}}} + C_{\pi^s_{\mathrm{poly}}} + C_{\pi^s_{\mathrm{trunc}}} + \mathcal{O}\left((\log \ell)(\kappa + \mathrm{loglog}\, \ell)\right)$. Finally, a slight optimisation regarding rounds is possible by invoking $\pi_{>?}$ and $\pi_{\leq ?}$ in parallel.

Active Security. The protocol in Figure 1 is only passively secure. However, obtaining active security is straightforward by executing appropriate ZK proofs. This increases the communication complexity by a constant factor.

5 The Constant-rounds Protocol

In this section we plug in protocols for the three sub-tasks. All protocols use a constant number of rounds and linear communication. Combined with the previous section this provides a constant-rounds protocol for division.

5.1 The constant-rounds π_{BL} protocol

In the full version of this paper [DNT12] we give a π^c_{BL} protocol that, somewhat surprising, does not rely on bit-decomposition. However, for clarity the π^c_{BL} protocol presented here in Figure 2 is composed of two protocols introduced in Section 2: π^c_{BD} and $\pi^c_{\text{pre-}\vee}$. To recap, given $[d]$ the former returns a vector of encrypted bits $[x_{\ell-1}], \ldots, [x_0]$ for which it holds that $\sum_{i=0}^{\ell-1} x_i \cdot 2^i = d$. The latter takes such a vector of encrypted bits and returns another such that $y_i = \bigvee_{j=i}^{\ell-1} x_j$.

A: sk_A $pk = M, [d]$ B: sk_B

$$\underset{\longleftrightarrow}{[x_{\ell-1}], \dots, [x_0] \leftarrow \pi^c_{\mathrm{BD}}([d])}$$

$$\underset{\longleftrightarrow}{[y_{\ell-1}], \dots, [y_0] \leftarrow \pi^c_{\text{pre-}\vee}([x_{\ell-1}], \dots, [x_0])}$$

$$\left[2^{\ell_d}\right] \leftarrow 1 + \textstyle\sum_{i=0}^{\ell-1} [y_i] \cdot 2^i$$

Fig. 2. Constant-rounds bit-length protocol, $\pi^c_{\mathrm{BL}}([d]) \mapsto \left[2^{\ell_d}\right]$

Correctness. By the correctness of the two sub-protocols we only have to argue the correctness of the final step. Note that the result of $\pi^c_{\text{pre-}\vee}$ is a set such that $y_i = 1$ if and only if $d \geq 2^i$. This means that $1 + \sum_{i=0}^{\ell-1} y_i \cdot 2^i$ is the desired 2^{ℓ_d}.

Privacy and Active Security. Follows immediately by the privacy and security guarantees of the two sub-protocols.

Complexity. Since the final step of π^c_{BL} is a local computation we simply have that $R_{\pi^c_{\mathrm{BL}}} = R_{\pi^c_{\mathrm{BD}}} + R_{\pi^c_{\text{pre-}\vee}} = \mathcal{O}(1)$ and $C_{\pi^c_{\mathrm{BL}}} = C_{\pi^c_{\mathrm{BD}}} + C_{\pi^c_{\text{pre-}\vee}} = \mathcal{O}(\ell)$.

5.2 The constant-rounds $\pi_{\mathbf{poly}}$ protocol

As shown in the protocol in Figure 3, we simply evaluate polynomial $A(x) = \sum_{i=0}^{\omega} x^i$ in point $p = \left(2^{\ell_d} - d\right) \cdot 2^{-\ell_d}$ using the prefix-product protocol $\pi^c_{\text{pre-}\Pi}$. This gives encryptions of $p^1, p^2, \dots, p^\omega$ – and knowing these, all there is left to do is to sum them together with $p^0 = 1$ to form $A(p)$.

A: sk_A $pk = M, [x]$ B: sk_B

$$\underset{\longleftrightarrow}{\left[p^1\right], \dots, \left[p^\omega\right] \leftarrow \pi^c_{\text{pre-}\Pi}([x], \dots, [x])}$$

$$[y] \leftarrow 1 + \textstyle\sum_{i=1}^{\omega} [p_i]$$

Fig. 3. Constant-rounds polynomial evaluation protocol, $\pi^c_{\mathrm{poly}}([x]) \mapsto [A(x)]$

Correctness, Privacy, Complexity, and Active Security. Noting that the second step of π^c_{poly} is a local computation, all properties directly reflect those of the $\pi^c_{\text{pre-}\Pi}$ subprotocol. Formally, $R_{\pi^c_{\mathrm{poly}}} = \mathcal{O}(1)$ and $C_{\pi^c_{\mathrm{poly}}} = \mathcal{O}(\omega)$.

5.3 The constant-rounds $\pi_{\mathbf{trunc}}$ protocol

Our constant-rounds protocol for truncation (shown in Figure 4) takes encryption $[\hat{q}]$ and public k as input and returns $[\tilde{q}]$ such that $\tilde{q} \approx \lfloor q/2^k \rfloor$. The result may have an additive error $c \leq 1$. It is possible to eliminate this error with a comparison $[c] \leftarrow ([\tilde{q}] \cdot 2^k >_? [\hat{q}])$, and computing the correct result as $[q] \leftarrow [\tilde{q}] - [c]$. However, instead of comparing two ℓ^2-bit numbers here, we handle the error in the main protocol with a comparison of two ℓ-bit numbers instead.

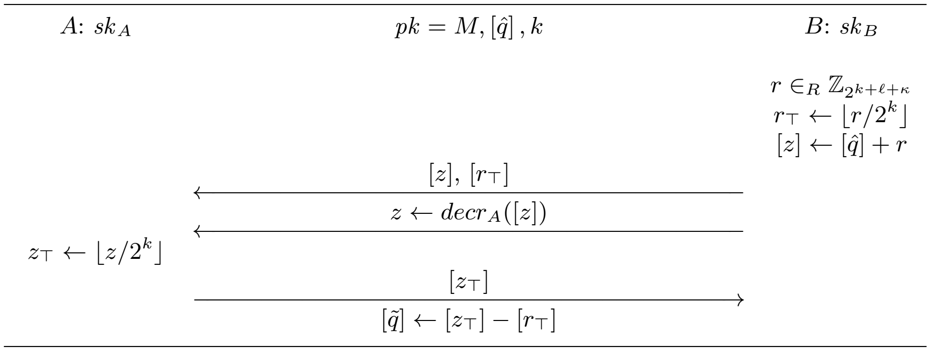
A: sk_A
$pk = M, [\hat{q}], k$
B: sk_B
$r \in_R \mathbb{Z}_{2^{k+\ell+\kappa}}$
$r_\top \leftarrow \lfloor r/2^k \rfloor$
$[z] \leftarrow [\hat{q}] + r$
$[z]$, $[r_\top]$
$z \leftarrow decr_A([z])$
$z_\top \leftarrow \lfloor z/2^k \rfloor$
$[z_\top]$
$[\tilde{q}] \leftarrow [z_\top] - [r_\top]$

Fig. 4. Constant-rounds truncation protocol, $\pi^c_{\text{trunc}}([\hat{q}], k) \mapsto \left[\lfloor \hat{q}/2^k \rfloor + c\right]$

To perform the truncation, party B first picks a random integer of a bit-length sufficient for using it as a mask for $\hat{q}$. He also stores the $\ell + \kappa$ most significent bits of r as $r_\top$ and computes an encryption of it. Upon receiving $[z]$, the masked value of $\hat{q}$, A and B now decrypt $[z]$ for A to see. After learning this value z, A can locally perform the truncation to form $z_\top$. She sends an encryption of this value to B and both can finally compute the output locally by $[z_\top] - [r_\top]$.

Correctness. When computing z it may happen that r causes a carry bit c from the k least significant bits to spill over into the $\ell + \kappa$ most significant bits. In this case the truncation of z will maintain this carry bit, causing the result of $z_\top - r_\top$ to be $\lfloor \hat{q}/2^k \rfloor + 1$ instead of $\lfloor \hat{q}/2^k \rfloor$. For efficiency we allow this error.

Privacy. The only point where information could potentially be leaked is through A seeing z. However, since r is chosen uniformly at random and κ bit longer than $\hat{q}$, z leaks information about $\hat{q}$ with probability negligible in κ.

Complexity. We see that the complexity of π^c_{trunc} is $R_{\pi^c_{\text{trunc}}} = 2 + R_{decr} = \mathcal{O}(1)$ where R_{decr} is the round complexity of a decryption, assumed to be constant. Likewise the communication complexity is $C_{\pi^c_{\text{trunc}}} = 3 + C_{decr} = \mathcal{O}(1)$.

Active Security. To obtain active security B must also send $\left[r_\perp = r \; mod \; 2^k\right]$ to A, who in turn must also send $\left[z_\perp = z \; mod \; 2^k\right]$. B can now append a zero-knowledge proof that $z = (r_\top \cdot 2^k + r_\perp) + \hat{q}$ as well as proofs that both $r_\top$ and $r_\perp$ are within the correct bounds. Similary, A also appends a proof of $z = z_\top \cdot 2^k + z_\perp$ and that $z_\top$ and $z_\perp$ are within bounds.

5.4 Combined Protocol and Analysis

By plugging the protocols introduced in this section into the π_{div} protocol of Section 4 we obtain our constant-rounds division protocol π^c_{div}. Correctness, privacy, and active security follow from the discussions above. Using the complexity expressions in Eq. 6 from Section 4 and the fact that $\omega = \ell$ we get:

$$R_{\pi^c_{\text{div}}} = R_{\pi^c_{\text{BL}}} + R_{\pi^c_{\text{poly}}} + R_{\pi^c_{\text{trunc}}} + \mathcal{O}(1) = \mathcal{O}(1)$$

$$C_{\pi^c_{\text{div}}} = C_{\pi^c_{\text{BL}}} + C_{\pi^c_{\text{poly}}} + C_{\pi^c_{\text{trunc}}} + \mathcal{O}(\ell) = \mathcal{O}(\omega) + \mathcal{O}(\ell) = \mathcal{O}(\ell).$$

6 The Sub-linear Protocol

In this section we give the protocols needed for giving the division protocol of Section 3 a sub-linear communication complexity. We can reuse the truncation protocol π^c_{trunc} from Section 5 and hence only present two new π_{BL} and π_{poly} protocols.

6.1 The sub-linear π_{BL} protocol

To compute $\left[2^{\ell_d}\right]$ from $[d]$ in sub-linear communication complexity we take inspiration from [Tof11] and perform, in a sense, a binary search. Assuming we have a protocol $\pi^s_{\leq ?}$ for performing comparison of two encrypted numbers, we give the protocol in Figure 5. For simplicity we assume that $\ell = 2^\gamma$ for some integer γ.

Intuitively, our construction recursively computes a pointer p into the binary representation of d. Initially p points to the first bit position ($p_0 = 2^0$). In the first round we then ask in which half of the binary representation of d the most significant 1 occurs and store the result in bit c_1. Next we update p to point to position $\ell/2^1$ if $c = 1$ (i.e. $p_1 = p_0 \cdot 2^{\ell/2^1}$) and to the same position as before if $c = 0$ (i.e. $p_1 = p_0 \cdot 1$). Iterating in this way p will eventually point to the position of the most significant bit of d. Shifting the position by one will give us integer 2^{ℓ_d}.

Correctness and Privacy. Correctness follows from the above description of the protocol, and privacy follows immediately from the sub-protocols as we only compute on encrypted values.

Complexity. The protocol requires $\gamma = \log_2 \ell$ iterations, each requiring one comparison and one multiplication (not counting multiplication by public values). Hence we get round complexity $R_{\pi^s_{\text{BL}}} = \gamma \cdot (R_{\pi^s_{\leq ?}} + R_{\pi_{\text{mult}}}) = \mathcal{O}(\log^2 \ell)$ and communication complexity $C_{\pi^s_{\text{BL}}} = \gamma \cdot (C_{\pi^s_{\leq ?}} + C_{\pi_{\text{mult}}}) = \mathcal{O}\left((\log^2 \ell)(\kappa + \text{loglog}\, \ell)\right)$.

Active Security. Since the sub-protocol is actively secure, we only have to append zero-knowledge proofs of correctness to every multiplication in order to make the protocol resistant against active attackers. This increases the number of messages communicated but only by a constant factor.

$$A\text{: } sk_A \qquad pk = M, [d] \qquad B\text{: } sk_B$$

$$[p_0] \leftarrow 1$$

$$[c_1] \leftarrow \pi^s_{\leq ?}\left(2^{\ell/2} \cdot [p_0]\,, [d]\right)$$

$$\longleftrightarrow$$

$$[p_1] \leftarrow [p_0] \cdot \left([c_1] \cdot (2^{\ell/2} - 1) + 1\right)$$

$$\longleftrightarrow$$

$$\vdots$$

$$[c_\gamma] \leftarrow \pi^s_{\leq ?}\left(2^{\ell/2^\gamma} \cdot [p_{\gamma-1}]\,, [d]\right)$$

$$\longleftrightarrow$$

$$[p_\gamma] \leftarrow [p_{\gamma-1}] \cdot \left([c_\gamma] \cdot (2^{\ell/2^\gamma} - 1) + 1\right)$$

$$\longleftrightarrow$$

$$\left[2^{\ell_d}\right] \leftarrow 2 \cdot [p_\gamma]$$

Fig. 5. Sub-linear bit-length protocol, $\pi^s_{\mathrm{BL}}([d]) \mapsto \left[2^{\ell_d}\right]$

6.2 The sub-linear $\pi_{\mathbf{poly}}$ protocol

Evaluating the $A(x) = \sum_{i=0}^{\omega} x^i$ polynomial at a point p can be done by a method similar to "square and multiply". We give the protocol in Figure 6 where for simplicity we have assumed that $\omega = 2^\gamma$ for some integer γ. The intuition behind the notation is that $\sigma_j = \sum_{i=1}^{2^j} x^i$ and $x_j = x^{2^j}$ – it is not hard to see that this is indeed the case. Specifically this gives us that $\sigma_\gamma = \sum_{i=1}^{2^\gamma} x^i$ and hence $\sigma_\gamma + 1 = \left(\sum_{i=1}^{\omega} x^i\right) + 1 = \sum_{i=0}^{\omega} x^i$ as required.

Correctness, Privacy, and Complexity. The first two follow respectively from the description above and from that fact that only arithmetical operations on encryptions are performed. For complexity we have that the protocol requires $\gamma = \log_2 \omega$ iterations with two multiplications in each. Hence the round complexity is $R_{\pi^s_{\mathrm{poly}}} = \gamma \cdot (2 \cdot R_{\pi_{\mathrm{mult}}}) = \mathcal{O}(\log \omega)$, and likewise for the communication complexity $C_{\pi^s_{\mathrm{poly}}} = \gamma \cdot (2 \cdot C_{\pi_{\mathrm{mult}}}) = \mathcal{O}(\log \omega)$.

Active Security. By appending zero-knowledge proofs of correctness to every multiplication we make the protocol resistant against active attackers. This increases the number of messages communicated but only by a constant factor.

6.3 The sub-linear $\pi_{\mathbf{trunc}}$ protocol

The truncation protocol π^c_{trunc} of Section 5 is efficient enought to be reused for the sub-linear protocol π^s_{trunc}: only a single operation is performed, namely the decryption of $[z]$. The remaining operations can be carried out locally.

A: sk_A $\qquad\qquad$ $pk = M, [x]$ $\qquad\qquad$ B: sk_B

$$[\sigma_0] \leftarrow [x]$$

$$[x_0] \leftarrow [x]$$
$$[\sigma_1] \leftarrow ([x_0] + 1) \cdot [\sigma_0]$$
$$\longleftrightarrow$$
$$\vdots$$
$$[x_{\gamma-1}] \leftarrow [x_{\gamma-2}] \cdot [x_{\gamma-2}]$$
$$[\sigma_\gamma] \leftarrow ([x_{\gamma-1}] + 1) \cdot [\sigma_{\gamma-1}]$$
$$\longleftrightarrow$$

$$\left[\sum_{i=0}^{\omega} x^i\right] \leftarrow [\sigma_\gamma] + 1$$

Fig. 6. Sub-linear polynomial evaluation protocol, $\pi^s_{\text{poly}}([x]) \mapsto [A(x)]$

6.4 Combined Protocol and Analysis

Our sub-linear division protocol π^s_{div} is obtained from the π_{div} protocol of Section 4. Correctness, privacy, and active security follow from the discussions in the previous sections and in this section. As for complexity, since $\omega = \ell$, we get:

$$R_{\pi^s_{\text{div}}} = R_{\pi^s_{\text{BL}}} + \cdots + \mathcal{O}(\log \ell) = \mathcal{O}(\log^2 \ell) + \mathcal{O}(\log \omega) + \mathcal{O}(\log \ell) = \mathcal{O}(\log^2 \ell)$$

$$C_{\pi^s_{\text{div}}} = C_{\pi^s_{\text{BL}}} + \cdots + \mathcal{O}\left((\log \ell)(\kappa + \text{loglog}\, \ell)\right) = \mathcal{O}\left((\log^2 \ell)(\kappa + \text{loglog}\, \ell)\right).$$

7 Variations and Extensions

The multiparty case. Though we have presented our protocols in the two-party setting, the ideas are also applicable to the multiparty case, based e.g. on the protocols of [CDN01]. Arithmetic operations on encrypted values are immediate, hence we must only consider π_{BL}, π_{trunc}, and the sublinear comparison $\pi_{>?}$.

For the constant-rounds protocol we may use the arithmetic-based comparison of [NO07] while π_{BL} is essentially the bit-decomposition of [RT10]. Thus, these immediately work in the multiparty setting. The π_{trunc} protocol in Figure 4 can be jointly played by the parties. Part A is played publicly and part B is played using the protocols of [CDN01]. First each party P_i $(1 \leq i \leq n)$ supplies an encryption of a random value $\left[r^{(i)}\right]$ as well as $\left[r^{(i)}_\top\right]$ with plaintext $\left\lfloor r^{(i)}/2^k \right\rfloor$. The parties then compute and decrypt $[z] \leftarrow [\hat{q}] + \sum_{i=1}^{n} \left[r^{(i)}\right]$ and in turn $[\hat{q}] \leftarrow \left\lfloor z/2^k \right\rfloor - \sum_{i=1}^{n} \left[r^{(i)}_\top\right]$. This is the right result plus an additive error originating from a carry in the addition of r. Since r is a sum itself, the possible error grows linearly in the number of parties. However, as in the main protocol (Figure 1) this may be corrected using a number of secure comparisons.

With the additional requirement of two named and mutually incorruptible parties, the sub-linear case follows analogously by the protocols of [Tof11]. Since

π_{BL} is based on comparison and arithmetic, and π_{trunc} is the same as the constant-rounds case, a sub-linear multiparty protocol is possible too.

Unconditionally secure integer division. Unconditionally secure variations of our protocols are possible, based e.g. on Shamir's secret sharing scheme and the protocols of Ben-Or et al. [Sha79,BGW88]. The construction is straightforward as all sub-protocols are applicable in this setting as well.

Improving the complexity of the sub-linear protocol. Using the other comparison protocol given in [Tof11] we may obtain slightly better bounds on our division protocol, namely $\mathcal{O}(\log \ell)$ rounds and $\mathcal{O}\left((\log \ell)\sqrt{\ell}(\kappa + \log \ell)\right)$ communications.

References

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432. Springer, 2002.

[BB89] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In P. Rudnicki, editor, *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 201–209, New York, 1989. ACM Press.

[BCD$^+$09] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In R. Dingledine and P. Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.

[Bou00] F. Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807, pages 431–444, 2000.

[CD09] O. Catrina and C. Dragulin. Multiparty computation of fixed-point multiplication and reciprocal. *Database and Expert Systems Applications, International Workshop on*, 0:107–111, 2009.

[CDN01] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045, pages 280–300, 2001.

[DFK$^+$06] I. Damgård, M. Fitzi, E. Kiltz, J. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In S. Halevi and T. Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304. Springer Berlin / Heidelberg, 2006.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.

[DNT12] M. Dahl, C. Ning, and T. Toft. On secure two-party integer division. Technical report, 2012. Available at `http://eprint.iacr.org/2012/164`.

[FJ05] S. From and T. Jakobsen. Secure multi-party computation on integers. Master's thesis, Aarhus University, 2005. Available at `http://users-cs.au.dk/tpj/uni/thesis/`.

[FSW02] P. Fouque, J. Stern, and J. Wackers. Cryptocomputing with rationals. In M. Blaze, editor, *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 136–146. Springer, 2002.

[GMS10] J. Guajardo, B. Mennink, and B. Schoenmakers. Modulo reduction for paillier encryptions and application to secure statistical analysis. In R. Sion, editor, *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 375–382. Springer, 2010.

[HAB02] W. Hesse, E. Allender, and D. A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695 – 716, 2002.

[HKS$^+$10] W. Henecka, S. Kögl, A. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: tool for automating secure two-party computations. In *CCS '10: Proceedings of the 17th ACM conference on Computer and communications security*, pages 451–462, New York, NY, USA, 2010. ACM.

[JW05] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In R. Grossman, R. J. Bayardo, and K. P. Bennett, editors, *KDD*, pages 593–599. ACM, 2005.

[KLM05] E. Kiltz, G. Leander, and J. Malone-Lee. Secure computation of the mean and related statistics. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 283–302. Springer, 2005.

[Lip03] H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894, pages 398–415, 2003.

[NO07] T. Nishide and K. Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 343–360, 2007.

[NX10] C. Ning and Q. Xu. Multiparty computation for modulo reduction without bit-decomposition and a generalization to bit-decomposition. In *ASIACRYPT*, pages 483–500, 2010.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, 1999.

[RT10] T. Reistad and T. Toft. Linear, constant-rounds bit-decomposition. In D. Lee and S. Hong, editors, *Information, Security and Cryptology - ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 245–257. Springer Berlin / Heidelberg, 2010.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Tof11] T. Toft. Sub-linear, secure comparison with two non-colluding parties. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 174–191. Springer, 2011.

[Veu10] Thijs Veugen. Encrypted integer division. In *IEEE Workshop on Information Forensics and Security (WIFS'10)*, Seattle, 2010. IEEE, IEEE.