Reducing Communication Channels in MPC

Marcel Keller[1], Dragos Rotaru[1,2], Nigel P. Smart[1,2], and Tim Wood[1,2]

[1] University of Bristol, Bristol, UK.
[2] imec-COSIC, KU Leuven, Leuven, Belgium.
mks.keller@gmail.com,Dragos.Rotaru@esat.kuleuven.be,
nigel.smart@kuleuven.be,t.wood@kuleuven.be

Abstract. We show that the recent, highly efficient, three-party honest-majority computationally-secure MPC protocol of Araki et al. can be generalised to an arbitrary Q_2 access structure. Part of the performance of the Araki et al. protocol is from the fact it does not use a complete communication network for the most costly part of the computation. Our generalisation also preserves this property. We present both passively- and actively-secure (with abort) variants of our protocol. In all cases we require fewer communication channels for secure multiplication than Maurer's "MPC-Made-Simple" protocol for Q_2 structures, at the expense of requiring pre-shared secret keys for Pseudo-Random Functions.

1 Introduction

Secret-sharing-based secure MPC (multi-party computation) is generally considered to lie in two distinct camps. In the first camp lies the information-theoretic protocols arising from the original work of Ben-Or, Goldwasser and Wigderson [4] and Chaum, Crepeau and Damgård [7]. In this line of work, adversarial parties are assumed to be computationally unbounded, and parties in an MPC protocol are assumed to be connected by a *complete network* of *secure channels*. Such a model was originally introduced in the context of threshold adversary structures, i.e. t-out-of-n secret-sharing schemes, which could tolerate up to t adversaries amongst n parties. We will call these access structures (n, t)-*threshold*. To obtain passively-secure protocols one requires $t < n/2$, and to obtain actively-secure protocols one requires $t < n/3$; these conditions are also sufficient. Passive adversaries follow the protocol but possibly try to learn information about other parties' inputs, whereas active adversaries may deviate arbitrarily from the protocol.

These results for threshold structures were extended to arbitrary access/adversary structures by Hirt and Maurer [14], in which case the two necessary and sufficient conditions become Q_2 and Q_3 respectively. These notions will be discussed in more detail later, but in brief an access structure is Q_ℓ if the union of no ℓ unqualified sets is the whole set of parties; for example, an (n, t)-threshold scheme is Q_ℓ if and only if $t < n/\ell$.

Another line of work which considered computationally-bounded adversaries started with [12,13]. Here the parties are connected by a *complete network* of *authenticated channels* and one can obtain actively-secure protocols in the threshold

case when $t < n/2$ (i.e. honest majority), and active security *with abort* when only one party is honest. Generally speaking, such computationally-secure protocols are less efficient than the information-theoretic protocols as they usually need some form of public-key cryptography.

In recent years there has been considerable progress in *practical* MPC by marrying the two approaches. For example, the BDOZ [5], VIFF [9], SPDZ [10] and Tiny-OT [17] protocols are computationally secure and use information-theoretic primitives in an online phase, but only computationally-secure primitives in an offline/pre-processing phase. The offline phase is used to produce so-called *Beaver triples* [2], which are then consumed in the online phase. In these protocols, parties are still connected by a *complete network* of *authenticated channels*, and they are usually in the full-threshold model (i.e. situations in which only one party is assumed to be honest). A key observation in much of the practical MPC work of the last few years is that communication costs in the *practically important online phase* are the main bottleneck.

However, recent work has provided a new method to unify information-theoretic and computationally-secure protocols. Araki et al. [1] provide a very efficient passively-secure MPC evaluation of the AES circuit in the case of a 1-out-of-3 adversary structure. This was then generalised to an actively secure protocol by Furukawa et al. [11]. Both protocols require a pre-processing phase making use of *symmetric-key* cryptographic primitives only; thus the pre-processing is much faster than for the full-threshold protocols mentioned above.

The passively-secure protocol of [1] makes use of a number of optimisations to the basic offline/online paradigm. Firstly, the offline phase is only used to produce additive sharings of zero. Additive sharings of zero can be easily produced using symmetric key primitives and pre-shared secrets. Secondly, the underlying network is *not* assumed to be *complete*: each party only sends data to *one* other party via a *secure channel*, and only receives data from *the third* party via a *secure channel*. Thirdly, parties need only transmit one finite-field element per multiplication. On the downside, however, each party needs to hold two finite-field elements per share, as opposed to using an ideal secret-sharing scheme (such as Shamir's) in which each party only holds one finite-field element per secret.

The underlying protocol of Araki et al., bar the use of the additive sharings of zero, is highly reminiscent of the Sharemind system [6], which also assumes a 1-out-of-3 adversary structure. Since both [1] and [6] are based on replicated secret-sharing, they are also closely related to the "MPC-Made-Simple" approach of Maurer [16]. Thus, for the case of this specific adversary structure, the work in [1] shows how to use cryptographic assumptions to optimise the information-theoretic approach of [16]. The active variant of the protocol given by Furakawa et al. [11] uses the passively-secure protocol (over an *incomplete network* of *secure channels*) to run an offline phase which produces the Beaver triples. These are then consumed in the online phase, by using the triples to check the passively-secure multiplication of actual secrets. The online phase also runs over an *incomplete network* of *authenticated channels*. The question therefore naturally arises as to whether the approach outlined in [1], [6] and [11] is

particularly tied to the 1-out-of-3 adversary structure, or whether it generalises to other access/adversary structures.

1.1 Our Work

In this paper we show that the basic passively-secure protocol of Araki et al. generalises to arbitrary Q_2 access structures and in the process hopefully shed some light onto the fundamental nature of what initially appear to be very specific constructions for 1-out-of-3 adversary structures. Moreover, the generalised protocol offers significant advantages in terms of communication cost when compared to the prior protocols in this setting.

In the full version we then show how to extend this to an actively-secure protocol (with abort) for any Q_2 access structure. We take a more traditional approach than [11] to obtain active security. In particular we utilise our passive protocol as an offline phase, and then in the online phase multiplication is performed via standard Beaver multiplication over an incomplete network of authenticated channels. We only require a full network of secure channels in the active protocol to obtain (verified) private output in the online phase and in a short setup phase.

The main challenge we meet in attempting to generalise the work of Araki et al. is that it is not immediately clear what the conditions on its shares mean in a wider context; more specifically, their protocol relies heavily on the fact that in the $(3, 1)$-threshold setting replicated shares are necessarily "consistent" and consequently their communication pattern allows errors to be detected in the active variant due to Furukawa et al. [11].

General, as opposed to threshold, access structures are practically interesting in situations where different groups of parties play different organisational roles. For example consider a financial application where one may have a computation performed between a number of banks and regulators; the required access structures for collaboration between the banks and the regulators may be different. Thus general access structures, such as the Q_2 structures considered in this paper, may have important real-world applications. All protocols have been implemented in the SCALE-MAMBA system[3].

We now proceed to give a high-level overview of our protocol and its main components. We divide the discussion into looking at the passively secure protocol first, and then give the changes needed to consider the actively secure (with abort) variant.

Passively Secure Protocol: If the access structure is Q_2 then the product of the shared values can be expressed as a linear combination of products of the values held by individual players. Hence, the product can be expressed as the sum of a single value held by each party. This is exploited in the protocol of Maurer to obtain a sharing (in the original replicated scheme) of the product,

[3] See `https://homes.esat.kuleuven.be/~nsmart/SCALE/` for details.

by each party producing a resharing of their component of the product. Thus multiplication of secrets in the passively-secure protocol of Maurer requires all parties to produce one secret-sharing.

In our protocol we take start as in Maurer's protocol in forming a representation of the product as a full threshold additive secret sharing. We then mask this using a pseudo-random zero sharing (PRZS), and then use the resulting full threshold sharing as a basis for the original replicated sharing. This means that each party need only communicate the share they hold to the other parties which need to replicate it. This produces savings in both the number of elements transmitted and the number of communication channels. In Sections 3.1 and 3.2 we outline and compare Maurer's and our protocol.

In a further optimisation, given in Section 3.3, we reduce the number of channels, which we denote by $\mathcal{G}_\Gamma$, and the required number of finite field elements transmitted, even further. This optimisation, however, comes at the expense of requiring more pre-distributed keys and PRF evaluations. But we present a simple six party access structure for which this optimization that gives a 93 percent saving on transmitted finite-field elements, and a 50 percent saving on the number of secure channels, compared to the original protocol of Maurer.

To obtain the output from our passively protocol we require a full set of either authenticated or secure channels (depending on the specific subprotocol being executed). However, these operations are not performed nearly as often as multiplication operations. It is the high bandwidth requirements of multiplication operations that form a bottleneck in many practical instantiations of MPC protocols.

Actively Secure Protocol: In the full version we then extend this basic protocol to the case of active security (with abort), again with the objective of minimising the number of pairwise connections and transmitted finite field elements. Our actively-secure protocol follows the paradigm of Furukawa et al. However, we need to make a few small changes to allow for arbitrary Q_2 access structures. We adopt a relatively standard three step approach to obtaining active security.

1. We use our passively-secure multiplication protocol in an offline phase to obtain so-called Beaver triples.
2. These triples are then checked using the trick of sacrificing (see e.g. [5]) to ensure that the triples are actually valid, and have not been tampered with by a malicious adversary. This requires communication over a reduced set of *authenticated* channels $\mathcal{H}_\Gamma$.
3. The triples are then used in a standard Beaver-like online phase which is executed over the same sub-network of *authenticated* channels.

Active security is obtained, as in [1], by each player hashing their view during a multiplication and comparing the resulting hashes at the end (which requires a complete graph of authenticated channels). We show that this obviates the need for *every* party holding a given share to send it to every party who does not. However, in generalising to arbitrary access structures it is no longer sufficient

to hash the view of the *values opened* in the multiplication sub-protocol: one also needs to hash the *vector of shares* used to produce these values. This hash-checking is analogous to the MAC checking in full threshold protocols such as SPDZ [10].

In this paper we are interested in evaluation of arithmetic circuits over an arbitrary finite field $\mathbb{F}_q$, which could include $q = 2$. We will assume, for the sacrifice step of our actively-secure protocol with abort, that q is sufficiently large to have a cheating detection probability of $1 - 2^{-\mathsf{sec}}$ for a suitable choice of sec; i.e. $q > 2^{\mathsf{sec}}$. If this is not the case, then by repeating our checking procedures $\mathsf{sec}/\log_2 q$ times, we can reduce the cheating probability to $2^{-\mathsf{sec}}$. We do not analyse this aspect in this paper so as to aid the reader in seeing the main concepts more clearly. This repetition and its generalisation to balls-and-bins experiments is relatively standard.

2 Preliminaries

In this section we recap on access structures, and in particular Q_2 access structures, and also look at pseudo-random zero sharings with respect to the additive secret sharing scheme. In this section we are working over an arbitrary finite field $\mathbb{F}_q$ where q is a prime power, although our protocols also work over any finite ring R. For any $n \in \mathbb{N}$ we denote the set $\{1, \ldots, n\}$ by $[n]$. We denote the computational security parameter by λ and the statistical security parameter by sec.

2.1 Access Structures and Secret Sharing

Access Structures. Let $\mathcal{P}$ denote the set of parties, $\mathcal{P} = [n]$, and let $\Gamma, \Delta \in 2^{\mathcal{P}}$. If $\Gamma \cap \Delta = \varnothing$ then we call the pair (Γ, Δ) an access structure. We call a set of parties $B \in \Gamma$ qualified, and a set in $A \in \Delta$ unqualified. As is typical in the literature, we assume monotonicity of the access structure: supersets of qualified sets are qualified, and subsets of unqualified sets are unqualified. The access structure is said to be *complete* if $\Delta = 2^{\mathcal{P}} \setminus \Gamma$, (i.e. every set of parties is either qualified or unqualified), and in this case we will sometimes just write Γ for the access structure instead of the pair.

A set of parties $A \in \Delta$ is called *maximally* unqualified if Δ contains no proper supersets of A. For a complete access structure, this implies that adding any party not already in A makes the set qualified. We denote by $\mathcal{M} \subseteq \Delta$ the set of maximally unqualified sets. Similarly, A set in Γ is called minimally qualified if it is qualified and every proper subset is unqualified. The set $\mathcal{M}$ and its structure is important for our protocol; however, it will be notationally simpler for us instead to consider the set of complements of maximally unqualified sets, which we denote by $\mathcal{B} = \{\mathcal{P} \setminus M : M \in \mathcal{M}\}$. Note that, in general, it is not true that the set $\mathcal{B}$ is equal to the set of minimally qualified sets.

Q_ℓ Access Structures. The set Δ, called the adversary structure, is said to be Q_ℓ (for **q**uorum), where $\ell \in \mathbb{N}$, if no set of ℓ sets in Δ cover $\mathcal{P}$. A result of Hirt and Maurer [15] says that every function can be computed securely in the presence of an adaptive, passive (resp. adaptive, active) computationally unbounded adversary if and only if the adversary structure is Q_2 (resp. Q_3).

It is clear that if Δ is Q_2, then so is any subset. In particular, the set of maximally unqualified sets $\mathcal{M}$ is also Q_2. In fact, if $\mathcal{M}$ is Q_2 then Δ is Q_2. Hence, for the set of complements $\mathcal{B}$ it holds that if $B_1, B_2 \in \mathcal{B}$ then $B_1 \cap B_2 \neq \varnothing$. A set $\mathcal{B}$ for which this property holds was called a quorum system by Beaver and Wool [3].

Let S denote a linear secret-sharing scheme which implements the Q_2 access structure (Γ, Δ). We use double square brackets, $[\![v]\!]$ to denote a sharing of the secret v according to this scheme. We let $S_{v,i}$ denote the set of elements which player i holds in representing the value v. Hirt and Maurer's result is realised by showing that if an access structure is Q_2 then it can be realised by a multiplicative secret sharing scheme, i.e. given two secret shared values $[\![a]\!]$ and $[\![b]\!]$, the product $a \cdot b$ can be represented as a linear combination of the elements in the *local* Schur products

$$S_{a,i} \otimes S_{b,i} = \{s_a \cdot s_b : s_a \in S_{a,i}, s_b \in S_{b,i}\}.$$

The fact that by local computations the parties each obtain one summand of the product is the reason one is able to build an MPC protocol secure against passive adversaries for any Q_2 access structure. For the details, we refer the reader to [15].

Replicated Secret Sharing. Given a monotone access structure (Γ, Δ), we will make extensive use of the replicated secret sharing scheme which respects it. Let $\mathcal{B}$ be, as above, the set of sets which are complements of maximally unqualified sets in the access structure. Then to share secret x, a set of shares $\{x_B\}_{B \in \mathcal{B}}$ is sampled uniformly at random from the field subject to $x = \sum_{B \in \mathcal{B}} x_B$ and x_B is given to each player in B. From now on, when writing $[\![x]\!]$ we will mean the secret sharing with respect to this scheme, and in particular the set $S_{x,i}$ above is given by $S_{x,i} = \{x_B : i \in B \text{ and } B \in \mathcal{B}\}$. Since every unqualified set is a (not necessarily proper) subset of a maximally unqualified set, every set of unqualified parties is missing at least one member of the set $\{x_B\}_{B \in \mathcal{B}}$, and hence these parties learn no information about the secret. Replicated secret-sharing is therefore *perfect*, which is defined to mean that no unqualified set of parties has any advantage over uniformly guessing the secret. Conversely, a qualified set A of parties is not a subset of any $M \in \mathcal{M}$ (i.e., for every $M \in \mathcal{M}$, A contains some i where $i \notin M$), and hence for every $B \in \mathcal{B}$, there is at least one party in A which receives the share x_B.

To see that a replicated secret-sharing scheme is multiplicative if the access structure it realises is Q_2, observe that given secrets x and y, for every pair of sets $B_1, B_2 \in \mathcal{B}$ there is some party i in $B_1 \cap B_2$, since the intersection of these sets is non-empty by definition of Q_2. Then party i can compute the cross terms

$x_{B_1} \cdot y_{B_2}$ and $x_{B_2} \cdot y_{B_1}$ (and also the diagonal terms $x_{B_1} \cdot y_{B_1}$ and $x_{B_2} \cdot y_{B_2}$). Thus the parties can together obtain all terms of $x \cdot y = \left(\sum_{B \in \mathcal{B}} x_B\right) \cdot \left(\sum_{B \in \mathcal{B}} y_B\right) = \sum_{B_1, B_2 \in \mathcal{B}} x_{B_1} \cdot y_{B_2}$ by local computations. Note that the parties do not, in general, have a correct sharing of the product after these local computations, since each party now holds only one share: the parties must somehow convert this additive share of the product into a sharing within the scheme. Minimising the number of communication channels required after the local computations to achieve this is the main goal of this paper. Note also that there may be multiple parties in the intersection of two sets in $\mathcal{B}$, but we only require one of these parties to include the term in their computation.

Example. We will use the following example later to demonstrate the savings which can result from our method and also to examine the communication channels in the next paragraph. Consider the following set of maximally unqualified sets for a six-party access structure, which we shall use as a running example throughout this section.

$$\mathcal{M} = \Big\{\{2,5,6\},\{3,5,6\},\{4,5,6\},\{1,2\},\{1,3\},\{1,4\},\{1,5\},\{1,6\},\\ \{2,3\},\{2,4\},\{3,4\}\Big\}.$$

Here the set $\mathcal{B}$ becomes

$$\mathcal{B} = \Big\{\{1,3,4\},\{1,2,4\},\{1,2,3\},\{3,4,5,6\},\{2,4,5,6\},\{2,3,5,6\},\\ \{2,3,4,6\},\{2,3,4,5\},\{1,4,5,6\},\{1,3,5,6\},\{1,2,5,6\}\Big\}.$$

As stated above, in replicated secret sharing a secret s is shared as an additive sum $s = \sum_{B \in \mathcal{B}} s_B$, with party i holding value s_B if and only if $i \in B$.

Redundancy. A redundant player is one whose shares are not *necessarily* needed to reconstruct the secret (if it is shared using replicated secret-sharing), and so one could define an MPC protocol achieving the same (passive) security by ignoring this player entirely in the computation and just providing it with the final output. To provide a more formal definition, consider the replicated scheme above: if there is a party $i \in \mathcal{P}$ for which there exists some other party $j \in \mathcal{P}$ such that for all $B \in \mathcal{B}$ we have $i \in B$ implies $j \in B$, then every share given to party i is also given to party j, and hence we consider party i redundant.

For an access structure Γ with set $\mathcal{M}$ of maximal unqualified sets, we define party i to be redundant if for every $M \in \mathcal{M}$ there exists $j \in \mathcal{P} \setminus \{i\}$ such that $i \notin M$ implies $j \notin M$, and non-redundant otherwise; equivalently, i is non-redundant if for every $j \in \mathcal{P}$ there exists $M \in \mathcal{M}$ such that $i \notin M$ but $j \in M$, and we say that Γ is non-redundant if every party in $\mathcal{P}$ is non-redundant.

For example, consider the set of maximally unqualified sets over $\mathcal{P} = [4]$ given by $\mathcal{M} = \{\{1\},\{2\},\{3,4\}\}$. We obtain the replicated scheme over this access structure by computing $\mathcal{B} = \{\{2,3,4\},\{1,3,4\},\{1,2\}\}$ and splitting a

secret s into three shares $s = s_{234} + s_{134} + s_{12}$; then we give player one the shares $\{s_{134}, s_{12}\}$, player two $\{s_{234}, s_{12}\}$, player three $\{s_{234}, s_{134}\}$ and player four $\{s_{234}, s_{134}\}$. Both shares obtained by player three are also obtained by player four, so we can essentially ignore player four in any protocol design and just provide the output to this player at the end.

Note that if any party is omitted from all sets in $\mathcal{M}$ then it is present in all sets in $\mathcal{B}$ and hence every party, but this party, is redundant, which makes the MPC protocol trivial: the omitted party can simply perform the entire computation itself and output the result to all parties.

Partition. In our protocol, we partition the set $\mathcal{B}$ into sets indexed by the parties $\{\mathcal{B}_i\}_{i\in\mathcal{P}}$ such that for every $i \in \mathcal{P}$ we have $B \in \mathcal{B}_i$ implies $i \in B$. To make this assignment of sets in $\mathcal{B}$ to parties, we consider all the maps $f : \mathcal{B} \to \mathcal{P}$ such that for every $i \in \mathcal{P}$, $f(B) = i$ implies $i \in B$, and choose an f such that $\mathrm{Im}(f)$ is as large as possible; then for each $i \in \mathcal{P}$ we let $\mathcal{B}_i = f^{-1}(i)$, where $f^{-1}(i)$ denotes the preimage of i under the map f.

Note that if f is not surjective then there is at least one set $\mathcal{B}_i$ (for some i) which is empty. For the rest of the main body of this paper, we assume that $\mathcal{B}_i$ is *not* empty for all i, since for small numbers of parties on a non-redundant Q_2 access structure, we can always find a surjective f. For the necessary adaptation to the protocol when this is not the case, and further relevant discussion, see Section 4.

Note that in general non-redundancy implies a lower bound on the size of $\mathcal{M}$: let n' be the number of parties which are *not* maximally unqualified sets as singleton sets, and let x be the number of sets in $\mathcal{M}$. The lower bound on number of sets there must be in $\mathcal{M}$ so that there is no redundancy amongst these n' parties is the number of ways of putting each party into at least two sets so that for every pair of parties there is a set containing one and not the other. Thus we require $\binom{x}{2} \geq n'$, which means that $x \geq \frac{1+\sqrt{1+8n'}}{2}$. Since there are more sets in $\mathcal{M}$ for non-redundant access structures, it becomes "easier" to find the surjective maps f required by our main protocol.

In our earlier six party example we could set the partition to be

$$
\begin{aligned}
\mathcal{B}_1 &= \{\{1,3,4\}\},\\
\mathcal{B}_2 &= \{\{1,2,4\}\},\\
\mathcal{B}_3 &= \{\{1,2,3\}\},\\
\mathcal{B}_4 &= \{\{2,3,4,5\}\},\\
\mathcal{B}_5 &= \{\{1,2,5,6\},\ \{1,3,5,6\},\ \{1,4,5,6\}\},\\
\mathcal{B}_6 &= \{\{2,3,4,6\},\ \{2,3,5,6\},\ \{2,4,5,6\},\ \{3,4,5,6\}\}.
\end{aligned}
$$

Channel Sets. Given the above partition of $\mathcal{B}$ we define the following graphs of channels:

$$\mathcal{G}_\Gamma = \bigcup_{i\in\mathcal{P}} \bigcup_{B\in\mathcal{B}_i} \bigcup_{j\in B\setminus\{i\}} \{(i,j)\}$$
$$\mathcal{H}_\Gamma = \bigcup_{i\in\mathcal{P}} \bigcup_{B\in\mathcal{B}_i} \bigcup_{j\notin B} \{(i,j)\}$$

Our (passively-secure) multiplication protocol makes use of the set of secure channels denoted by $\mathsf{SC}(\mathcal{G}_\Gamma)$, namely $(i,j) \in \mathsf{SC}(\mathcal{G}_\Gamma)$ implies that party i is connected to party j by a uni-directional secure channel. The sacrificing step and online multiplication protocol in our actively secure protocol requires communication over an authentic set of channels $\mathsf{AC}(\mathcal{H}_\Gamma)$, where $(i,j) \in \mathsf{AC}(\mathcal{H}_\Gamma)$ implies that party i is connected to party j by an authenticated channel.

The key operation in both sacrificing and the online phase is being able to open a value to all parties in an authenticated manner. Publicly opening a secret requires every party to receive every share it does not have from at least one other party holding that share. Thus the definition of $\mathcal{H}_\Gamma$.

2.2 Pseudo-Random Zero Sharing for Additive Secret Sharing Schemes

At various points we will need to use an additive secret sharing over all players $\mathcal{P} = \{1,\ldots,n\}$. This shares a value $v \in \mathbb{F}_q$ as an additive sum $v = \sum_{i=1}^{n} v_i$ and gives player i the value v_i. We denote such a sharing by $\langle v\rangle$. This type of secret-sharing does not respect a Q_2 access structure since all shares are required to determine the secret, but it will play a crucial role in our protocols.

Improving on the protocol of [3] and [16] requires us to sacrifice the information-theoretic security for a cryptographic assumption. In particular, we require the parties to engage in a pre-processing phase in which they share keys for a pseudo-random function (PRF) in order to generate (non-interactively) pseudo-random zero sharings (PRZSs) for the additive secret sharing scheme $\langle v\rangle$, and pseudo-random secret sharings (PRSSs) for the replicated scheme $[\![v]\!]$. Note, we could produce these using additional interaction, but recall our goal is to reduce communication. In particular, we make black-box use of the functionality given in Figure 1. Pseudo-random secret sharings, and pseudo-random zero sharings in particular, for arbitrary access structures can involve a set-up phase requiring the agreement of exponentially-many keys in general. The protocol is given in [8] and so we omit it here, though the reader may refer to the full version for an overview of our variant (specialised for replicated secret-sharing).

3 Passively-Secure MPC Protocol

In this section we outline our optimisation of Maurer's protocol. As remarked earlier, our protocol, instead of being in the information-theoretic model, uses

The Functionality $\mathcal{F}_{\mathsf{Rand}}$

Set-up: The functionality accepts Initialise or Abort from all parties and the adversary. If any party inputs Abort, the functionality sends the message Abort to all parties.

PRZS:

- On input PRZS(count) from all parties, if the counter value is the same for all parties and has not been used before, for each $i \in [n-1]$ the functionality samples $t_i \xleftarrow{\$} \mathbb{F}$ uniformly at random, fixes $t_n \leftarrow -\sum_{i=1}^{n-1} t_i$ and sends t_i to party i for each $i \in \mathcal{P}$.

PRSS:

- On input PRSS(count) from all parties, if the counter value is the same for all parties and has not been used before, the functionality samples a set $\{r_B\}_{B \in \mathcal{B}} \xleftarrow{\$} \mathbb{F}$ and for each $B \in \mathcal{B}$ sends r_B to all $i \in B$.

Figure 1. The Functionality $\mathcal{F}_{\mathsf{Rand}}$

PRFs to obtain additive sharings of zero non-interactively. We assume throughout that we start with an access structure which does not contain any redundant players. As stated in Section 2, we will assume we can define a partition $\{\mathcal{B}_i\}$ of $\mathcal{B}$ such that $\mathcal{B}_i \neq \varnothing$ and $B \in \mathcal{B}_i$ implies $i \in B$. We call such a partition (where $\mathcal{B}_i \neq \varnothing$ for all i) a *surjective partition*; when this is not possible we provide the requisite alterations to the protocol in Section 4. We consider $\mathcal{B}_i$ to be the set of sets for which party i will be "responsible".

3.1 Maurer's "MPC-Made-Simple" Protocol

The information-theoretic protocol we describe is based on one due to Maurer [16]. Maurer's protocol is itself a variant of the protocol of Beaver and Wool [3] but specialised to the case of replicated secret-sharing. For comparison with our protocol, we explain Maurer's protocol here.

We assume a Q_2 access structure (Γ, Δ), and we share data values x via the replicated secret-sharing $[\![x]\!]$, where $x = \sum_{B \in \mathcal{B}} x_B$. Since this secret-sharing scheme is linear, addition of secret-shared values comes "for free", i.e. it requires no interaction and parties just need to add their local shares together.

The real difficulty in creating an MPC protocol given a linear secret-sharing scheme is in performing secure multiplication of secret-shared values, $[\![x]\!]$ and $[\![y]\!]$. With this goal, we begin by following [3] and define a *surjective* function $\rho : \mathcal{B}^2 \to \mathcal{P}$ such that $\rho(B_1, B_2) = i$ implies that $i \in B_1 \cap B_2$; the existence of such a function follows from the fact that the access structure is Q_2. Note that there are possibly multiple choices for ρ, and that it is certainly not true in general that $i = \rho(B_1, B_2)$ implies $B_1 \cap B_2 = \{i\}$ (though clearly $B_1 \cap B_2 \supseteq \{i\}$). Note that party $\rho(B_1, B_2)$ holds a copy of share x_{B_1} and y_{B_2}. We will put player

$\rho(B_1, B_2)$ "in charge" of computing the cross term $x_{B_1} \cdot y_{B_2}$ in the following multiplication protocol:

1. Party i computes
$$v_i \leftarrow \sum_{B_1, B_2 \in \mathcal{B} \,:\, \rho(B_1,B_2)=i} x_{B_1} \cdot y_{B_2}$$
2. Party i creates a sharing $[\![v_i]\!]$ of the value v_i and distributes the different summands securely to the appropriate parties according to the replicated secret-sharing scheme.
3. The parties now locally compute
$$[\![z]\!] \leftarrow \sum_{i=1}^{n} [\![v_i]\!].$$

It is clear that each party i, in sharing v_i, needs to generate $|\mathcal{B}|$ different finite-field elements, each of which is sent to every member of a given set of parties in $\mathcal{B}$. In particular this means that each party has to maintain a secure connection to each other party, assuming a non-redundant access structure. If we let l denote the average size of $B \in \mathcal{B}$, i.e. $l = \sum_{B \in \mathcal{B}} |B|/|\mathcal{B}|$, then it is clear that the total communication required is $n \cdot |\mathcal{B}| \cdot l$ finite-field elements.

In fact each party i sends a total of
$$\sum_{B \in \mathcal{B}: B \ni i} (|B| - 1) + \sum_{B \in \mathcal{B}: B \not\ni i} |B| = \sum_{B \in \mathcal{B}} |B| - \sum_{B \in \mathcal{B}: B \ni i} 1$$
finite-field elements, and hence the total communication (for all parties) for one multiplication is
$$\sum_{i=1}^{n} \left(\sum_{B \in \mathcal{B}} |B| - \sum_{B \in \mathcal{B}: B \ni i} 1 \right) = (n-1) \cdot \sum_{B \in \mathcal{B}} |B|$$
finite-field elements over $n \cdot (n-1)$ uni-directional secure channels[4]. For our example Q_2 access structure this translates into sending $(6-1) \cdot 41 = 205$ finite-field elements over $6 \cdot 5 = 30$ secure channels. Note that the same finite-field element will be sent to multiple parties (every set of parties $B \in \mathcal{B}$ obtains a share common to them all), but we count these elements as distinct when analysing communication costs.

3.2 New Protocol

Our protocol is largely the same as Maurer's, with one major difference: in our protocol, the parties do not each create a replicated sharing of the partial

[4] Note, as is common in security systems we assume channels are uni-directional; as good security practice is to have different secret keys securing communication in different directions so as to avoid various reflection attacks etc. This is exactly how TLS and IPSec secure channels are configured.

product v_i – instead, they do the following. Notice that the v_i form an additive sharing $\langle z \rangle$ of the sum. Our basic idea is first to re-randomise this sum using a PRZS, and then to consider each re-randomised v_i as one share of the new secret (namely, the product of the previous two secrets), i.e. consider each share v_i as z_B indexed by some B containing i, which should then be distributed to all other parties in B. There are some minor technical caveats but this is the essential idea.

Our method directly generalises the method used by [1], which concentrated on the case of the finite field $\mathbb{F}_2$ and a 1-out-of-3 adversary structure. It results in each party not needing to be connected to each other party by a secure channel. The total number of distinct finite field elements transmitted for a threshold structure via this method is then $O(n \cdot 2^n)$, as opposed to the $O(n^2 \cdot 2^n)$ of Maurer's protocol. For other Q_2 structures the saving in communication is more significant, as our earlier example demonstrates.

As in Maurer's "MPC-Made-Simple" protocol, we assume a Q_2 access structure (Γ, Δ) and share data values x via the replicated secret-sharing $[\![x]\!]$, so that $x = \sum_{B \in \mathcal{B}} x_B$. We also retain the assignment which tells player $i = \rho(B_1, B_2)$ to compute the product $x_{B_1} \cdot y_{B_2}$. However, our basic multiplication procedure is given by the following:

1. Party i computes

$$v_i \leftarrow \sum_{B_1, B_2 \in \mathcal{B} \,:\, \rho(B_1, B_2) = i} x_{B_1} \cdot y_{B_2}$$

   We think of v_i as an additive sharing $\langle v \rangle$ of the product.
2. The parties obtain an additive sharing of zero $\langle t \rangle$ using the PRZS from Figure 1; thus party i holds t_i such that $\sum_{i=1}^{n} t_i = 0$.
3. Party i samples u_B for $B \in \mathcal{B}_i$ such that $\sum_{B \in \mathcal{B}_i} u_B = v_i + t_i$.
4. Party i sends, for all $B \in B_i$, the value u_B to party j for all $j \in B$.

Notice that the parties do not need to perform local computations after the communication as in Maurer's protocol, and that the total number of elements transmitted is $\sum_{B \in \mathcal{B}} (|B| - 1)$. Also notice that we obtain a valid sharing of the product as we have assumed $\mathcal{B}_i \neq \varnothing$, and thus every share v_i has been utilised in the final sharing.

The key observation for security is that the PRZS masks the Schur product terms, so after choosing the u_B's and sending these to the appropriate parties, not even qualified sets of parties can learn any information about these terms, despite being able to compute the secret.

Given this informal description, we now give a full description of our MPC protocol, which is the analogue of [1] for arbitrary Q_2 access structures and arbitrary finite fields; see Figure 3 for details. One can think of the passively-secure protocol as being in the pre-processing model in which the offline phase simply involves some key agreement. The online phase is then a standard MPC protocol in which parties can compute an arithmetic circuit on their combined (secret) inputs, using the multiplication procedure described above, so as to

Passively Secure MPC Functionality $\mathcal{F}_{\mathsf{PMPC}}$

Input: On input (Input, x_i) by party i, the functionality stores (id, x_i) in memory.

Add: On input $(\mathsf{Add}, \mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_3)$ from all parties, the functionality retrieves (id_1, x) and (id_2, y) and stores $(\mathsf{id}_3, x + y)$.

Multiply: On input $(\mathsf{Multiply}, \mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_3)$ from all parties, the functionality retrieves (id_1, x) and (id_2, y) and stores $(\mathsf{id}_3, x \cdot y)$.

Output: On input $(\mathsf{Output}, \mathsf{id}, i)$ from all parties, the functionality retrieves (id, x) and returns x to all parties if $i = 0$, and to player i only otherwise.

Figure 2. Passively Secure MPC Functionality $\mathcal{F}_{\mathsf{PMPC}}$

implement the functionality in Figure 2. That the protocol securely implements this functionality is given by the following theorem, whose proof is given in the full version.

Theorem 1. *Suppose we have a non-redundant Q_2 access structure with a surjective partition $\{\mathcal{B}_i\}$ of the set $\mathcal{B}$. Then the protocol Π_{PMPC} securely realises the functionality $\mathcal{F}_{\mathsf{PMPC}}$ against passive adversaries in the $\mathcal{F}_{\mathsf{Rand}}$-hybrid model*[5]*.*

Assuming a surjective partition, the protocol requires at most $\sum_{B \in \mathcal{B}}(|B| - 1)$ field elements of communication, over $|\mathcal{G}_\Gamma|$ secure channels, per multiplication gate, and the same number to perform the input procedure.

In the output procedure we require that the parties be connected by a complete network of bilateral secure channels (i.e. $n \cdot (n-1)$ uni-directional channels) if all players are to receive distinct private outputs, and instead a complete network of authenticated channels if only public output is required.

Note that the above theorem is given for non-redundant access structures. To apply the protocol in the case of redundant access structures, we simply remove redundant players from the computation phase and only require interaction with them in the input and output phases. To avoid explaining this (trivial) extra complication we specialise to the case of non-redundant access structures.

In our previous six party example we have

$$\mathsf{SC}(\mathcal{G}_\Gamma) = \Big\{(1,3),(1,4),(2,1),(2,4),(3,1),(3,2),(4,2),(4,3),(4,5),\\ (5,1),(5,2),(5,3),(5,4),(5,6),(6,2),(6,3),(6,4),(6,5)\Big\}.$$

Thus in this example we need to send 30 finite-field elements over 18 uni-directional secure channels per multiplication operation, thus giving a saving of 85 percent on the number of finite-field elements transmitted, and 40 percent on the number of secure channels needed.

[5] The alterations to the protocol for when there is no surjective partition are discussed in Section 4.

Protocol Π_{PMPC}

The set $\mathcal{B}_i$ denotes the set of the partition $\mathcal{B} = \{\mathcal{B}_i\}_{i\in\mathcal{P}}$ containing sets associated to party i (though note that it is usually a *strict* subset of the sets containing i).

Set-up: The parties set $\mathsf{count} \leftarrow 0$.

Input: For party i to provide input x,

1. The parties call $\mathcal{F}_{\mathsf{Rand}}$ with input $\mathsf{PRZS}(\mathsf{count})$ so that each player $j \in \mathcal{P}$ obtains t_j such that $\sum_{j\in\mathcal{P}} t_j = 0$.
2. Party i samples $\{u_B\}_{B\in\mathcal{B}_i} \leftarrow \mathbb{F}$ such that $\sum_{B\in\mathcal{B}_i} u_B = x + t_i$.
3. For each $j \in \mathcal{P}\setminus\{i\}$, party j samples $\{u_B\}_{B\in\mathcal{B}_j} \leftarrow \mathbb{F}$ such that $\sum_{B\in\mathcal{B}_j} u_B = t_j$.
4. For each $j \in \mathcal{P}$, for each $B \in \mathcal{B}_j$, for each $k \in B$, party j sends u_B securely to party k.
5. The parties increment count by one.

Add:

1. For each $B \in \mathcal{B}$, each party $i \in B$ locally computes $x_B + y_B$ so that collectively the parties obtain $[\![x + y]\!]$.

Multiply:

1. For each $i \in \mathcal{P}$, party i computes $v_i \leftarrow \sum_{B_1,B_2\in\mathcal{B}\,:\,\rho(B_1,B_2)=i} x_{B_1} \cdot y_{B_2}$.
2. The parties call $\mathcal{F}_{\mathsf{Rand}}$ with input $\mathsf{PRZS}(\mathsf{count})$ so that each player $i \in \mathcal{P}$ obtains t_i such that $\sum_{i\in\mathcal{P}} t_i = 0$.
3. For each $i \in \mathcal{P}$, party i samples $\{u_B\}_{B\in\mathcal{B}_i} \leftarrow \mathbb{F}$ such that $\sum_{B\in\mathcal{B}_i} u_B = v_i + t_i$.
4. For each $i \in \mathcal{P}$, for each $B \in \mathcal{B}_i$, for each $j \in B \setminus \{i\}$, party i securely sends the value u_B to party j.
5. The parties increment count by one.

Output$([\![x]\!], i)$:

1. If $i \neq 0$, for each $j \in \mathcal{P}$, for each $B \in \mathcal{B}_j$, party j securely sends x_B to i if $i \notin B$. If $i = 0$, each player j instead sends to *all* players i for which $i \notin B$. In the latter case the communication need not be done securely.
2. Player i (or all players if $i = 0$) computes $x \leftarrow \sum_{B\in\mathcal{B}} x_B$.

Figure 3. Protocol Π_{PMPC}

3.3 An Optimisation

We end this section by presenting a minor optimisation of our passively secure multiplication protocol, which can result in a further reduction in both the number of communication channels and the number of finite-field elements that need to be sent. However, this comes at the expense of needing further PRF evaluations.

Recall that to each player i we associated a set $\mathcal{B}_i$, of sets B for which player i is "responsible" for producing the sharing u_B during the multiplication protocol. In our optimisation we make player i responsible for only a single set, which we

call B_i, which is an element of $\mathcal{B}_i$. All other values u_B for $B \in \mathcal{B}_i \setminus \{B_i\}$ are generated by a PRF evaluation.

We informally describe the extensions needed here in the case of a surjective partition; the extension to non-surjective partitions is immediate. First we extend the $\mathcal{F}_{\mathsf{Rand}}$ functionality so that it contains an additional command $\mathcal{F}_{\mathsf{Rand}}.\mathbf{Rand}(B)$. This command, on input of a set of players B, will output the same uniformly random value z_B to all players in B. Clearly, this additional command is a component of the existing command $\mathcal{F}_{\mathsf{Rand}}.\mathbf{PRSS}$, and so can be implemented in the same way.

Our optimisation of the multiplication protocol is then given in Figure 4. It is then clear that we need to transmit only n distinct, finite-field elements over the set

$$\widehat{\mathcal{G}_\Gamma} = \bigcup_{i \in \mathcal{P}} \bigcup_{j \in B_i \setminus \{i\}} \{(i,j)\} \subseteq \mathcal{G}_\Gamma$$

of secure channels, which we denote by $\mathsf{SC}(\widehat{\mathcal{G}_\Gamma})$. The total number of (non-distinct) finite fields elements that need to be sent is equal to $\sum_{i=1}^{n}(|B_i| - 1)$.

Optimised Passively Secure Multiplication Protocol

Multiply:

1. For each $i \in \mathcal{P}$, party i computes $v_i \leftarrow \sum_{B_1, B_2 \in \mathcal{B} \,:\, \rho(B_1,B_2)=i} x_{B_1} \cdot y_{B_2}$.
2. The parties call $\mathcal{F}_{\mathsf{Rand}}.\mathbf{PRZS}$ so that each player $i \in \mathcal{P}$ obtains t_i such that $\sum_{i \in \mathcal{P}} t_i = 0$.
3. For each $B \in \mathcal{B}_i \setminus \{B_i\}$ the players execute $\mathcal{F}_{\mathsf{Rand}}.\mathbf{Rand}(B)$, so that each player $j \in B$ obtains a uniformly random element u_B.
4. Party i defines u_{B_i} by setting $u_{B_i} \leftarrow v_i + t_i - \sum_{B \in \mathcal{B}_i \setminus \{B_i\}} u_B$.
5. For each $i \in \mathcal{P}$, party i sends the value u_{B_i} securely to party j for all $j \in B_i$.

Figure 4. Optimised Passively Secure Multiplication Protocol

When specialised to our six-party example from the introduction, and taking $B_5 = \{1,2,5,6\}$ and $B_6 = \{2,3,4,6\}$ (with the obvious definition of B_1, B_2, B_3, and B_4), we find

$$\widehat{\mathcal{G}_\Gamma} = \Big\{(1,3),(1,4),(2,1),(2,4),(3,1),(3,2),(4,2),(4,3),(4,5),\\ (5,1),(5,2),(5,6),(6,2),(6,3),(6,4)\Big\}.$$

Thus we need to send only 15 finite fields elements over 15 uni-directional secure channels. This equates to a bandwidth saving of an additional 50 percent over our initial protocol, and a 17 percent saving over the number of secure channels. Compared to the initial protocol of Maurer we obtain a saving of 93 percent in the number of transmitted finite field elements, and a saving of 50 percent in the number of secure channels.

4 Passive Multiplication Protocol when f is not Surjective

We now describe the modifications to our basic protocol when we cannot find a partition of the set $\mathcal{B}$ into non-empty sets $\{\mathcal{B}_i\}_{i\in[n]}$ such that $i \in B$ for all $B \in \mathcal{B}_i$. We also work out how this change affects our overall consumption of bandwidth, and the number (and type) of communication channels. For efficiency, we first select any map $f : \mathcal{B} \longrightarrow \mathcal{P}$ for which $\mathrm{Im}(f)$ is as large as possible.

Recall that our basic protocol works in the case that $\mathrm{Im}(f) = \mathcal{P}$. The modification is simply to apply the standard protocol for all $i \in \mathrm{Im}(f)$, and apply Maurer's protocol when $i \notin \mathrm{Im}(f)$. The multiplication protocol then becomes:

1. For each $i \in \mathcal{P}$, party i computes $v_i \leftarrow \sum_{\rho(B_1,B_2)=i} x_{B_1} \cdot y_{B_2}$.
2. The parties call $\mathcal{F}_{\mathsf{Rand}}$.**PRZS** so that each player $i \in \mathcal{P}$ obtains t_i such that $\sum_{i\in\mathcal{P}} t_i = 0$.
3. For each $i \in \mathrm{Im}(f)$
   (a) Party i samples $\{u_B\}_{B\in\mathcal{B}_i} \leftarrow \mathbb{F}$ such that $\sum_{B\in\mathcal{B}_i} u_B = v_i + t_i$.
   (b) Party i sends, for all $B \in \mathcal{B}_i$, the value u_B securely to party j for all $j \in B \setminus \{i\}$.
4. For each $i \notin \mathrm{Im}(f)$
   (a) Party i samples $\{s^i_B\}_{B\in\mathcal{B}} \leftarrow \mathbb{F}$ such that $\sum_{B\in\mathcal{B}} s^i_B = v_i + t_i$. Note that the sum is over all $B \in \mathcal{B}$ not $B \in \mathcal{B}_i$ (which by assumption is empty).
   (b) Party i sends, for all $B \in \mathcal{B}$, the value s^i_B securely to party j for all $j \in B \setminus \{i\}$. Note, the transmission is over all $B \in \mathcal{B}$ not $\mathcal{B}_i$.
5. Party i for each $B \in \mathcal{B}$ with $i \in B$ computes

$$z_B = u_B + \sum_{j\notin\mathrm{Im}(f)} s^j_B.$$

The fact that the multiplication protocol is correct and secure can be easily verified. The only issue is to adapt our formulae for the number of secure and authenticated channels needed. Instead of the graph $\mathcal{G}_\Gamma$, we have

$$\widetilde{\mathcal{G}_\Gamma} = \left(\bigcup_{i\in\mathrm{Im}(f)} \bigcup_{B\in\mathcal{B}_i} \bigcup_{j\in B\setminus\{i\}} \{(i,j)\} \right) \bigcup \left(\bigcup_{i\notin\mathrm{Im}(f)} \bigcup_{B\in\mathcal{B}} \bigcup_{j\in B\setminus\{i\}} \{(i,j)\} \right).$$

and hence we require the set $\mathsf{SC}(\widetilde{\mathcal{G}_\Gamma})$ of secure channels. The number of finite-field elements needed to be transmitted in our passively secure protocol above becomes

$$\left(\sum_{B\in\mathcal{B}} (|B|-1) \right) + \sum_{i\notin\mathrm{Im}(f)} \left(\sum_{B\in\mathcal{B}:B\ni i} (|B|-1) + \sum_{B\in\mathcal{B},B\not\ni i} |B| \right).$$

Recall that for the set of authenticated channels $\mathcal{H}_\Gamma$, needed in the actively secure variant, we just need to guarantee that every party receives one share from at least one player. Hence, each party in $\mathcal{P} \setminus \mathrm{Im}(f)$ can receive all their

required values from any one of the parties in $\mathrm{Im}(f)$. Thus, instead of $\mathcal{H}_\Gamma$, we have

$$\widetilde{\mathcal{H}_\Gamma} = \left(\bigcup_{i \in \mathrm{Im}(f)} \bigcup_{B \in \mathcal{B}_i} \bigcup_{j \notin B} \{(i,j)\} \right).$$

and hence a set $\mathsf{AC}(\widetilde{\mathcal{H}_\Gamma})$ of authenticated channels is needed in place of $\mathcal{H}_\Gamma$ in our actively secure protocol.

5 Summary

To make clear what channels are required when, and how many, we provide Table 1. Following the standard mathematical notation, we use $\mathcal{K}_n$ to denote the complete graph on n vertices (i.e. parties) so that, for example, $\mathsf{SC}(\mathcal{K}_n)$ means that the n parties are connected in a complete network of secure channels. The table presents the costs in terms of the sets of edges $\mathcal{K}_n$, $\mathcal{G}_\Gamma$ and $\mathcal{H}_\Gamma$. Apart from the first set, the cardinalities of these sets depend crucially on the precise access structure one is considering, so it is not possible to give formulae describing their size. However, since $\mathcal{G}_\Gamma$ and $\mathcal{H}_\Gamma$ are strict subsets of $\mathcal{K}_n$, we always obtain benefits over the naive protocol(s).

Protocol	Procedure	Channels required
Π_{Rand}	Set-up	$\mathsf{SC}(\mathcal{K}_n)$
	PRSS	n/a
	PRZS	n/a
Passive Protocol	Input	$\mathsf{SC}(\mathcal{G}_\Gamma)$
	Multiplication	$\mathsf{SC}(\mathcal{G}_\Gamma)$
	Output to one	$\mathsf{SC}(\mathcal{K}_n)$
	Output to all	$\mathsf{AC}(\mathcal{K}_n)$
Active Offline Protocol	Triple Gen.	$\mathsf{SC}(\mathcal{G}_\Gamma)$
	Triple Sac.	$\mathsf{AC}(\mathcal{H}_\Gamma)$
	Authentication check	$\mathsf{AC}(\mathcal{K}_n)$
Active Online Protocol	Input	$\mathsf{SC}(\mathcal{H}_\Gamma) + \mathsf{AC}(\mathcal{K}_n)$
	Multiplication	$\mathsf{AC}(\mathcal{H}_\Gamma)$
	Output to one	$\mathsf{SC}(\mathcal{H}_\Gamma) + \mathsf{AC}(\mathcal{K}_n)$
	Output to all	$\mathsf{AC}(\mathcal{H}_\Gamma) + \mathsf{AC}(\mathcal{K}_n)$

Table 1. Number of channels needed at each point in the computation. The channels for "Output to one" assumes every party will receive private output. Notice that the active variant of our protocol never needs a complete network of secure channels in the online phase and that it only requires a complete authenticated network for the hash-comparison stage only.

The set-up of the protocol Π_{Rand} is a one-time offline phase used to generate sharings of random values at various points for zero communication cost. While it requires a complete network of secure channels, the main bottleneck

in secret-sharing-based MPC is in multiplication, for which our protocol significantly reduces the communication cost.

It should be noted that our online phase methodology can actually be executed using other secret-sharing schemes, assuming the Beaver triples in the offline phase are produced with respect to the corresponding secret-sharing scheme. In particular in the (n, t)-threshold case it turns out that we would obtain, using Shamir sharing, an online phase which only requires $n \cdot t$ authenticated channels, as opposed to $n \cdot (n-1)$ authenticated channels using the naïve protocol.

Acknowledgements

This work has been supported in part by ERC Advanced Grant ERC-2015-AdG-IMPaCT, by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070, and by EPSRC via grants EP/M012824 and EP/N021940/1.

References

1. T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 805–817, Vienna, Austria, Oct. 24–28, 2016. ACM Press.
2. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press.
3. D. Beaver and A. Wool. Quorum-based secure multi-party computation. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 375–390, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.
4. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.
5. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.
6. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In S. Jajodia and J. López, editors, *ESORICS 2008: 13th European Symposium on Research in Computer Security*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206, Málaga, Spain, Oct. 6–8, 2008. Springer, Heidelberg, Germany.
7. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19, Chicago, IL, USA, May 2–4, 1988. ACM Press.

8. R. Cramer, I. Damgård, and Y. Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In J. Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362, Cambridge, MA, USA, Feb. 10–12, 2005. Springer, Heidelberg, Germany.
9. I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous multiparty computation: Theory and implementation. In S. Jarecki and G. Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 160–179, Irvine, CA, USA, Mar. 18–20, 2009. Springer, Heidelberg, Germany.
10. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Heidelberg, Germany.
11. J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In J. Coron and J. B. Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 225–255, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.
12. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In A. Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.
13. S. Goldwasser and Y. Lindell. Secure computation without agreement. In D. Malkhi, editor, *Distributed Computing, 16th International Conference, DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*, volume 2508 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2002.
14. M. Hirt and U. M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In J. E. Burns and H. Attiya, editors, *16th ACM Symposium Annual on Principles of Distributed Computing*, pages 25–34, Santa Barbara, CA, USA, Aug. 21–24, 1997. Association for Computing Machinery.
15. M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.
16. U. M. Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154(2):370–381, 2006.
17. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 681–700, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Heidelberg, Germany.