Efficient General-Adversary Multi-Party Computation

Martin Hirt, Daniel Tschudi*

ETH Zurich
{hirt,tschudid}@inf.ethz.ch

Abstract. Secure multi-party computation (MPC) allows a set $\mathcal{P}$ of n players to evaluate a function f in presence of an adversary who corrupts a subset of the players. In this paper we consider active, general adversaries, characterized by a so-called adversary structure $\mathcal{Z}$ which enumerates all possible subsets of corrupted players. In particular for small sets of players general adversaries better capture real-world requirements than classical threshold adversaries.

Protocols for general adversaries are "efficient" in the sense that they require $|\mathcal{Z}|^{\mathcal{O}(1)}$ bits of communication. However, as $|\mathcal{Z}|$ is usually very large (even exponential in n), the exact exponent is very relevant. In the setting with perfect security, the most efficient protocol known to date communicates $\mathcal{O}(|\mathcal{Z}|^3)$ bits; we present a protocol for this setting which communicates $\mathcal{O}(|\mathcal{Z}|^2)$ bits. In the setting with statistical security, $\mathcal{O}(|\mathcal{Z}|^3)$ bits of communication is needed in general (whereas for a very restricted subclass of adversary structures, a protocol with communication $\mathcal{O}(|\mathcal{Z}|^2)$ bits is known); we present a protocol for this setting (without limitations) which communicates $\mathcal{O}(|\mathcal{Z}|^1)$ bits.

Keywords: Secure Multiparty Computation, General Adversaries, Efficiency

1 Introduction

Secure Multi-Party Computation Secure Multi-Party Computation (MPC) allows a set $\mathcal{P}$ of n players to securely evaluate a function f even when a subset of the players is corrupted by a central adversary. MPC was introduced by Yao [Yao82]. A first solution (with computational security) was given by Goldreich, Micali, and Wigderson [GMW87]. Later solutions [BGW88, CCD88, RB89] provide statistical and even perfect security. All these protocols consider threshold adversaries (characterized by an upper bound t on the number of corrupted parties). This was generalized in [HM00] by considering so-called general adversaries, characterized by an adversary structure $\mathcal{Z} = \{Z_1, \ldots, Z_\ell\}$, which enumerates all possible subsets of corrupted players.
In the setting with perfect active security, MPC is achievable if and only if $t < \frac{n}{3}$, respectively $\mathcal{Q}^3(\mathcal{P}, \mathcal{Z})$ (the union of no *three* sets in $\mathcal{Z}$ covers $\mathcal{P}$). In the setting with statistical or cryptographic active security, MPC is achievable if and only if $t < \frac{n}{2}$, respectively $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$ (the union of no *two* sets in $\mathcal{Z}$ covers $\mathcal{P}$).

* Research supported in part by the Swiss National Science Foundation (SNF), project no. 200020-132794

K. Sako and P. Sarkar (Eds.) ASIACRYPT 2013, Part II, LNCS 8270, pp. 181200, 2013.

Threshold vs. General Adversaries Clearly, general adversaries are more flexible, which is relevant in particular when the set of players is not very large. However, general-adversary protocols are typically by orders of magnitude less efficient than threshold protocols; more specifically, threshold protocols usually communicate $\mathrm{Poly}(n)$ bits per multiplication, whereas general-adversary protocols require $\mathrm{Poly}(|Z|)$ bits. As typically $|\mathcal{Z}|$ is exponential in n, this is a huge drawback. However, in some scenarios (e.g. with very different types of players), threshold protocols are not applicable, and general-adversary protocols must be used. In these settings, the concrete communication complexity of the general-adversary protocol is highly relevant: For example for $n = 25$, $|\mathcal{Z}|$ is expected to be around one million, and a protocol communicating $|\mathcal{Z}| \cdot \mathrm{Poly}(n)$ might be acceptable, whereas a protocol communicating $|\mathcal{Z}|^3 \cdot \mathrm{Poly}(n)$ might be useless.

Contributions In the statistically-secure model, one can tolerate at most adversary structures satisfying $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$. The most efficient protocol known to date, which is also optimal in terms of resilience, requires $|\mathcal{Z}|^3 \cdot \mathrm{Poly}(n, \kappa)$ bits of communication (where κ is the security parameter) [Mau02, HMZ08]. There exists a protocol with communication complexity of $|\mathcal{Z}|^2 \cdot \mathrm{Poly}(n, \kappa)$ [PSR03]. But this results is non-optimal in terms of resilience, as it tolerates only adversaries satisfying $\mathcal{Q}^3$.
Using a new approach for multiplication, we construct a protocol communicating $|\mathcal{Z}| \cdot \mathrm{Poly}(n, \kappa)$ bits and tolerating $\mathcal{Q}^2$ adversary structures. This protocol is optimal both in terms of overall efficiency and resilience. We stress that even with cryptographic security, $\mathcal{Q}^2$ is necessary and complexity linear in $|\mathcal{Z}|$ is required at least with respect to the computation (see [Hir01]).
Furthermore, we present a perfectly secure protocol (with no error probability) with communication complexity of $|\mathcal{Z}|^2 \cdot \mathrm{Poly}(n)$. It is optimal in terms of resilience ($\mathcal{Q}^3$) and also the most efficient protocol up to date in the model with perfect security.

Setting	Cond.	Bits / Mult.	Reference
passive perfect	$\mathcal{Q}^2$	$\|\mathcal{Z}\| \cdot \mathrm{Poly}(n)$	[Mau02]
active perfect	$\mathcal{Q}^3$	$\|\mathcal{Z}\|^3 \cdot \mathrm{Poly}(n)$	[Mau02]
active perfect	$\mathcal{Q}^3$	$\|\mathcal{Z}\|^2 \cdot \mathrm{Poly}(n)$	our result
active unconditional	$\mathcal{Q}^2$	$\|\mathcal{Z}\|^3 \cdot \mathrm{Poly}(n, \kappa)$	[Mau02]/[HMZ08]
active unconditional	$\mathcal{Q}^3$	$\|\mathcal{Z}\|^2 \cdot \mathrm{Poly}(n, \kappa)$	[PSR03]
active unconditional	$\mathcal{Q}^2$	$\|\mathcal{Z}\| \cdot \mathrm{Poly}(n, \kappa)$	our result

Table 1. Communication complexity of different protocols

2 Preliminaries

Players and Computation Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n players. The players in $\mathcal{P}$ want to compute a function f over some finite field $\mathbb{F}$. The function is specified

by a circuit $\mathcal{C}$ consisting of input, output, random, addition, and multiplication gates. In an ideal world, a trusted party does all the computation. In the real world, players are connected by a complete network of secure (private and authentic) synchronous channels. There exist authenticated broadcast channels (they can be simulated by the players, see e.g. [FM98] or [PW96]). In order to compute the function f, the players simulate the trusted party by using some MPC-protocol Π.

Adversary and Adversary Structure Dishonesty is modeled in terms of a central adversary $\mathcal{A}$ who corrupts players. During the execution of the protocol the adversary can access the internal state of corrupted players and make them deviate from the protocol at will. We allow that the adversary is computationally unbounded. Before the execution of the protocol the adversary has to specify the players he wants to corrupt. His choice is limited by means of an adversary structure $\mathcal{Z} = \{Z_1, \ldots, Z_\ell\} \subseteq 2^{\mathcal{P}}$, i.e. all corrupted players must be part of an adversary set in $\mathcal{Z}$. We denote the chosen set by Z^*. Note that Z^* is not known to the honest players and is solely used for ease of notation. We say that $\mathcal{Z}$ satisfies the $\mathcal{Q}^k(\mathcal{P}, \mathcal{Z})$ property if $\mathcal{P} \not\subseteq Z_1 \cup \cdots \cup Z_k \quad \forall Z_1, \ldots, Z_k \in \mathcal{Z}$.

Security We say a protocol is $\mathcal{Z}$-secure if anything the adversary achieves during the execution of the protocol can be achieved in the ideal world as well. More precisely, for every adversary in the real world there exists an adversary in the ideal world such that both the information the adversary gets and the output of honest players are statistically indistinguishable for perfect security respectively statistically close for unconditional security. The main result from [HM97] states that $\mathcal{Q}^3(\mathcal{P}, \mathcal{Z})$ resp. $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$ are the necessary and sufficient conditions for the existence of perfectly resp. unconditionally $\mathcal{Z}$-secure protocols considering active adversaries. For simplicity we assume that all messages sent during the execution of Π are from the right domain. If a player receives a message where this is not the case, he replaces it with an arbitrary element from the right domain. If a player receives an unexpected message, he ignores it.

3 Perfect Protocol

In this section we present a perfectly $\mathcal{Z}$-secure protocol for an arbitrary adversary structure $\mathcal{Z}$ satisfying the $\mathcal{Q}^3$ property. The communication complexity of the protocol is quadratic in $|\mathcal{Z}|$. The efficiency gain is due to an improved multiplication protocol. The sharing is (up to presentation) the same as in [Mau02].

3.1 Secret Sharing

Secret sharing allows a player to distribute a secret value among the player set, such that only qualified subsets of players are able reconstruct it. The secret sharing used for our protocol is based on the one from [Mau02] / [BFH+08]. It is characterized by a *sharing specification* $\mathbb{S} = (S_1, \ldots, S_h)$, which is a tuple of subsets of $\mathcal{P}$.

Definition 1. *A value s is* shared *with respect to sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$ if the following holds:*

a) *There exist shares* $s_1, \ldots, s_h$ *such that* $s = \sum_{q=1}^{h} s_q$
b) *Each* s_q *is known to every (honest) player in* S_q

We denote the sharing of a value s by $[s]$ and use $[s]_q$ as notation for s_q, the q-th share. A sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$ is called $\mathcal{Z}$*-private* if for every $Z \in \mathcal{Z}$ there is an $S \in \mathbb{S}$ such that $Z \cap S = \emptyset$. A sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$ and an adversary structure $\mathcal{Z}$ satisfy $\mathcal{Q}^k(\mathbb{S}, \mathcal{Z})$ if $S \not\subseteq Z_1 \cup \cdots \cup Z_k \quad \forall Z_1, \ldots, Z_k \in \mathcal{Z}\ S \in \mathbb{S}$. If $\mathbb{S}$ is $\mathcal{Z}$-private, a sharing $[s]$ does not leak information to the adversary, as all shares known by the adversary are statistically independent of s. The players can compute a sharing of any linear combination of shared values (with respect to a sharing specification $\mathbb{S}$) by locally computing the linear combination of their shares. This property is called the linearity of the sharing. The following protocol Share allows a dealer P_D to correctly share value s among the players in $\mathcal{P}$.

Protocol Share($\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s$) [Mau02]

0: The dealer P_D takes s as input.
1: P_D splits s into random shares $s_1, \ldots, s_{|\mathbb{S}|}$ subject to $s = \sum_{q=1}^{|\mathbb{S}|} s_q$.
2: **for all** $q \in \{1, \ldots, |\mathbb{S}|\}$ **do**
3: P_D sends s_q to every player in S_q.
4: Each player in S_q forwards the received value to every player in S_q.
5: Each player in S_q checks that the received values are all the same and broadcasts OK, or NOK accordingly.
6: If a player in S_q broadcast NOK, the dealer broadcasts s_q and the players in S_q take this value (resp. some default value if the dealer does not broadcast) as share. Otherwise every player in S_q takes the value he received in Step 3 as share.
7: **end for**
8: The players in $\mathcal{P}$ collectively output $[s]$.

Lemma 1. *For any adversary structure* $\mathcal{Z}$ *the protocol* Share$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ *securely computes a sharing* $[s']$. *For honest* P_D *it holds that* $s' = s$. *The protocol communicates at most* $|\mathbb{S}|\,(n^2 + n) \log |\mathbb{F}|$ *bits and broadcasts at most* $|\mathbb{S}|\,(\log |\mathbb{F}| + n)$ *bits.*

Proof. Correctness: For each s_q either all the honest players in S_q hold the same value after Step 3, or one of them complains and they receive a consistent value in Step 6. Hence the protocol outputs a (consistent) sharing $[s']$. If the dealer is honest he is able to ensure in Steps 3 and 6 that the honest players use the intended value for s_q such that $s = s'$. Privacy: Let the dealer be honest, as otherwise secrecy is trivially fulfilled. All a player learns beyond his designated output are values broadcast in Step 6. However this does not violate secrecy as these values are already known to the adversary (from Step 3). Complexity: For each share at most $n + n^2$ values are sent and at most $n + \log |\mathbb{F}|$ bits broadcast. □

For publicly known value s the players can invoke DefaultShare to get a sharing $[s]$ without having to communicate.

Protocol DefaultShare($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, s)
0: Every player takes s as input.
1: The share s_1 is set to s and all other shares are set to 0.
2: The players in $\mathcal{P}$ collectively output $[s]$.

Lemma 2. DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, s)$ *securely computes a sharing* $[s]$ *where* s *is a publicly known value. The protocol does not communicate.*

Proof. Correctness: In Step 1 every honest player in S_q takes the same value for share s_q. As the sum of all shares is s, the protocol outputs a consistent sharing $[s]$. Privacy: During the protocol no communication occurs, hence the adversary does not obtain new information. □

The protocol ReconstructShare allows the reconstruction of a share $[s]_q$ to the players in some set R. This implies that the players can reconstruct a shared value $[s]$ by invoking ReconstructShare for each share.

Protocol ReconstructShare($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, $[s]_q$, R)
0: The players in S_q take the share $[s]_q$ as input.
1: Every player P_i in S_q sends $[s]_q$ to every player in R.
2: For each player $P_j \in R$ let $v_{j,i}$ be the value received from P_i. Then P_j outputs some value v_j such that there exists a $Z \in \mathcal{Z}$ with $v_{j,i} = v_j$ for all $P_i \in S_q \backslash Z$.

Lemma 3. *If* S_q *and* $\mathcal{Z}$ *satisfy* $\mathcal{Q}^2(S_q, \mathcal{Z})$*, the protocol* ReconstructShare *securely reconstructs the share* $[s]_q$ *to the players in* R*, such that every (honest) player outputs* $[s]_q$*. The protocol communicates at most* $n^2 \log |\mathbb{F}|$ *bits.*

Proof. Correctness: In Step 1 all honest player will send the same value $[s]_q$, which is a suitable choice for v_j for an (honest) player $P_j \in R$ in Step 2. For the sake of contradiction suppose there exist two values $v_1 \neq v_2$ with corresponding $Z_1, Z_2 \in \mathcal{Z}$ such that the condition of Step 2 holds for both of them. Hence $(S_q \backslash Z_1) \cap (S_q \backslash Z_2) = \emptyset$ and thus $S_q \subseteq Z_1 \cup Z_2$ which contradicts $\mathcal{Q}^2(S_q, \mathcal{Z})$. Therefore every honest players outputs the value $[s]_q$. Privacy: The adversary learns at most $[s]_q$ (if a malicious player is part of R). Complexity: Each player in S_q sends his value to at most n players. □

Protocol Reconstruct($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, $[s]$, R) [Mau02]
0: The players in $\mathcal{P}$ take collectively $[s]$ as input.
1: $\forall q \in \{1, \ldots, |\mathbb{S}|\}$ protocol ReconstructShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s]_q, R)$ is invoked.
2: The players in R locally sum up the obtained shares and output the sum s.

Lemma 4. *If* $\mathbb{S}$ *and* $\mathcal{Z}$ *satisfy* $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$ *and* $[s]$ *is a sharing of the value* s*, then* Reconstruct$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$ *securely reconstructs* s *to the players in* R*. The protocol communicates at most* $|\mathbb{S}|\, n^2 \log |\mathbb{F}|$ *bits.*

Proof. Correctness and privacy follow directly from Lemma 3. As ReconstructShare is invoked $|\mathbb{S}|$ times the complexity follows as well. □

3.2 Multiplication

We present a protocol for the perfectly-secure computation of the (shared) product of two shared values $[a]$ and $[b]$ (with respect to a sharing specification $\mathbb{S}$). Along the lines of [Mau02] the fundamental idea of multiplication is to assign each local product $a_p b_q$ to a player in $S_p \cap S_q$, who computes and shares his designated products. The sum of all these sharings is a sharing of ab as long as no player actively cheated. So each player is mapped to a collection of local products, formalized by a function $I : [n] \to 2^{\{(p,q) \mid 1 \le p,q \le |\mathcal{Z}|\}}$ with the constraint that $\forall (p,q)\ \exists!\ i$ such that $(p,q) \in I(i)$. W.l.o.g let $I(i) := \{(p,q) \mid P_i = \min_P\{P \in S_p \cap S_q\}\}$. We first show an optimistic multiplication protocol which takes an additional parameter Z and computes the correct product if the actual adversary set Z^* is a subset of Z. In this protocol local products are assigned to players in $\mathcal{P} \setminus Z$ only. Clearly this is possible if and only if for each local product a player in $\mathcal{P} \setminus Z$ holds both involved shares, i.e. $\forall S_p, S_q \in \mathbb{S} : S_p \cap S_q \setminus Z \neq \emptyset$. So for each $Z \in \mathcal{Z}$ let I_Z be a mapping as above with the additional constraint that $\forall P_i \in Z\ I_Z(i) = \emptyset$. Without loss of generality, let $I_Z(i) := \{(p,q) \mid P_i = \min_P\{P \in S_p \cap S_q \setminus Z\}\}$.

Protocol OptimisticMult($\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], Z$)

0: The players in $\mathcal{P}$ take collectively $[a]$, $[b]$ and Z as input.

1:

a) Each player $P_i \in \mathcal{P} \setminus Z$ (locally) computes his designated products and shares the sum $c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$.

b) For each $P_i \in Z$ DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, 0)$ is invoked to share $c_i = 0$.

2: The players collectively output $([c_1], \ldots, [c_n])$ and $[c] = \sum_{i=1}^{n} [c_i]$.

Lemma 5. *Let $Z \subseteq \mathcal{P}$ such that $\forall S_p, S_q \in \mathbb{S} : S_p \cap S_q \setminus Z \neq \emptyset$. Then the protocol* OptimisticMult *securely computes sharings $[c], ([c_1], \ldots, [c_n])$. If no player in $\mathcal{P} \setminus Z$ actively cheats (in particular, if $Z^* \subseteq Z$), then $\forall i\ c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$ and $c = ab$. The protocol communicates at most $\mathcal{O}(|\mathbb{S}|\, n^3 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}|\, (n \log |\mathbb{F}| + n^2))$ bits.*

Proof. Correctness: The properties of the sharing protocol guarantee that the outputs are valid sharings. If none of the players in $\mathcal{P} \setminus Z$ cheated actively, it holds for each P_i that $c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$. The condition $\forall S_p, S_q \in \mathbb{S} : S_p \cap S_q \setminus Z \neq \emptyset$ guarantees that $ab = \sum_{i=1}^{n} \sum_{(p,q) \in I_Z(i)} a_p b_q$. Hence it follows that $c = ab$. Privacy / Complexity: Follow directly from Lemmas 1 and 2. □

As the players do not know the actual adversary set Z^*, they invoke OptimisticMult once for each set $Z \in \mathcal{Z}$ (Step 1 of the Multiplication protocol). This guarantees that at least one of the resulting sharings is correct. By comparing them the players can determine whether cheating occurred (Step 2 of the Multiplication protocol). If all sharings are equal, no cheating occurred and any of the sharings can serve as sharing of the product. Otherwise at least one player cheated. In this case the (honest) players can identify him and remove all sharings where he was involved in computation,

as these sharings are potentially tampered (Step 3 of the Multiplication protocol). This checking and removing is repeated until all remaining sharing are equal (and hence correct). As the identification of cheaters does not reveal any information to the adversary, Multiplication allows the secure computation of the product of two shared secret values.

Protocol Multiplication$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b])$

0: Set $M = \emptyset$.

1: Invoke OptimisticMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], Z)$ to compute $([c_1^{(Z)}], \ldots, [c_n^{(Z)}])$ and $[c^{(Z)}]$ for each $Z \in \mathcal{Z}$.

2: Set $\mathcal{Z}_M := \{Z \in \mathcal{Z} \mid M \subseteq Z\}$, fix some $\widetilde{Z} \in \mathcal{Z}_M$ and reconstruct the differences $[c^{(\widetilde{Z})}] - [c^{(Z)}]$ $\forall Z \in \mathcal{Z}_M$.

3: If all differences are zero, output $[c^{(\widetilde{Z})}]$ as sharing of the product.

Otherwise let $([d_1], \ldots, [d_n]) := ([c_1^{(\widetilde{Z})}], \ldots, [c_n^{(\widetilde{Z})}])$, $([e_1], \ldots, [e_n]) := ([c_1^{(Z)}], \ldots, [c_n^{(Z)}])$, $D := I_{\widetilde{Z}}$ and $E := I_Z$, where $[c^{(\widetilde{Z})}] - [c^{(Z)}] \neq 0$.

a) Each P_i shares the $2n$ values $d_{i,j} = \sum_{(p,q) \in D(i) \cap E(j)} a_p b_q$ and $e_{i,j} = \sum_{(p,q) \in E(i) \cap D(j)} a_p b_q$

b) For each player P_i reconstruct the differences $[d_i] - \sum_{j=1}^{n} [d_{i,j}]$ and $[e_i] - \sum_{j=1}^{n} [e_{i,j}]$. If one of them is non-zero set $M \leftarrow M \cup \{P_i\}$ and continue at Step 2.

c) For each (ordered) pair (P_i, P_j) of players reconstruct the difference $[d_{i,j}] - [e_{j,i}]$. If it is non-zero, reconstruct $[d_{i,j}]$,$[e_{j,i}]$ and all shares $\{a_p, b_q \mid (p,q) \in D(i) \cap E(j)\}$ to find the cheater $P \in \{P_i, P_j\}$. Set $M \leftarrow M \cup \{P\}$ and continue at Step 2.

Lemma 6. *If* $\mathbb{S}$ *and* $\mathcal{Z}$ *satisfy* $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$ *the protocol* Multiplication *yields a sharing* $[c] = [ab]$. *No information is leaked to the adversary.* Multiplication *communicates at most* $\mathcal{O}(|\mathbb{S}|\,|\mathcal{Z}|\,n^3 \log|\mathbb{F}| + |\mathbb{S}|\,n^5 \log|\mathbb{F}|)$ *bits and broadcasts at most* $\mathcal{O}(|\mathbb{S}|\,|\mathcal{Z}|\,(n \log|\mathbb{F}| + n^2) + |\mathbb{S}|\,(n^3 \log|\mathbb{F}| + n^4))$ *bits.*

Proof. Correctness: By invoking OptimisticMult for each $Z \in \mathcal{Z}$ it holds for Z^* that $[c^{(Z^*)}] = [ab]$ (due to $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$ $\forall S_p, S_q \in \mathbb{S} : S_p \cap S_q \setminus Z \neq \emptyset$ holds) . If for every $Z \in \mathcal{Z}_M$ the difference in Step 2 is zero, then $[c^{(Z)}] = [ab]$ $\forall Z \in \mathcal{Z}_M$ ($M = \emptyset$ at the beginning). Hence the protocol terminates successfully outputting a sharing of ab. Otherwise there exists $[c^{(\widetilde{Z})}] - [c^{(Z)}] \neq 0$ and thus $\sum_{i=1}^{n} [d_i] \neq \sum_{i=1}^{n} [e_i]$. In Step 3a) each player is supposed to share a partition of his shares. Hence one of the following cases must occur: There exists a player P_i such that $[d_i] \neq \sum_{j=1}^{n} [d_{i,j}]$ or $[e_i] \neq \sum_{j=1}^{n} [e_{i,j}]$. Or there exists a pair of players (P_i, P_j) such that $[d_{i,j}] \neq [e_{j,i}]$. In the first case P_i will be detected as cheater in Step 3b). In the second case the cheater will be detected in Step 3c). In both cases $M \subseteq \mathcal{P}$ is strictly increased, hence the protocol will terminate after at most n iterations. It holds that $M \subseteq Z^*$ and thus $Z^* \in \mathcal{Z}_M$. Therefore the correct sharing $[c^{(Z^*)}]$ is always used in Step 2 and the protocol will output the correct result. *Privacy*: By the properties of the sharing scheme and Lemma 5 the invocation of Share, Reconstruct, OptimisticMult does not violate privacy. The

adversary learns the differences reconstructed in Steps 2 and 3 of Multiplication, which are all zero unless the adversary cheats. In case of cheating the reconstructed values depends solely on the inputs of the adversary and are thus already known to him, thus privacy is not violated. All values further reconstructed in Step 3c) are known to the adversary before, as either P_i or P_j is corrupted. *Complexity*: Follows from Lemmas 1, 4 and 5 by counting the number of invocations of the corresponding sub-protocols. □

3.3 MPC Protocol

Combining Share, Reconstruct, and Multiplication the players can securely compute a circuit $\mathcal{C}$ over $\mathbb{F}$, where all intermediate values are shared according to Definition 1.

Protocol MPC$(\mathcal{P}, \mathcal{Z}, \mathcal{C})$

0: The players take $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ as sharing specification.

1: For every gate of C being evaluated do the following:

- *Input gate for P_D:* Share$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ is invoked to share s, where P_D is the input-giving player.
- *Linear gate:* The linear combination of the corresponding shares is computed locally using the linearity of the sharing.
- *Random gate:* Each player shares a random value. The sum of these values is used as output of the gate.
- *Multiplication gate:* Multiplication$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b])$ is used to multiply $[a]$ and $[b]$.
- *Output gate:* The players invoke Reconstruct$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$ to reconstruct the sharing $[s]$ to players in R.

Theorem 1. *Let $\mathcal{P}$ be a set of n players, $\mathcal{C}$ a circuit over $\mathbb{F}$ and $\mathcal{Z}$ an adversary structure satisfying $\mathcal{Q}^3(\mathcal{P}, \mathcal{Z})$, then* MPC$(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ *perfectly $\mathcal{Z}$-securely evaluates $\mathcal{C}$. It communicates $|\mathcal{C}|\,|\mathcal{Z}|^2 \cdot \mathrm{Poly}(n, \log|\mathbb{F}|)$ bits.*

Proof. It is easy to see that $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ is a sharing specification satisfying $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$. Hence by the properties of the sharing scheme and Lemma 6 the statement follows. The protocol communicates $\mathcal{O}(|\mathcal{C}|\,|\mathcal{Z}|^2\, n^3 \log|\mathbb{F}| + |\mathcal{C}|\,|\mathcal{Z}|\, n^5 \log|\mathbb{F}|)$ bits and broadcasts $\mathcal{O}(|\mathcal{C}|\,|\mathcal{Z}|^2\,(n \log|\mathbb{F}| + n^2) + |\mathcal{C}|\,|\mathcal{Z}|\,(n^3 \log|\mathbb{F}| + n^4))$ bits. Broadcast can be simulated with the protocol in [FM98], which communicates $\mathrm{Poly}(n)$ bits in order to broadcast one bit. This yields the claimed communication complexity. □

4 Unconditional Protocol

Our main result is an MPC protocol unconditionally $\mathcal{Z}$-secure for an $\mathcal{Q}^2$ adversary structure $\mathcal{Z}$. Its communication complexity is linear in $|\mathcal{Z}|$. This is the first protocol reaching the optimal lower bound of $\Omega(|\mathcal{Z}|)$ on the computational complexity (see Section 6).

4.1 Information Checking

In the perfect model, $\mathcal{Q}^3$ enables the honest players to securely reconstruct shares, as it assures that every share is held by enough honest players. Here, $\mathcal{Q}^2$ only ensures that each share is held by at least one honest player. Correctness is achieved by the use of information checking, a technique that prevents (malicious) players from announcing wrong values (see [RB89, Bea91a, CDD+99, HMZ08]). The following information-checking protocol is a slight variation of [CDD+99]. It is a three party protocol between a sender P_i, a recipient P_j and a verifier P_k. The sender P_i provides P_j with some authentication tag and P_k with some verification tag, such that P_j later can prove the authenticity of a value s to the verifier P_k. We assume that each pair P_i, P_k of players knows a fixed secret value $\alpha_{i,k} \in \mathbb{F} \setminus \{0, 1\}$.

Definition 2. *A vector (s, y, z, α) is 1-consistent if there exists a polynomial f of degree 1 over $\mathbb{F}$ such that $f(0) = s, f(1) = y, f(\alpha) = z$. We say a value s is (P_i, P_j, P_k)-*authenticated *if P_j knows s and some authentication tag y and P_k knows a verification tag z such that $(s, y, z, \alpha_{i,k})$ is 1-consistent. The vector $(y, z, \alpha_{i,k})$ is denoted by $A_{i,j,k}(s)$.*

Lemma 7. *A (P_i, P_j, P_k)-authenticated value s does not leak information to P_k.*

Proof. The verification tag z is statistically independent of the value s. □

Lemma 8. *Let s be (P_i, P_j, P_k)-authenticated, i.e. $(s, y, z, \alpha_{i,k})$ is 1-consistent. Then for P_j being able to find an authentication tag y' for a value $s' \neq s$ such that $(s', y', z, \alpha_{i,k})$ is 1-consistent is equivalent to finding $\alpha_{i,k}$.*

Proof. If both $(s, y, z, \alpha_{i,k})$ and $(s', y', z, \alpha_{i,k})$ are 1-consistent, then also $(s - s', y - y', 0, \alpha_{i,k})$ is 1-consistent. The corresponding polynomial of degree 1 is not parallel to the x-axis, as $s - s' \neq 0$. Thus it has an unique root at $\alpha_{i,k} = \frac{s-s'}{s-s'-y+y'}$.

Lemma 9. *The players P_j and P_k can locally compute an authentication and a verification tag of any linear combination of (P_i, P_j, P_k)-authenticated values (for fixed P_i). This is called the linearity of the authentication.*

Proof. Let s_a and s_b be (P_i, P_j, P_k)-authenticated with authentication tags y_a, y_b and verification tags z_a, z_b and the (fixed) point $\alpha_{i,k}$ and let L be a linear function. Then $L(s_a, s_b)$ is (P_i, P_j, P_k)-authenticated with authentication tag $y = L(y_a, y_b)$ and verification tag $z = L(z_a, z_b)$. This works as the polynomials of degree 1 over $\mathbb{F}$ form a vector space, hence $(L(s_a, s_b), L(y_a, y_b)\ , L(z_a, z_b), \alpha_{i,k})$ is 1-consistent. □

Let s be a value known to P_j and P_k. Then these players can use the protocol Default Authenticate to (P_i, P_j, P_k)-authenticate s without communication for arbitrary P_i. Note that P_i does not play an (active) role in this protocol.

Protocol DefaultAuthenticate(P_i, P_j, P_k, s)
0: P_j, P_k take the value s as input.
1: P_j outputs authentication tag $y = s$. P_k outputs verification tag $z = s$.

Lemma 10. *If the value s is known to the honest players in $\{P_j, P_k\}$ protocol* Default Authenticate(P_i, P_j, P_k, s) *securely (P_i, P_j, P_k)-authenticates s without any communication.*

Proof. Correctness: $(s, s, s, \alpha_{i,k})$ is 1-consistent for any $\alpha_{i,k}$. *Privacy/Communication:* No communication occurs. □

The non-robust protocol Authenticate allows to securely (P_i, P_j, P_k)-authenticate a (secret) value s.

Protocol Authenticate(P_i, P_j, P_k, s)

0: P_i and P_j take the value s as input.
1: P_i chooses random values $(y, z) \in \mathbb{F}$ such that $(s, y, z, \alpha_{i,k})$ is 1-consistent and random values $(s', y', z') \in \mathbb{F}$ such that $(s', y', z', \alpha_{i,k})$ is 1-consistent and sends (s', y, y') to P_j and (z, z') to P_k
2: P_k broadcasts random $r \in \mathbb{F}$.
3: P_i broadcasts $s'' = rs + s'$ and $y'' = ry + y'$.
4: P_j checks if $s'' = rs + s'$ and $y'' = ry + y'$ and broadcast OK or NOK accordingly. If NOK was broadcast the protocol is aborted.
5: P_k checks if $(s'', y'', rz + z', \alpha_{i,k})$ is 1-consistent. If yes P_k sends OK to P_j otherwise he sends $(\alpha_{i,k}, z)$ to P_j, who adjusts y such that $(s, y, z, \alpha_{i,k})$ is 1-consistent.
6: P_j outputs y and P_k outputs z.

Lemma 11. *If P_k is honest and s is known to the honest players in $\{P_i, P_j\}$. Then* Authenticate(P_i, P_j, P_k, s) *either securely (P_i, P_j, P_k)-authenticates s or aborts except with error probability of at most $\frac{1}{|\mathbb{F}|}$. In the case of an abort a player in $\{P_i, P_j\}$ is corrupted. The protocol communicates at most $7 \log |\mathbb{F}|$ bits and broadcasts at most $3 \log |\mathbb{F}| + 1$ bits.*

Proof. Correctness: If the protocol was aborted, either $s'' \neq rs + s'$ or $y'' \neq ry + y'$ meaning P_i is corrupted, or P_j misleadingly accused P_i. Otherwise, the players use some $(s, y, z, \alpha_{i,k})$ as authentication of s. The probability that $(s, y, z, \alpha_{i,k})$ is not 1-consistent is $|\mathbb{F}|^{-1}$, as for a fixed r there is exactly one way to choose y, z such that the inconsistency is not detected. *Privacy:* The verification tag z, the values s'' and y'' are statistically independent of the value s. Also $\alpha_{i,k}$ is sent only to P_j if either P_i or P_k is malicious. *Communication:* Seen by counting the number of messages sent or broadcast during the protocol. □

Remark 1. If the (honest) players P_i and P_j do not know the same s the protocol will abort as well.

Assume that P_k knows a candidate s' for a (P_i, P_j, P_k)-authenticated value s. If P_j wants to prove the authenticity of s' (i.e. that $s' = s$) the players invoke the protocol Verify. If P_k accepts the proof he outputs s', otherwise he outputs $\perp$.

Protocol Verify($P_i, P_j, P_k, s', A_{i,j,k}(s)$)
0: Let $A_{i,j,k}(s) = (y, z, \alpha_{i,k})$. P_j takes y as input and P_k takes s', z as input.
1: P_j sends y to P_k
2: P_k outputs s' if $(s', y, z, \alpha_{i,k})$ is 1-consistent otherwise $\perp$.

Lemma 12. *Assume s is (P_i, P_j, P_k)-authenticated and let P_k be an honest player knowing s'. If P_j is honest and $s' = s$, P_k will output s in* Verify. *Otherwise P_k will output $\perp$ or s except with error probability of at most $\frac{1}{|\mathbb{F}|-2}$. The protocol communicates at most $\log |\mathbb{F}|$ bits.*

Proof. Correctness: Let P_k be an honest player,let $A_{i,j,k}(s) = (y, z, \alpha_{i,k})$ be consistent with s, i.e. $(s, y, z, \alpha_{i,k})$ is 1-consistent and assume that $s' = s$. If P_j sends the right y the vector $(s', y, z, \alpha_{i,k})$ is 1-consistent and P_k will output s. Otherwise P_k always outputs $\perp$. So assume $s' \neq s$. Then the probability of finding y' such that the vector $(s', y', z, \alpha_{i,k})$ is 1-consistent is at most $\frac{1}{|\mathbb{F}|-2}$, thus P_k outputs $\perp$ except with error probability of at most $\frac{1}{|\mathbb{F}|-2}$. *Privacy/Communication:* No information except y is sent. □

4.2 Unconditional Secret Sharing

Starting from the secret sharing of Section 3.1 we construct a sharing scheme for the $\mathcal{Q}^2$ case using the information-checking scheme of the previous section.

Definition 3. *A value s is* shared *with respect to the sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$, if the following holds:*

a) *There exist shares $s_1, \ldots, s_h$ such that $s = \sum_{q=1}^{h} s_q$*
b) *Each s_q is known to every (honest) player in S_q*
c) *$\forall P_i, P_j \in S_q \; P_k \in \mathcal{P}$ s_q is (P_i, P_j, P_k)-authenticated.*

We denote the sharing of a value s by $[s]$. Let $[s]_q = (s_q, \{A_{i,j,k}(s_q)\})$, where s_q is the q-th share and $\{A_{i,j,k}(s_q)\}$ the set of all associated authentications. As the perfect sharing from Section 3.1 this sharing is linear and does not leak information to the adversary (for a $\mathcal{Z}$-private $\mathbb{S}$).
The following protocol allows a dealer P_D to securely share a secret value s.

Protocol Share($\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s$)
0: The dealer P_D takes s as input.
1: P_D splits s into random shares $s_1, \ldots, s_{|\mathbb{S}|}$ subject to $s = \sum_{q=1}^{|\mathbb{S}|} s_q$.
2: **for all** $q \in \{1, \ldots, |\mathbb{S}|\}$ **do**
3: P_D sends share s_q to every player in S_q.
4: $\forall P_i, P_j \in S_q$ and $\forall P_k \in \mathcal{P}$ invoke Authenticate(P_i, P_j, P_k, s_q).
If (for fixed q) any Authenticate(P_i, P_j, P_k, s_q) was aborted
P_D broadcasts s_q, the players in S_q replace there share and
DefaultAuthenticate(P_i, P_j, P_k, s_q) is invoked $\forall P_i, P_j \in S_q \; \forall P_k \in \mathcal{P}$.
5: **end for**
6: The players in $\mathcal{P}$ collectively output $[s]$.

Lemma 13. *For any adversary structure $\mathcal{Z}$ the protocol* $\mathsf{Share}(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ *securely computes a sharing* $[s']$ *except with error probability of at most* $\frac{1}{|\mathbb{F}|} n^3 |\mathbb{S}|$ *and if* P_D *is honest* $s' = s$*. The protocol communicates at most* $|\mathbb{S}| (7n^3 + n) \log |\mathbb{F}|$ *bits and broadcasts at most* $|\mathbb{S}| ((3n^3 + 1) \log |\mathbb{F}| + n^3)$ *bits.*

Proof. Correctness: Assume that P_D does not send the same value to the (honest) players in S_q (Step 3). In this case at least one invocation of Authenticate will abort (see Remark 1) and P_D must broadcast the value. Otherwise all (honest) player use the same value s_q in Step 3. We have to show that every (honest) P_j gets his authentications $A_{i,j,k}(s_q)$. If all instances of Authenticate do not abort the statement follows from Lemma 11. Otherwise s_q is broadcast and the players use DefaultAuthenticate resulting in the proper sharing state (c.f. Lemma 10). Note that a single invocation of Authenticate has an error probability of at most $\frac{1}{|\mathbb{F}|}$, so the above upper bound on the error probability follows. *Privacy:* We only have to check that broadcasting s_q in Step 4 does not violate privacy. But s_q is only broadcast when at least one Authenticate was aborted. In this case either P_D or a player in S_q is malicious, hence s_q is known to the adversary before the broadcast (Lemma 11 and Remark 1). *Communication:* Follows directly by counting the numbers of messages sent or broadcast (c.f. Lemmas 11 and 10) □

If a value is publicly known the player can use DefaultShare to obtain a sharing of it.

Protocol DefaultShare($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, s)

0: Every player takes s as input.
1: The share s_1 is set to s and all other shares are set to 0.
2: DefaultAuthenticate(P_i, P_j, P_k, s_q) is invoked $\forall S_q \forall P_i, P_j \in S_q \; \forall P_k \in \mathcal{P}$.
3: The players in $\mathcal{P}$ collectively output $[s]$.

Lemma 14. $\mathsf{DefaultShare}(\mathcal{P}, \mathcal{Z}, \mathbb{S}, s)$ *securely computes a sharing* $[s]$ *of* s*. The protocol does not communicate.*

Proof. The statement follows from Lemmas 2 and 10. □

The protocol ReconstructShare allows reconstruction of a share from some sharing $[s]$ to players in $R \subseteq \mathcal{P}$. Hence the players can reconstruct s by invoking protocol ReconstructShare for each share of $[s]$.

Protocol ReconstructShare($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, $[s]_q$, R)

0: The players in S_q take collectively $[s]_q = (s_q, \{A_{i,j,k}(s_q)\})$ as input.
1: Every player P_j in S_q sends s_q to every player in R.
2: **for all** $P_j \in S_q, P_k \in R$ **do**
3: Invoke $\mathsf{Verify}(P_i, P_j, P_k, s_q^{(j)}, A_{i,j,k}(s_q)) \; \forall P_i \in S_q$ where $s_q^{(j)}$ is the value received by P_k from P_j in Step 1. If P_k output $s_q^{(j)}$ in each invocation he acccepts it as value for s_q.
4: **end for**
5: Each P_k outputs some value he accepted in Step 3 (or $\perp$ if never accepted a value).

Lemma 15. *Assume S_q and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(S_q, \mathcal{Z})$ and let $[s]_q$ be a consistent share. Every honest player in R outputs s_q in* ReconstructShare *except with error probability of at most $\frac{1}{|\mathbb{F}|-2} n\,|S_q|$. The protocol communicates at most $(n^3 + n^2)\log|\mathbb{F}|$ bits and does not broadcast.*

Proof. Correctness: As S_q and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(S_q, \mathcal{Z})$ there exists at least one honest player P_j in S_q, who sends the right value s_q to $P_k \in R$ in Step 1. Hence every (honest) P_k will accept s_q in Step 3 from P_j, as P_j has a valid authentication for s_q from every player in S_q (c.f. Lemma 12). On the other hand a malicious player does not have a valid authentication for $s'_q \neq s_q$ from every player in S_q (one of them is honest!). So no honest player will accept $s'_q \neq s_q$ in Step 3 and thus P_k output s_q in the last step except with error probability of at most $\frac{1}{|\mathbb{F}|-2}\,|S_q|$ (c.f. Lemma 12). As there are at most n players in R the overall error probability follows. *Privacy:* Follows from Lemma 12. *Communication:* Follows directly by counting the numbers of messages sent (c.f. Lemma 12) □

Protocol Reconstruct$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$
0: The players in $\mathcal{P}$ take collectively $[s]$ as input.
1: **for all** $q = 1, \ldots, |\mathbb{S}|$ **do**
2: ReconstructShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s]_q, R)$ is invoked.
3: **end for**
4: The players locally sum up the shares to obtain and output s.

Lemma 16. *Assume $\mathbb{S}$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$ and let $[s]$ be a sharing of the value s. Every honest player in R outputs s in* Reconstruct *except with error probability of at most $\frac{1}{|\mathbb{F}|-2} n^2\,|\mathbb{S}|$. The protocol communicates at most $|\mathbb{S}|\,(n^3 + n^2)\log|\mathbb{F}|$ bits and does not broadcast.*

Proof. The statement follows directly from Lemma 15, as the players invoke the protocol ReconstructShare for each share. □

4.3 Multiplication

We present a protocol for the unconditionally-secure computation of the (shared) product of two shared values $[a]$ and $[b]$. The idea is, as in the perfect case, to use an optimistic multiplication. The protocol BasicMult takes a set M of (identified) malicious players as input and outputs the correct product given that no player in $\mathcal{P} \setminus M$ actively cheated. In a next step a probabilistic check is used to determine whether the product computed in BasicMult is correct. This allows us to detect malicious behaviour. If cheating occured, all involved sharings (from BasicMult) are reconstructed to identify a cheater in $\mathcal{P} \setminus M$. These reconstructions violate the privacy of the involved factors the protocol is not used directly in the actual circuit computation. Instead we use it to multiply two random values and make use of circuit randomization from [Bea91b] for actual multiplication gates.

Protocol BasicMult($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, $[a]$, $[b]$, M)

0: The players in $\mathcal{P}$ take collectively $[a]$, $[b]$ and M as input.
1: $\forall S_q : S_q \cap M \neq \emptyset$ invoke ReconstructShare to reconstruct a_q and b_q.
2: *a*) Each player $P_i \in \mathcal{P} \setminus M$ (locally) computes his designated products and shares the sum $c_i = \sum_{(p,q)\in I(i)} a_p b_q$.
   b) For each $P_i \in M$ DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, c_i)$ is invoked where $c_i = \sum_{(p,q)\in I(i)} a_p b_q$.
3: The players collectively output $([c_1], \ldots, [c_n])$ and $[c] = \sum_{i=1}^{n} [c_i]$.

Lemma 17. *Let $M \subseteq Z^*$ be a set of (identified) malicious players and assume that $\mathcal{Z}$ and $\mathbb{S}$ satisfy $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$. Then* BasicMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], M)$ *securely computes sharings $[c], ([c_1], \ldots, [c_n])$ except with error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|} n^4 |\mathbb{S}|)$. If no player in $\mathcal{P} \setminus M$ actively cheats, then $\forall i\ c_i = \sum_{(p,q)\in I_Z(i)} a_p b_q$ and $c = ab$. The protocol communicates at most $\mathcal{O}(|\mathbb{S}|\, n^4 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}|\, n^4 \log |\mathbb{F}|)$ bits.*

Proof. Correctness: The properties of the sharing protocol guarantee that the outputs are valid sharings except with error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|} n^4 |\mathbb{S}|)$. The $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$ property allows the players to securely reconstruct shares and grants that there exists a proper assignment of players in $\mathcal{P}$ to the local products. If none of the players in $\mathcal{P} \setminus M$ cheated, it holds for each P_i that $c_i = \sum_{(p,q)\in I_Z(i)} a_p b_q$ (for players in M DefaultShare is used on reconstructed values). Privacy: All reconstructed shares a_q,b_q are known to players in M. Complexity: Follow directly from the properties of the sharing scheme (c.f. Lemmas 13, 14 and 15). □

Detectable Random Triple Generation The following unconditionally secure protocol takes a set M of malicious players as an additional input and computes a random multiplication triple $([a], [b], [c])$ where $c = ab$ given that no player in $\mathcal{P} \setminus M$ actively cheats. Otherwise it outputs a set of malicious players M' such that $M \subsetneq M'$. This protocol uses a probabilistic check to detect cheating. First the players generate a shared random challenge $[r]$ and a blinding $[b']$. Then they use BasicMult to compute the sharings $[c] = [a][b]$, $[c'] = [a][b']$ and check whether $[a](r[b] + [b']) = (r[c] + [c'])$. If this is the case the multiplication triple $([a], [b], [c])$ is output. Otherwise the players identify (at least) one cheater in $\mathcal{P} \setminus M$ by reconstructing $[a], [b], [b'], [c], [c']$.

Lemma 18. *If $\mathbb{S}$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$ and $M \subseteq Z^*$, the protocol* RandomTriple *outputs either a random multiplication triple $([a], [b], [c])$ or set $M' \subseteq Z^*$ where $M \subsetneq M'$ except with error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|} |\mathbb{S}|\, n^4) + \frac{1}{|\mathbb{F}|}$. No information is leaked to the adversary.* RandomTriple *communicates at most $\mathcal{O}(|\mathbb{S}|\, n^4 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}|\, n^4 \log |\mathbb{F}|)$ bits.*

Proof. Correctness: In Step 2, the players compute $[c]$ and $[c']$. Given that no player in $\mathcal{P} \setminus M$ actively cheated it holds that $c = ab$ and $c' = ab'$. In this case $[a](r[b] + [b']) - r[c] - [c']$, which is computed in Step 3, is zero for all r and the players reconstruct the random multiplication triple $([a], [b], [c])$. If $c \neq ab$ the difference $[a](r[b]+[b'])-r[c]-$

$[c']$ is non-zero except for at most one r and the players go to Step 5 with probability at least $(1 - \frac{1}{|\mathbb{F}|})$ (assuming that no errors happen in sharing and reconstruction of values). For at least one player $P_i \in \mathcal{P} \setminus M$ it must hold that $rc_i + c'_i \neq \sum_{(p,q)\in I(i)} r(a_p b_q) + (a_p b'_q)$. By opening all involved sharing it is easy to find these players. Thus it holds that $M \subsetneq M'$ and $M' \subseteq Z^*$. The overall error probability is composed of the error probability of the sharing scheme and the one of the random challenge check in Step 3. *Privacy:* Neither the protocol BasicMult nor the sharing scheme do violate privacy (c.f. Lemma 17). The values e is statistically independent of $([a], [b], [c])$, as b' acts as blinding. If no cheating occurred the value d is always zero. If Step 5 is invoked, the reconstructed values are not used, and privacy is met. *Communication:* Follows from counting the number of messages sent (c.f. Lemmas 13, 16 and 17). □

Protocol RandomTriple($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, M)

0: The players take the set $M \subseteq \mathcal{P}$ as input.
1: The players generate random shared values $[a], [b], [b'], [r]$ by summing up shared random values (one from each player) for each value.
2: Invoke BasicMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], M)$ to compute the sharing $[c]$ and the vector $([c_1], \ldots, [c_n])$ and invoke BasicMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b'], M)$ to compute the sharing $[c']$ and the vector $([c'_1], \ldots, [c'_n])$.
3: Reconstruct $[r]$ and (locally) compute $[e] = r[b] + [b']$ and reconstruct it to obtain e. Then $[d] = e[a] - r[c] - [c']$ is computed (locally) and reconstructed.
4: If the value d is zero the players output $([a], [b], [c])$.
5: Otherwise reconstruct the sharings $[a], [b], [b'], [c_1], \ldots, [c_n], [c'_1], \ldots, [c'_n]$. The players output $M' = M \cup \{P_i : rc_i + c'_i \neq \sum_{(p,q)\in I(i)} r(a_p b_q) + (a_p b'_q)\}$.

Multiplication with Circuit Randomization The actual multiplication is based on circuit randomization [Bea91b]. It allows players to compute the product $[xy]$ of two shared values $[x]$ and $[y]$ at the cost of two reconstructions given a random multiplication triple $([a], [b], [c])$, where $ab = c$. The trick is to use that $xy = ((x - a) + a)((y - b) + b)$. By reconstructing $d = x - a$ and $e = y - b$ the players can compute $[xy]$ as $de + d[b] + [a]e + [c]$. This does not violate the secrecy of $[x]$ or $[y]$ as the random values $[a]$ and $[b]$ act as blinding.

Protocol Multiplication($\mathcal{P}$, $\mathcal{Z}$, $\mathbb{S}$, $[x]$, $[y]$)

0: The players in $\mathcal{P}$ take collectively $[x], [y]$ as input and set $M := \emptyset$.
1: Invoke RandomTriple$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, M)$. If the protocol outputs a set M', set $M \leftarrow M'$ and repeat Step 1. Otherwise use the output as random multiplication triple $([a], [b], [c])$.
2: Compute and reconstruct $[d_x] = [x] - [a]$ and $[d_y] = [y] - [b]$. Compute $d_x d_y + d_x[b] + d_y[a] + [c] = [xy]$ to obtain a sharing of xy.

Lemma 19. $\mathsf{Multiplication}(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [x], [y])$ *is an unconditional secure multiplication protocol given that* $\mathbb{S}$ *and* $\mathcal{Z}$ *satisfy* $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$*. The protocol has an error probability of* $\mathcal{O}(\frac{1}{|\mathbb{F}|}\,|\mathbb{S}|\,n^5)$*, communicates at most* $\mathcal{O}(|\mathbb{S}|\,n^5 \log|\mathbb{F}|)$ *bits and broadcasts at most* $\mathcal{O}(|\mathbb{S}|\,n^5 \log|\mathbb{F}|)$ *bits.*

Proof. Correctness: Assume that RandomTriple in Step 1 outputs a set M', then we have that $M \subsetneq M' \subseteq \mathcal{P}$. Hence this step is repeated less then n times and results in a random multiplication triple $([a], [b], [c])$ (c.f. Lemma 18). The rest of the protocol is just the multiplication from [Bea91b]. *Privacy:* Follows from [Bea91b] and Lemma 18. *Communication:* Follows from counting the number of messages sent (c.f. Lemmas 16 and 18). □

4.4 Unconditional MPC Protocol

The combination of Share, Reconstruct and Multiplication directly gives the following unconditionally secure MPC protocol.

Protocol $\mathsf{MPC}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$

0: The players take $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ as sharing specification.

1: For every gate of $\mathcal{C}$ being evaluated do the following:

- *Input gate for P_D:* $\mathsf{Share}(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ is invoked to share s
- *Linear gate:* The linear combination of the corresponding shares is computed locally using the linearity of the sharing.
- *Random gate:* Each player shares a random value. The sum of these values is used as output of the gate.
- *Multiplication gate:* $\mathsf{Multiplication}(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [x], [y])$ is used to multiply $[x]$ and $[y]$.
- *Output gate:* The players invoke $\mathsf{Reconstruct}(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$ to reconstruct s for players in R.

Theorem 2. *Let* $\mathcal{C}$ *be a circuit over* $\mathbb{F}$*, where* $|F| \in \Omega(2^\kappa)$ *and* κ *is a security parameter, and let* $\mathcal{Z}$ *be an adversary structure satisfying* $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$*, then* $\mathsf{MPC}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ *$\mathcal{Z}$-securely evaluates $\mathcal{C}$ with an error probability of* $2^{-\kappa}\,|\mathcal{C}|\,|\mathcal{Z}|\cdot\mathrm{Poly}(n, \kappa)$*. It communicates* $|\mathcal{C}|\,|\mathcal{Z}|\cdot\mathrm{Poly}(n, \kappa)$ *bits and broadcasts* $|\mathcal{C}|\,|\mathcal{Z}|\cdot\mathrm{Poly}(n, \kappa)$ *bits within* $\mathrm{Poly}(n, \kappa)\cdot d$ *rounds, where* d *denotes the multiplicative depth of* C*.*

Proof. It is easy to see that $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ is a sharing specification satisfying $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$. Hence by the properties of the sharing scheme and Lemma 19 correctness and the bound on the error probability follow. The claimed communication and broadcast complexity follow directly from the used subprotocols. Inspection of the subprotocols also shows that it is possible to evaluate gates on the same multiplicative depth of C in parallel. As each subprotocol only requires $\mathrm{Poly}(n, \kappa)$ rounds, the total number of rounds follows. □

Note that broadcast can be (unconditionally secure) simulated using the protocol from [PW96], which communicates $\mathrm{Poly}(n, \kappa)$ bits in order to broadcast one bit (with error probability of $\mathcal{O}(2^{-\kappa})$). This results in an MPC protocol with the same efficiency and error probability as stated in Theorem 2.
The error probability of the presented protocol grows linearly in the size of the adversary structure $\mathcal{Z}$. As $|\mathcal{Z}|$ is typically exponential in n, the security parameter κ must be chosen accordingly (such that $|\mathcal{Z}| \in \mathrm{Poly}(\kappa)$). This results in a huge security parameter and therefore in inefficient protocols. We therefore provide an extension of the previous protocol in which the error probability only depends on $\log |Z|$, and hence a reasonably large security parameter κ is sufficient.

5 Unconditional Protocol for Superpolynomial $|\mathcal{Z}|$

The protocol from the previous section has an error probability linear in $|\mathcal{Z}|$, which is problematic for large adversary structures $\mathcal{Z}$. In this section, we present modifications to the protocol that reduce the dependency to $\log |\mathcal{Z}|$, which is in $\mathrm{Poly}(n)$.
The reason for the error probability being dependent on $|\mathcal{Z}|$ is twofold: Firstly, the protocol requires $\Omega(|\mathcal{Z}|)$ probabilistic checks, in each of them a cheating party might remain undetected with probability $2^{-\kappa}$. Secondly, the protocol requires $\Omega(|\mathcal{Z}|)$ broadcasts, each of them having a small probability of failure.

5.1 Information Checking

In each invocation of Authenticate / Verify, a cheating attempt of a malicious player P_i is not detected with probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|})$ (c.f. Section 4.1). As these protocols are invoked $\Theta(|\mathcal{Z}|)$ times per sharing, the resulting error probability depends linearly on $|\mathcal{Z}|$. To avoid this we use local dispute control to deal with detected cheaters.
More formally, each player P_k locally maintains a list $\mathcal{L}_k$ of players whom he distrusts. At the beginning of the MPC protocol these lists are empty. Protocol Authenticate is modified, such that P_j puts P_i on his list $\mathcal{L}_j$ if the check in Step 4 fails. Once $P_i \in \mathcal{L}_j$, P_j behaves in all future invocations of the protocol as if the check in Step 4 failed independently whether this is the case or not. Similarly P_k puts P_i on his list $\mathcal{L}_k$ if the check in Step 5 fails. As soon as $P_i \in \mathcal{L}_k$, P_k behaves in Step 5 as if the corresponding check failed. Furthermore, in protocol Verify, P_k puts P_j on his list $\mathcal{L}_k$ if the check in Step 2 failed. Again P_k behaves for all $P_j \in \mathcal{L}_k$ as if the check failed independently whether this is the case or not.
In both protocols the adversary has a chance of $\mathcal{O}(\frac{1}{|\mathbb{F}|})$ to cheat successfully, but if he fails (with probability $\Omega(1 - \frac{1}{|\mathbb{F}|})$) one corrupted player P_i is put on the list $\mathcal{L}_k$ of an honest player P_k. From then on P_i is never able to cheat in instances of both protocols when P_k takes part (in the right position). This means that the adversary actually has at most n^2 attempts to cheat. Hence total error probability of arbitrary many instances of Verify and Authenticate is at most $\mathcal{O}(\frac{1}{|\mathbb{F}|}n^2)$ and no longer depends on $\mathcal{Z}$.
Note that the parallel invocation of Authenticate, as it is used in Share, requires special care. For example if in one of the parallel invocations of Authenticate (with P_i and

P_k) the consistency check fails P_k must assume that all other parallel checks failed. Analogous modifications are made in Verify and Multiplication.

Lemma 20. *The modified* Authenticate *and* Verify *protocols have a total error probability of* $\mathcal{O}(\frac{1}{|\mathbb{F}|}n^2)$ *independent of the number of invocations.*

5.2 Broadcast

Although broadcast is only needed in Share, the total number of broadcast calls is in $\Theta(|\mathcal{Z}|)$. If [PW96] is used, the resulting overall error probability depends linearly on $|\mathcal{Z}|$. To avoid this problem, the number of broadcast calls must be reduced.
To reach this goal we use the fact that the Share protocol only has constantly many rounds. In each round a player P_S must broadcast $\Theta(|\mathcal{Z}|)$ many messages of size $\mathcal{O}(\log|\mathbb{F}|)$. Instead of broadcasting these messages in parallel, P_S sends their concatenation to the other players, who then check that they received the same message. If an inconsistency is detected the protocol is repeated. To limit the number of repetitions we use the concept of dispute control from [BH06] which prevents the malicious players from repetitive cheating. Dispute control is realized by a publicly known *dispute set* $\Gamma \subseteq \mathcal{P} \times \mathcal{P}$, a set of unordered pairs of players. If $\{P_i, P_j\} \in \Gamma$ it means that there is a dispute between P_i and P_j and thus at least one of them is corrupted. Note that from P_i's view all player in $\{P_j | \{P_i, P_j\} \in \Gamma\}$ are malicious and thus he no longer trust them. At the beginning of the MPC protocol Γ is empty.

Protocol OptimisticBroadcast($\mathcal{P}, \mathcal{Z}, P_S, m$)

0: The player P_S takes $m \in \{0,1\}^w$ as input.
1: $\forall \{P_i, P_S\} \notin \Gamma$ P_S sends m as m_i to P_i.
2: $\forall \{P_i, P_j\} \notin \Gamma$ P_i sends m_i as m_{ij} to P_j.
3: $\forall P_i$ if all received values are the same P_i is happy, otherwise unhappy. P_i broadcasts using [PW96] his happy bit.
4: If all players are happy, each P_i outputs the value he holds.
Otherwise, an unhappy player P_i (e.g. the one with the smallest index) broadcasts j, j', z, b where m_{ji} differs from $m_{j'i}$ at bit-position z and b is the bit of m_{ji} at position z. Then $P_S, P_j, P_{j'}$ broadcast their versions of the bit at position z. Using this information the players localize a dispute between two players of $\{P_i, P_S, P_j, P_{j'}\}$. Then the protocol is repeated with updated Γ.

Lemma 21. *The protocol* OptimisticBroadcast$(\mathcal{P}, \mathcal{Z}, P_S, m)$ *achieves the broadcast of a message* $m' \in \{0,1\}^w$. *The protocol communicates at most* $w \cdot \mathrm{Poly}(n, \kappa)$ *bits and broadcasts at most* $\log w \cdot \mathrm{Poly}(n, \kappa)$.

Proof. The properties of Γ guarantee that honest players will exchange in Step 2 their received messages from P_S. So if all honest player are happy they all will output the same message m'. For an honest P_S this also ensures that $m' = m$. If a player is unhappy, at least one player misbehaved. The actions taken in Step 4 then ensure that

the honest players will find at least one dispute. The protocol will terminate, as the number of repetition is limited by $n(n-1)$. As the broadcast of z requires $\log w$ bits, the communication and broadcast complexities follow. □

For a message of length $\Theta(|\mathcal{Z}|)$ the above protocol only needs to broadcast $\log|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$ bits, hence the total number of broadcast calls per invocation of Share is reduced to $\log|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$.

Lemma 22. *The modified* Share *protocol communicates* $|\mathcal{C}|\,|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$ *bits and broadcasts* $|\mathcal{C}|\log|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$ *bits.*

5.3 Summary

The combination of the above extension results in the following Lemma:

Lemma 23. *Let* $\mathcal{C}$ *be a circuit over* $\mathbb{F}$*, where* $|\mathbb{F}| \in \Omega(2^\kappa)$ *and* κ *is a security parameter, and let* $\mathcal{Z}$ *be an adversary structure satisfying* $\mathcal{Q}^2(\mathcal{P},\mathcal{Z})$*, then the modified protocol* $\mathsf{MPC}(\mathcal{P},\mathcal{Z},\mathcal{C})$ $\mathcal{Z}$*-securely evaluates* $\mathcal{C}$ *with an error probability of* $2^{-\kappa}\,|\mathcal{C}| \cdot \mathrm{Poly}(n,\kappa)$*. It communicates* $|\mathcal{C}|\,|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$ *bits and broadcasts* $|\mathcal{C}|\log|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$ *bits. The number of rounds is* $\mathrm{Poly}(n,\kappa) \cdot d$*, where* d *denotes the multiplicative depth of* C*.*

Proof. Follows directly from Theorem 2 and Lemmas 20 and 22. □

By replacing broadcast with the simulated one from [PW96], one gets for $|\mathcal{Z}| \in \mathcal{O}(2^n)$ and $|\mathcal{C}| \in \mathrm{Poly}(\kappa)$ the following theorem.

Theorem 3. *Let* $\mathcal{C}$ *be a circuit over* $\mathbb{F}$*, where* $|\mathbb{F}| \in \Omega(2^\kappa)$ *and* κ *is a security parameter, and let* $\mathcal{Z}$ *be an adversary structure satisfying* $\mathcal{Q}^2(\mathcal{P},\mathcal{Z})$*, then* $\mathsf{MPC}(\mathcal{P},\mathcal{Z},\mathcal{C})$ $\mathcal{Z}$*-securely evaluates* $\mathcal{C}$ *with an error probability of* $2^{-\kappa} \cdot \mathrm{Poly}(n,\kappa)$*. It communicates* $|\mathcal{Z}| \cdot \mathrm{Poly}(n,\kappa)$ *bits.*

6 Lower Bound on the Efficiency

The following theorem states that there exists a family of circuits and $\mathcal{Q}^2$ adversary structures such that the length of unconditionally secure protocols tolerating these adversaries grows exponentially in the number of players. This implies that the computational complexity of our protocol from the previous section is optimal, as there exists no protocol with a computational complexity in $\mathrm{o}(|\mathcal{Z}|)$.

Theorem 4. *[Hir01] Let* $\mathcal{C}$ *be the circuit which takes inputs from* P_1 *and* P_2 *and outputs the product to* P_1*. Then there exists a family* $\mathcal{Z}_2,\mathcal{Z}_3,\ldots$ *of* $\mathcal{Q}^2$ *adversary structures for player sets* $\mathcal{P}_2,\mathcal{P}_3,\ldots$ *(*$|\mathcal{P}_n| = n$*) such that the length of the shortest unconditionally* $\mathcal{Z}_n$*-secure protocol for* $\mathcal{C}$ *grows exponentially in* n*.*

References

[Bea91a] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[Bea91b] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432. Springer-Verlag, 1991.

[BFH+08] Zuzana Beerliova-Trubiniova, Matthias Fitzi, Martin Hirt, Ueli Maurer, and Vassilis Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In *TCC*, volume 4948 of *LNCS*, pages 231–250. Springer-Verlag, March 2008.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10. ACM, 1988.

[BH06] Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In *TCC*, volume 3876 of *LNCS*, pages 305–328. Springer-Verlag, March 2006.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *STOC*, pages 11–19. ACM, 1988.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary, 1999.

[FM98] Matthias Fitzi and Ueli Maurer. Efficient Byzantine agreement secure against general adversaries. In *DISC*, volume 1499 of *LNCS*, pages 134–148. Springer-Verlag, September 1998.

[GMW87] S. Goldwasser, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with an honest majority. In *STOC*, volume 87, pages 218–229, 1987.

[Hir01] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, September 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[HM97] Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *PODC*, pages 25–34, August 1997.

[HM00] Martin Hirt and Ueli Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, April 2000. Extended abstract in *Proc. 16th of ACM PODC '97*.

[HMZ08] Martin Hirt, Ueli Maurer, and Vassilis Zikas. MPC vs. SFE: Unconditional and computational security. In *ASIACRYPT*, volume 5350 of *LNCS*, pages 1–18. Springer-Verlag, December 2008.

[Mau02] Ueli Maurer. Secure multi-party computation made simple. In *SCN*, volume 2576 of *LNCS*, pages 14–28. Springer-Verlag, September 2002.

[PSR03] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In *SCN*, pages 342–353. Springer-Verlag, 2003.

[PW96] Birgit Pfitzmann and Michael Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. In *Research report*. IBM Research, 1996.

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

[Yao82] A.C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.