On Fully-Secure Honest Majority MPC without n^2 Round Overhead

Daniel Escudero[1] and Serge Fehr[2]

[1] J.P. Morgan AI Research & J.P. Morgan AlgoCRYPT CoE, New York, U.S.A.
[2] CWI Amsterdam, The Netherlands

Abstract. Fully secure multiparty computation (or guaranteed output delivery) among n parties can be achieved with perfect security if the number of corruptions t is less than $n/3$, or with statistical security with the help of a broadcast channel if $t < n/2$. In the case of $t < n/3$, it is known that it is possible to achieve linear communication complexity, but at a cost of having a round count of $\Omega(\mathsf{depth}(C) + n)$ in the worst case. The number of rounds can be reduced to $O(\mathsf{depth}(C))$ by either increasing communication, or assuming some correlated randomness (a setting also known as the preprocesing model). For $t < n/2$ it is also known that linear communication complexity is achievable, but at the cost of $\Omega(\mathsf{depth}(C) + n^2)$ rounds, due to the use of a technique called dispute control. However, in contrast to the $t < n/3$ setting, it is not known how to reduce this round count for $t < n/2$ to $O(\mathsf{depth}(C))$, neither allowing for larger communication, or by using correlated randomness.

In this work we make progress in this direction by taking the second route above: we present a fully secure protocol for $t < n/2$ in the preprocessing model, that achieves linear communication complexity, and whose round complexity is only $O(\mathsf{depth}(C))$, without the additive n^2 term that appears from the use of dispute control. While on the $t < n/3$ such result requires circuits of width $\Omega(n)$, in our case circuits must be of width $\Omega(n^2)$, leaving it as an interesting future problem to reduce this gap. Our $O(\mathsf{depth}(C))$ round count is achieved by avoiding the use of dispute control entirely, relying on a different tool for guaranteeing output. In the $t < n/3$ setting when correlated randomness is available, this is done by using error correction to reconstruct secret-shared values, but in the $t < n/2$ case the equivalent is robust secret-sharing, which guarantees the reconstruction of a secret in spite of errors. However, we note that a direct use of such tool would lead to *quadratic* communication, stemming from the fact that each party needs to authenticate their share towards each other party. At the crux of our techniques lies a novel method for reconstructing a batch of robustly secret-shared values while involving only a linear amount of communication per secret, which may also be of independent interest.

1 Introduction

Secure multiparty computation (MPC) is a set of techniques that enable a set of parties $P_1, \ldots, P_n$, each P_i having a private input x_i, to securely compute

a function $z = f(x_1, \ldots, x_n)$ while only involving communication among each other, in such a way that they learn the output z, and not even an *adversary* corrupting a subset of t parties can learn any information about the inputs x_i from non-corrupted/honest parties, besides from what is inherently leaked by the result z.

Multiple MPC protocols exist depending on different factors like the power of the adversary, or the level of security required. A first important distinction is whether the adversary is *passive*, meaning that the behavior of corrupt parties is not affected (*i.e.* they follow the protocol specification), or whether he is *active*, meaning that the corrupt parties can deviate arbitrarily from the protocol execution. In this work we only consider active security, so from now on we assume the adversary under consideration is malicious, or active. Also, security can be computational, meaning that it is based on the hardness of some computational problem, or it can be information-theoretic, which means that security holds even against computationally unbounded adversaries. In the information-theoretic case there is a further distinction between perfect and statistical security, where the latter allows for a negligible error probability while the former achieves zero error probability.

Different techniques are used—and different results can be obtained—conditioning on the amount of parties t that the adversary is assumed to corrupt. If $t < n/3$ then we can achieve very efficient MPC with perfect security and with guaranteed output delivery (also abbreviated G.O.D., which means that the honest parties are guaranteed to receive the correct output z in spite of arbitrary adversarial behavior) [BTH08], and if $t < n/2$ then we can achieve the same but with statistical security, assuming a broadcast channel, which can be shown to be necessary [GSZ20]. Finally, if we do not assume any bound on t, then we must rely on computational assumptions, and the G.O.D. property (or even fairness, where the honest parties get the output *if* the adversary gets it) cannot be obtained in general. The focus of this work is the setting $t < n/2$, also known as *honest majority*. Furthermore, we assume the maximal case $n = 2t + 1$. From now on, whenever we refer to the setting $t < n/2$, we mean statistical security with an assumed broadcast channel, while for $t < n/3$ we mean perfect security without broadcast. In both cases we require full security, or G.O.D.

Several works have studied MPC in the honest majority setting. As we have mentioned, it is known that in this case G.O.D. with statistical security is attainable, with the help of a broadcast channel. Furthermore, this can be done while maintaining a total communication complexity of $O(n|C|)$, where $|C|$ is the size of the circuit to be computed (measured in the number of multiplication gates) [GSZ20]. This means that the total communication scales linearly with the number of parties, or put differently, the communication per party (*i.e.* after dividing by n) is constant in n. This is crucial for scalability, since this ensures each party does not communicate more as more parties join. Unfortunately, there is a major drawback of these works: the round complexity is given by $\Omega(\mathsf{depth}(C) + n^2)$. In contrast, weaker notions such as security with abort in the honest majority setting allow for a round complexity of $O(\mathsf{depth}(C))$, without

any dependency in n. This is good for cases where the underlying communication has high latency, and the number of parties is large enough so that n^2 extra rounds become too much of an overhead.

Dispute control. To understand where this additive term of n^2 originates from, it is worth discussing at a high level the core ideas involved in existing G.O.D. constructions like [GSZ20,BSFO12,BTH06]. In short, these works operate by letting the parties compute the circuit in a gate-by-gate fashion, where the parties maintain the invariant that for each wire they hold secret-shares of the underlying value. Multiplication gates require interaction, and a malicious party may cause an error by misbehaving. To ensure that the computation can recover from this failure, a technique called dispute control, introduced in [BTH06], is used. With this method, upon detecting a failure, the parties execute a "localization" protocol whose purpose is to identify a pair of parties $\{P_i, P_j\}$ where at least one is guaranteed to be corrupt (we say that P_i and P_j are *dispute* with each other). At this point, the computation is attempted once again, but this time some machinery is put on top so that it is ensured that either (1) the computation succeeds, or (2) a *new* dispute is identified, that is, the same pair $\{P_i, P_j\}$ will not fall in a dispute again.

The techniques used for securely computing multiplication gates require linear communication complexity $O(n)$ per gate, with interaction at each layer of the circuit, so in particular if there are no failures, the total communication is $O(n|C|)$ distributed across $O(\mathsf{depth}(C))$ rounds. Unfortunately, an active adversary can cause all new attempts to fail, generating the maximal number of disputes, which is $\Omega(n^2)$. This would lead to a communication of $n^2 \cdot \Omega(n|C|) = \Omega(n^3|C|)$ and $n^2 \cdot \Omega(\mathsf{depth}(C))$ rounds, but fortunately this can be addressed by partitioning the circuit into segments and verifying the correctness of each of them, instead of checking the whole circuit. This way, at most n^2 segments are repeated, and by choosing segments of size $|C|/n^2$ we see that the overall communication remains $O(n|C|)$. In terms of round count, letting d be the depth of each segment, the number of rounds is $O(\mathsf{depth}(C))$ from the optimistic case, plus $\Omega(n^2 d)$ from the possible n^2 segments that are repeated in the worse case. Hence, the round complexity in the worst case becomes $\Omega(\mathsf{depth}(C) + n^2)$. We note that this is particularly prohibitive when n is moderately large, with the overhead being more and more noticeable for not so large values of $\mathsf{depth}(C)$ (in particular when $\mathsf{depth}(C) = o(n^2)$).

On the $< n/3$ setting. We recall that honest majority protocols that achieve weaker notions such as security with abort achieve a round complexity of $O(\mathsf{depth}(C))$. Currently, it is not known whether G.O.D. protocols with $O(\mathsf{depth}(C))+o(n^2)$ rounds exist, which shows that, in the honest majority case, there is an efficiency gap between security with abort and guaranteed output delivery. In contrast, the landscape turns out to be different for the $t < n/3$ case. There it is also possible to achieve linear communication complexity by making use of a tool called player elimination. This is similar to dispute control (and in fact,

player elimination predates dispute control), with the difference that when a pair $\{P_i, P_j\}$ with at least one corrupt party is identified, both of these parties can be removed from the computation, before a reattempt.[3] This way, a segment is repeated at most $O(n)$ times (since for every iteration there is one less corrupt party), so by taking segments of size $|C|/n$, linear communication is achieved with a round count of $\Omega(\mathsf{depth}(C) + n)$ in the worst case.

In stark contrast with the honest majority case, for $t < n/3$ it is actually possible to remove the overhead in n in the round complexity to obtain a protocol with $O(\mathsf{depth}(C))$ rounds, and this can be done in two different ways:

1. Pay more in terms of communication, leading to works such as [AAPP22,AAY21] that have superlinear communication.[4]
2. Maintain linear communication, but assume correlated randomness, or in other words, consider a protocol in the preprocessing model.

The second approach above is based on the observation that, in player-elimination-based protocols such as [BTH08], the adversary can only cause the repetition of a segment in a preprocessing phase in which some correlated randomness is established. Once this is completed, the online phase is guaranteed to provide output "in one go", without the need of repeating any segment (or splitting the remaining of the computation into segments, for that matter). We emphasize again that it is not known how to achieve $O(\mathsf{depth}(C))$ round count for the honest majority case, not even giving up on linear communication complexity, or even assuming correlated randomness. Hence, this motivates the following question:

> *Can we design fully secure honest majority MPC protocols with statistical security that require $O(\mathsf{depth}(C))$ rounds to securely compute a circuit C, either by having superlinear communication or assuming correlated randomness?*

1.1 Our Contribution

As previously mentioned, for the $t < n/3$ setting with perfect security, such result is only known to be achievable by either allowing for superlinear communication complexity, or having linear communication but assuming correlated randomness. In this work we take the second route in the question above, and we present a statistically and fully secure MPC protocol for honest majority, that has a round complexity $O(\mathsf{depth}(C))$ that is independent of n, and has *linear* communication

[3] This is not possible in the case of $n = 2t + 1$ since one out of the $t + 1$ honest parties may end up being removed, and the t remaining honest parties cannot "keep the state" of the computation (since otherwise the set of t corrupt parties should be able to also determine such state).

[4] In fact, the very recent work of [AAPP23] shows how to obtain *expected* $O(\mathsf{depth}(C))$ rounds while still achieving linear communication complexity, for certain class of circuits. We do not discuss this work further since our interest is on deterministic $O(\mathsf{depth}(C))$ rounds.

complexity. One drawback of our protocol is that the claimed linear communication holds for circuits of width $\Omega(n^2)$, while in the $t < n/3$ case this width requirement is a factor of n smaller. We leave it as interesting future work to further extend our techniques to tolerate circuits of width $\Omega(n)$, hence narrowing down the gap between results for $t < n/2$ and $t < n/3$.

In more detail, we present an MPC protocol to securely compute a circuit $|C|$ over a finite field $\mathbb{F}$, that has the following features:

- The protocol is set in the *preprocessing model* where the parties have access to some *correlated randomness*;
- The communication complexity is $O(n|C| + n^3\mathsf{depth}(C))$, which is linear if $\mathsf{depth}(C) = O(|C|/n^2)$, that is, each layer should contain $\Omega(n^2)$ multiplication gates.
- The round complexity is $O(\mathsf{depth}(C))$.

Our result is achieved by introducing novel ideas that remove the need of using dispute control techniques, in the preprocessing model. Qualitatively speaking, our approach offers several advantages with respect to protocols based on dispute control: the protocol proceeds in a gate-by-gate fashion (batching multiplication gates in the same layer), and handling each gate is guaranteed to be completed "in one go", without the need of checking its correctness or repeating its computation. Furthermore, our protocol is also arguably simpler than dispute-control-based protocols, as it does not need to localize and handle disputes, repeat segments, or incur in any of the related complexities of using dispute control. In fact, one important advantage of our protocol we only require a broadcast channel for the input phase, while the rest of the computation can happen over point to point channels; in contrast, dispute-control-based protocols like [BTH06] require a broadcast channel throughout the whole computation.

At the heart of our techniques lies a new and non-trivial technique for reconstructing a batch of n^2 values that are secret-shared in a robust manner, while involving a communication complexity of $O(n^3)$, which ultimately leads to a communication of $O(n)$ per reconstructed value. This may find several other applications in contexts where robust secret-sharing is used, like secure distributed storage.

1.2 Related Work

We already discussed some related work above, but here we present a more thorough discussion on relevant literature. In [HMP00], which is set in the $t < n/3$ setting, the idea of splitting the circuit into segments, together with the tool of player elimination, were introduced. This work makes use of verifiable secret-sharing and re-sharing *à la* [BOGW88,GRR98] in order to securely compute a given arithmetic circuit. This was later improved in [BTH08] by making use of multiplication triples, which, on top of achieving linear communication complexity, pushes the use of player elimination to the offline phase thanks to the use of error-correction techniques for the online phase. Any work that uses player

elimination introduces an additive overhead of $\Omega(n)$ in terms of the number of rounds, stemming from the need of re-running each segment every time a new semi-corrupt pair (a pair of parties where at least one of them is guaranteed to be corrupt) is identified. If one wishes to avoid this overhead, works such as [AAPP22,AAY21] achieve a round count of $O(\mathsf{depth}(C))$, albeit with superlinear communication complexity. Substantial progress towards improving this trade-off was made in [AAPP23], where a perfectly secure protocol for $t < n/3$ with linear communication complexity (for certain class of circuits) and *expected* $O(\mathsf{depth}(C))$ rounds was given.

In [BTH06] the dispute control framework, which is inspired by the player elimination framework, is introduced for the setting of $t < n/2$. This time, instead of removing an identified semi-corrupt pair, a mechanism is set in place so that these parties can keep participating in the protocol, and subsequent cheating leads to a *new* semi-corrupt pair being identified. To enable the identification of semi-corrupt pairs whenever cheating takes place, the secret-sharing structure must be enhanced by letting the parties hold shares of their shares, along with extra additional information in the form of tags under a message authentication code (MAC). Similarly to the player elimination technique, the use of dispute control introduces an additive overhead in terms of the number of rounds, this time of n^2, which corresponds to the maximum number of disputes that can occur.

The protocol in [BTH06] achieves a communication complexity of $O(|C|n^2\kappa)$ (ignoring non-dominant terms and broadcast calls), where κ is the statistical security parameter. This was later improved in [BSFO12] to $O(|C|(n+\kappa))$.[5] Concrete constants (plus certain quadratic term that is affected by the depth of the circuit) are further improved in [GSZ20]. We remark, however, that these two works still make use of the core ideas introduced in [BTH06] regarding dispute control, splitting the circuit into segments and repeating each of these in case of cheating. As a result, they suffer from a n^2 overhead in terms of the number of rounds. This also holds for the work of [BGIN20], which is based on replicated secret-sharing.

Another important work is [IKP+16], where a transformation that takes a semi-honest honest majority protocol together with a fully-secure protocol with a potentially smaller threshold, and produces a fully-secure honest majority protocol, is presented. This achieved by adapting the IPS compiler [IKP+16] from the computational to the information-theoretic setting. However, their approach once again makes use of similar ideas as player elimination and dispute control, and hence they incur in an additive overhead of $\mathsf{poly}(n)$ in terms of the number of rounds.[6]

[5] Here we count the number of field elements, although the only constructions known in the literature require a field whose size grows linearly with the number of parties.

[6] Interestingly, here the overhead if n instead of n^2, since the authors do not use dispute control but instead a technique that is closer to player elimination.

1.3 Overview of our Techniques

For aiding in terms of readability, we provide a high level overview of our techniques.

A protocol for $t < n/3$ in the preprocessing model. In order to motivate our protocol, we begin by presenting a G.O.D. protocol (with perfect security) in the $t < n/3$ setting, that achieves linear communication complexity and only requires $O(\mathsf{depth}(C))$ number of rounds. This is a simplified version of the protocol from [BTH08]. First, we introduce some notation: we consider a finite field $\mathbb{F}$, and use $[\![x]\!]_d$ to denote Shamir secret-sharing with degree d. At a high level, the protocol proceeds as the majority of secret-sharing-based protocols: the parties start with degree-t sharings of the inputs of the computation, and then they proceed in a gate-by-gate fashion by computing shares of the result of each gate, until shares of the output are obtained, which can then be reconstructed. Addition gates are processed locally with the help of the linearity properties of Shamir secret-sharing. For multiplication gates, the parties make use of the preprocessing model in the following way: given $[\![x]\!]_t$ and $[\![y]\!]_t$ to be multiplied, and given correlated randomness of the form $([\![a]\!]_t, [\![b]\!]_t, [\![a \cdot b]\!]_t)$, where $a, b \in \mathbb{F}$ are uniformly random, (1) the parties compute locally $[\![d]\!]_t = [\![x]\!]_t - [\![a]\!]_t$ and $[\![e]\!]_t = [\![y]\!]_t - [\![b]\!]_t$, (2) they reconstruct these values to obtain d and e, and (3) the parties compute locally $[\![xy]\!]_t = e[\![a]\!]_t + d[\![b]\!]_t + [\![ab]\!]_t + de$.

The protocol is private given that a (and b) entirely hides x (and y) when reconstructing d (and e). Communication-wise, the complexity depends on how d and e are reconstructed. One way of doing this is letting each party announce their share of $[\![d]\!]_t$ (and $[\![e]\!]_t$) to each other party, who then reconstruct d (and e) from the received shares. A crucial property of Shamir secret-sharing with degree t is that, if $t < n/3$, then the t potentially incorrect shares provided by the corrupt parties can be error-corrected, and the receiving parties are guaranteed to be able to reconstruct the right underlying secret. In particular, every gate is guaranteed to succeed, so no repetitions are required and hence the number of rounds is $O(\mathsf{depth}(C))$.

Getting linear communication complexity. The approach sketched above, however, suffers from one major issue: since each party sends a share to every other party, communication is $\Omega(n^2)$. We are interested in linear communication, so a different approach is required. A common idea to achieve linearity when reconstructing secret-shared values is to use a "relay": all parties send their shares to a chosen party, say P_1, who reconstructs and forwards the result to the other parties. Unfortunately nothing prevents a corrupt P_1 from forwarding an incorrect value of their choice to the other parties. To handle this, a clever solution is given in [DN07], which consists of using multiple relays in such a way that, intuitively, there is always at least one honest relay who is guaranteed to reconstruct the right secret. However, special care is needed to really reduce communication to linear.

To do this reconstruction efficiently—while also guaranteeing successful reconstruction—the trick is to batch a collection of $t+1$ values to be reconstructed $[\![s_0]\!]_t, \ldots, [\![s_t]\!]_t$, define the polynomial $f(\mathsf{Z}) = \sum_{\ell=0}^{t} s_\ell \cdot \mathsf{Z}^\ell$, compute (locally) $[\![f(j)]\!]_t$ for $j \in [n]$, and reconstruct each $f(j)$ towards each party P_j, who then relays this reconstructed value to the other parties. The main observation is that the parties can again apply error correction to $(f(1), \ldots, f(n))$ in order to recover the polynomial $f(\mathsf{Z}) = \sum_{\ell=0}^{t} s_\ell \cdot \mathsf{Z}^\ell$, from which they can recover the secrets $s_0, s_1, \ldots, s_t$. Notice that communication is still quadratic in n, but crucially, $t+1$ values have been reconstructed. Hence, if $t+1 = \Theta(n)$ (*e.g.* if $n = 3t+1$), then this is $O(n^2/(t+1)) = O(n)$ communication per secret, as desired. One can interpret $(f(1), \ldots, f(n))$ as "Shamir sharings", except that we are not interested in the zero point $f(0)$, but rather in the polynomial $f(\mathsf{Z})$ itself.

A first $\Omega(n^2|C|)$ protocol for honest majority. With the ideas for G.O.D. with $t < n/3$ in the preprocessing model, we are ready to tackle the $t < n/2$ case. We begin by presenting a construction that achieves quadratic communication, and then discuss how we improve this initial protocol to achieve linear communication complexity. First, we follow a similar template as the protocol above: the parties start with Shamir sharings $[\![\cdot]\!]_t$ of each input, proceed in a gate-by-gate manner, handling addition gates locally and multiplication gates using triples $([\![a]\!]_t, [\![b]\!]_t, [\![ab]\!]_t)$, and finally reconstruct the output. Recall that, with this paradigm, a major bottleneck is the reconstruction of a secret-shared value $[\![s]\!]_t$, which in the $t < n/3$ setting is done while exploiting the error-correcting properties of Shamir secret-sharing. Unfortunately, for $t < n/2$, degree-t Shamir sharings do not satisfy error correction, and instead they only satisfy error detection, which ensures that either the correct secret is reconstructed by the receiver, or possibly no secret is reconstructed at all. At this point, one can obtain a protocol with abort, but it is not clear how to leverage this for G.O.D., besides making use of dispute control techniques to identify disputes and re-run some computation, which we want to avoid in order to only use $O(\mathsf{depth}(C))$ rounds.

Our first observation towards solving this issue is that, even though in the honest majority setting it is not possible to perform error correction, there is a related notion that provides a similar functionality, and this is *robust secret-sharing*. In a robust secret-sharing scheme there is a share algorithm that enables the distribution of a secret into multiple shares, and there is a rec(onstruction) algorithm that, on input n shares among which t of them are potentially maliciously modified, outputs the correct underlying secret, with overwhelming probability. There are multiple robust secret-sharing constructions in the literature (*cf.* [BPRW16,FY19]). Normally, these schemes operate by letting each party obtain Shamir shares of the secret s, together with some authentication tags that enable each share owner to prove to the intended receiver that the share they send is correct. To check correctness of the received shares and tags, each party also receives some keys. Let us abstract this notion and denote by $\langle x \rangle$ when a value x is robustly secret-shared. It is common that these constructions

satisfy some notion of linearity, meaning that the parties can locally compute $\langle x + y \rangle$ from $\langle x \rangle$ and $\langle y \rangle$. We will make use of this in what follows.

We can use $\langle \cdot \rangle$ as a building block instead of plain Shamir $[\![\cdot]\!]_t$ in the template above to obtain an honest majority G.O.D. protocol, where the preprocessing produces correlations of the form $(\langle a \rangle, \langle b \rangle, \langle ab \rangle)$. In its basic form, reconstruction would be done by letting the parties use the rec procedure towards every other party, which enables each party to successfully reconstruct each value needed for handling multiplication gates.[7] However, this basic protocol is insufficient for our goals here, simply due to the reason that the resulting communication complexity is $\Omega(n^2|C|)$. This stems from the fact that, whenever a shared value must be learned by all parties (which again, happens for every single multiplication gate), each single party must send their share and tags to each other single party.

Reducing to $O(n|C|)$ for $t < n/2$. Our core contribution consists of improving the approach from above, that achieves G.O.D. in the preprocessing model using quadratic communication via robust secret-sharing, to linear communication. $\Omega(n^2)$ communication stems from the need of reconstructing a robustly-shared value towards all parties. Recall that a similar issue was faced when we sketched the protocol for $t < n/3$, and in page 7 we discussed a solution to this problem originating in the ideas from [DN07]. Hence, our first approach is to try and adapt this idea from the secret-sharing scheme used there, namely plain Shamir secret-sharing $[\![\cdot]\!]$, to the robust scheme $\langle \cdot \rangle$ used here. Such adaptation would look like this: to reconstruct $t+1$ robustly-shared values $\langle s_0 \rangle, \ldots, \langle s_t \rangle$, (1) the parties define the polynomial $f(\mathtt{Z}) = \sum_{\ell=0}^{t} s_\ell \cdot \mathtt{Z}^\ell$, (2) they compute $\langle f(j) \rangle$ for $j \in [n]$ (which can be done using the homomorphic properties of $\langle \cdot \rangle$), (3) they reconstruct each $f(j)$ towards each party P_j, who successfully reconstructs this $f(j)$ using the robustness properties of $\langle \cdot \rangle$. At this point, P_j is supposed to send $f(j)$ to the other parties, who then recover $f(\mathtt{Z})$ from these received values. However, here we face a fundamental issue: while this works in the $t < n/3$ case due to error correction, in our $t < n/2$ case the polynomial $f(\mathtt{Z})$ cannot necessarily be recovered from the values $(f(1), \ldots, f(n))$! This is precisely why robust secret-sharing was introduced: in order to guarantee reconstruction.

Our idea is to ensure that the parties are able to get not only "Shamir shares of $f(\mathtt{X})$", as above, but actual "robust shares of $f(\mathtt{X})$". In a bit more detail, our approach is to ensure the parties can obtain authentication information on the "Shamir shares" $(f(1), \ldots, f(n))$, which guarantees that each receiver can successfully filter out incorrect $f(i)$'s, and hence reconstruct the polynomial $f(\mathtt{Z})$, and with it the secrets $s_0, \ldots, s_t$. An important problem here, however, is that we would need to devise a way to enable each P_j to obtain tags on $f(j)$, one for each receiver P_k. We now discuss at an intuitive level how this is addressed in our work.

[7] We note that this is the approach taken in, for example, Bedoza [BDOZ11], which is set in the dishonest majority setting $t < n$.

Providing P_j with authentication information. Recall that P_j holds $f(j)$, which is a "Shamir share" of the polynomial $f(\mathsf{Z})$, and P_j sends $f(j)$ to every other recipient P_k. We are missing a method by which P_k can check the correctness of this $f(j)$. To this end, imagine that $(f(1), \ldots, f(n))$ were *robust* sharings of $f(\mathsf{Z})$, that is, in addition to the share $f(j)$, each party P_j also holds a set of authentication tags $(\tau_{1j}, \ldots, \tau_{nj})$ on $f(j)$. To reconstruct, P_j provides P_k with $f(j)$ and τ_{kj}, and P_k uses τ_k to verify the correctness of $f(j)$. Now, in our setting $f(j)$ is not a "sharing" per se, as it is not the result of a dealer distributing robust shares, or linear combinations of these. Instead, P_j obtained $f(j)$ via a robust reconstruction of $\langle f(j) \rangle$.

Our idea is to enhance the robust secret-sharing scheme to allow for "nested" reconstruction: we let the parties also hold shares of $(\langle \tau_{1j} \rangle, \ldots, \langle \tau_{nj} \rangle)$, which can be reconstructed towards P_j so that this party can obtain the tags $(\tau_{1j}, \ldots, \tau_{nj})$ on $f(j)$, and hence prove correctness of $f(j)$ to each other party P_k. The problem with this approach is that communication grows to n^3, given that each party must reconstruct n values towards each other party.

To alleviate this issue, consider a larger number of shares to be reconstructed, say n groups of $t+1$ sharings each (hence, there are $O(n^2)$ total shared values). If we first apply the idea above to each of the n groups, we obtain a communication of n^4, or $n^4/n^2 = n^2$ per reconstruction. However, here we make the crucial observation that, among these n^4 messages, the amount related to transmitting shares not related to authentication is n^3. Indeed, if authentication was not an issue, for each group, each P_j would receive one share from each other party P_i, and each P_j would send one share to each P_k, leading to n^2 elements, which is n^3 when the number of groups is factored in. As a result, the n^4 overhead is only coming from the transmission of authentication-related information. In the notation of the sketch above, this is originating from the reconstruction of the authentication tags $(\langle \tau_{1j} \rangle, \ldots, \langle \tau_{nj} \rangle)$ towards P_j (one for each group), whose sole purpose is to enable P_j to prove the correctness of $f(j)$ towards P_k.

To achieve linear communication complexity, we note that *all authentication information can be compressed across the n groups using random linear combinations*, or in other words, the parties can distribute the authentication data of the n groups at the same cost of one single group. This results in n^3 communication in total for the n^2 reconstructions, or $n^3/n^2 = n$ per reconstruction, as desired. One must be careful when developing this idea in detail. First, a corrupt party can easily overcome a check that uses random linear combinations if he/she knows the random coefficients before adding the errors. To address this, in contrast to vanilla robust secret-sharing where each party can send their Shamir share at the same time as the authentication information, we require each party to "commit" to their errors by first sending their Shamir shares before sampling the random coefficients, and only then they distribute the associated authentication data. However, this new approach introduces another complication, which is that the random linear combination used to convince each P_j of the reconstruction of $f(j)$ cannot be the same as the one P_j will use to convince each P_k of the correctness of $f(j)$. To this end, after P_j has sent $f(j)$ to each P_k, new random

coefficients are sampled, and the parties robustly reconstruct towards P_j the necessary authentication data (using these coefficients) to convince P_k of the correctness of $f(j)$.

This high level idea is materialized in detail in Section 3, where we show how to efficiently and robustly reconstruct secret-shared values. The robust secret-sharing scheme we use is introduced in Section 2.

1.4 Notation

We let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > \mathsf{poly}(n) \cdot 2^{\kappa}$, where κ is the statistical security parameter. We use $[k]$ to denote the set $\{1, \ldots, k\}$. $\mathbb{F}_{\leq d}[\mathtt{X}]$ denotes the vector space of polynomials over $\mathbb{F}$ of degree at most d, on the variable $\mathtt{X}$. For security definitions in MPC we refer the reader to standard references such as [CDN15].

2 Robust Secret-Sharing

The main tool we make use of in our work is that of robust secret-sharing, which enables the properties of error correction, a secret to be distributed into multiple shares. Concretely, we introduce the following construction.

Definition 1. *We define the sharing $\langle x \rangle$ for a secret $x \in \mathbb{F}$ to consist of*

- *a random* sharing polynomial $F_0(\mathtt{X}) \in \mathbb{F}_{\leq t}[\mathtt{X}]$ *subject* $F_0(0) = x$,
- *random* randomizer polynomials $F_1(\mathtt{X}), \ldots, F_t(\mathtt{X}) \in \mathbb{F}_{\leq t}[\mathtt{X}]$
- *random* key polynomials $A_0(\mathtt{Y}), \ldots, A_t(\mathtt{Y}) \in \mathbb{F}_{\leq t}[\mathtt{Y}]$, *and*
- *the* checking polynomial $C(\mathtt{X}, \mathtt{Y}) \in \mathbb{F}_{\leq t, \leq t}[\mathtt{X}, \mathtt{Y}]$ *given by*

$$C(\mathtt{X}, \mathtt{Y}) = F_0(\mathtt{X}) \cdot A_0(\mathtt{Y}) + F_1(\mathtt{X}) \cdot A_1(\mathtt{Y}) + \cdots + F_t(\mathtt{X}) \cdot A_t(\mathtt{Y}) \,. \tag{1}$$

Every party P_i is given $F_0(i), F_1(i), \ldots, F_t(i)$ and $A_0(i), A_1(i), \ldots, A_t(i)$ as well as the (coefficients of the) polynomial $C(\mathtt{X}, i)$.

With the definition above, we note that the *v*iew of party P_i is given by

$$\mathsf{view}_i(\langle x \rangle) = \big(A_0(i), A_1(i), \ldots, A_t(i), F_0(i), F_1(i), \ldots, F_t(i), C(\mathtt{X}, i)\big);$$

similarly, $\mathsf{view}_{\mathcal{A}}(\langle x \rangle)$ denotes the joint view of a set $\mathcal{A}$ of parties. For *multiple* secrets, their sharings are defined as above, but with *the same* key polynomials $A_0(\mathtt{Y}), \ldots, A_t(\mathtt{Y})$ yet random and independent sharing and randomizer polynomials. In other words, the key polynomials $A_0(\mathtt{Y}), \ldots, A_t(\mathtt{Y})$ are sampled uniformly at random once and for all, and the sharing and randomizer polynomials are sampled freshly for each x uniformly at random subject to the given constraint, *i.e.*, $F_0(0) = x$.

To have simpler notation and more concise expressions, we introduce the *polynomial vectors* $\boldsymbol{F}(\mathtt{X}) = \big(F_0(\mathtt{X}), \ldots, F_t(\mathtt{X})\big) \in \mathbb{F}_{\leq t}[\mathtt{X}]^{t+1}$ and $\boldsymbol{A}(\mathtt{X}) = \big(A_0(\mathtt{Y}), \ldots, A_t(\mathtt{Y})\big) \in \mathbb{F}_{\leq t}[\mathtt{Y}]^{t+1}$, which allows us to re-write (1) very compactly as

$$C(\mathtt{X}, \mathtt{Y}) = \boldsymbol{F}(\mathtt{X}) \cdot \boldsymbol{A}(\mathtt{Y}) \,. \tag{2}$$

A sharing $\langle x \rangle$ is then a random triple $\big(\boldsymbol{A}(\mathtt{Y}), \boldsymbol{F}(\mathtt{X}), C(\mathtt{X},\mathtt{Y})\big)$ subject to $F_0(0) = x$ and (2), and the view of P_i (and similar for a set of parties) becomes $\mathsf{view}_i(\langle x \rangle) = \big(\boldsymbol{A}(i), \boldsymbol{F}(i), C(\mathtt{X}, i)\big)$.

It follows immediately from (2) and the fact that $\boldsymbol{A}(\mathtt{Y})$ is reused for different sharings, that linear functions can be computed on shared values by obvious local computations. We will use the notation $\langle x + y \rangle \leftarrow \langle x \rangle + \langle y \rangle$ for local additions, and similarly for more general affine combinations.

Lemma 1 below ensures that a sharing $\langle x \rangle$ of a secret x leaks no information on x to any t parties, except with negligible probability.

Lemma 1. *For any set $\mathcal{A}$ of t (or fewer) parties, and for a random key polynomial vector $\boldsymbol{A}(\mathtt{Y})$, the following holds except with probability $1/|\mathbb{F}|$ over the choice of $\boldsymbol{A}(\mathtt{Y})$: The distribution of $\mathsf{view}_{\mathcal{A}}(\langle x \rangle)$ conditioned on $\boldsymbol{A}(\mathtt{Y})$ does not depend on the value of x.*

Proof. Without loss of generality, we may assume $\mathcal{A} = \{P_1, \ldots, P_t\}$. We consider an arbitrary but fixed choice of $\boldsymbol{A}(\mathtt{Y})$, and we show the claimed independence to hold unless the $(t \times t)$-matrix with entries $A_i(j)$ for $i, j \in [t]$ is singular, which happens with probability $1/|\mathbb{F}|$ for a random $\boldsymbol{A}(\mathtt{Y})$.

Let $K_0(\mathtt{X}) \in \mathbb{F}_{\leq t}[\mathtt{X}]$ be the (unique) polynomial with $K_0(0) = 1$ yet $K_0(1) = \cdots = K_0(t) = 0$. Also, let $K_1(\mathtt{X}), \ldots, K_t(\mathtt{X})$ be such that also here $K_\ell(1) = \cdots = K_\ell(t) = 0$ for $\ell \in [t]$, but now $K_1(0) \cdot A_1(j) + \cdots + K_t(0) \cdot A_t(j) = -A_0(j)$ for $j \in [t]$. This exists due to the assumption on $\boldsymbol{A}(\mathtt{Y})$. The above conditions ensure that $K_1(\mathtt{X}) \cdot A_1(j) + \cdots + K_t(\mathtt{X}) \cdot A_t(j) = -K_0(\mathtt{X}) \cdot A_0(j)$, and thus $\boldsymbol{K}(\mathtt{X}) \cdot \boldsymbol{A}(j) = 0$, for $j \in [t]$. Then, for any $\delta \in \mathbb{F}$, the pair consisting of $\boldsymbol{F}'(\mathtt{X}) := \boldsymbol{F}(\mathtt{X}) + \delta \cdot \boldsymbol{K}(\mathtt{X})$ and $C'(\mathtt{X}, \mathtt{Y}) := \boldsymbol{F}'(\mathtt{X}) \cdot \boldsymbol{A}(\mathtt{Y})$, together with $\boldsymbol{A}(\mathtt{Y})$, forms a sharing $\langle x' \rangle$ for the secret $x' = x + \delta$, for which the parties $P_1, \ldots, P_t$ have the same view; namely $\boldsymbol{F}'(i) := \boldsymbol{F}(i)$ and $C'(\mathtt{X}, i) = \big(\boldsymbol{F}(\mathtt{X}) + \delta \cdot \boldsymbol{K}(\mathtt{X})\big) \cdot \boldsymbol{A}(i) = \boldsymbol{F}(\mathtt{X}) \cdot \boldsymbol{A}(i) = C(\mathtt{X}, i)$ for all $i \in [t]$. Furthermore, the above mapping from $\big(\boldsymbol{F}(\mathtt{X}), C(\mathtt{X}, \mathtt{Y})\big)$ to $\big(\boldsymbol{F}'(\mathtt{X}), C'(\mathtt{X}, \mathtt{Y})\big)$ is bijective, which proves the claim of the statement. □

Recall that P_i's *share vector* $\boldsymbol{s}_i = \boldsymbol{F}(i)$ satisfies $\boldsymbol{s}_i \cdot \boldsymbol{A}(j) = C(i, j)$, and so any *incorrect* share vector $\boldsymbol{s}'_i \neq \boldsymbol{s}_i$ satisfies $\boldsymbol{s}'_i \cdot \boldsymbol{A}(j) = C(i, j)$ if and only if $(\boldsymbol{s}_i - \boldsymbol{s}'_i) \cdot \boldsymbol{A}(j) = 0$, which happens with probability $1/|\mathbb{F}|$ only when $\boldsymbol{A}(j)$ is random. Thus, Lemma 2 below implies that the set $\mathcal{A}$ of corrupt parties will find an incorrect share vector that will be accepted by honest P_j with probability $1/|\mathbb{F}|$ only, even if they get to see the entire sharing polynomial vector $\boldsymbol{F}(\mathtt{X})$. Hence, any honest P_j can filter out all incorrect share vectors $\boldsymbol{s}'_i \neq \boldsymbol{s}_i$, allowing him to reconstruct the polynomial vector $\boldsymbol{F}(\mathtt{X})$, and thus $\boldsymbol{F}(0)$ and the actual secret $x = F_0(0)$.

Lemma 2. *For any set $\mathcal{A}$ of t (or fewer) parties, for any $j \notin \mathcal{A}$, and for any $x \in \mathbb{F}$, the key vector $\boldsymbol{A}(j)$ is uniformly random and independent of the pair $\big(\mathsf{view}_{\mathcal{A}}(\langle x \rangle), \boldsymbol{F}(\mathtt{X})\big)$.*

Proof. Again, we may assume $\mathcal{A} = \{P_1, \ldots, P_t\}$. Let $K(\mathtt{Y}) \in \mathbb{F}_{\leq t}[\mathtt{Y}]$ be such that $K(1) = \cdots = K(t) = 0$ and $K(j) = 1$. Then, for any vector $\boldsymbol{\delta} \in \mathbb{F}^{t+1}$ the triple

consisting of $\boldsymbol{A}'(\mathtt{Y}) = \boldsymbol{A}(\mathtt{Y}) + K(\mathtt{Y}) \cdot \boldsymbol{\delta}$, $\boldsymbol{F}(\mathtt{X})$ and $C'(\mathtt{X}, \mathtt{Y}) = \boldsymbol{F}(\mathtt{X}) \cdot \boldsymbol{A}'(\mathtt{Y})$ forms a sharing $\langle x \rangle$ of x for which the parties $P_1, \ldots, P_t$ have the same view, but now with P_j having key vector $\boldsymbol{A}'(j) = \boldsymbol{A}(j) + \boldsymbol{\delta}$. Furthermore, the above mapping from $\big(\boldsymbol{A}(\mathtt{Y}), \boldsymbol{F}(\mathtt{X}), C(\mathtt{X}, \mathtt{Y})\big)$ to $\big(\boldsymbol{A}'(\mathtt{Y}), \boldsymbol{F}(\mathtt{X}), C'(\mathtt{X}, \mathtt{Y})\big)$ is bijective, which proves the claim of the statement. □

From the above, we see that a secret-shared value $\langle x \rangle$ can be reconstructed towards a given receiver P_j, in such a way that P_j is guaranteed to obtain the correct secret. More precisely, each party P_i sends their share vector $\boldsymbol{s}_i = \boldsymbol{F}(i)$ to the receiver P_j, who checks that $\boldsymbol{s}_i \cdot \boldsymbol{A}(j) = C(i, j)$ for every $i \in [n]$. There are at least $t + 1$ honest shares $\boldsymbol{s}_i$ that will pass the check, and due to Lemma 2 above, any incorrect share will fail the check with overwhelming probability. Hence, P_j will have sufficient correct shares to reconstruct the right secret: from the $\geq t + 1$ shares $\boldsymbol{s}_i = (F_0(i), \ldots, F_t(i))$ that pass the check, P_j uses the corresponding $F_0(i)$'s to reconstruct the secret $x = F_0(0)$. We call this procedure $\pi_{\mathsf{QuadRec}}(\langle x \rangle)$.

3 Efficient Reconstruction

It is possible to securely evaluate any arithmetic circuit over $\mathbb{F}$ with using our robust secret-sharing solution and multiplication triples $(\langle a \rangle, \langle b \rangle, \langle ab \rangle)$ from the preprocessing model: to add two robustly shared values the parties use the linearity properties of $\langle \cdot \rangle$, and to multiply they make use of the multiplication triples by first opening $d \leftarrow \langle x \rangle - \langle a \rangle$ and $e \leftarrow \langle y \rangle - \langle b \rangle$, and then computing locally $\langle xy \rangle \leftarrow e\langle a \rangle + d\langle b \rangle + \langle ab \rangle + de$. Reconstruction is guaranteed to result in the correct d and e, which ensures correctness, which is also the same property that guarantees the final output can be reconstructed correctly.

Since our goal is to achieve linear communication, we must design a way of reconstructing a series of shared values robustly and with linear communication per secret, which is precisely what we discuss in this section. Jumping ahead, our protocol will reconstruct n^2 secrets with $O(n^3)$ communication per secret, which means $O(n)$ per secret amortized. In the MPC protocol sketched above[8] each multiplication gate requires two reconstructions, and all reconstructions corresponding to multiplication gates in a single layer can be batched together. This means we need $n^2/2 = O(n^2)$ multiplication gates per layer (in average) to get the linear communication benefits from our reconstruction procedure, which is where the circuit width requirements of our results come from.

3.1 Towards Efficient Reconstruction

The naive approach of reconstructing a shared value by every party sending their share to every other party has quadratic complexity even when ignoring additional information, like tags etc., that are needed to filter out incorrect shares. In other words, just communicating the actual Shamir shares produces a too

[8] Even though this approach is quite standard in the literature, we provide a formal description and a security proof in the full version.

large overhead. For passive security, one can reconstruct a Shamir-shared value by sending the shares to a single party, who reconstructs and acts as a relay by sending the result to the other parties. In the actively secure setting, the work of [DN07] introduces a technique to achieve linear communication when reconstructing a batch of Shamir-shared values, essentially by using a different honest party to reconstruct each different value. However, for guaranteed output delivery this trick requires error correction, which is possible when $t < n/3$, but does not work in the $t < n/2$ regime, where only error detection is possible.

Here, we show how to reconcile the trick from [DN07] with the sharing $\langle x \rangle$ introduced above so that the (amortized) communication of the actual Shamir shares $s_i = F_0(i)$ becomes *linear*. However, the resulting reconstruction approach will still involve quadratic communication, but crucially, the super-linear overhead will be "only" due to the tags, *i.e.*, the remaining coordinates of $\boldsymbol{F}(i)$. We address this in Section 3.2, where we show how this quadratic communication can be taken care of. This is done, in essence, by batching together sufficiently many openings and using a single set of tags to verify a random linear combination.

Consider $t+1$ sharings $\langle x^{(0)} \rangle, \ldots, \langle x^{(t)} \rangle$ that need to be reconstructed. They are given by $t+1$ triples $\big(\boldsymbol{A}(\mathtt{Y}), \boldsymbol{F}^{(\ell)}(\mathtt{X}), C^{(\ell)}(\mathtt{X},\mathtt{Y})\big)$ with the same $\boldsymbol{A}(\mathtt{Y})$ and with $C^{(\ell)}(\mathtt{X},\mathtt{Y}) = \boldsymbol{F}^{(\ell)}(\mathtt{X}) \cdot \boldsymbol{A}(\mathtt{Y})$. Inspired by the basic idea from [DN07], we consider $\boldsymbol{F}(\mathtt{X},\mathtt{Z}) = \sum_{\ell=0}^{t} \mathtt{Z}^\ell \cdot \boldsymbol{F}^{(\ell)}(\mathtt{X})$ and $C(\mathtt{X},\mathtt{Y},\mathtt{Z}) = \sum_{\ell=0}^{t} \mathtt{Z}^\ell \cdot C^{(\ell)}(\mathtt{X},\mathtt{Y})$ which satisfy $C(\mathtt{X},\mathtt{Y},\mathtt{Z}) = \boldsymbol{F}(\mathtt{X},\mathtt{Z}) \cdot \boldsymbol{A}(\mathtt{Y})$. The reconstruction then proceeds as follows:

1. Each P_i sends $\boldsymbol{F}(i,j) = \sum_\ell j^\ell \boldsymbol{F}^{(\ell)}(i)$ to P_j.
2. Each P_j checks that $C(i,j,j) = \boldsymbol{F}(i,j) \cdot \boldsymbol{A}(j)$ for $i \in [n]$, and P_j reconstructs $\boldsymbol{F}(0,j)$ from the values that pass the check.
3. Each P_j sends to each P_k the vector $\boldsymbol{F}(0,j)$.
4. Each P_k checks that $C(0,k,j) = \boldsymbol{F}(0,j) \cdot \boldsymbol{A}(k)$ for each $j \in [n]$, and then P_k reconstructs $\boldsymbol{F}(0,\mathtt{Z})$ from the values $\boldsymbol{F}(0,j)$ that pass the check. From $\boldsymbol{F}(0,\mathtt{Z})$ one can then read out $\boldsymbol{F}^{(0)}(0), \ldots, \boldsymbol{F}^{(t)}(0)$ and thus $x^{(0)}, \ldots, x^{(t)}$.

The key idea in our protocol above is the following. When P_j receives the vectors $\boldsymbol{F}(1,j), \ldots, \boldsymbol{F}(n,j)$ in step 1, P_j can filter out incorrect shares and hence reconstruct $\boldsymbol{F}(0,j)$, so in particular P_j obtains the "secret" $F_0(0,j)$, which is relayed to each other party P_k. However, the main observation is that P_j also obtained as a "byproduct" the other points $(F_1(0,j), \ldots, F_t(0,j))$, and it turns out these can be used for P_j to convince each receiver P_k of the correctness of $F_0(0,j)$. In a bit more detail, we make the crucial observation that $\boldsymbol{F}(\mathtt{X},j)$ are the share vectors corresponding to the sharing $\big(\boldsymbol{A}(\mathtt{Y}), \boldsymbol{F}(\mathtt{X},j), C(\mathtt{X},\mathtt{Y},j)\big)$. So, from the discussion before Lemma 2, an honest P_j will indeed be able to recover $\boldsymbol{F}(0,j)$. Similarly the $\boldsymbol{F}(0,j)$'s sent to P_k in step 3. are the share vectors corresponding to the sharing $\big(\boldsymbol{A}(\mathtt{Y}), \boldsymbol{F}(0,\mathtt{Z}), C(0,\mathtt{Y},\mathtt{Z})\big)$, allowing each P_k to recover $\boldsymbol{F}(0,\mathtt{Z})$.

3.2 Batched Verification

Unfortunately, the above reconstruction still has quadratic amortized complexity. This originates in the fact that, in the first (and third) step, each party sends

a length-$(t+1)$ vector to each other party. However, the crucial observation is that if we do *not* count the information that is "only" sent for checking purposes, *e.g.*, if in step 1. we only count the first coordinate $F_0(i,j)$ of $\boldsymbol{F}(i,j)$, then we actually *have* linear amortized complexity.

Due to this observation, we can get the aspired amortized linear complexity by doing the verification in batches, that is, compressing the checking information of a number of reconstructions without increasing the associated communication costs. Concretely, we consider the reconstruction of n groups of $t+1$ secrets each: $\langle x^{(m,0)}\rangle, \ldots, \langle x^{(m,t)}\rangle$ for $m \in \{0,\ldots,n-1\}$. Intuitively, our protocol with linear communication complexity is obtained by running, for each $m \in \{0,\ldots,n-1\}$, the solution from the previous section, but ignoring the checking information. That is, step 1. from the previous section is modified by letting each P_i compute $\boldsymbol{F}^{(m)}(i,\mathtt{Z}) = \sum_{\ell=0}^{t} \mathtt{Z}^\ell \boldsymbol{F}^{(m,\ell)}(i)$, but P_i only sends the first coordinate $F_0^{(m)}(i,j)$ to P_j. For each other coordinate $h \in [t]$, P_i sends a *compressed* version $F_h(i,j) = \sum_{m=0}^{n-1} \xi^m F_h^{(m)}(i,j)$, where $\xi \in \mathbb{F}$ is a fresh uniformly random value known by all parties.[9]This can still be used by P_j to filter out incorrect shares and hence reconstruct $F_0^{(m)}(0,j)$ for $m \in \{0,\ldots,n-1\}$. Then, P_j relays these values to each other party P_k.

The challenge now is that, to interpolate $F_0^{(m)}(0,\mathtt{Z})$ and hence learn the reconstructed secrets, P_k requires certain checking information to verify the validity of the values $F_0^{(m)}(0,j)$ sent by P_j. In step 3. from Section 3.1, such information corresponds to $(F_1^{(m)}(0,j),\ldots,F_t^{(m)}(0,j))$, but P_j does not have the means to send this to P_k as P_j only received $(F_1(0,j),\ldots,F_t(0,j))$, which is a compressed version of these values (and moreover, even if P_j had this data, sending it to each P_k would be too costly). The solution once again is to apply compression. A first thought would be to let P_j send $(F_1(0,j),\ldots,F_t(0,j))$ to P_k, who can use these values to check the correctness of $F_0^{(0)}(0,j),\ldots,F_0^{(n-1)}(0,j)$ by verifying the correctness $F_0(0,j) = \sum_{m=0}^{n-1} \xi^m F_0^{(m)}(0,j)$ instead. However, this does not work since P_j already knows ξ before sending each $F_0^{(m)}$, so P_j can correct any error present in these terms.

The solution here is to sample a new fresh random challenge ω, after P_j has "committed" to the values $F_0^{(m)}$ by sending them to each P_k, and use the compressed check as above but with this new term for the linear combination. The problem now is that, for $h \in [t]$, P_j holds $F_h(0,j) = \sum_{m=0}^{n-1} \xi^m F_h^{(m)}(0,j)$, but not the necessary $F'_h(0,j) = \sum_{m=0}^{n-1} \omega^m F_h^{(m)}(0,j)$ to convince each receiver P_k. To address this we simply let the parties run the "checking part" of the first part of the protocol above but using the challenge ω instead of ξ. More precisely, each P_i sends $F'_h(i,j) = \sum_{m=0}^{n-1} \omega^m F_h^{(m)}(0,j)$ to P_j, who uses this values to interpolate $F'_h(0,j)$, which P_j sends to P_k. The details of this protocol are provided below.

[9] This is done by reconstructing, using the procedure π_{QuadRec} from Sect. 3.1, a preshared random $\langle \xi \rangle$ provided by the preprocessing functionality.

π_{LinRec}: Reconstruction with linear communication

Input: $(t+1)\cdot n$ secrets $(\langle x^{(m,\ell)}\rangle)$, for $\ell \in \{0,\dots,t\}$ and $m \in \{0,\dots,n-1\}$, each given by polynomials $(\boldsymbol{A}(\mathtt{Y}), \boldsymbol{F}^{(m,\ell)}(\mathtt{X}), C^{(m,\ell)}(\mathtt{X},\mathtt{Y}))$.
Output: Each party P_k learns all $(x^{(m,\ell)})_{m,\ell}$.
Preprocessing: A functionality $\mathcal{F}_{\mathsf{Prep}}$ that distributes sharings $\langle r\rangle$, where $r \in \mathbb{F}$ is uniformly random and unknown to the adversary.

For each $j \in [n]$, each P_k obtains $\{F_0^{(m)}(0,j)\}_{m=0}^{n-1}$:

1. For $m \in \{0,\dots,n-1\}$, each P_i computes $\boldsymbol{F}^{(m)}(i,\mathtt{Z}) = \sum_{\ell=0}^{t} \mathtt{Z}^\ell \boldsymbol{F}^{(m,\ell)}(i)$, and P_i sends $F_0^{(m)}(i,j)$ to each P_j.
2. The parties call $\mathcal{F}_{\mathsf{Prep}}$ to obtain $\langle\xi\rangle$, where $\xi \in \mathbb{F}$ is uniformly random and unknown to any party, and the parties execute the procedure $\pi_{\mathsf{QuadRec}}(\langle\xi\rangle)$, so that all parties learn ξ.
3. For $\ell \in \{0,\dots,t\}$ and $h \in [t]$, each P_i computes $F_h(i,\mathtt{Z}) = \sum_{m=0}^{n-1} \xi^m F_h^{(m)}(i,\mathtt{Z})$, and sends to each P_j the vector $(F_1(i,j),\dots,F_t(i,j))$.
4. Each P_j computes, for $i \in [n]$, $F_0(i,j) = \sum_{m=0}^{n-1} \xi^m F_0^{(m)}(i,j)$, and upon receiving $(F_1(i,j),\dots,F_t(i,j))$ from P_i, P_j checks that
$$\boldsymbol{F}(i,j)\cdot\boldsymbol{A}(j) = \sum_{m=0}^{n-1}\sum_{\ell=0}^{t} \xi^m j^\ell \cdot C^{(m,\ell)}(i,j).$$
5. Let $\mathcal{I} \subseteq [n]$ be the set of indexes i's for which the check above did not fail. P_j interpolates $\boldsymbol{F}(\mathtt{X},j)$ from $(\boldsymbol{F}(i,j))_{i\in\mathcal{I}}$.
6. Each P_j sends $\{F_0^{(m)}(0,j)\}_{m=0}^{n-1}$ to each P_k

Each P_j receives checking information:

7. The parties call $\mathcal{F}_{\mathsf{Prep}}$ to obtain $\langle\omega\rangle$, where $\omega \in \mathbb{F}$ is uniformly random and unknown to any party, and the parties execute the procedure $\pi_{\mathsf{QuadRec}}(\langle\omega\rangle)$, so that all parties learn ω.
8. Each P_i computes $F'_h(i,\mathtt{Z}) = \sum_{m=0}^{n-1} \omega^m F_h^{(m)}(i,\mathtt{Z})$ for $h \in [t]$. Then P_i sends $(F'_1(i,j),\dots,F'_t(i,j))$ to each P_j.
9. Each P_j computes, for $i \in [n]$, $F'_0(i,j) = \sum_{m=0}^{n-1} \omega^m F_0^{(m)}(i,j)$, and upon receiving $(F'_1(i,j),\dots,F'_t(i,j))$ from P_i, P_j checks that
$$\boldsymbol{F}'(i,j)\cdot\boldsymbol{A}(j) = \sum_{m=0}^{n-1}\sum_{\ell=0}^{t} \omega^m j^\ell \cdot C^{(m,\ell)}(i,j).$$
10. Let $\mathcal{I} \subseteq [n]$ be the set of indexes i's for which the check above did not fail. P_j interpolates $\boldsymbol{F}'(\mathtt{X},j)$ from $(\boldsymbol{F}'(i,j))_{i\in\mathcal{I}}$.

Each P_j sends checking information to each P_k, who then reconstruct:

11. Each P_j sends $(F'_1(0,j),\ldots,F'_t(0,j))$ to each P_k.
12. Upon receiving these values, each P_k computes $F'_0(0,j) = \sum_{m=0}^{n-1} \omega^m \cdot F^{(m)}(0,j)$ and checks that

$$\boldsymbol{F}'(0,j) \cdot \boldsymbol{A}(j) = \sum_{m=0}^{n-1} \sum_{\ell=0}^{t} \omega^m j^\ell \cdot C^{(m,\ell)}(0,j),$$

    for each $j \in [n]$
13. Let $\mathcal{J} \subseteq [n]$ be the set of indexes j's for which the check above did not fail. For each $m \in \{0,\ldots,n-1\}$, P_k interpolates $F_0^{(m)}(0,\mathtt{Z}) = \sum_{\ell=0}^{t} x^{(m,\ell)}\mathtt{Z}^\ell$ from $(F_0^{(m)}(0,j))_{j\in\mathcal{J}}$, and outputs $(x^{(m,\ell)})_{m,\ell}$.

Theorem 1. *After executing procedure π_{LinRec} on input $(\langle x^{(m,\ell)}\rangle)_{\ell\in\{0,\ldots,t\},m\in\{0,\ldots,n-1\}}$, each party P_k outputs the correct secrets $x^{(m,\ell)}$, except with probability $3t(n+1)/|\mathbb{F}|$. Moreover, the protocol requires linear communication complexity and makes use of a constant number of rounds.*

Proof. The claim on the number of rounds is verified by inspection. It is also easy to check that the total communication is $\Theta(n^3)$, and when we divide by the $(t+1)n = \Theta(n^2)$ elements being reconstructed, we obtain an amortized communication of $\Theta(n)$ per secret, as required.

Now, we prove the correctness and security of the protocol. To this end, we begin with the following claim.

Claim. In step 5, each P_j interpolates the correct $\boldsymbol{F}(\mathtt{X},j)$, except with probability $t(n+1)/|\mathbb{F}|$.

Proof (of claim). Let us consider a malicious party P_i who sends incorrect $\{F_0^{(m)}(i,j)+\epsilon_0^{(m)}\}_{m=0}^{n-1}$, and $(F_1(i,j)+\epsilon_1,\ldots,F_t(i,j)+\epsilon_t)$, to P_j. Assume that at least one $\epsilon_0^{(m)}$ is not zero. The check that P_j performs is

$$\sum_{h=0}^{t}(F_h(i,j)+\epsilon_h)\cdot A_h(j) = \sum_{m=0}^{n-1}\sum_{\ell=0}^{t}\xi^m j^\ell \cdot C^{(m,\ell)}(i,j),$$

where $\epsilon_0 = \sum_{m=0}^{n-1}\xi^m\epsilon_0^{(m)}$. Notice that the distribution of $\{\epsilon_0^{(m)}\}_{m=0}^{n-1}$ is independent of ξ since P_i sent $\{F_0^{(m)}(i,j)+\epsilon_0^{(m)}\}_{m=0}^{n-1}$ to P_j before the value ξ was opened. Hence, since at least one $\epsilon_0^{(m)}$ is not zero, Schwartz-Zippel lemma implies that ϵ_0 is also not zero except with probability at most $(n-1)/|\mathbb{F}|$.

It can be checked that the right hand side is equal to $\sum_{h=0}^{t} F_h(i,j)\cdot A_h(j)$, so in particular the check passes if and only if $\sum_{h=0}^{t}\epsilon_h\cdot A_h(j) = 0$. From the above, except with probability at most $|\mathbb{F}|^{-1}$, the vector $\boldsymbol{\epsilon}$ is not zero. Furthermore, from Lemma 2 we have that, except with probability $|\mathbb{F}|^{-1}$, the vector $\boldsymbol{A}(j)$ looks uniformly random to the adversary. Hence, we see that except with probability

$(n-1)/|\mathbb{F}| + 1/|\mathbb{F}| = n/|\mathbb{F}|$, the adversary passes the check if and only if a dot product between a random vector and a non-zero vector results in zero. This can happen only with probability $1/|\mathbb{F}|$. Hence, except with probability $1-(n+1)/|\mathbb{F}|$, the shares received by P_i are rejected.

From the above we see that the probability that P_j accepts an incorrect share is at most $(n+1)/|\mathbb{F}|$. Since there are at most t malicious parties, we have that the probability that there is at least one incorrect share accepted by P_j is at most $t(n+1)/|\mathbb{F}|$. Since the check for every honest party passes, and there are at least $t+1$ honest parties, P_j successfully reconstructs the correct $\boldsymbol{F}(\mathtt{X}, j)$, except with the probability above. This completes the proof of the claim.

With the claim at hand, we see that with overwhelming probability, every honest party P_j sends the correct $F_0^{(m)}(0, j)$ to each P_k in step 6. In a completely similar way as the proof of the claim above, we can prove the following:

Claim. In step 10, each P_j interpolates the correct $\boldsymbol{F}'(\mathtt{X}, j)$, except with probability $t(n+1)/|\mathbb{F}|$.

Proof (of claim). We proceed in the same way as in the claim above, but replacing ξ by ω, and $(F_1(i,j) + \epsilon_1, \ldots, F_t(i,j) + \epsilon_t)$ by $(F'_1(i,j) + \delta_1, \ldots, F'_t(i,j) + \delta_t)$, where δ_h are the possible errors introduced by P_i in step 8. The same proof works given that, as before, the error $\epsilon_0^{(m)}$ on $F_0^{(m)}$ is chosen by the adversary before sampling ω. We do not write down the rest of the details.

This claim shows then that, with overwhelming probability, each honest P_j will send to each P_k the correct $(F'_1(0,j), \ldots, F'_t(0,j))$ in Step 11, so in particular P_k receives at least $t+1$ correct shares. This turns out to be enough for an honest P_k to interpolate $F_0^{(m)}(0, \mathtt{Z})$ correctly since, as the following claim illustrates, P_k can filter out incorrect shares with overwhelming probability.

Claim. In step 13, P_k interpolates the correct $F_0^{(m)}(0, \mathtt{Z})$, except with probability at most $t(n+1)/|\mathbb{F}|$.

Proof (of claim). The proof is similar to that of the previous two lemmas 1 and 2. Consider a malicious P_j who sends $\{F_0^{(m)}(0,j) + \delta_0^{(m)}\}_{m=0}^{n-1}$ to P_k in step 6, and also $(F'_1(0,j) + \delta_1, \ldots, F'_t(0,j) + \delta_t)$ in step 11. The check that P_k carries out is then

$$\sum_{h=0}^{t} (F'_h(0,j) + \delta_h) \cdot A_h(j) = \sum_{m=0}^{n-1} \sum_{\ell=0}^{t} \omega^m j^\ell \cdot C^{(m,\ell)}(0,j),$$

where $\delta_0 = \sum_{m=0}^{n-1} \omega^m \delta_0^{(m)}$. The right-hand side equals $\sum_{h=0}^{t} F'_h(0,j) \cdot A_h(j)$, so the check passes if and only if $\sum_{h=0}^{t} \delta_h \cdot A_h(j) = 0$. Here, we proceed as with the proofs of the previous claims, noticing that $\{\delta_0^{(m)}\}_{m=0}^{n-1}$ is chosen by the adversary before seeing the challenge ω, so $\delta_0 \neq 0$ with probability at least $1 - (n-1)/|\mathbb{F}|$. Following similar steps as the previous proofs, we obtain that P_k accepts an incorrect share with probability at most $(n+1)/|\mathbb{F}|$. Hence, the probability that P_k reconstructs an incorrect $F_0^{(m)}(0, \mathtt{Z})$ is at most $t(n+1)/|\mathbb{F}|$, as stated in the claim.

Putting together what we have seen above, we obtain that, except with probability $3t(n+1)/|\mathbb{F}|$, each P_k reconstructs the correct secrets. Thus, the theorem is proven. □

Acknowledgments

This paper was prepared in part for information purposes by the Artificial Intelligence Research Group and the AlgoCRYPT CoE of JPMorgan Chase & Co and its affiliates ("JP Morgan") and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy, or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer, or solicitation for the purchase or sale of any security, financial instrument, financial product, or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

AAPP22. Ittai Abraham, Gilad Asharov, Shravani Patil, and Arpita Patra. Asymptotically free broadcast in constant expected time via packed vss. In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I*, pages 384–414. Springer, 2022.

AAPP23. Ittai Abraham, Gilad Asharov, Shravani Patil, and Arpita Patra. Detect, pack and batch: Perfectly-secure mpc with linear communication and constant expected time. In *Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part II*, pages 251–281. Springer, 2023.

AAY21. Ittai Abraham, Gilad Asharov, and Avishay Yanai. Efficient perfectly secure computation with optimal resilience. In *Theory of Cryptography Conference*, pages 66–96. Springer, 2021.

BDOZ11. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–188. Springer, 2011.

BGIN20. Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero-knowledge proofs. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 244–276. Springer, 2020.

BOGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, 1988.

BPRW16. Allison Bishop, Valerio Pastro, Rajmohan Rajaraman, and Daniel Wichs. Essentially optimal robust secret sharing with maximal corruptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 58–86. Springer, 2016.

BSFO12. Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Annual Cryptology Conference*, pages 663–680. Springer, 2012.

BTH06. Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In *Theory of Cryptography Conference*, pages 305–328. Springer, 2006.

BTH08. Zuzana Beerliova-Trubiniova and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In *Theory of Cryptography Conference*, pages 213–230. Springer, 2008.

CDN15. Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation*. Cambridge University Press, 2015.

DN07. Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Annual International Cryptology Conference*, pages 572–590. Springer, 2007.

FY19. Serge Fehr and Chen Yuan. Towards optimal robust secret sharing with security against a rushing adversary. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III*, pages 472–499. Springer, 2019.

GRR98. Rosario Gennaro, Michael O Rabin, and Tal Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pages 101–111, 1998.

GSZ20. Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority mpc. In *Annual International Cryptology Conference*, pages 618–646. Springer, 2020.

HMP00. Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multiparty computation. In *International conference on the theory and application of cryptology and information security*, pages 143–161. Springer, 2000.

IKP+16. Yuval Ishai, Eyal Kushilevitz, Manoj Prabhakaran, Amit Sahai, and Ching-Hua Yu. Secure protocol transformations. In *Annual International Cryptology Conference*, pages 430–458. Springer, 2016.