Highly Efficient OT-Based Multiplication Protocols

Iftach Haitner[1,2], Nikolaos Makriyannis[3], Samuel Ranellucci[4], and Eliad Tsfadia[1,5]

[1] School of Computer Science, Tel Aviv University
Research supported by Israel Science Foundation grant 666/19
{iftachh,eliadtsf}@tau.ac.il
[2] Member of the Check Point Institute for Information Security
[3] Fireblocks, nikos@fireblocks.com
[4] Coinbase, samuel.ranellucci@coinbase.com
[5] Google Research

Abstract. We present a new OT-based two-party multiplication protocol that is almost as efficient as Gilboa's semi-honest protocol (Crypto '99), but has a high-level of security against malicious adversaries without further compilation. The achieved security suffices for many applications, and, assuming DDH, can be cheaply compiled into full security.

1 Introduction

In a two-party multiplication protocol, each party's output is a random additive share of the multiplication of the parties' private inputs. Two-party multiplication is a fundamental building-block of arithmetic secure computation, holding a role analogous to that oblivious transfer (OT) has in Boolean secure computation. We present a new, highly efficient (maliciously secure) OT-based two-party multiplication protocol below, but first start with some background.

1.1 Background on OT-Based Two-Party Multiplication

There are a several known techniques to obtain two-party multiplication, historically falling in one of two categories: protocols based on homomorphic encryption (HE), or protocols based on (Boolean) OT. The two classes of protocols offer different tradeoffs between efficiency and underlying security assumption; HE-based protocols are typically more efficient communication-wise, while OT-based are more efficient computation-wise. Also, HE-based protocols typically require stronger assumptions. In recent years, new paradigms [14, 6, 4, 1, 5] have emerged for realizing two-party multiplication,[6] where the underlying "machinery" is based on *homomorphic* [7, 8] or *function* [6] secret sharing. The two notions may be viewed as analogues of respectively HE and *functional encryption*

[6] Actually, most papers in the space focus on the related functionalities of OLE and VOLE, discussed later on.

[3] in the secret sharing realm. In this paper, we focus on OT-based protocols, and we refer the reader to Section 1.4 for further discussion on protocols that do not rely on OT.

Recall that OT is the functionality that takes two inputs $x_0, x_1 \in \mathbb{Z}_q$ from the sender, a bit β from the receiver, and returns x_β to the receiver (and nothing to the sender). To the best of our knowledge, there are essentially two basic templates for honest-but-curious OT-based multiplication: the Gilboa [15] protocol, and the Ishai, Prabhakaran, and Sahai [19] protocol. We refer the reader to Figure 1 for a side by side comparison of the two protocols. For clarity of exposition, we focus our attention on multiplications over the field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ for an odd prime q (i.e., the arithmetic field of integers modulo an odd prime).

Malicious Security. As far as we know, all OT-based multiplication protocols only achieve honest-but-curious (passive) security.[7] To achieve malicious security, these protocols can be compiled in a number of generic ways, e.g., using SNARKSs, cut-and-choose, and/or MPC-in-the-head techniques. For concrete efficiency, however, it is often preferable to design tailor-made solutions [20, 14]. For instance, motivated by applications to MPC in the preprocessing model, Keller et al. [20] (MASCOT) design various cut-and-choose techniques, on top of Gilboa's protocol, for maliciously realizing various useful functionalities in the preprocessing model. We discuss MASCOT in detail in Section 1.3.

1.2 Our Contributions

We present a new OT-based two-party multiplication protocol that achieves a high level of security against malicious adversaries. The protocol may be viewed as a noisy generalization of Gilboa [15]'s protocol (or, alternatively, as a hybrid between Gilboa [15] and Ishai et al. [19] protocols).

Let $a, b \in \mathbb{Z}_q$ be the inputs of P_1 and P_2, respectively, and let $n = \lceil \log q \rceil + \kappa$ for a (statistical) security parameter κ. Our protocol requires no initialization stage, and the parties make n parallel OT-calls. In the i^{th} call, P_2's input index is a random value $t_i \leftarrow \{-1, 1\}$ (i.e., we switch conventions regarding the OT-receiver's input),[8] and P_1's input pair is $(-a + \delta_i, a + \delta_i)$ for a random mask $\delta_i \leftarrow \mathbb{Z}_q$. Notice that this differs from Ishai et al. [19] protocol in which P_1's input in for OT-calls depends on the vectors sent by P_2. After these calls are done, P_2 uniformly samples $\boldsymbol{v} = (v_1, \ldots, v_n) \leftarrow \mathbb{Z}_q^n$ subject to $b = \sum_i v_i t_i$, and sends $\boldsymbol{v}$, but not the t_i's, to P_1. See Protocol 1 for a more detailed description.

Protocol 1 (Our OT-based multiplication protocol $(\mathsf{P}_1, \mathsf{P}_2)$)

[7] The OT-based protocol of Ghosh, Nielsen, and Nilges [14] does achieve malicious security (without further compilation), but its security proof relies on an additional hardness assumption (a rather non-standard coding assumption). Interestingly, the security analysis in [14] is somewhat reminiscent of the security analysis of our protocol.

[8] The choice of $\{-1, 1\}$ instead of $\{0, 1\}$ significantly simplifies our security analysis, but it is also what limits it to fields of characteristic greater than two (see Theorem 2).

Gilboa's Protocol

– **Init.** Let $\ell = \lceil \log q \rceil$.
  P_2 sets $t_1, \ldots t_\ell \in \{0,1\}$ to the bit-decomposition of $b = \sum_i t_i \cdot 2^{i-1}$.

– **OT.** The parties make ℓ parallel OT-calls, P_1 as sender, P_2 as receiver.
  In the i^{th} call:
  1. P_1 uses input $(\delta_i, a + \delta_i)$, for $\delta \leftarrow \mathbb{Z}_q$. (It receives no output).
  2. P_2 uses input index t_i.
     It receives output $z_i \in \mathbb{Z}_q$.
– **Outputs.**
  1. P_1 outputs $-\sum_i \delta_i \cdot 2^{i-1}$.
  2. P_2 outputs $\sum_i z_i \cdot 2^{i-1}$.

Ishai et al.'s Protocol

– **Init.** Let $\ell = \lceil \log q \rceil$ and $n = \ell + \kappa$.
  1. P_2 samples $\boldsymbol{u}_0, \boldsymbol{u}_1 \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{t} \leftarrow \{0,1\}^n$, subject to $b = \sum_i u_{t_i,i}$.
  2. P_2 sends $(\boldsymbol{u}_0, \boldsymbol{u}_1)$ to P_1.
– **OT.** The parties make n parallel OT-calls, P_1 as sender, P_2 as receiver.
  In the i^{th} call:
  1. P_1 uses input $(au_{0,i} + \delta_i, au_{1,i} + \delta_i)$, for $\delta_i \leftarrow \mathbb{Z}_q$. (It receives no output).
  2. P_2 uses input index t_i.
     It receives output $z_i \in \mathbb{Z}_q$.
– **Outputs.**
  1. P_1 outputs $-\sum_i \delta_i$.
  2. P_2 outputs $\sum_i z_i$.

Fig. 1: Honest-But-Curious multiplication protocols between party P_1, holding input $a \in \mathbb{Z}_q$, and party P_2, holding input $b \in \mathbb{Z}_q$. Gilboa's protocol consists of $\ell = \lceil \log(q) \rceil$ parallel OT-calls, where Ishai et al. [19]' protocol consists on $n = \ell + \kappa$ calls, where κ is a (statistical) security parameter. We remark that Gilboa's protocol can be cast as a variant of Ishai et al. [19]' protocol, where the pair of vectors $(\boldsymbol{u}_0, \boldsymbol{u}_1)$, which P_2 uses for encoding its input in Ishai et al. [19] are implicitly hardcoded as $\boldsymbol{u}_0 = (0, \ldots, 0)$ and $\boldsymbol{u}_1 = (1, 2^1, 2^2, \ldots, 2^{\ell-1})$. Gilboa [15], however, dispenses of the communication round prior to the OT, since the two vectors are known in advance to both parties, and achieves perfect security (in the OT-hybrid model).

– ***Inputs.*** *The parties hold common input 1^κ. Party P_1 holds private input $a \in \mathbb{Z}_q$, and party P_2 holds private input $b \in \mathbb{Z}_q$. Let $n = \lceil \log q \rceil + \kappa$.*
– ***OT.*** *The parties makes n parallel OT-calls. In the i-th call:*
  1. *P_1, as the sender, inputs pair $(-a + \delta_i, a + \delta_i)$ for a uniform $\delta_i \leftarrow \mathbb{Z}_q$. (It receives no output.)*
  2. *P_2, as the receiver, inputs index $t_i \leftarrow \{-1, 1\}$, and receives output $z_i \in \mathbb{Z}_q$.*
– ***Outputs.***
  1. *P_2 samples $\boldsymbol{v} = (v_1, \ldots, v_n) \leftarrow \mathbb{Z}_q^n$ subject to $b = \sum_i v_i \cdot t_i$. It sends $\boldsymbol{v}$ to P_1.*

2. P_1 *outputs* $-\sum_i \delta_i \cdot v_i$.
3. P_2 *outputs* $\sum_i z_i \cdot v_i$.

Before we discuss the merits of our protocol, we briefly touch on the correctness and security analysis. It is easy to see that the protocol is correct (when invoked by honest parties). Indeed,

$$\begin{aligned} s_2 = \langle \boldsymbol{v}, (z_1, \ldots, z_n) \rangle &= \langle \boldsymbol{v}, \underbrace{(\delta_1, \ldots, \delta_n)}_{\boldsymbol{\delta}} + a \cdot \underbrace{(t_1, \ldots, t_n)}_{\boldsymbol{t}} \rangle \\ &= a \cdot \langle \boldsymbol{v}, \boldsymbol{t} \rangle + \langle \boldsymbol{v}, \boldsymbol{\delta} \rangle = a \cdot b - s_1, \end{aligned}$$

making $s_1 + s_2 = a \cdot b$. Second, (similarly to Gilboa's protocol mentioned earlier) the protocol is fully secure for a malicious P_2: the only way P_2 may deviate from the protocol is by choosing a different value for $\boldsymbol{v}$ (unrelated to b) at the last stage of the protocol. This behavior, however, is equivalent to choosing a different input, and thus does not violate the security of the protocol. The analysis for a malicious P_1 is more involved. Effectively, P_1 is limited to choosing inconsistent inputs for the OT-calls: instead of using (a_i, a'_i) of the form $(\delta_i - a, \delta_i + a)$, a corrupted P_1 may choose pairs of inputs which are not consistent across different OT-calls i.e., for some $i \neq j$, it holds that $a_i - a'_i \neq a_j - a'_j$, and it seems this attack cannot be simulated using access to the (standard) multiplication functionality.[9] Instead, we show that it exhibits the following useful dichotomy: depending on the number of inconsistent inputs in the OT-calls provided by P_1, either the execution can be simulated using the standard multiplication functionality (with $2^{-\kappa/4}$ statistical-closeness), or, P_2's output has *min-entropy* at least $\kappa/4$, when conditioning jointly on P_2's input and P_1's view. That is, P_2's output is highly unpredictable, even when knowing its input. This property is technically captured by the following informally stated theorem.

Theorem 2 (Security of our multiplication protocol, informal). *For adversary* A *corrupting* P_1*, consider a random execution of Protocol 1 in the presence of* A*, where* P_2 *is holding input* b*, and let* $\mathrm{out}_2^{\mathsf{A}}(b)$ *denote* P_2*'s output and* $\mathrm{view}^{\mathsf{A}}(b)$ *denote* A*'s view in this execution. Assume* $q \geq 2^{\kappa/2}$*,* [10] *then at least one of the following holds (depending on its inputs to the* OT*-calls):*

1. A *can be simulated given access to the perfect (standard) multiplication functionality. (By extracting the input to the perfect multiplication from* A*'s inputs to the* OT*-calls.)*

[9] It is not too hard to get convinced that our protocol does not realize the multiplication functionality with *statistical* security (in the OT-hybrid model), but we defer the rather tedious proof of this fact to the next version of this paper. It seems plausible, however, that under the right *Subset-Sum* hardness assumption, the protocol does realize the multiplication functionality with *computational* security. Proving it is an intriguing open question.

[10] We discuss how our results extend to arbitrary fields of characteristic greater than two in Section 2.

2. $\mathrm{H}_\infty\left(\mathrm{out}_2^{\mathsf{A}}(b) \mid \mathrm{view}^{\mathsf{A}}(b), b\right) \geq \kappa/4$. *(i.e.,* $\mathsf{P}_2(b)$*'s output is* unpredictable *from* A*'s point of view, even if* A *knows* b*.)*

We prove Theorem 2 by showing that our protocol realizes a "weak" ideal multiplication functionality that formally captures the two conditions above (see Section 4 for details). The above security guarantee makes our protocol very desirable for a number of reasons, enumerated below.

1. First, via a simple reduction from (standard) designated-input multiplication to random-input multiplication, we can compile our protocol into a maliciously secure protocol by performing an *a posteriori* check on the shares. Such a check does not seem to exist for Gilboa [15], Ishai et al. [19] protocols.
2. Second, and more importantly, we claim that the security notion achieved out-of-the-box by our protocol is sufficient for a number of applications, e.g., within protocols where some kind of correctness check is performed obliviously on the parties' outputs. For instance, in the threshold ECDSA of Lindell and Nof [23], the output is released only after it is checked for correctness. Consequently our protocol can readily be used as a multiplication protocol therein.

Batching. We show that our protocol enjoys the following performance improvement when performing m multiplications with P_1 using the same input in each instance; this task essentially corresponds to the important VOLE functionality discussed in Section 1.3. Instead of running the protocol m times (and thus paying $m \cdot n = m \cdot (\ell + \kappa)$ OT's), our protocol can be batched so that it requires only $\kappa + m \cdot \ell$ calls to the underlying OT functionality. The batched version of our protocol exhibits a similar dichotomy to the non-batched version: either the protocol is secure (with $2^{-\kappa/4}$ closeness to the ideal world), or, if not, each one of the honest outputs has *min-entropy* at least $\kappa/4$, *even when conditioning on all of the honest party's inputs* (albeit there may be dependencies between the outputs). For large m, our approach almost matches the number of OT-calls from Gilboa's honest-but-curious protocol, while achieving a stronger security notion. Moreover, in the Random Oracle Model (ROM), it is possible to also bring down the communication complexity of our protocol to match [15] by instructing P_2 to communicate $\boldsymbol{v} = (v_1, \ldots, v_n)$ succinctly via the oracle, e.g., by sending a short seed instead of the entire vector. Furthermore, for malicious security, it is enough to perform a single *a posteriori* check on the shares of only one of the underlying multiplications (say the first multiplication). Indeed, our dichotomy result guarantees that the check is successful only if the attack can be simulated in the ideal world (and thus all outputs are well-formed).

As a concrete efficiency example, for a prime q for which there exists a q-size group where DDH is assumed to hold (say secp256k1 – the Bitcoin curve – with prime $q \approx 2^{256}$), we instantiate the correctness-check using El-Gamal commitments (these commitments were thoroughly used in [23] in the context of threshold ECDSA). We estimate that the correctness-check requires computational-complexity of around 30 exponentiations in the group and communication-complexity of 20 group elements (assuming the encodings of field elements and

group elements have essentially the same size). Since this penalty is independent of the number of multiplications in the batch, performing a batch of m multiplications with (full) malicious security $2^{-\kappa/4}$ in the ROM incurs the following cost:

OT's	Communication (bits)	Computation (group exp.)
$m \cdot \ell + \kappa$	$(m + 20) \cdot \ell$ bits	30

Hence, even with the correctness-check, the complexity-penalty of our protocol compared to Gilboa's honest-but-curious protocol is insignificant for large m.[11]

1.3 Applications

In this section, we discuss several applications where our protocol may be of interest.

OLE & VOLE. The oblivious linear evaluation (OLE) functionality may be viewed as a variant of two-party multiplication where one party (say P_2) has full control over its share. Namely, on input a for P_1 and (b, σ) for P_2, the functionality returns $ab + \sigma$ to party P_1 and nothing to party P_2. An important generalization of OLE is *vector* oblivious linear evaluation (VOLE), where it is now assumed that P_2 holds a pair of vectors $(\boldsymbol{b}, \boldsymbol{\sigma})$ and P_1 learns the combination $a\boldsymbol{b} + \boldsymbol{\sigma}$. There is a straightforward reduction from OLE and VOLE to multiplication and batch-multiplication respectively and thus our protocol (compiled for malicious security) can readily be used for this purpose.

MACs & Multiplication Triplets. Motivated by applications of arithmetic MPC in the preprocessing model, i.e., generating function-independent correlated random data that can be later used by the parties to achieve statistically secure MPC for any functionality, there is a rich line of work ([2, 22, 13, 10, 11, 20] to name but a few) for generating message authentication codes (MACs) and authenticated multiplication triplets. For convenience, we recall the definition of each notion. On secret input x from P_1 (only one party provides input), the two-party MAC functionality returns $\tau \in \mathbb{Z}_q$ to P_1 and a pair $(k, \sigma) \in \mathbb{Z}_q^2$ to P_2 such that $\tau = x \cdot k + \sigma$. Thus, a corrupted P_1 is effectively committed to x which can be authenticated by revealing the pair (x, τ). Notice that P_2 accepts the decommitment if and only if $\tau = x \cdot k + \sigma$ which uniquely determines x (unless P_1 can guess k, which happens with negligible probability). For reference, σ and τ are referred to as the MAC shares and k is referred to as the MAC key. Next, we define authenticated multiplication triplets. On empty inputs, the authenticated multiplication triplets functionality (Beaver) returns (a_1, b_1, c_1) and (a_2, b_2, c_2) to P_1 and P_2 respectively such that $(a_1 + a_2) \cdot (b_1 + b_2) = c_1 + c_2$, together with MAC keys and shares for all the relevant data, i.e., P_2 holds a key k and shares

[11] Without the oracle the penalty is rather noticeable, since there is a $(\ell \cdot m + \kappa)$-multiplicative blowup in the communication complexity.

$\sigma, \sigma', \sigma''$, and P_1 holds τ, τ', τ'' as MAC data for the triplet (a_1, b_1, c_1), and the MAC data for P_2's triplet (a_2, b_2, c_2) is analogously defined (where the parties' roles are reversed). It goes without saying, our base protocol can be used to generate MACs and triplets in a straightforward way (explained further below). For comparison, we briefly outline MASCOT [20], the only purely OT-based work for generating triplets with malicious security.

MASCOT [20]. To realize the two functionalities described above in the presence of malicious adversaries, [20] employs a number of cut-and-choose techniques on top of Gilboa's protocol. Specifically, for the MAC functionality, the authors propose the following process: P_2 samples a random MAC key k and the parties run Gilboa's protocol twice; once with inputs (x, k) and once with inputs (x_0, k) where x_0 denotes a random dummy input sampled by P_1. At the end of the protocol the parties (are supposed to) obtain MAC shares for both x and x_0 under key k. To verify that P_1 behaved honestly (as we discussed earlier, only P_1 is capable of cheating), P_1 is instructed to reveal a random combination of x_0 and x as well as the same random combination of its MAC shares. If P_2 accepts, then, with all but negligible probability, P_2 is holding the right MAC data for x. The protocol for the Beaver functionality follows a similar template, however the added redundancy and check procedure (to verify correctness) is more involved. For brevity, we do not describe it here but we mention that it requires 6 or 8 executions (depending on the target security) of Gilboa's protocol on top of the required runs to obtain the MAC data (In total, Gilboa's protocol is ran 18 or 20 times depending on the target security for a single authenticated multiplication triple).

Using our protocol to generate MACs & Triplets maliciously. MAC-generation essentially coincides with batch-multiplication (where a single k is used as a MAC-key to authenticate many values x_1, x_2, ...). Thus, our batch-multiplication protocol (with the correctness-check) can readily be used for this purpose. Next, we turn to the triplets.

Analogously to standard multiplication, if we allow for an a posteriori check on the shares (more involved than the one presented earlier), we show how our protocol can be used to generate triplets. In particular, a single triplet can be generated by running our base protocol 2 times in its non-batched version (to generate the triplet) and 2 times in the batched version with batches of size 3 (to generate all the MAC-data), and then performing a correctness-check on the shares. For concreteness, we instantiate this check for prime q when there is an accompanying group where DDH is hard. We estimate that the correctness-check requires computational-complexity of around 90 exponentiations in the group and communication-complexity of 60 group elements. In total, this process incurs the following costs for generating a single triplet in the random oracle model.[12]

[12] Since it is not the focus of our paper, we have not examined how to optimize the protocol or correctness-check when many triplets are being generated, and we speculate that several optimizations are possible.

OT's	Communication (bits)	Computation (group exp.)
$4\kappa + 8\ell$	70ℓ	90

As an example, for $\ell \approx 512$, our protocol is 53% cheaper in usage of the underlying OT compared to MASCOT when aiming for security 2^{-64}.

Comparison to 2PC Multiplication from [12]. We note that our multiplication protocol may also improve the efficiency of the threshold ECDSA protocol of Doerner et al. [12]. In more detail, the core two-party multiplication protocol in [12] is a variant of MASCOT where the parties multiply (random) dummy values which are then opened in a cut-and-choose way to check for correctness. Specifically, for each (designated-input) multiplication, [12] instructs the parties to perform *two* random multiplications using the OT. Our protocol only prescribes one random multiplication and avoids this redundancy. Thus, our protocol enjoys an x2 improvement in the underlying use of OT.[13]

1.4 Related Work

Multiplication from noisy encoding. Drawing from [24], Ishai et al. [19] generalize their protocol so that it supports many types of encodings for P_2 input. Thus, instead of the two $\boldsymbol{u}$-vectors from Figure 1, P_2 may use different *noisy encoding* to encode its input prior to the OT. Under various coding assumption (e.g., [21]), Ishai et al. [19] show that several coding schemes give rise to honest-but-curious multiplication protocols with much improved complexity. As mentioned earlier, this approach was later shown to be sufficient by [14] for achieving malicious security under a specific coding assumption.

Non OT-based multiplication. Here we distinguish between HE-based and the more recent approaches based on homomorphic and function secret sharing. HE-Multiplication can be based on either somewhat homomorphic encryption or fully homomorphic encryption. We refer the reader to [25] for a discussion on HE-based multiplication in the context of a specific general-purpose MPC (the SPDZ protocol [11]). The work on the two newer notions (homomorphic and function secret sharing) is motivated by applications to correlated data generation in the prepossessing model (in the spirit of multiplication triplets). For instance, Boyle et al. [5] show how to generate OLE-correlations using homomorphic secret sharing (under various coding assumptions), and Boyle et al. [4] show how to generate long VOLE instances (again under various coding assumptions). These new approaches offer improvements over previous ones, especially in communication costs.

[13] When using OT-extensions, this improvement automatically translates into an x2 improvement in communication complexity, which is the most expensive resource in [12].

Paper Organization

In Section 2, we describe the high-level approach for analyzing the security of P_2 in Protocol 1, as stated in Theorem 2. Notations, definitions and general statements used throughout the paper are given in Section 3. Theorem 2 is formally stated and proved in Section 4, and its batching extension is formally stated in Section 5. Finally, in Section 6, we show how to compile our protocol generically for a number of applications (including, e.g., perfect multiplication). We note that we also provide (non-generic) group-theoretic instantiations in the supplementary material.

2 Our Techniques

In this section, we describe the high-level approach for analyzing the security of P_2 in Protocol 1, as stated in Theorem 2. For the formal proof of this theorem see Section 4.

Recall that a malicious A corrupting P_1 can deviate from the protocol by providing inputs to the OT-calls that are not consistent with any $a \in \mathbb{Z}_q$. Our security proof consist of a case-by-case analysis depending on how "far from consistent" A's inputs to the OT are. Let (w_i^-, w_i^+) denote the inputs that A uses in the i^{th} OT-call, let $a_i = (w_i^+ - w_i^-)/2$ and let $\delta_i = w_i^+ - a_i$. Let $\hat{a}$ be the value that appears the most often in $\boldsymbol{a} = (a_1, \ldots, a_n)$, and let $\boldsymbol{d} = \boldsymbol{a} - \hat{a} \cdot \mathbf{1}$. Intuitively, the hamming distance of $\boldsymbol{d}$ from $\mathbf{0}$ measures how much A deviates from honest behaviour. In particular, $\boldsymbol{d} = \mathbf{0}$ if P_1 uses the same a in all OT-calls, and the hamming weight of $\boldsymbol{d}$ is $n-1$ if P_1 never uses the same input twice. Let $\boldsymbol{t} = (t_1, \ldots, t_n)$, $\boldsymbol{z} = (z_1, \ldots, z_n)$ and $\boldsymbol{v}$ be the values that are sampled/obtained by P_2 in the execution, and let s_2 denote its final output. By definition, it holds that

$$\begin{aligned} s_2 = \langle \boldsymbol{v}, \boldsymbol{z} \rangle = \langle \boldsymbol{v}, \boldsymbol{\delta} + \boldsymbol{a} * \boldsymbol{t} \rangle &= \langle \boldsymbol{v}, \boldsymbol{\delta} + \hat{a} \cdot \boldsymbol{t} \rangle + \langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle \\ &= (\langle \boldsymbol{v}, \hat{a} \cdot \boldsymbol{t} \rangle + \langle \boldsymbol{v}, \boldsymbol{\delta} \rangle) + \langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle \\ &= (\hat{a} \cdot b + \langle \boldsymbol{v}, \boldsymbol{\delta} \rangle) + \langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle, \end{aligned}$$

letting $*$ stand for point-wise multiplication and $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)$. The last equation holds by the definition of $\boldsymbol{v}$. Thus, given P_1's view along with the value of b, notice that the value of s_2 is the addition of the following two summands: the *constant*[14] $(\hat{a} \cdot b + \langle \boldsymbol{v}, \boldsymbol{\delta} \rangle)$ (viewed as a single summand) and $\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle$.

We say that $\boldsymbol{a} \in \mathbb{Z}_q^n$ is *m-polychromatic*, if for every $y \in \mathbb{Z}_q$ it holds that $\mathrm{Ham}(\boldsymbol{d}, y^n) \geq m$ (e.g., $(0, 1, 2, 3, 0)$ is 3-polychromatic but not 4-polychromatic). We show that if $\boldsymbol{a}$ is *not* $\kappa/2$-polychromatic, hereafter *almost monochromatic*, then the execution of the protocol can be simulated using oracle-access to the perfect (i.e., standard) multiplication functionality (which provides the right share to each party, without any offset). Otherwise, if $\boldsymbol{a}$ is $\kappa/2$-polychromatic,

[14] given P_1's view and P_2's input

hereafter *polychromatic*, then $\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle$ has high min-entropy, given A's view and the value of b.

Before we further elaborate on each of the above two cases, we introduce the following notation. To distinguish between the values fixed adversarially by A and those sampled (honestly) by P_2, in the remainder we treat the adversary's inputs as fixed values and the honest party's input as random variables. Namely, it is assumed that $\boldsymbol{a} \in \mathbb{Z}_q^n$ is fixed (and thus also the vector $\boldsymbol{d}$), and we let $\boldsymbol{V}$ and $\boldsymbol{T}$ denote the random variables where $\boldsymbol{v}$ and $\boldsymbol{t}$ are drawn from (i.e., uniform distribution over $\mathbb{Z}_q^n$ and $\{-1,1\}^n$, respectively).

Almost-Monochromatic $\boldsymbol{a}$ yields statistical security. We prove this part by showing that, given $\boldsymbol{V}$, the value of $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle$ is close to being *independent* of b. Namely, for any $b, b' \in \mathbb{Z}_q$,

$$\mathrm{SD}\big((\boldsymbol{V}, \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle)|_{\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle = b},\ (\boldsymbol{V}, \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle)|_{\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle = b'}\big) \leq 2^{-\kappa/4} \tag{1}$$

Equation (1) yields that the simulation of P_2 in the ideal world, given access to the perfect multiplication functionality, can be simply done by emulating P_2 on an arbitrary input.

To see why Equation (1) holds, let $\mathcal{I} := \{i \in [n] \colon \boldsymbol{d}_i \neq 0\}$, and assume $\boldsymbol{T}_{\mathcal{I}}$ (the value of $\boldsymbol{T}$ in the coordinate of $\mathcal{I}$) is fixed to some $\boldsymbol{s} \in \{-1,1\}^{|\mathcal{I}|}$. Since, given this fixing, $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle = \langle \boldsymbol{V}_{\mathcal{I}}, \boldsymbol{d}_{\mathcal{I}} * \boldsymbol{s} \rangle$ is a deterministic function of $\boldsymbol{V}$, proving the monochromatic case is reduced to proving that

$$\mathrm{SD}(\boldsymbol{V}|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}, \boldsymbol{V}) \leq 2^{-\kappa/4} \tag{2}$$

Since $\boldsymbol{d}$ is almost-monochromatic, then, given the above fixing of $\boldsymbol{T}_{\mathcal{I}}$, it still holds that $\mathrm{H}_\infty(\boldsymbol{T}) \geq n - |\mathcal{I}| \geq \lceil \log q \rceil + \kappa/2$. Thus, by the leftover hash lemma

$$\mathrm{SD}((\boldsymbol{V}, \langle \boldsymbol{V}, \boldsymbol{T} \rangle), (\boldsymbol{V}, U)) \leq 2^{-\kappa/4} \tag{3}$$

for a uniformly sampled $U \leftarrow \mathbb{Z}_q$. In other words, the value of $\boldsymbol{V}$ is $2^{-\kappa/4}$-close to uniform given $\langle \boldsymbol{V}, \boldsymbol{T} \rangle$, and Equation (2) follows by a not-too-complicated chain of derivations (see proof of Lemma 3).

Polychromatic $\boldsymbol{a}$ yields unpredictable offset. Fix $b \in \mathbb{Z}_q$, and for $\boldsymbol{t} \in \{-1,1\}^n$ let $W^{\boldsymbol{t}}$ be the indicator random variable of the event $\{\langle \boldsymbol{V}, \boldsymbol{t} \rangle = b\}$, and let $W := \sum_{\boldsymbol{t} \in \{-1,1\}^n} W^{\boldsymbol{t}}$. In addition, for $\boldsymbol{t} \in \{-1,1\}^n$ and $x \in \mathbb{Z}_q$, let $Z_x^{\boldsymbol{t}}$ be the indicator random variable of the event $\{\langle \boldsymbol{V}^b, \boldsymbol{t} \rangle = b \wedge \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{t} \rangle = x\}$, and let $Z_x := \sum_{\boldsymbol{t} \in \{-1,1\}^n} Z_b^{\boldsymbol{t}}$. We show that for a polychromatic $\boldsymbol{a}$, with probability $1 - 2^{-\kappa/4}$ over $\boldsymbol{V}$ it holds that

$$Z_x / W \leq 2^{-\kappa/4} \tag{4}$$

for *every* $x \in \mathbb{Z}_q$ (simultaneously). It follows that for such vector $\boldsymbol{a}$, with high probability over $\boldsymbol{V}$, the probability that $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle = x$, for *any* value of x, is small. In other words, $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle$ has high min-entropy given $(\boldsymbol{V}, b)$.[15]

[15] Actually, since the value of $\boldsymbol{v}$ sent to P_1 is not uniform, but rather distributed according to $\boldsymbol{V}^b := \boldsymbol{V}|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}$, to argue about the security of the protocol one needs

We prove Equation (4) by upper-bounding $\mathrm{E}[W^3]$ and $\mathrm{E}[Z_x^3]$, for any x, and then we use a third moment concentration inequality to derive Equation (4). The harder part is bounding $\mathrm{E}[Z_x^3]$. To get the gist of this bound, we give the intuition for bounding $\mathrm{E}[Z_x^2]$. This bound is derived by proving that the number of pairs $(\boldsymbol{t}, \boldsymbol{t}')$ with $\mathrm{E}[Z_x^{\boldsymbol{t}} \cdot Z_x^{\boldsymbol{t}'}] > 1/q^4$ is small. These pairs are identified by relating the correlation of the indicator random variables of the events $\{\langle \boldsymbol{V}, \boldsymbol{t} \rangle = b\}$, $\{\langle \boldsymbol{V}, \boldsymbol{t}' \rangle = b\}$, $\{\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{t} \rangle\}$ and $\{\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{t} \rangle\}$ to the dimension of space spanned by the vectors in $\mathcal{S}_{\boldsymbol{t},\boldsymbol{t}'} := \{\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{d} * \boldsymbol{t}, \boldsymbol{d} * \boldsymbol{t}'\}$. In particular, it is not hard to see that

$$\mathrm{rank}(\mathcal{S}_{\boldsymbol{t},\boldsymbol{t}'}) = j \implies \mathrm{E}[Z_x^{\boldsymbol{t}} \cdot Z_x^{\boldsymbol{t}'}] \leq 1/q^j$$

Hence, upper-bounding $\mathrm{E}[Z_x^2]$ reduces to upper-bounding to number of pairs $(\boldsymbol{t}, \boldsymbol{t}')$ with $\mathrm{rank}(\mathcal{S}_{\boldsymbol{t},\boldsymbol{t}'}) < 4$. Upper-bounding the number of such pairs is done using linear algebra arguments, exploiting the fact that $\boldsymbol{d}$ has at least $\kappa/2$ non-zero elements (since it is polychromatic). Specifically, we show that the number of pairs $(\boldsymbol{t}, \boldsymbol{t}')$ with $\mathrm{E}[Z_x^{\boldsymbol{t}} \cdot Z_x^{\boldsymbol{t}'}] < 1/q^4$ decreases *exponentially* with the weight of $\boldsymbol{d}$. This bound is sufficient for calculating the second moment of Z_x (deducing a weaker bound than Equation (4), cf., Section 4.2). Calculating the third moment of Z_x, however, for deriving Equation (4) is more involved, and requires a more detailed case-by-case analysis in the counting argument, cf., the full version of this paper [17].

Extension to Arbitrary Fields. Our results extend trivially to *large* finite fields (i.e., of size greater than $2^{\kappa/2}$). Next, we briefly explain how to use our protocol for multiplying in a small field, denoted $\mathbb{F}$. Unfortunately, as is, the protocol does not enjoy the same unpredictability under attack since the entropy of the offset is constrained by the size of the field, i.e., the offset has min-entropy at most $\log(|\mathbb{F}|)$. To circumvent this issue, we instruct the parties to embed $\mathbb{F}$ into a larger field $\mathbb{H}$ of size $2^{\kappa/2}$ and perform the multiplication in $\mathbb{H}$ (of course, the parties' shares then reside in the larger field).

To obtain additive shares over the smaller field $\mathbb{F}$, it is enough to perform a local transformation to the output. This way, we enjoy the unpredictability under attack (and thus the correctness-check can be performed over the larger field) and we obtain correct shares of the output in $\mathbb{F}$.

3 Preliminaries

3.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables, lowercase for values and functions, and boldface for vectors. All logarithms considered here are in base 2. For a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ and a set $\mathcal{I} \subseteq [n]$, let $\boldsymbol{v}_{\mathcal{I}}$ be the *ordered sequence* $(v_i)_{i \in \mathcal{I}}$, let $\boldsymbol{v}_{-\mathcal{I}} := \boldsymbol{v}_{[n] \setminus \mathcal{I}}$, and let $\boldsymbol{v}_{-i} := \boldsymbol{v}_{-\{i\}}$

to argue about the min-entropy of $\langle \boldsymbol{V}^b, \boldsymbol{d} * \boldsymbol{T} \rangle$ given $(b, \boldsymbol{V}^b)$. We ignore this subtlety in this informal exposition.

(i.e., $(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$). For two vectors $\boldsymbol{u} = (u_1, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_n)$, let $\boldsymbol{u} * \boldsymbol{v} := (u_1 \cdot v_1, \ldots, u_n \cdot v_n)$, and let $\langle \boldsymbol{u}, \boldsymbol{v} \rangle := \sum_{i=1}^{n} u_i v_i$. Let $\boldsymbol{b^n}$ denote the the n-size all b vector, or just $\boldsymbol{b}$ when the size is clear from the context. For a field $\mathbb{F}$ and a sequence of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{F}^n$, let $\text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\} := \{\sum_{j=1}^{m} \lambda_j \boldsymbol{v}_j : \lambda_1, \ldots, \lambda_m \in \mathbb{F}\}$ (i.e., the vector space that is spawn by vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$), and let $\text{rank}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ denote the dimension of $\text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$. For a function f taking $1^\kappa \in \mathbb{N}$ as its first input, we let $f_\kappa(\cdot)$ stand for $f(1^\kappa, \cdot)$. Let PPT stand for probabilistic polynomial time, and PPTM stand for PPT (uniform) algorithm Turing Machine).

3.2 Distributions and Random Variables

The support of a distribution P over a finite set $\mathcal{S}$ is defined by $\text{Supp}(P) := \{x \in \mathcal{S} : P(x) > 0\}$. For a (discrete) distribution D, let $d \leftarrow D$ denote that d is sampled according to D. Similarly, for a set $\mathcal{S}$, let $x \leftarrow \mathcal{S}$ denote that x is drawn uniformly from $\mathcal{S}$. The statistical distance (also known as, variation distance) of two distributions P and Q over a discrete domain $\mathcal{X}$ is defined by $\text{SD}(P, Q) := \max_{\mathcal{S} \subseteq \mathcal{X}} |P(\mathcal{S}) - Q(\mathcal{S})| = \frac{1}{2} \sum_{x \in \mathcal{S}} |P(x) - Q(x)|$. The min-entropy of a distribution P over a discrete domain $\mathcal{X}$ is defined by $H_\infty(P) := \min_{x \in \text{Supp}(P)} \{\log(1/P(x))\}$.

3.3 Two-Party Protocols and Functionalities

A two-party protocol consists of two *interactive* Turing Machines (TMs). In each round, only one party sends a message. At the end of protocol, each party outputs some value. This work focuses on static adversaries: before the beginning of the protocol, the adversary corrupts one of the parties that from now on may arbitrarily deviate from the protocol. Thereafter, the adversary sees the messages sent to the corrupted party and controls its messages. A party is honest, with respect to a given protocol, if it follows the prescribed protocol. A party is semi-honest, if it follows the prescribed protocol, but might output additional values.

We mark inputs to protocols and functionalities as **optional**, if they do not have to be defined by the caller, and in this case they are set to $\perp$.

3.3.1 Security We define the security of our two-party protocols in the *real* vs. *ideal* paradigm [9, 16]. In this paradigm, the *real-world model*, in which protocols is executed, is compared to an *ideal model* for executing the task at hand. The latter model involves a trusted party whose functionality captures the security requirements of the task. The security of the real-world protocol is argued by showing that it "emulates" the ideal-world protocol, in the following sense: for any real-life adversary A, there exists an ideal-model oracle-aided adversary (also known as, simulator) S, such that the global output of an execution of the protocol with A in the real-world model is distributed similarly to the global output of running S^{A} in the ideal model. In the following we only consider *non*-reactivate functionalities, i.e., random functions.

The ideal model. In the ideal execution model, the parties do not interact, but rather make a single joint call to a two-party functionality. An ideal execution of a two-party functionality f with respect to an adversary A taking the role of P_1 and inputs $(1^\kappa, x_1, x_2)$, denoted by $\mathrm{IDEAL}^f_{\mathsf{P}_1}(\mathsf{A}, \kappa, x_1, x_2)$, is the output of A and that of the trusted party, in the following experiment (the case of malicious P_2 is analogously defined):

Experiment 3 (Ideal execution)

1. *On input* $(1^\kappa, x_1)$, A *sends an arbitrary message* $\widehat{x}_1$ *to the trusted party.*
2. *The trusted party computes* $(y_1, y_2) = f(1^\kappa, \widehat{x}_1, x_2)$ *and sends* y_1 *to* $\mathsf{A}(1^\kappa, x_1)$.
3. A *sends the message* Continue/ Abort *to the trusted party, and locally outputs some value.*
4. *If* A *instructs* Abort, *the trusted party outputs* $\perp$. *Otherwise, it outputs* y_2.

The real model. We focus on security of protocols in the *g-hybrid model*, in which the parties are given access to two-party functionality g. In executions of such protocols, a malicious party can instruct the functionality g to abort after seeing its output (which it gets first). Let $\Pi = (\mathsf{P}_1, \mathsf{P}_2)$ be an two-party protocol in the g-hybrid model, and let A be an adversary controlling party P_1 (the case of malicious P_2 is analogously defined). We define $\mathrm{REAL}^{\Pi}_{\mathsf{P}_1}(\mathsf{A}, \kappa, x_1, x_2)$ as the output of A (i.e., without loss of generality its view: its random input, the messages it received, and the output of the g calls) and the prescribed output of P_2, in a random execution of $(\mathsf{A}^g(x_1), \mathsf{P}_2^g(x_2))(1^\kappa)$.

Hybrid-model security.

Definition 1 (α-security). *A two-party protocol* $\Pi = (\mathsf{P}_1, \mathsf{P}_2)$ *(black-boxly)* α-computes a two-party functionality f in the g-hybrid model with respect to input domain $\mathcal{D}_1 \times \mathcal{D}_2$, *if there exists a* PPT *oracle-aided algorithm* S *(simulator), such that for every adversary* A, $\kappa \in \mathbb{N}$ *and inputs* $(x_1, x_2) \in \mathcal{D}_1 \times \mathcal{D}_2$, *it holds that*

$$\mathrm{SD}\Big(\mathrm{REAL}^{\Pi}_{\mathsf{P}_1}(\mathsf{A}, \kappa, x_1, x_2)),\ \mathrm{IDEAL}^f_{\mathsf{P}_1}\big(\mathsf{S}^{\mathsf{A}}, \kappa, x_1, x_2\big)\Big) \leq \alpha(\kappa).$$

Furthermore, if A *is semi-honest then so is* S^{A}*: it sends its (real) input to the trusted party, and does not ask to abort. Security is defined analogously for* P_2.

Extension to UC security. The above security notions are defined in the so-called "standalone" model. However, we mention that the security analysis for our main results (realizing WeakMult and WeakBatch) as well as for our applications (e.g. Realizing PerfectMult from WeakMult and auxiliary "helper" functionalities) uses *straightline simulators* exclusively, i.e., the simulator does not rewind the adversary at any point of the simulation. Therefore, our results can be extended to the UC setting.

3.3.2 Oblivious Transfer (OT) We use the (perfect) one-out-two oblivious transfer functionality (OT) defined as follows: on input (σ_{-1}, σ_1) sent by the first party (the sender), and input $i \in \{-1, 1\}$ sent by the second party (the receiver), it sends σ_i to the receiver. The functionality gets no security parameter.

3.3.3 Two-Party Multiplication In multiplication over the field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, where q is an odd prime, party P_1 holds private input $a \in \mathbb{Z}_q$, party P_2 holds private input $b \in \mathbb{Z}_q$, and the goal is to securely computes random shares $s_1, s_2 \in \mathbb{Z}_q$ for P_1 and P_2 (respectively), such that $s_1 + s_2 = a \cdot b$ (for the ease of notation, we assume that operations are made over the field $\mathbb{Z}_q$, i.e., modulo q). The following is what we address as the *perfect multiplication functionality*.

Functionality 4 (PerfectMult)
P_1*'s input:* $a \in \mathbb{Z}_q$.
P_2*'s input:* $b \in \mathbb{Z}_q$ *and* ***optional*** $s_2 \in \mathbb{Z}_q$.
Operation:

1. *If* $s_2 = \perp$, *sample* $s_2 \leftarrow \mathbb{Z}_q$.
2. *Output* (s_1, s_2) *for* $s_1 \leftarrow a \cdot b - s_2$.

Note that it is always holds that $s_1 + s_2 = a \cdot b$. Also note that an adversary controlling P_1 can do no harm, and adversary controlling party P_2 may choose the value of its share s_2, but no information about the other party's input is leaked. It seems that allowing one party to control its output is unavoidable, and is also harmless fo all the applications we are aware of.

3.3.4 Batching In a *batch-multiplication*, a single input provided by one party is multiplied with several inputs provided by the other party. Such multiplication is interesting if the batching is more efficient than parallel executions of the (single input per party) multiplication protocol. For this case, we define the *perfect batch-multiplication functionality* below.

Functionality 5 (PerfectMultBatching)
P_1*'s input:* $a \in \mathbb{Z}_q$.
P_2*'s input:* $\boldsymbol{b} = (b_1, \ldots, b_m) \in \mathbb{Z}_q^m$ *and* ***optional*** $(s_2^1, \ldots, s_2^m) \in \mathbb{Z}_q^m$.
Operation:

1. *If* $(s_2^1, \ldots, s_2^m) = \perp$, *sample* $(s_2^1, \ldots, s_2^m) \leftarrow \mathbb{Z}_q^m$.
2. *Output* $(s_1^1, \ldots, s_1^m)$ *to* P_1 *and* $(s_2^1, \ldots, s_2^m)$ *to* P_2 *for* $(s_1^1, \ldots, s_1^m) \leftarrow a \cdot \boldsymbol{b} - (s_2^1, \ldots, s_2^m)$.

3.4 Some Inequalities

We use the following inequalities.

Lemma 1 (Chebyshev's inequality). *Let X be a random variable with $\mathrm{E}[X] \in (-\infty, \infty)$ and $\mathrm{Var}(X) \in (0, \infty)$. Then*

$$\forall k > 0 : \quad \Pr[|X - \mathrm{E}[X]| \geq k] \leq \mathrm{Var}(X)/k^2.$$

Definition 2 (Universal hash functions). *A family $\mathcal{H} = \{h \colon \mathcal{D} \to \mathcal{R}\}$ of (hash) functions is called* universal *if for every $x, y \in \mathcal{D}$ with $x \neq y$,*

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(y)] \leq 1/|\mathcal{R}|.$$

Lemma 2 (The leftover hash lemma [18]). *Let X be a random variable over a universe $\mathcal{D}$, let $\mathcal{H} = \{h \colon \mathcal{D} \to \mathcal{R}\}$ be a universal hash family. Then for $H \leftarrow \mathcal{H}$ it holds that*

$$\mathrm{SD}((H, H(X)),\ (H, U)) \leq 2^{-(H_\infty(X) - \log|\mathcal{R}|)/2},$$

where $U \leftarrow \mathcal{R}$ (independent of H).

The following lemma is similar both in statement and proof to [19, Lemma 1]. It states that for a uniform universal hash function H conditioned on its output for a uniform input X, does not affect its distribution by much. This is in a sense the converse of the leftover hash lemma that states that $(H, H(X))$ is close to uniform. For simplicity, we only state the lemma for the inner-product hash family.

Lemma 3. *Let $(\mathcal{R}, +, \cdot)$ be a finite ring of size r, let $n = \lceil \log r \rceil + \kappa$, let $\boldsymbol{d} \in \mathcal{R}^n$, let $\ell = \mathrm{dist}(\boldsymbol{d}, 0^n)$ and let $\boldsymbol{V} \leftarrow \mathcal{R}^n$ and $\boldsymbol{T} \leftarrow \{-1, 1\}^n$ be two independent random variables. Then for every $x \in \mathcal{R}$ it holds that:*

$$\mathrm{SD}(\boldsymbol{V}, \boldsymbol{V}|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = x}) \leq 2^{-(\kappa - 1)/2}.$$

The proof of the above can be found in the full-version of this paper [17].

4 Multiplication with Unpredictable Output Under Attack

In this section, we formally describe our "weak" OT-based multiplication protocol introduced in Section 1; we state and analyze its security guarantee. We show that our protocol securely realizes a multiplication functionality that guarantees *unpredictable honest-party output under attack*, which, for lack of a better short name, we will address as WeakMult. Intuitively, WeakMult allows the adversary to either act honestly, or to induce an unpredictable offset on the honest party's output. As discussed in the introduction, such a security guarantee suffices in many settings where "secure multiplication" is needed, and, with some additional effort (see Section 6), can be compiled into perfect i.e., standard multiplication.

In Section 4.1, we define the WeakMult functionality and analyze the security guarantee it provides. In Section 4.2, we formally define our OT-based multiplication protocol, and we prove that it securely realizes WeakMult. Hereafter, we fix $q \in \mathsf{PRIMES}_{>2}$ (i.e., the size of the field), and all arithmetic operations are done over the field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ (i.e., modulo q). Let $\mathrm{Ham}(\boldsymbol{x}, \boldsymbol{y})$ stand for the hamming distance between the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

4.1 The Ideal Functionality

We start by describing the ideal functionality WeakMult. Recall that PerfectMult is the perfect (standard) multiplication functionality defined in Section 3.3.3.

Definition 3 (polychromatic vector). *A vector $\boldsymbol{d} \in \mathbb{Z}_q^n$ is m-*polychromatic *if for every $y \in \mathbb{Z}_q$ it holds that $\mathrm{Ham}(\boldsymbol{d}, y^n) \geq m$.*

Functionality 6 (WeakMult)
Common input: a security parameter 1^κ. Let $n = \lceil \log q \rceil + \kappa$.
*P_1's input: $a \in \mathbb{Z}_q$, and **optional** $\boldsymbol{d} \in \mathbb{Z}_q^n$.*
*P_2's input: $b \in \mathbb{Z}_q$, and **optional** $s_2 \in \mathbb{Z}_q$.*
Operation:

*If $\boldsymbol{d}$ is **not** $\kappa/2$-polychromatic (or $\boldsymbol{d} = \perp$), act according to* PerfectMult$(a, (b, s_2))$.

Else:

1. *Sample $(\boldsymbol{v}, \boldsymbol{t}) \leftarrow \mathbb{Z}_q^n \times \{-1, 1\}^n$ such that $\langle \boldsymbol{v}, \boldsymbol{t} \rangle = b$.* [16]
2. *Sample $s_2 \leftarrow \mathbb{Z}_q$.*
3. *Output $((s_1, \boldsymbol{v}),\ s_2)$ for $s_1 = a \cdot b - s_2 + \langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle$.*

It is clear that WeakMult outputs the shares of $a \cdot b$ correctly on a non $\kappa/2$-polychromatic $\boldsymbol{d}$. The following lemma states the security guarantee of WeakMult against a "cheating" P_1 that uses a $\kappa/2$-polychromatic vector $\boldsymbol{d}$.

Lemma 4. *Let $q \in \mathsf{PRIMES}_{>2}$, $\kappa \in \mathbb{N}$ and $n := \lceil \log q \rceil + \kappa$. Let $\boldsymbol{d} \in \mathbb{Z}_q^n$, let $\ell = \min_{y \in \mathbb{Z}_q} \{\mathrm{Ham}(\boldsymbol{d}, y^n)\}$, let $\lambda := \min\{\ell, \kappa - 5, \log q, n/3\}$, and let $(\boldsymbol{V}, \boldsymbol{T}) \leftarrow \mathbb{Z}_q^n \times \{-1, 1\}^n$. Then for every $b \in \mathbb{Z}_q$, with probability $1 - 2^{-\lambda/2+3}$ over $\boldsymbol{v} \leftarrow \boldsymbol{V}|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}$, it holds that*

$$\mathrm{H}_\infty(\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{T} \rangle \mid \langle \boldsymbol{v}, \boldsymbol{T} \rangle = b) \geq \lambda/2 - 4.$$

When $\lambda \geq \kappa/2$ (by the definition of λ this happens when the field is not too small), for a $\kappa/2$-polychromatic $\boldsymbol{d}$, Lemma 4 yields that for such $\boldsymbol{d}$, conditioned on $\langle \boldsymbol{v}, \boldsymbol{T} \rangle = b$, the min-entropy of $\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{T} \rangle$ is at least $\kappa/4 - 4$. The rather tedious proof of Lemma 4 is given in the full version of this paper [17]. Below, we state and prove a weaker, but easier to read, variant.

Lemma 5 (A weak variant of Lemma 4). *Let $\kappa, n, \boldsymbol{d}, \ell, \boldsymbol{V}, \boldsymbol{T}$ be as in Lemma 4, and let $\lambda := \min\{\ell, \kappa, \log q, n/3\}$. Then for any $b \in \mathbb{Z}_q$, with probability $1 - 2^{-\lambda/3+2}$ over $\boldsymbol{v} \leftarrow \boldsymbol{V}|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}$, it holds that*

$$\mathrm{H}_\infty(\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{T} \rangle \mid \langle \boldsymbol{v}, \boldsymbol{T} \rangle = b) \geq \lambda/3 - 4.$$

In words, compared to Lemma 4, Lemma 5 yields a slightly smaller min-entropy guarantee which occurs with a slightly smaller probability.

[16] This sampling can be done efficiently by sampling the two item uniformly, and then adjusting one coordinate of $\boldsymbol{v}$.

Proof. We assume without loss of generality that

$$\operatorname*{argmax}_{x\in\mathbb{Z}_q}|\{i\in[n]\colon d_i=x\}|=0,$$

i.e., 0 is the most common element in $\boldsymbol{d}$. (Otherwise, we prove the lemma for the vector $\boldsymbol{d}'=\boldsymbol{d}-y^n$, where $y\in\mathbb{Z}_q$ be the most common element). We also assume that $\boldsymbol{d}$ is not the all-zero vector, as otherwise the proof trivially holds.

Let $\kappa, n, \boldsymbol{d}, \ell, \lambda, \boldsymbol{V}, \boldsymbol{T}$ be as in Lemma 4, and fix $b\in\mathbb{Z}_q$. In addition, for $\boldsymbol{t}\in\{-1,1\}^n$, let $W^{\boldsymbol{t}}$ be the indicator random variable for the event $\{\langle\boldsymbol{V},\boldsymbol{t}\rangle=b\}$, and let $W:=\sum_{\boldsymbol{t}\in\{-1,1\}^n}W^{\boldsymbol{t}}$. For $\boldsymbol{t}\in\{-1,1\}^n$ and $x\in\mathbb{Z}_q$, let $Z_x^{\boldsymbol{t}}$ be the indicator random variable for the event $\{\langle\boldsymbol{V},\boldsymbol{t}\rangle=b\ \wedge\ \langle\boldsymbol{V},\boldsymbol{d}*\boldsymbol{t}\rangle=x\}$, and let $Z_x:=\sum_{\boldsymbol{t}\in\{-1,1\}^n}Z_x^{\boldsymbol{t}}$. We start by proving that with high probability over $\boldsymbol{V}$, for every $x\in\mathbb{Z}_q$, it holds that

$$Z_x/W\le 2^{-\lambda/3+4} \tag{5}$$

and we will complete the proof of the lemma by showing that the above inequality still holds when defining Z_x and W with respect to the random variable $\boldsymbol{V}^b:=\boldsymbol{V}|_{\langle\boldsymbol{V},\boldsymbol{T}\rangle=b}$ (rather than with respect to $\boldsymbol{V}$). We prove Equation (5) by bounding the variance of W and Z_x, and then use Chebyshev's inequality (Lemma 1). Specifically, we use the following claims (proven below).

Claim 7 *For every* $x\in\mathbb{Z}_q$: $\mathrm{E}[Z_x]=2^n/q^2$ *and* $\mathrm{Var}(Z_x)\le 2^{2n-\lambda+4}/q^3$.

Claim 8 $\mathrm{E}[W]=2^n/q$ *and* $\mathrm{Var}(W)\le 2^{n+1}/q$.

By Chebyshev's inequality and Claim 7, for every $x\in\mathbb{Z}_q$:

$$\Pr\Big[\big|Z_x-2^n/q^2\big|\ge 2^{n-\lambda/3+2}/q\Big]\le\frac{q^2\cdot\mathrm{Var}(Z_x)}{2^{2n-2\lambda/3+4}}\le\frac{2^{-\lambda/3}}{q},$$

and thus by a union bound

$$\Pr\Big[\exists x\ \text{ s.t. }\ \big|Z_x-2^n/q^2\big|\ge 2^{n-\lambda/3+2}/q\Big]\le 2^{-\lambda/3}. \tag{6}$$

Applying Chebyshev's inequality with respect to Claim 8, we get that

$$\Pr\big[W\le 2^{n-1}/q\big]\le\Pr\big[|W-2^n/q|\ge 2^{n-1}/q\big]\le\frac{q^2\cdot\mathrm{Var}(W)}{2^{2n-2}}\le 2^{-\kappa+3}, \tag{7}$$

where the last inequality holds since, be definition, $n\ge\log q+\kappa$. Combining Equations (6) and (7) yields that with probability at least $1-(2^{-\lambda/3}+2^{-\kappa+3})\ge 1-2^{-\lambda/3+1}$ over $\boldsymbol{v}\leftarrow\boldsymbol{V}$, it holds that:

1. $\forall x\in\mathbb{Z}_q:\ Z_x\le 2^{n-\lambda/3+3}/q$, and
2. $W\ge 2^{n-1}/q$.

Note that for every $\boldsymbol{v}$ satisfying Items 1 and 2, and every $x \in \mathbb{Z}_q$, it holds that

$$\begin{aligned}\Pr[\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{T} \rangle = x \mid \langle \boldsymbol{v}, \boldsymbol{T} \rangle = b] &= \frac{\Pr[\langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{T} \rangle = x \wedge \langle \boldsymbol{v}, \boldsymbol{T} \rangle = b]}{\Pr[\langle \boldsymbol{v}, \boldsymbol{T} \rangle = b]} \\ &= \frac{Z_x}{W}|_{\boldsymbol{V}=\boldsymbol{v}} \\ &\leq 2^{-\lambda/3+4}.\end{aligned} \tag{8}$$

We now turn to the distribution $\boldsymbol{V}^b = \boldsymbol{V}|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}$. Applying Lemma 3 with respect to the ring $\mathcal{R} = \mathbb{Z}_q$ with addition and multiplication modulo q, yields that

$$\mathrm{SD}(\boldsymbol{V}, \boldsymbol{V}^b) \leq 2^{-(\kappa-1)/2} \tag{9}$$

It follows that Equation (8) holds with probability at least $1 - 2^{-\lambda/3+1} - 2^{-(\kappa-1)/2} \geq 1 - 2^{-\lambda/3+2}$ over $\boldsymbol{v} \leftarrow \boldsymbol{V}^b$, as required.

4.1.1 Proving Claim 8

Proof. Recall that $W := \sum_{\boldsymbol{t} \in \{-1,1\}^n} W^{\boldsymbol{t}}$ for $W^{\boldsymbol{t}}$ being the indicator random variable for the event $\{\langle \boldsymbol{V}, \boldsymbol{t} \rangle = b\}$. Therefore, it is clear that $\mathrm{E}[W] = 2^n/q$, and a simple calculation yields that

$$\begin{aligned}\mathrm{Var}(W) &= \mathrm{Var}\left(\sum_{\boldsymbol{t} \in \{-1,1\}^n} W^{\boldsymbol{t}}\right) \\ &= \sum_{\boldsymbol{t} \in \{-1,1\}^n} (\mathrm{E}[(W^{\boldsymbol{t}} - 1/q)^2] + \mathrm{E}[(W^{\boldsymbol{t}} - 1/q) \cdot (W^{-\boldsymbol{t}} - 1/q)]) \\ &\leq 2 \cdot \sum_{\boldsymbol{t} \in \{-1,1\}^n} \mathrm{Var}(W^{\boldsymbol{t}}) \\ &\leq 2^{n+1}/q,\end{aligned} \tag{10}$$

as required. The second equality holds since for every $\boldsymbol{t}, \boldsymbol{t}'$ with $\boldsymbol{t}' \notin \{-\boldsymbol{t}, \boldsymbol{t}\}$, the random variables $W^{\boldsymbol{t}}$ and $W^{\boldsymbol{t}'}$ are independent (because $\boldsymbol{t}$ and $\boldsymbol{t}'$ are linearly independent).

4.1.2 Proving Claim 7 Recall that $Z_x := \sum_{\boldsymbol{t} \in \{-1,1\}^n} Z_x^{\boldsymbol{t}}$ for $Z_x^{\boldsymbol{t}}$ being the indicator random variable for the event $\{\langle \boldsymbol{V}, \boldsymbol{t} \rangle = b \wedge \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{t} \rangle = x\}$. For any $\boldsymbol{t} \in \{-1,1\}^n$, since the vectors $\boldsymbol{t}$ and $\boldsymbol{d} * \boldsymbol{t}$ are linearly independent (recall that $\boldsymbol{d}$ contains zero and non-zero elements) it holds that $\mathrm{E}[Z_x^{\boldsymbol{t}}] = 1/q^2$, and therefore, $\mathrm{E}[Z_x] = 2^n/q^2$. It is left to bound $\mathrm{Var}(Z_x)$. For $j \in [4]$, let

$$\mathcal{B}_j := \{(\boldsymbol{t}, \boldsymbol{t}') \in \{-1,1\}^{2n} \colon \mathrm{rank}\{\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{d} * \boldsymbol{t}, \boldsymbol{d} * \boldsymbol{t}'\} = j\}$$

Note that the only possible values for $\mathrm{E}[Z_x^{\boldsymbol{t}} \cdot Z_x^{\boldsymbol{t}'}]$ are $\{0\} \cup \{1/q^j\}_{j=1}^4$, where $\mathrm{E}[Z_x^{\boldsymbol{t}} \cdot Z_x^{\boldsymbol{t}'}] = 1/q^j \implies (\boldsymbol{t}, \boldsymbol{t}') \in \mathcal{B}_j$. We relate $\mathrm{Var}\left(\sum_{\boldsymbol{t} \in \{-1,1\}^n} Z_x^{\boldsymbol{t}}\right)$ to size $\{\mathcal{B}_j\}$ as follows:

$$\begin{aligned} \mathrm{Var}(Z_x) &= \sum_{\boldsymbol{t}, \boldsymbol{t}' \in \{-1,1\}^n} \mathrm{E}[(Z_x^{\boldsymbol{t}} - 1/q^2)(Z_x^{\boldsymbol{t}'} - 1/q^2)] \\ &\leq \sum_{\boldsymbol{t}, \boldsymbol{t}' \in \{-1,1\}^n} \mathrm{E}[Z_x^{\boldsymbol{t}} \cdot Z_x^{\boldsymbol{t}'}] \\ &\leq \sum_{j=1}^{4} |\mathcal{B}_j|/q^j. \end{aligned} \tag{11}$$

We complete the proof by bounding the size of $\mathcal{B}_j$ for each $j \in [3]$ (for $\mathcal{B}_4$ we use the trivial bound $|\mathcal{B}_4| \leq 2^{2n}$).

Claim 9 $|\mathcal{B}_1| = 0$.

Proof. Since $\boldsymbol{d}$ contains zeros and non-zeros elements, the vectors $\boldsymbol{t}$ and $\boldsymbol{d} * \boldsymbol{t}$, for any $\boldsymbol{t} \in \{-1,1\}^n$, are linearly independent over $\mathbb{Z}_q^n$, yielding that $|\mathcal{B}_1| = 0$.

Claim 10 $|\mathcal{B}_2| \leq 2^{n+2}$.

Proof. Since there are exactly 2^{n+1} linearly dependent pairs $(\boldsymbol{t}, \boldsymbol{t}')$, i.e., the pairs $\cup_{\boldsymbol{t} \in \{-1,1\}^n} \{(\boldsymbol{t}, \boldsymbol{t}), (\boldsymbol{t}, -\boldsymbol{t})\}$, we deduce the bound by proving that there are at most 2^{n+1} independent pairs $(\boldsymbol{t}, \boldsymbol{t}')$ in $\mathcal{B}_2$.

Fix an independent pair $(\boldsymbol{t}, \boldsymbol{t}') \in \mathcal{B}_2$, let $\mathcal{E} = \{i \in [n] \colon t_i = t'_i\}$ and let $\mathcal{N} = [n] \setminus \mathcal{E}$. Up to reordering of the coordinates, we can write $\boldsymbol{t} = (\boldsymbol{t}_\mathcal{E}, \boldsymbol{t}_\mathcal{N})$, $\boldsymbol{t}' = (\boldsymbol{t}_\mathcal{E}, -\boldsymbol{t}_\mathcal{N})$ and $\boldsymbol{d} = (\boldsymbol{d}_\mathcal{E}, \boldsymbol{d}_\mathcal{N})$. It is easy to verify that

$$\mathrm{span}\{\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{d} * \boldsymbol{t}, \boldsymbol{d} * \boldsymbol{t}'\} = \mathrm{span}\{(\boldsymbol{t}_\mathcal{E}, \boldsymbol{0}), (\boldsymbol{0}, \boldsymbol{t}_\mathcal{N}), (\boldsymbol{d}_\mathcal{E} * \boldsymbol{t}_\mathcal{E}, \boldsymbol{0}), (\boldsymbol{0}, \boldsymbol{d}_\mathcal{N} * \boldsymbol{t}_\mathcal{N})\}.$$

Since $(\boldsymbol{t}, \boldsymbol{t}')$ are independent and $\mathrm{rank}\{\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{d} * \boldsymbol{t}, \boldsymbol{d} * \boldsymbol{t}'\} = 2$, the above yields that

$$\boldsymbol{d}_\mathcal{E} \in \mathrm{span}\{\boldsymbol{1}\} \ \wedge \ \boldsymbol{d}_\mathcal{N} \in \mathrm{span}\{\boldsymbol{1}\} \tag{12}$$

Since, by assumption, $\boldsymbol{d}$ is $\boldsymbol{d}$ is not the all-zero vector, Equation (12) yields that $(\boldsymbol{d}_\mathcal{E}, \boldsymbol{d}_\mathcal{N}) = (u \cdot \boldsymbol{1}, \boldsymbol{0})$ or $d = (\boldsymbol{0}, u \cdot \boldsymbol{1})$, for some $u \in \mathbb{Z}_q \setminus \{0\}$.

Assuming that $\mathcal{B}_2$ contains an independent pair, otherwise we are done, the above yields that the non-zero coordinates of $\boldsymbol{d}$ are all equal to some $u \in \mathbb{Z}_q \setminus \{0\}$. It follows that for each vector $\boldsymbol{t} \in \{-1,1\}^n$ there are at most *two* vectors $\boldsymbol{t}^1$ and $\boldsymbol{t}^2$, such that $(\boldsymbol{t}, \boldsymbol{t}^j)$ is an independent pair in $\mathcal{B}_2$ (actually, each $\boldsymbol{t}$ has exactly two such vectors, with $\boldsymbol{t}^1 = -\boldsymbol{t}^2$). We conclude that the number of independent pairs $(\boldsymbol{t}, \boldsymbol{t}') \in \mathcal{B}_2$ is at most 2^{n+1}.

Claim 11 $|\mathcal{B}_3| \leq 2^{2n - \min\{n/3, \ell\} + 2}$ *(recall that* $\ell = \mathrm{Ham}(\boldsymbol{d}, \boldsymbol{0})$*).*

Proof. Let $\mu := \min\{n/3, \ell\}$, fix $(\boldsymbol{t}, \boldsymbol{t}') \in \mathcal{B}_3$, let $\mathcal{E} = \{i \in [n] \colon t_i = t'_i\}$ and let $\mathcal{N} = [n] \setminus \mathcal{E}$. Up to reordering of the coordinates, we can write $\boldsymbol{t} = (\boldsymbol{t}_\mathcal{E}, \boldsymbol{t}_\mathcal{N})$, $\boldsymbol{t}' = (\boldsymbol{t}_\mathcal{E}, -\boldsymbol{t}_\mathcal{N})$ and $\boldsymbol{d} = (\boldsymbol{d}_\mathcal{E}, \boldsymbol{d}_\mathcal{N})$. It holds that

$$\operatorname{span}\{\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{d} * \boldsymbol{t}, \boldsymbol{d} * \boldsymbol{t}'\} = \operatorname{span}\{(\boldsymbol{t}_\mathcal{E}, \boldsymbol{0}), (\boldsymbol{0}, \boldsymbol{t}_\mathcal{N}), (\boldsymbol{d}_\mathcal{E} * \boldsymbol{t}_\mathcal{E}, \boldsymbol{0}), (\boldsymbol{0}, \boldsymbol{d}_\mathcal{N} * \boldsymbol{t}_\mathcal{N})\}.$$

Since the assumed dimension is 3, then

$$\boldsymbol{d}_\mathcal{E} \in \operatorname{span}\{\boldsymbol{1}\} \ \vee \ \boldsymbol{d}_\mathcal{N} \in \operatorname{span}\{\boldsymbol{1}\} \tag{13}$$

We next show how to partition the coordinates of $\boldsymbol{d}$ into sets $\mathcal{I}_0$ and $\mathcal{I}_1$, each of size at least μ, such that for all $i \in \mathcal{I}_0$ it holds that $d_i \notin \{d_j : j \in \mathcal{I}_1\}$ and vice versa. If $\ell \leq n - \mu$, then we are done by taking $\mathcal{I}_0 = \{i \colon d_i = 0\}$ and $\mathcal{I}_1 = [n] \setminus \mathcal{I}_0$. Assume that $\ell > n - \mu$, which implies that $\mu \leq n - 2\mu < 2\ell - n$. For $\alpha \in \mathbb{Z}_q$ define $\mathcal{J}_\alpha = \{i \colon d_i = \alpha\}$ and notice that $|\mathcal{J}_\alpha| < (n - \mu)/2$ because otherwise

$$|\mathcal{J}_\alpha| \geq (n - \mu)/2 > (n - (2\ell - n))/2 = n - \ell$$

which contradicts the definition of ℓ (recall that 0 is the element with maximal number of appearances in $\boldsymbol{d}$, and there are exactly $n - \ell$ zero coordinates). Finally, define $s \in \mathbb{Z}_q$ to be the minimal value such that $\cup_{\alpha=0}^{s} \mathcal{J}_\alpha \geq \mu$ and let $\mathcal{I}_0 = \cup_{\alpha=0}^{s} \mathcal{J}_\alpha$ and $\mathcal{I}_1 = [n] \setminus \mathcal{I}_0$. By definition, $\mathcal{I}_0$ is bigger than μ and it remains to show that $\mathcal{I}_1 \geq \mu$. It holds that

$$|\mathcal{I}_1| = n - |\mathcal{I}_0| = n - \left|\cup_{\alpha=0}^{s-1} \mathcal{J}_\alpha\right| - |\mathcal{J}_s| \geq n - \mu - (n - \mu)/2 \geq \mu.$$

Back to the proof, Equation (13) yields that either $\mathcal{E} \subseteq \mathcal{I}_0$, or $\mathcal{E} \subseteq \mathcal{I}_1$, or $\mathcal{N} \subseteq \mathcal{I}_0$, or $\mathcal{N} \subseteq \mathcal{I}_1$. Since $|\mathcal{I}_0|, |\mathcal{I}_1| \geq \mu$, the number of pairs $(\boldsymbol{t}, \boldsymbol{t}') \in \{-1, 1\}^n$ that satisfy this condition is at most $4 \cdot 2^{2n-\mu}$, which ends the proof of the claim.

Putting it together. Given the above claims, we are ready to prove Claim 7.

Proof (Proof of Claim 7). Recall that $\lambda := \min\{\ell, \kappa, \log q, n/3\}$. By Equation (11) and Claims 9 to 11, we conclude that

$$\begin{aligned}
\operatorname{Var}(Z_x) &\leq \sum_{j=1}^{4} |\mathcal{B}_j|/q^j \\
&\leq 2^{n+2}/q^2 + 2^{2n-\lambda+2}/q^3 + 2^{2n}/q^4 \\
&\leq 2^{2n-\lambda+4}/q^3,
\end{aligned}$$

as required. The last inequality holds since $\lambda \leq \kappa$ implies that $2^{n+2}/q^2 \leq 2^{2n-\lambda+2}/q^3$, and $\lambda \leq \log q$ implies that $2^{2n}/q^4 \leq 2^{2n-\lambda+2}/q^3$.

4.2 The OT-Based Protocol

In the following we describe our OT-based implementation of the functionality WeakMult. Recall that throughout this section we fix a field size $q > 2$ and assume that all operation are made over the field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ (i.e., modulo q).

Protocol 12 ($\Pi = (\mathsf{P}_1, \mathsf{P}_2)$)
Oracle: (one-out-of-two) OT.
Common input: security parameter 1^κ. *Let* $n = \lceil \log q \rceil + \kappa$.
P_1*'s private input:* $a \in \mathbb{Z}_q$.
P_2*'s private input:* $b \in \mathbb{Z}_q$.
Operations:

1. *For each* $i \in [n]$, *in parallel:*
   (a) P_1 *samples* $\delta_i \leftarrow \mathbb{Z}_q$, *and* P_2 *samples* $t_i \leftarrow \{-1, 1\}$.
   (b) *The parties jointly call* $\mathrm{OT}((\delta_i - a, \delta_i + a), t_i)$.
   Let z_i *be the output obtained by* P_2 *in this call.*
2. P_2 *samples* $\boldsymbol{v} \leftarrow \mathbb{Z}_q^n$ *such that* $\langle \boldsymbol{v}, (t_1, \ldots, t_n) \rangle) = b$, *samples* $\sigma \leftarrow \mathbb{Z}_q$, *and sends* $(\boldsymbol{v}, \sigma)$ *to* P_1.
3. P_1 *outputs* $s_1 := -\langle \boldsymbol{v}, \boldsymbol{\delta} \rangle - \sigma$.
4. P_2 *outputs* $s_2 := \langle \boldsymbol{v}, (z_1, \ldots, z_n) \rangle + \sigma$.

Note that, unlike in the simplified version of the protocol presented in the introduction, party P_2 in the above adds an additional mask σ to the shares. The role of this additional mask is rather technical, but it appears necessary for simulating of the above protocol using WeakMult (Functionality 6).

Lemma 6 (Security). *Protocol 12* ($\alpha(\kappa) := 2^{-\kappa/4+1.5}$)*-computes* WeakMult *in the* OT*-hybrid model with respect to input domain* $\mathbb{Z}_q \times \mathbb{Z}_q$. *Furthermore, if both parties act honestly, then their joint output equals the output of* WeakMult *on their joint inputs.*

Proof. We start with proving correctness (correct output when acting honestly). Indeed, for any possible values of $a, b, \kappa, s_2, \boldsymbol{\delta} = (\delta_1, \ldots, \delta_n), \boldsymbol{t} = (t_1, \ldots, t_n), \boldsymbol{z} = (z_1, \ldots, z_n), \boldsymbol{v}$ an σ in a honest execution of $\Pi(a, b)(1^\kappa)$, it holds that

$$s_2 = \langle \boldsymbol{v}, \boldsymbol{z} \rangle + \sigma = \langle \boldsymbol{v}, \boldsymbol{\delta} + a \cdot \boldsymbol{t} \rangle + \sigma = a \cdot \langle \boldsymbol{v}, \boldsymbol{t} \rangle + \langle \boldsymbol{v}, \boldsymbol{\delta} \rangle + \sigma = a \cdot b - s_1,$$

and thus $s_1 + s_2 = a \cdot b$.

For security, fix a security parameter $\kappa \in \mathbb{N}$ and inputs $a, b \in \mathbb{Z}_q$.

We only prove security for corrupted P_1 (the proof for corrupted P_2 is straightforward and can be found in the full version of this paper [17]).

Corrupted P_1: Given an oracle access to (the next-message function of) an interactive adversary A controlling P_1, its ideal-model simulator S, which uses the functionality WeakMult, is described as follows:

Algorithm 13 (Ideal-model S)
Inputs: 1^κ *and* $a \in \mathbb{Z}_q$.
Oracles: (real-model) attacker A.
Operations:

1. *Simulate a random execution of* $(\mathsf{A}(a), \mathsf{P}_2(0))(1^\kappa)$ *till the end of Step 1.*

2. *If the simulation ends prematurely (e.g., on invalid behavior), send* Abort *to* $\mathsf{WeakMult}_\kappa$*, output* A*'s output and halt the execution.*
3. *Let* (w_i^-, w_i^+) *and* t_i *denote the inputs that* A *and* P_2 *use (respectively) in the* i^{th} OT *execution of the simulation (Step 1b). Let* $a_i = (w_i^+ - w_i^-) \cdot 2^{-1}$ *(where* 2^{-1} *stands for the inverse of* 2 *in* $\mathbb{Z}_q$*), let* $\boldsymbol{a} = (a_1, \ldots, a_n)$*, let* $\boldsymbol{\delta} = (w_1^+ - a_1, \ldots, w_n^+ - a_n)$*, let* $\hat{a} \in \mathbb{Z}_q$ *denote the value that appears the most often in* $\boldsymbol{a}$*, and let* $\boldsymbol{d} = \boldsymbol{a} - \hat{a} \cdot \mathbf{1}$*.*
4. *If* $\mathrm{Ham}(\boldsymbol{d}, 0^n) < \kappa/2$*:*
   (a) *Send* $(\hat{a}, \boldsymbol{d})$ *to* $\mathsf{WeakMult}_\kappa$*.*
   (b) *Receive* s_1 *from* $\mathsf{WeakMult}_\kappa$*.*
   (c) *Sample* $\boldsymbol{v} \leftarrow \mathbb{Z}_q^n$ *such that* $\langle \boldsymbol{v}, (t_1, \ldots, t_n) \rangle = 0$*, and send* $(\boldsymbol{v}, \sigma := -\langle \boldsymbol{v}, \boldsymbol{\delta} \rangle - \langle \boldsymbol{v}, \boldsymbol{d} * \boldsymbol{t} \rangle - s_1)$ *to* A*.*
5. *Else:*
   (a) *Send* $(\hat{a}, \boldsymbol{d})$ *to* $\mathsf{WeakMult}_\kappa$*.*
   (b) *Receive* $(s_1, \hat{\boldsymbol{v}})$ *from* $\mathsf{WeakMult}_\kappa$*.*
   (c) *Send* $(\hat{\boldsymbol{v}}, \sigma := -s_1 - \langle \hat{\boldsymbol{v}}, \boldsymbol{\delta} \rangle)$ *to* A*.*
6. *Output* A*'s output in the simulation.*

It is clear that S is efficient. We next bound the statistical distance between $\mathrm{REAL}^{\Pi}_{\mathsf{P}_1}(\mathsf{A}, \kappa, a, b)$ and $\mathrm{IDEAL}^{\mathsf{WeakMult}}_{\mathsf{P}_1}(\mathsf{S}^{\mathsf{A}}, \kappa, a, b)$. Assuming without loss of generality that A is deterministic (a randomized adversary is just a convex combination of deterministic adversaries), the values of $\boldsymbol{d}$, $\hat{a}$ and $\boldsymbol{\delta}$ that it uses are fixed, and it either uses an $\kappa/2$-polychromatic $\boldsymbol{d}$, or not (i.e., an almost all-zeros $\boldsymbol{d}$). We handle each of these cases separately. In the following let $\boldsymbol{V} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{T} \leftarrow \{-1, 1\}^n$ and $S_1 \leftarrow \mathbb{Z}_q$ be independent random variables.

Polychromatic $\boldsymbol{d}$. If A uses an $\kappa/2$-polychromatic $\boldsymbol{d}$, then $\mathrm{REAL}^{\Pi}_{\mathsf{P}_1}(\mathsf{A}, \kappa, a, b)$, the view of A and the output of P_2 in the real execution $(\mathsf{A}(a), \mathsf{P}_2(b))(1^\kappa)$, are jointly distributed according to

$$((\boldsymbol{V}, -S_1 - \langle \boldsymbol{V}, \boldsymbol{\delta} \rangle),\ \hat{a} \cdot b - S_1 + \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b} \tag{14}$$

Let $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{t}})$ be the pair that is sampled in Step 1 of $\mathsf{WeakMult}_\kappa$. Since this pair is sampled according to $(\boldsymbol{V}, \boldsymbol{T})|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}$, in the ideal execution it holds that $\mathrm{IDEAL}^{\mathsf{WeakMult}}_{\mathsf{P}_1}(\mathsf{S}^{\mathsf{A}}, \kappa, a, b)$ (A's view and the output of the trusted party in the ideal execution) are jointly distributed according to Equation (14). This concludes the proof of this case.

Almost-monochromatic $\boldsymbol{d}$. Assume A uses a non $\kappa/2$-polychromatic vector $\boldsymbol{d}$, i.e., ℓ, the hamming distance of $\boldsymbol{d}$ from 0^n, is less than $\kappa/2$. In this case, A's view in the real execution, i.e., the pair $(\boldsymbol{v}, \sigma)$, and the output s_2 of P_2, are jointly distributed according to $((\boldsymbol{V}, \Sigma), \hat{a} \cdot b - S_1)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b}$, for $\Sigma = -S_1 - \langle \boldsymbol{V}, \boldsymbol{\delta} \rangle - \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle$. On the other hand, the output of S^{A} and that of the trusted party in the

ideal execution, are jointly distributed according to $((\boldsymbol{V}, \Sigma),\ \hat{a} \cdot b - S_1)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = 0}$ (i.e., now the conditioning is over $\langle \boldsymbol{V}, \boldsymbol{T} \rangle$ equals 0 and not b). Therefore

$$\begin{aligned} &\mathrm{SD}\Big(\mathrm{REAL}^{\Pi}_{\mathsf{P}_1}(\mathsf{A}, \kappa, a, b),\ \mathrm{IDEAL}^{\mathsf{WeakMult}}_{\mathsf{P}_1}(\mathsf{S}^{\mathsf{A}}, \kappa, a, b)\Big) \\ &\quad = \mathrm{SD}\big(((\boldsymbol{V}, \Sigma),\ \hat{a} \cdot b - S_1)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b},\ (((\boldsymbol{V}, \Sigma),\ \hat{a} \cdot b - S_1)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = 0}\big) \\ &\quad \leq \mathrm{SD}\big((\boldsymbol{V},\ \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b},\ (\boldsymbol{V},\ \langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle)|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = 0}\big). \end{aligned} \tag{15}$$

The inequality holds since each pair is a randomized function of $\boldsymbol{V}$ and $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle$ (recall that $\hat{a}, b, \boldsymbol{\delta}$ are fixed, S_1 is independent, and Σ is a function of S_1, $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle$ and $\langle \boldsymbol{V}, \boldsymbol{\delta} \rangle$). Recall that $\ell = \mathrm{Ham}(\boldsymbol{d}, 0^n) < \kappa/2$, and let $\mathcal{I} := \{i \in [n] \colon d_i \neq 0\}$. Since $\langle \boldsymbol{V}, \boldsymbol{d} * \boldsymbol{T} \rangle$ is a deterministic function of $\boldsymbol{V}$ and $\boldsymbol{T}_{\mathcal{I}}$, it suffices to prove that

$$\mathrm{SD}((\boldsymbol{V}, \boldsymbol{T}_{\mathcal{I}})|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = b},\ (\boldsymbol{V}, \boldsymbol{T}_{\mathcal{I}})|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = 0}) \leq 2^{-(\kappa - \ell - 3)/2} \tag{16}$$

Since $\mathcal{I} \subsetneq [n]$, for every $x \in \mathbb{Z}_q$ it holds that

$$(\boldsymbol{V}_{\mathcal{I}}, \boldsymbol{T}_{\mathcal{I}})|_{\langle \boldsymbol{V}, \boldsymbol{T} \rangle = x} \equiv (\boldsymbol{V}_{\mathcal{I}}, \boldsymbol{T}_{\mathcal{I}}) \tag{17}$$

Hence, it suffices to prove that Equation (16) holds for every fixing of $(\boldsymbol{V}_{\mathcal{I}}, \boldsymbol{T}_{\mathcal{I}}) = (\boldsymbol{v}_{\mathcal{I}}, \boldsymbol{t}_{\mathcal{I}})$. Indeed,

$$\begin{aligned} &\mathrm{SD}(\boldsymbol{V}_{-\mathcal{I}}|_{\langle \boldsymbol{V}_{-\mathcal{I}}, \boldsymbol{T}_{-\mathcal{I}} \rangle = x},\ \boldsymbol{V}_{-\mathcal{I}}|_{\langle \boldsymbol{V}_{-\mathcal{I}}, \boldsymbol{T}_{-\mathcal{I}} \rangle = x'}) \\ &\leq \mathrm{SD}(\boldsymbol{V}_{-\mathcal{I}},\ \boldsymbol{V}_{-\mathcal{I}}|_{\langle \boldsymbol{V}_{-\mathcal{I}}, \boldsymbol{T}_{-\mathcal{I}} \rangle = x}) + \mathrm{SD}(\boldsymbol{V}_{-\mathcal{I}},\ \boldsymbol{V}_{-\mathcal{I}}|_{\langle \boldsymbol{V}_{-\mathcal{I}}, \boldsymbol{T}_{-\mathcal{I}} \rangle = x'}) \\ &\leq 2 \cdot 2^{-(\kappa - \ell - 1)/2} \\ &= 2^{-(\kappa - \ell - 3)/2}. \end{aligned}$$

The second inequality holds by applying Lemma 3 with a vector size $\tilde{n} = n - \ell = \lceil \log q \rceil + (\kappa - \ell)$, over the ring $\mathcal{R} = \mathbb{Z}_q$ with addition and multiplication modulo q.

5 Batching

In this section we consider the case that the parties $\hat{\mathsf{P}}_1$ and $\hat{\mathsf{P}}_2$ would like to perform $m > 1$ multiplications, where $\hat{\mathsf{P}}_1$ uses the same input $a \in \mathbb{Z}_q$ and $\hat{\mathsf{P}}_2$ uses different inputs $b_1, \ldots, b_m \in \mathbb{Z}_q$. A naive solution is to perform m independent executions of our single multiplication protocol Π (Protocol 12), where the overall cost is $m \cdot (\log q + \kappa)$ OT calls. In this section we present our batching protocol which performs m such multiplications using only $m \cdot \log q + \kappa$ OT calls, at the cost of relaxing the security requirement. In Section 5.1 we describe the relaxed ideal functionality $\mathsf{WeakBatch}$ that we consider for our batching task, and in Section 5.2 we describe our OT-Based implementation (Protocol 15).

5.1 The Ideal Functionality

In the following we describe the ideal functionality WeakBatch.

Functionality 14 (WeakBatch)
Parameters: Multiplications number $m \in \mathbb{N}$ *and a security parameter* $\kappa \in \mathbb{N}$.
Let $n := \lceil m \cdot \log q \rceil + \kappa$.
$\hat{\mathsf{P}}_1$*'s input:* $a \in \mathbb{Z}_q$, *and* ***optional*** $\boldsymbol{d} \in \mathbb{Z}_q^n$.
$\hat{\mathsf{P}}_2$*'s input:* $\boldsymbol{b} = (b_1, \ldots, b_m) \in \mathbb{Z}_q^m$, *and* ***optional*** $\boldsymbol{s}_2 = (s_2^1, \ldots, s_2^m) \in \mathbb{Z}_q^m$.
Operation:
If $\boldsymbol{d}$ *is* **not** $\kappa/2$*-polychromatic (or* $\boldsymbol{d} = \bot$*), act according to* PerfectMultBatching$(a, (\boldsymbol{b}, \boldsymbol{s}_2))$.

Else:
1. *Sample* $(\boldsymbol{v}^1, \ldots, \boldsymbol{v}^m, \boldsymbol{t}) \leftarrow (\mathbb{Z}_q^n)^m \times \{-1, 1\}^n$ *such that* $\forall i \in [m] : \langle \boldsymbol{v}^i, \boldsymbol{t} \rangle = b_i$.
2. *Sample* $\boldsymbol{s}_2 = (\boldsymbol{s}_2^1, \ldots, \boldsymbol{s}_2^m) \leftarrow \mathbb{Z}_q^m$.
3. *Output* $(\{(s_1^i, \boldsymbol{v}^i)\}_{i=1}^m, \{(s_2^i)\}_{i=1}^m)$ *for* $s_1^i = a \cdot b_i - s_2^i + \langle \boldsymbol{v}^i, \boldsymbol{d} * \boldsymbol{t} \rangle$.

Note that for $m = 1$, WeakBatch is identical to WeakMult (Section 4.1). For $m > 1$, WeakBatch achieves perfect correctness and security whenever $\boldsymbol{d}$ is not $\kappa/2$-polychromatic. In particular, when $\hat{\mathsf{P}}_1$ is honest (i.e., $\boldsymbol{d} = \bot$), the functionality is perfectly secure against a cheating $\hat{\mathsf{P}}_2$. As in WeakMult, the more complicated security guarantee is against a cheating $\hat{\mathsf{P}}_1$, which may use a $\kappa/2$-polychromatic vector $\boldsymbol{d}$.

The security guarantee against a cheating $\hat{\mathsf{P}}_1$ that chooses an $\kappa/2$-polychromatic $\boldsymbol{d}$ is characterized by the following result.

Lemma 7. *Let* $q \in \mathsf{PRIMES}_{>2}$, $\kappa \in \mathbb{N}$, $m \in \mathbb{N}$ *and* $n := \lceil m \cdot \log q \rceil + \kappa$. *Let* $\boldsymbol{d} \in \mathbb{Z}_q^n$, *let* $\ell := \min_{y \in \mathbb{Z}_q} \{\mathrm{Ham}(\boldsymbol{d}, y^n)\}$ *and let* $\lambda := \min\{\ell, \kappa - 5, \log q, n/3\}$. *Let* $(\boldsymbol{V} = (\boldsymbol{V}^1, \ldots, \boldsymbol{V}^m), \boldsymbol{T}) \leftarrow (\mathbb{Z}_q^n)^m \times \{-1, 1\}^n$. *Then for any* $b_1, \ldots, b_m \in \mathbb{Z}_q$, *w.p.* $1 - m \cdot 2^{-\lambda/2+3}$ *over* $\boldsymbol{v} = (\boldsymbol{v}^1, \ldots, \boldsymbol{v}^m) \leftarrow \boldsymbol{V}|_{\forall j \in [m] :\ \langle \boldsymbol{V}^j, \boldsymbol{T} \rangle = b_j}$, *it holds that*

$$\forall i \in [m] \colon \mathrm{H}_\infty(\langle \boldsymbol{v}^i, \boldsymbol{d} * \boldsymbol{T} \rangle \mid \forall j \in [m] : \langle \boldsymbol{v}^j, \boldsymbol{T} \rangle = b_j) \geq \lambda/2 - 4.$$

We remark that the security guarantee that is obtained by Lemma 7 is weaker than m independent calls to WeakMult, i.e., the functionality $\mathsf{WeakMults}_{m,\kappa}((a, \boldsymbol{d}), (\boldsymbol{b}, \boldsymbol{s}_2)) := (\mathsf{WeakMult}_\kappa((a, \boldsymbol{d}), (b_i, s_2^i)))_{i=1}^m$. The reason is that Lemma 7 does not guarantee independence between the m shares of $\hat{\mathsf{P}}_2$. While each share, without knowing the other shares, has high min-entropy, it might be that this is not the case when revealing some of the other shares.

The proof of Lemma 4 is given in the full version of this paper [17].

5.2 The OT-Based Protocol

In the following we describe our OT-based implementation of the functionality WeakBatch. We remind that throughout this section we fix a field size $q \in \mathsf{PRIMES}_{>2}$ and assume that all operation are made over the field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ (i.e., modulo q).

Protocol 15 ($\Gamma = (\hat{\mathsf{P}}_1, \hat{\mathsf{P}}_2)$)
Oracles: One-out-of-two OT protocol OT.
Common inputs: $m \in \mathbb{N}$ *and* 1^κ *for* $\kappa \in \mathbb{N}$. *Let* $n = \lceil m \cdot \log q \rceil + \kappa$.
$\hat{\mathsf{P}}_1$*'s private input:* $a \in \mathbb{Z}_q$.
$\hat{\mathsf{P}}_2$*'s private inputs:* $b_1, \ldots, b_m \in \mathbb{Z}_q$.
Operations:

1. *For each* $i \in [n]$, *in parallel:*
   (a) $\hat{\mathsf{P}}_1$ *samples* $\delta_i \leftarrow \mathbb{Z}_q$, *and* $\hat{\mathsf{P}}_2$ *samples* $t_i \leftarrow \{-1, 1\}$.
   (b) *The parties jointly call* $\mathrm{OT}((\delta_i - a, \delta_i + a), t_i)$. .
   Let z_i *be the output obtained by* $\hat{\mathsf{P}}_2$ *in this call.*
2. $\hat{\mathsf{P}}_2$ *samples* $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^m \leftarrow \mathbb{Z}_q^n$ *such that* $\forall i \in [m] : \langle \boldsymbol{v}^i, \boldsymbol{t} \rangle = b_i$, *samples* $\sigma_1, \ldots, \sigma_m \leftarrow \mathbb{Z}_q$, *and sends* $(\boldsymbol{v}^1, \sigma_1), \ldots, (\boldsymbol{v}^m, \sigma_m)$ *to* $\hat{\mathsf{P}}_1$.
3. $\hat{\mathsf{P}}_1$ *outputs* $(s_1^1, \ldots, s_1^m)$ *for* $s_1^i = -\langle \boldsymbol{v}^i, \boldsymbol{\delta} \rangle - \sigma_i$.
4. $\hat{\mathsf{P}}_2$ *outputs* $(s_2^1, \ldots, s_2^m)$ *for* $s_2^i = \langle \boldsymbol{v}^i, \boldsymbol{z} \rangle + \sigma_i$.

Namely, as in Protocol 12 (single multiplication), $\hat{\mathsf{P}}_1$ samples random values $(\delta_1, \ldots, \delta_n)$ and $\hat{\mathsf{P}}_2$ samples random values $(t_1, \ldots, t_n)$, and the OT calls (i.e., Step 1) are performed the same (except from the fact that in Protocol 15, the value of n is larger than the one used in Protocol 12). But now, in Step 2, instead of sampling a single vector $\boldsymbol{v}$ a single σ, $\hat{\mathsf{P}}_2$ now samples m independent random vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^m$, where each $\boldsymbol{v}^i$ satisfy $\langle \boldsymbol{v}^i, \boldsymbol{t} \rangle = b_i$, and samples m independent random offsets $\sigma_1, \ldots, \sigma_m$ (instead of a single one).

Lemma 8 (Security). *For every* $m \in \mathbb{N}$, $\Gamma_m = (\hat{\mathsf{P}}_1, \hat{\mathsf{P}}_2)(m, \cdot)$ *(Protocol 12)* $(\alpha(\kappa) := 2^{-\kappa/4+1.5})$*-computes* $\mathsf{WeakBatch}_m = \mathsf{WeakBatch}(m, \cdot, \cdot, \cdot)$ *in the* OT*-hybrid model, with respect to input domain* $\mathbb{Z}_q \times \mathbb{Z}_q^m$. *Furthermore, if both parties act honestly, then their joint output equals* $\mathsf{WeakBatch}_m$*'s output on their joint input.*

The proof of Lemma 8 is given in the full version of this paper [17].

6 Applications

In this section, we show how our protocol can be used in several applications. To be more precise, we show how to realize several functionalities of interest (Perfect Multiplication, OLE, VOLE, MACs, Authenticated Triplets) in a hybrid model with oracle access to the functionality $\mathsf{WeakMult}$, which can be compiled into a real-world protocol by substituting the oracle with our protocol (as per the composition theorem of Canetti [9]).

6.1 Realizing Perfect Multiplication

We begin by showing how to realize perfect batch-multiplication maliciously where the definition of perfect batch-multiplication is according Functionality 5 (It is stressed that perfect multiplication is simply a special case). We will distinguish between large and small fields (where a field $\mathbb{F}$ is small if $|\mathbb{F}| < 2^{\kappa/2}$). Thus, we will assume here that $q \geq 2^{\kappa/2}$. In the full version of this paper [17] we discuss the technicalities for small fields (it is stressed that our results extend trivially to large fields that are not prime order).

To realize malicious security for Functionality 5, we will be needing the following "helper" functionalities: One commitment functionality denote $\mathcal{F}_{\mathsf{com}}$ (Functionality 16) that allows the parties to commit to certain values that can be revealed at a later time, and another functionality ShareCheck (Functionality 17) that enables the parties to verify whether their shares where computed correctly. In Section 6.1.2 we define our protocol in the hybrid model with ideal access to WeakBatch, ShareCheck and $\mathcal{F}_{\mathsf{com}}$ and we prove that it realizes PerfectMultBatching.[17] In the full version of this paper [17], we show how to realize ShareCheck cheaply using group-theoretic cryptography. A real world protocol with minimal overhead can thus be derived by substituting the oracles with the relevant protocols herein.[18]

6.1.1 Ideal Commitment & Share-Correctness Functionalities The functionality below receives one input from each party. These values are revealed at a later time once the functionality receives approval by both parties.

Functionality 16 (Commitment Functionality $\mathcal{F}_{\mathsf{com}}$)
- P_1*'s input:* $\alpha \in \mathbb{Z}_q$.
- P_2*'s input:* $\beta \in \mathbb{Z}_q$
- *Operation: Upon receiving* continue *from both parties,* $\mathcal{F}_{\mathsf{com}}$ *outputs* β *to* P_1 *and* α *to* P_2.

The functionality below receives one input and one share from each party. It simply checks whether the additive shares sum up to the product of the inputs.

Functionality 17 (ShareCheck)
P_1*'s input:* $(x_1, s_1) \in \mathbb{Z}_q^2$.
P_2*'s input:* $(x_2, s_2) \in \mathbb{Z}_q^2$
Operation: Output 1 *if* $x_1 \cdot x_2 = s_1 + s_2$ *and* 0 *otherwise.*

6.1.2 Secure Multiplication Protocol

Protocol 18 ($\Psi = (\mathsf{P}_1, \mathsf{P}_2)$)
Oracles: WeakBatch *and* ShareCheck

[17] We note that the definition of $\mathcal{F}_{\mathsf{com}}$ is reactive. This feature does not interfere with composition [9].
[18] Typically, $\mathcal{F}_{\mathsf{com}}$ is realized via a hash function modelled as a random oracle.

Parameters: Multiplications number $m \in \mathbb{N}$ *and a security parameter* $\kappa \in \mathbb{N}$.
Let $n := \lceil m \cdot \log q \rceil + \kappa$.
P_1*'s input:* $a \in \mathbb{Z}_q$.
P_2*'s input:* $\boldsymbol{b} = (b_1, \ldots, b_m) \in \mathbb{Z}_q^m$.
Operations:

1. P_1 *samples* $x \leftarrow \mathbb{Z}_q$*, sets* $\alpha = a - x$ *and sends* α *to* $\mathcal{F}_{\mathsf{com}}$.
2. P_2 *samples* $y \leftarrow \mathbb{Z}_q$*, sets* $\beta = b_1 - y$ *and sends* β *to* $\mathcal{F}_{\mathsf{com}}$.
3. P_1 *and* P_2 *invoke* WeakBatch *on inputs* $(1^\kappa, x)$ *and* $(1^\kappa, y, b_2, \ldots, b_m)$ *respectively. Let* $(\hat{s}_1^1, \ldots, \hat{s}_1^m)$, $(\hat{s}_1^2, \ldots, \hat{s}_2^m)$ *denote the respective outputs.*
4. P_1 *and* P_2 *invoke* ShareCheck *on inputs* $(1^\kappa, x, \hat{s}_1^1)$ *and* $(1^\kappa, y, \hat{s}_2^1)$ *respectively.*
5. P_1 *and* P_2 *send* continue *to* $\mathcal{F}_{\mathsf{com}}$.
6. P_1 *locally outputs* $(x \cdot \beta + \hat{s}_1^1, \hat{s}_1^2, \ldots, \hat{s}_1^m)$ *and* P_2 *locally outputs* $(b_1 \cdot \alpha + \hat{s}_2^1, \ldots, b_m \cdot \alpha + \hat{s}_2^m)$.

Theorem 19. *Protocol 18* α*-computes* PerfectMultBatching *(Functionality 5) for*

$$\alpha(\kappa) = 2^{-\kappa/4+4}.$$

The proof of Theorem 19 can be found in the suplementary material.

6.1.3 Realizing OLE & VOLE Recall that in VOLE (OLE is just single-instance VOLE), P_1 holds an input a and P_2 holds $\boldsymbol{b}, \boldsymbol{\sigma} \in \mathbb{Z}_q^m$, and the functionality returns $a\boldsymbol{b} + \boldsymbol{\sigma}$ to P_1 and nothing to P_2. Using a straightforward reduction from VOLE to batch-multiplication, it is enough to run Protocol 18 with parties using inputs a and $\boldsymbol{b}$ respectively. Then, once the protocol concludes, we instruct P_2 to add $\boldsymbol{\sigma}$ to its output and reveal the result to P_1. The resulting protocol is a secure realization of VOLE (or OLE for $m = 1$). We omit the formal details since they are rather straightforward.

6.2 Generating Correlated Data in the Preprocessing Model

In this section, we show how to use our protocol for generating correlated preprocessed data for general purpose MPC (namely MACs and Beaver Triplets). For an informal discussion of the two concepts, we refer the reader to the introduction (Section 1.3). Since MACs are just a special instance of batch-multiplication (and thus Protocol 18 can readily be used for this purpose) we only focus here on Beaver triplets. Similarly to PerfectMult, we will be using another "helper" functionality denote BeaverCheck which is analogous the ShareCheck, except that it is more complicated because it involves many more checks. Still, in the full version of this paper [17], we show that it can be cheaply realized using group-theoretic cryptography.

Functionality 20 (Beaver)
Inputs: Empty for both parties with the following optional inputs.

1. P_1*'s optional input* opt_1*:* $(x_1^1, x_1^2, x_1^3, k_1) \in \mathbb{Z}_q^2$ *and* $(\sigma_1^i, \tau_1^i) \in \mathbb{Z}_q^2$ *for* $i \in [3]$.
2. P_2*'s optional input* opt_2*:* $(x_2^1, x_2^2, x_2^3, k_2) \in \mathbb{Z}_q^2$ *and* $(\sigma_2^i, \tau_2^i) \in \mathbb{Z}_q^2$ *for* $i \in [3]$.

Operation:

- *Verify* $\mathsf{opt}_1 = \perp$ *or* $\mathsf{opt}_2 = \perp$*, otherwise abort (wlog say* $\mathsf{opt}_1 \neq \perp$*).*
- *Sample* $(x_2^1, x_2^2, k_2) \leftarrow \mathbb{Z}_q^3$.
- *Output* $(x_i^1, x_i^2, x_i^3, k_i, \sigma_i^1, \sigma_i^2, \sigma_i^3, \tau_i^1, \tau_i^2, \tau_i^3)$ *to* P_i *where unassigned values are set subject to*

$$\begin{cases} (x_1^1 + x_2^1)(x_1^2 + x_2^2) = x_1^3 + x_2^3 \\ \tau_i^j = k_{3-i} x_i^j + \sigma_{3-i}^j \quad \text{for } i \in \{1,2\},\ j \in \{1,2,3\} \end{cases}.$$

Functionality 21 (BeaverCheck)
Common input: 1^κ *for a security parameter* $\kappa \in \mathbb{N}$.
P_1*'s input:* $(x_1^1, x_1^2, x_1^3, k_1) \in \mathbb{Z}_q^2$ *and* $(\sigma_1^i, \tau_1^i) \in \mathbb{Z}_q^2$ *for* $i \in \{1,2,3\}$.
P_2*'s input:* $(x_2^1, x_2^2, x_2^3, k_2) \in \mathbb{Z}_q^2$ *and* $(\sigma_2^i, \tau_2^i) \in \mathbb{Z}_q^2$ *for* $i \in \{1,2,3\}$.
Operation: Output 1 *if the inputs satisfy the following (output 0 otherwise)*

$$\begin{cases} (x_1^1 + x_2^1)(x_1^2 + x_2^2) = x_1^3 + x_2^3 \\ \tau_i^j = k_{3-i} x_i^j + \sigma_{3-i}^j \quad \text{for } i \in \{1,2\},\ j \in \{1,2,3\} \end{cases}.$$

6.2.1 Authenticated (Beaver) Triplets Protocol As mentioned in the introduction, the protocol below simply preforms two weak multiplications to calculate the triplet and two (weak) batch-multiplications each to obtain all the MAC data. In the end, the parties perform the correctness-check on their shares.

Protocol 22 ($\Phi = (\mathsf{P}_1, \mathsf{P}_2)$)
Oracles: WeakMult, WeakBatch *and* BeaverCheck.
Inputs: Statistical parameter κ.
Operations:

1. *Each* P_i *samples* $k_i, a_i, b_i \leftarrow \mathbb{Z}_q$.
2. P_1 *and* P_2 *invoke* WeakMult (a_1, b_2) *and* WeakMult (b_1, a_2).
   Write γ_1, δ_1 *and* γ_2, δ_2 *for their respective outputs.*
3. *Each* P_i *sets* $c_i = a_i b_i + \gamma_i + \delta_i$.
4. P_1 *and* P_2 *invoke* $\mathsf{WeakBatch}(k_1, (a_2, b_2, c_2))$ *and* $\mathsf{WeakBatch}(k_2, (a_1, b_1, c_1))$.
   Write $(\tau_i, \tau_i', \tau_i'')$*, and* $(\sigma_i, \sigma_i', \sigma_i'')$ *for* P_i*'s outputs in each execution.*
5. P_1 *and* P_2 *invoke* BeaverCheck *on the relevant inputs.*
6. P_i *outputs* $(a_i, b_i, c_i, k_i, \tau_i, \tau_i', \tau_i'', \sigma_i, \sigma_i', \sigma_i'')$.

Theorem 23. *Protocol 22* α*-computes* Beaver *(Functionality 20) for*

$$\alpha(\kappa) = 2^{-\kappa/4+4}.$$

The proof of the above is very similar to the proof of Theorem 19 and it is ommited.

Bibliography

[1] Baum, C., Escudero, D., Pedrouzo-Ulloa, A., Scholl, P., Troncoso-Pastoriza, J.R.: Efficient protocols for oblivious linear function evaluation from ring-lwe. In: Security and Cryptography for Networks - 12th International Conference, SCN 2020. vol. 12238, pp. 130–149. Springer (2020)

[2] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference. vol. 576, pp. 420–432. Springer (1991)

[3] Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011. vol. 6597, pp. 253–273. Springer (2011)

[4] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018. pp. 896–912. ACM (2018)

[5] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019. pp. 291–308. ACM (2019)

[6] Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques. vol. 9057, pp. 337–367. Springer (2015)

[7] Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference. vol. 9814, pp. 509–539. Springer (2016)

[8] Boyle, E., Kohl, L., Scholl, P.: Homomorphic secret sharing from lattices without FHE. In: Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques. vol. 11477, pp. 3–33. Springer (2019)

[9] Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)

[10] Damgard, I., Orlandi, C.: Multiparty computation for dishonest majority: From passive to active security at low cost. In: Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference. vol. 6223, pp. 558–576. Springer (2010)

[11] Damgard, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference. vol. 7417, pp. 643–662. Springer (2012)

[12] Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ECDSA from ECDSA assumptions: The multiparty case. In: 2019 IEEE Symposium on Security

and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019. pp. 1051–1066. IEEE (2019)
[13] Frederiksen, T.K., Pinkas, B., Yanai, A.: Committed MPC - maliciously secure multiparty computation from homomorphic commitments. In: Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography. vol. 10769, pp. 587–619. Springer (2018)
[14] Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security. vol. 10624, pp. 629–659. Springer (2017)
[15] Gilboa, N.: Two party RSA key generation. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference. vol. 1666, pp. 116–129. Springer (1999)
[16] Goldreich, O.: Foundations of Cryptography – VOLUME 2: Basic Applications. Cambridge University Press (2004)
[17] Haitner, I., Makriyannis, N., Ranellucci, S., Tsfadia, E.: Highly efficient ot-based multiplication protocols. Cryptology ePrint Archive, Report 2021/1373 (2021), https://ia.cr/2021/1373
[18] Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions. In: Proceedings of the twenty-first annual ACM symposium on Theory of computing. pp. 12–24 (1989)
[19] Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009. vol. 5444, pp. 294–314. Springer (2009)
[20] Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 830–842. ACM (2016)
[21] Kiayias, A., Yung, M.: Cryptographic hardness based on the decoding of reed-solomon codes. IEEE Trans. Inf. Theory 54(6), 2752–2769 (2008)
[22] Lindell, Y., Nof, A.: A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017. pp. 259–276. ACM (2017)
[23] Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018. pp. 1837–1854. ACM (2018)
[24] Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006)
[25] Rotaru, D., Smart, N.P., Tanguy, T., Vercauteren, F., Wood, T.: Actively secure setup for SPDZ. IACR Cryptol. ePrint Arch. p. 1300 (2019)