Asymptotically Quasi-Optimal Cryptography

Leo de Castro[1], Carmit Hazay[2], Yuval Ishai[3], Vinod Vaikuntanathan[1], and Muthu Venkitasubramaniam[4]

[1] MIT
[2] Bar-Ilan University
[3] Technion
[4] Georgetown University

Abstract. The question of minimizing the *computational overhead* of cryptography was put forward by the work of Ishai, Kushilevitz, Ostrovsky and Sahai (STOC 2008). The main conclusion was that, under plausible assumptions, most cryptographic primitives can be realized with *constant* computational overhead. However, this ignores an additive term that may depend polynomially on the (concrete) computational security parameter λ. In this work, we study the question of obtaining optimal efficiency, up to polylogarithmic factors, for *all* choices of n and λ, where n is the size of the given task. In particular, when $n = \lambda$, we would like the computational cost to be only $\widetilde{O}(\lambda)$. We refer to this goal as *asymptotically quasi-optimal* (AQO) cryptography.

We start by realizing the first AQO semi-honest batch oblivious linear evaluation (BOLE) protocol. Our protocol applies to OLE over small fields and relies on the near-exponential security of the ring learning with errors (RLWE) assumption. Building on the above and on known constructions of AQO PCPs, we design the first AQO zero-knowledge (ZK) argument system for Boolean circuit satisfiability. Our construction combines a new AQO ZK-PCP construction that respects the AQO property of the underlying PCP along with a technique for converting statistical secrecy into soundness via OLE reversal. Finally, combining the above results, we get AQO secure computation protocols for Boolean circuits with security against malicious parties under RLWE.

1 Introduction

The work of Ishai, Kushilevitz, Ostrovsky and Sahai [IKOS08] put forward the goal of minimizing the *computational overhead* of cryptography. This was defined as the asymptotic ratio between the amount of work (say, Boolean circuit size) required to securely realize a given cryptographic task of size n and the amount of work required to realize the same task without any security at all. Here, n denotes the size of a Boolean circuit specifying a functionality, for primitives such as secure computation or zero-knowledge proofs, and just message size for simpler primitives such as encryption or commitment. The main conclusion of [IKOS08] is that, under plausible assumptions, most cryptographic primitives can be realized with *constant* computational overhead.

However, this ignores the significant additive term that may depend polynomially on the (concrete) computational security parameter λ.[5] That is, the computational cost of the constant overhead protocol could be $O(n + \lambda^c)$ for some constant $c > 1$. As a consequence, amortized efficiency may only kick in when $n \gg \lambda$, namely when the problem size is very big. For smaller instances, efficiency (measured in terms of communication, computation, or other resources) can be far from optimal. This is not only a theoretical concern, but also a practical concern for many primitives that have good amortized efficiency.

Asymptotically Quasi-Optimal Cryptography. The question that motivates our work is whether this is inherent. Can we get close to the best possible efficiency for *all* choices of n and λ, in particular when $n = \lambda$? We refer to the goal of achieving this up to polylogarithmic factors as *asymptotically quasi-optimal* (AQO) cryptography. AQO requires that solving a problem of size n with λ bits of security has computational cost (measured by Boolean circuit size) of $\widetilde{O}(n+\lambda)$. Modulo polylogarithmic factors, this represents the *best possible* solution, as the costs of n and λ are both inherent for natural primitives.

We will sometimes also refer to the relaxed goal of *AQO communication*, where the communication (or ciphertext size) is $\widetilde{O}(n+\lambda)$, but the computational cost may be larger. Here n refers to the *communication complexity* of realizing the same task without security requirements.

We view AQO as a clean theoretical abstraction of a practically relevant question, with an appealingly simple one-line description:

Solve a size-n cryptographic problem with efficiency $\widetilde{O}(n + \lambda)$.

AQO Cryptography: What's Known and What Isn't. In the domain of *symmetric cryptography*, the AQO goal is relatively easy to achieve. For instance, natural generalizations of popular block ciphers such as AES are conjectured to be AQO [MV15]. In fact, even with a *constant* (rather than polylogarithmic) overhead, most symmetric primitives can be realized under plausible hardness assumptions. This includes one-way functions [Gol00], pseudorandom generators [AIK08, BIO14], collision-resistant hashing [AHI+17], and pseudorandom functions [BIP+18]. For public-key encryption and statistically binding commitments, we have AQO schemes from Ring-LWE [LPR10, LPR13, LS, GHKW17], and for collision-resistant hashing and statistically hiding commitments, we have AQO schemes from Ring-SIS [Mic02, PR06, LM06].

Besides these, the case for other central cryptographic primitives such as zero-knowledge proofs and secure computation, seems to be wide open and for good

[5] Throughout this paper, the security parameter λ refers to bits of *concrete security*, requiring that no adversary of circuit size 2^λ can gain better than $2^{-\lambda}$ advantage. This is a natural and robust notion of concrete security. An alternative notion that settles for *negligible* advantage is not as robust, analogously to relaxing standard security definitions by requiring that every polynomial-time adversary has $o(1)$ advantage (rather than negligible in the sense of sub-polynomial).

Primitive	Assumption	Reference
Secret-key Encryption	Generalized AES	[DR02, MV15]
	Ring-LWE	[LPR10]
	Ring-LPN	[HKL+12]
	Mod-2/Mod-3 PRFs*	[BIP+18]
Public-key Encryption	Elliptic Curve CDH/DDH†	[Gam85]
	Ring-LWE	[LPR10]
String-OT	Elliptic Curve DDH†	[NP01, AIR01]
	Ring-LWE	Folklore‡
Batch-OT	Ring-LWE	This work
Single-OLE	Ring-LWE	This work
Batch-OLE	Ring-LWE	This work
Additively Homomorphic Encryption**	Ring-LWE	[LPR10]
(Malicious-Verifier) Zero-Knowledge	Ring-LWE	This work
(Malicious) Two-party Computation	Ring-LWE	This work

Table 1: Some representative examples in the AQO Landscape. Entry labeled with * was conjectured to achieve asymptotically optimal security (i.e., same as AQO but without the polylog factors).
† denotes solutions with AQO communication but not AQO computation, e.g. elliptic curves that employ exponentiation, making their computational complexity at least quadratic.
‡ The folklore protocol for string-OT uses the fact that the [LPR10] Ring-LWE-based PKE is additively homomorphic which gives us a leaky-OT. This can in turn be corrected via randomness extraction.
** indicates measuring the complexity of (non-function-private) homomorphic pointwise addition or scalar multiplication of two vectors of plaintexts.

reasons, as we discuss below. We refer to Table 1 for more examples in the AQO landscape. By "batch-OT" and "batch-OLE" we refer to a semi-honest secure two-party protocol for n copies of oblivious transfer (OT) or its generalization to oblivious linear-function evaluation (OLE) modulo $p \leq \mathsf{poly}(\lambda)$, which will be discussed in detail later.

Where do previous techniques fail? The main technical challenge in achieving AQO is the question of how to *amplify soundness or privacy without naïve repetition (or similar techniques)*. Traditional techniques such as statistical noise-flooding (for lattice-based OLE), arithmetization over big fields, or cut-and-choose (for zero knowledge and secure computation) all fall short of this goal. Even techniques that do achieve this goal in an amortized sense (such as "constant-overhead" semi-honest 2PC from a local PRG [IKOS08] or ZK based on robust honest-majority MPC [IKOS07, DIK10]) incur quadratic (in λ) additive terms, seemingly for inherent reasons.

An additional challenge for public-key AQO cryptography is that standard "number-theoretic" constructions fail for two reasons. First, common number-theoretic operations, such as modular exponentiation over a λ-bit modulus, are only known to have circuits of size $\widetilde{O}(\lambda^2)$. Second, factoring λ-bit integers or discrete logarithm modulo a λ-bit prime can be done in time 2^{λ^c} for $c < 1$, which requires working with numbers of size $\lambda^{c'}$ for $c' > 1$.

This leaves us with essentially elliptic curve discrete logarithms and (ring) learning with errors. In the case of elliptic curves, computations typically re-

quire exponentiation, and in the case of learning with errors, computations typically require matrix multiplication, both of which require superlinear time which rules out *computational* quasi-optimality. This leaves us the ring learning with errors assumption [LPR10, LPR13] with the unique status of helping us go beyond communication AQO. On the one hand, the problem is believed to be quasi-exponentially hard in the bit-length of the instance; and on the other hand, operations typically involve multiplication of two $O(\lambda)$-degree polynomials over a number field, which can be performed in quasi-linear $\widetilde{O}(\lambda)$ time using the (number-field version of) fast Fourier transform. Finally, Ring LWE has proven itself to be versatile not only in theory, having helped us construct fully homomorphic encryption [BV11b, BGV12, GHS12], but also in practice with the NIST standardization effort for post-quantum-secure public-key cryptography [Moo16].

Challenges for secure computation. In the semi-honest model, AQO secure computation reduces to an AQO batch-OT via classical techniques [GMW87, Gol04]. This in turn reduces to AQO batch-OLE, which is a more natural target in the context of lattice-based constructions. Several Ring-LWE based batch-OLE protocols have been proposed in the literature. The vanilla batch-OLE from Ring-LWE uses noise-flooding to ensure sender privacy, and this causes the communication and computation to be of size $O(n \cdot \lambda)$ where λ is a statistical security parameter, and hence is not AQO. Alternative techniques for circuit-private FHE without noise-flooding also fall short of the AQO goal: [DS16] involves λ iterations of bootstrapping to "rinse out" the noise, and [BPMW16] requires homomorphic evaluation of a branching program of size $\Omega(\lambda)$. Even ignoring AQO, these approaches do not seem attractive from a *concrete efficiency* viewpoint.

Challenges for security against malicious parties. Going beyond semi-honest security and achieving security against *malicious* parties in the AQO setting poses additional challenges. Common cut-and-choose techniques in zero-knowledge proofs and secure computation protocols [LPS08, HKE13, Lin16, WRK17] incur a *multiplicative overhead* of at least λ to achieve simulation error $2^{-\lambda}$. This is also the case when embedding a Boolean computation into an arithmetic computation over a big field $\mathbb{F}$ to achieve soundness $O(1/|\mathbb{F}|)$ [BCG+17, XZZ+19, CDI+].

MPC-in-the-head techniques [IKOS07, IPS08, AHIV17, HIMV19] can improve over standard cut-and-choose techniques by achieving a better *amortized* overhead, as low as polylog(λ) [DIK10]. However, the underlying MPC protocols incur an additive communication overhead of at least $\Omega(\lambda^2)$. This also applies to all known *succinct* zero-knowledge argument systems, including those based on classical PCPs and IOPs (e.g., [Kil92, BCS16, BBHR19, BCR+19, ZXZS20, CY21, RR21]) and linear PCPs [IKO07, Gro10, GGPR13, BCI+13, BISW18].

Finally, there is a rich and productive literature constructing zero-knowledge protocols for $\mathcal{NP}$ statements assuming RLWE [BKLP15, LS18, BBC+18, BLS19, BLNS20]. To the best of our knowledge, these protocols cannot even achieve AQO communication, let alone computation. The reason is subtle. Focusing on ZK protocols for RLWE statements, the protocols seem to fall into one of two

categories. The first type incurs large soundness error which is then repeated $\widetilde{O}(\lambda)$ times to get $2^{-\lambda}$ soundness, and thus has at least a quadratic in λ overhead. The second type is a direct proof with exponentially small soundness error and AQO efficiency, but for a weaker $\mathcal{NP}$ statement related to RLWE which does not seem directly applicable to constructing ZK proofs for general $\mathcal{NP}$ statements.[6] We refer the reader to an extensive discussion in [BLS19] who, with a great deal of ingenuity, reduce the number of parallel repetitions required in the first type of protocols in order to achieve $2^{-\lambda}$ soundness from λ to $\lambda/\mathsf{log}\lambda$.

1.1 Our Results and Techniques

Our Results, in a Nutshell. Our results are three-fold. First, we show AQO protocols for batch-oblivious linear evaluation (batch-OLE) and batch-oblivious transfer (batch-OT)[7] which are secure against semi-honest corruptions under the Ring Learning with Errors (RLWE) assumption.

Secondly, in the case of batch-OLE, we improve this to obtain asymptotic download rate that approaches 1. This gives an AQO variant of recent rate-1 constructions from [DGI+19, BDGM19, GH19], which require uncompressing batched ciphertexts. In contrast, our construction is a simple tweak on an old encryption scheme due to Peikert, Vaikuntanathan and Waters [PVW08], a batched version of Regev's encryption [Reg05]. The high rate of our construction gives rise (via a simple extractor-based transformation [BGI+17]) to an AQO construction of statistically sender-private (SSP) 2-message OT from RLWE. Beyond the AQO feature, this gives an alternative route to recent lattice-based constructions of SSP OT [BD18, DGI+19, MS20].

Finally, we use the batch-OT to construct AQO zero-knowledge and secure computation protocols for Boolean circuits with security against malicious parties. These protocols too are secure under RLWE.

Our goal was to answer a clean theoretical question. However, as it turned out, our solution for AQO batch-OLE is competitive in practice, especially for small instance sizes (and small fields). While the *AQO definition* may not say anything about concrete efficiency for real-world parameters, we view this empirical data point of correlation between "AQO security" and "concrete efficiency" as a promising sign.

We now proceed to describe our results and techniques in more detail.

Semi-Honest Batch-OLE: AQO and Concretely Efficient. Oblivious linear evaluation (OLE) is a protocol between two parties S, the sender who has a linear function $L_{\alpha,\beta}(x) = \alpha x + \beta$ over a finite ring $\mathcal{R}$, and R, the receiver who has an input $m \in \mathcal{R}$. At the end of the protocol, R gets $L_{\alpha,\beta}(m) = \alpha m + \beta$, and

[6] We remark that the statements we want are proofs (of knowledge) of a short secret s such that $As = t$ over a ring. On the other hand, the second type of protocols prove that there is a short secret s such that *As equals a short multiple of t*.

[7] Recall that Batch-OT / OLE refers to multiple OT / OLE instances carried out in parallel.

the sender gets nothing. OLE is a generalization of oblivious transfer (OT) over fields (and rings) larger than $\mathbb{F}_2$, and has numerous applications, notably in secure computation. We consider the n-fold repetition of the OLE functionality, called *batch-OLE*, where the sender has $\alpha, \beta \in \mathcal{R}^n$, the receiver has $m \in \mathcal{R}$ and gets $\alpha \circ m + \beta \in \mathcal{R}^n$ where $\circ$ denotes a coordinate-wise product. Here we consider the case of batch-OLE over a polynomial-size modulus, namely where $\mathcal{R} = \mathbb{Z}_p$ for $p \leq \mathsf{poly}(\lambda)$.

We show the first construction of a semi-honest batch-OLE protocol which is asymptotically quasi-optimal. Our protocol has minimal interaction of just two rounds, and its security is based on the ring learning with errors (RLWE) assumption. In the parameter regimes in which our protocol has competitive concrete efficiency, it can be useful for realizing the distributed seed generation of pseudorandom correlation generators (PCGs) for OLE and multiplication triples based on Ring-LPN [BCG+20].

Our starting point is a folklore batch-OLE scheme using a batched version of the classical Lyubashevsky-Peikert-Regev [LPR10] encryption scheme (henceforth called LPR encryption). The encryption scheme works over a message space $\mathcal{R}_p := \mathbb{Z}_p[x]/(x^k + 1)$ where k is a power of 2 and $x^k + 1$ factors completely into linear factors mod p.[8] To encrypt a vector $m \in \mathbb{Z}_p^k$, we first find, using the number theoretic transform (NTT), a polynomial $\widehat{m}$ such that $\widehat{m}(\zeta_i) = m_i \pmod p$ for all $i \in [k]$. Here, $\zeta_i \in \mathbb{Z}_p$ are the k roots of the polynomial $x^k + 1 \pmod p$ which exist by our choice of $p = 1 \pmod{2k}$. The ciphertext

$$ct = (a, as + e + \Delta\widehat{m}) := (a, b)$$

where $\Delta = \lceil q/p \rceil$,[9] s is a random ring element and e is a short ring element.

The receiver in our OLE generates a ciphertext ct that encrypts his input m and sends ct to the sender. By the semantic security of LPR encryption, which relies on RLWE, the sender learns nothing about m.

The sender has $\alpha, \beta \in \mathbb{Z}_p^k$ and wishes to homomorphically compute the (linear) OLE function. They do this by computing and returning to the receiver

$$ct_{eval} = (\widehat{\alpha}a, \widehat{\alpha}b + \Delta\widehat{\beta})$$

which one can easily check is an encryption of $\alpha \circ m + \beta$.

The main problem with this idea is the lack of *function privacy*. That is, the homomorphically evaluated ciphertext could contain information not just about $\alpha \circ m + \beta$, but about α and β themselves. Indeed, this is not hard to see as the first component of ct' reveals α already. This can be solved using rerandomization: the receiver can send a rerandomization key that allows the sender to generate a pair $(a', b' := a's + e')$ with a (statistically close to) random a', which they add to ct_{eval}. This results in a rerandomized ciphertext

$$ct'_{eval} = (\widehat{\alpha}a + a', \widehat{\alpha}b + b' + \Delta\widehat{\beta})$$

[8] We denote the Ring-LWE dimension by k, and the OLE batch-size by n.

[9] We typically pick q to be a multiple of p so the rounding is not necessary.

where the first component is statistically close to random. Still, we are not done: the receiver who knows s can retrieve terms such as $\widehat{\alpha}e + e'$ which could reveal significant information about α.

The typical way to get around this problem is to add to ct' a very-high-noise encryption of 0 (namely, one with a very large e') that will mask such terms. In particular, one appeals to the so-called noise-flooding lemmas [Gen09, AJL+12, GKPV10] that requires adding noise that is a factor 2^λ larger than $\widehat{\alpha}e$ to achieve $2^{-\lambda}$ (statistical) security. Unfortunately, this blows up ciphertext sizes by a multiplicative factor of λ, resulting in a communication of $O(k\lambda)$, violating the demands of AQO efficiency. (Recall that we need $\widetilde{O}(k + \lambda)$.)

At a high level, our main observation is that noise flooding is too strong a hammer to achieve function privacy and therefore sender privacy. Instead, our main contribution is a *gentle noise-flooding* procedure that gives us AQO efficiency. We start by imagining what happens if we only add a small amount of noise to each coordinate, in fact, just a constant factor larger than $\|\widehat{\alpha}e\|_\infty$.

To illustrate this concretely, imagine that each coordinate t of $\widehat{\alpha}e$ lives in the interval $[0, 10]$ and we add a noise term η chosen randomly from the interval $[0, 20]$. If $r = t + \eta$ lands up in the interval $[10, 20]$, it reveals no information about what t was to begin with! Indeed, all values of t in the interval $[0, 10]$ are equally likely conditioned on such a "good" r. In other words, by adding noise that is a constant factor larger than $\|\widehat{\alpha}e\|_\infty$, one could hope to hide a constant fraction of the coordinates of $\widehat{\alpha}e$. This is formalized as our gentle noise-flooding lemma (Lemma 3).

This is still not enough as leaking a constant fraction of $\widehat{\alpha}e'$ is not acceptable. However, this predicament we are in should point to secret-sharing as a possible way out. Indeed, our solution is to use a suitable modification of the OLE extractors of [IKOS09a, BMN18a] to extract fresh OLE instances from these "leaky" OLE instances. In a nutshell, to achieve AQO efficiency, we instantiate the compiler of Block, Gupta, Maji and Nguyen [BGMN18] with a Reed-Solomon code which admits quasi-linear time encoding and erasure-decoding. We can also achieve AQO batch-OT by embedding OT into OLE using a standard technique.

Finally, we show a simple modification of (a ring version of) the PVW encryption scheme [PVW08] which, when used in place of LPR encryption, gives us a download-rate-$(1 - \epsilon)$ batch-OLE. Namely, the sender message has length $(1 + \epsilon)k\log\, p$ for any constant $\epsilon > 0$. Note that this is smaller than the total length of the sender input (namely, $2k\log\, p$), thus the sender input is somewhat *statistically hidden* even if the receiver is malicious.

On Asymptotic Quasi-Optimality vs. Concrete Efficiency. Asymptotic quasi-optimality is a theoretical framework to capture efficiency of cryptographic protocols, *with an eye towards practicality*. To demonstrate the latter, we provide an implementation of our batch-OLE protocol and benchmark it against several competing approaches [BDOZ11, dCJV21, BEP+20], demonstrating that it achieves as good or better communication and/or computational overhead than the competing approaches. Due to lack of space, we defer detailed performance results and comparisons to the full version of this paper. The computational

and communication complexity is as good as the rounding protocol in [dCJV21] and considerably better than other competing approaches [BDOZ11, BEP+20]. For example, doing 10,000 OLEs over a 16-bit field requires a communication of 1.17 MB in our protocol versus 1.31 MB in the protocol of [dCJV21] and 2.09 MB in the protocol of [BEP+20].

AQO Zero Knowledge. We show the first construction of a zero-knowledge proof for all of $\mathcal{NP}$ that is asymptotically quasi-optimal in both computation and communication. Furthermore, our protocol is constant-round. Our starting point is an asymptotically quasi-optimal PCP that is implicit in prior works on near-optimal PCPs [BS08,BCGT13]. We abstract such an AQO PCP via a gap version of Cook's theorem (cf. Theorem 4). Our construction proceeds in three steps:

1. We first compile the AQO PCP into an honest-verifier AQO zero-knowledge PCP (ZKPCP). Recall that in such a ZKPCP, the view of an honest verifier can be simulated without knowing the witness. Our construction relies on the "MPC-in-the-head" technique from [IKOS07,IW14]. For the compilation, we design a specialized MPC protocol that preserves the AQO property of the underlying PCP.
2. In the next step, we construct an AQO honest-verifier ZK from our ZKPCP, using batch-OT to emulate the PCP queries. As we are constructing a proof (with unconditional soundness), we need the batch-OT to be unconditionally secure against a malicious sender. However, we are unable to obtain such a protocol directly from our semi-honest batch-OT protocol (in fact, our malicious batch-OT protocol will rely on the zero-knowledge proof system designed here). Instead, we design a new AQO batch-OT protocol based on Ring-LWE with two caveats: it is unconditionally secure against a malicious receiver as opposed to the sender, and it is only *entropically secure* in the sense that a malicious receiver obtains some bounded leakage on the sender's input. By reversing the OT direction [WW06], we solve the problem of getting security against a malicious sender. Finally, we show that the entropy loss in the sender's input only reduces the ZKPCP soundness in our honest-verifier ZK proof system by essentially the loss in entropy. By appropriately instantiating the parameters, we preserve AQO in this reduction.
3. Finally, to handle a malicious verifier, we have the verifier commit to its randomness for the honest-verifier ZK system and reveal it to demonstrate honest behavior. As the verifier can abort at reveal, we need to ensure that the actual proof is not learned by the verifier before it demonstrates honest behavior. We achieve this by having the prover commit to the proof and reveal it after the verifier reveals its randomness. By using an AQO commitment, we ensure this step preserves the AQO property.

Thus, a key insight in this construction is that a leaky (entropically secure) batch-OLE scheme is good enough because we only use it for *soundness*.

AQO Secure Function Evaluation. Finally, we discuss how to achieve AQO secure function evaluation (SFE) for Boolean circuits in the presence of semi-honest and malicious adversaries. Loosely speaking, semi-honest SFE is implied

directly by instantiating a variant of the classic GMW protocol [GMW87,Gol04] with our AQO semi-honest batch-OT. Next, to compile it to achieve malicious security, we first compile our semi-honest batch-OT protocol to be secure against malicious parties using our AQO ZK. Our protocol then relies on the semi-honest GMW protocol where the OTs are instantiated using our maliciously secure OT protocol. Next, we rely on the observation from [GIP+14] that if we remove the final output reconstruction round from the *semi-honest* GMW protocol in the OT-hybrid model, then it does not reveal any information even to malicious parties. This allows us to use a *single* zero-knowledge proof (rather than one in each step of the protocol) to be provided just before the output shares are revealed. As a corollary, we get an AQO *single* OLE over an arbitrary modulus.

1.2 Perspectives and Open Problems

Theoretical Motivation. Our original motivation for this work was to design efficient solutions when the instance size n was small, i.e. $n = O(\lambda)$ where λ is the security parameter. We expect that studying this question will lead to *creative* ways to solve problems such as OT, OLE, ZK, and MPC.

Our optimism is based on past examples. Several lines of research have started from clean questions of this kind and turned out to have unexpected theoretical and practical applications. Some examples include lattice-based cryptography, black-box reductions and, closer to our work, low-complexity cryptography. A common feature is that a new theoretical challenge has led to a rich landscape of new techniques, which have then found other applications.

Practical Motivation. As already mentioned, our (semi-honest) batch-OLE protocol gives a promising evidence for relevance of the asymptotic AQO question to concrete efficiency. Batch-OLE can serve as a useful building block for secure arithmetic computation, and can be used to bootstrap pseudorandom correlation generators for OLE [BCG+20]. In contrast, our current AQO zero-knowledge protocol is impractical because of its reliance on a classical PCP.

Open Problems. The central creative challenge in achieving AQO is to find new ways of amplification. While we succeeded in some cases, many questions about AQO cryptography remain open and motivate future research. We include here some open questions.

First, while there are AQO constructions of minicrypt objects from a variety of assumptions, the only AQO public-key encryption scheme we are aware of is based on Ring-LWE. There are likely to be be other ways to achieve AQO cryptomania, and we believe this is an interesting challenge for future research. A second question is obtaining concretely efficient AQO zero-knowledge proofs. A possible route is by employing a suitable AQO variant of a *linear* PCP (such as the one of Gennaro et al. [GGPR13]), where the field size is kept small and soundness is amplified by using λ queries, but with only a polylogarithmic increase in computation. Third, the notion of *AQO reductions* (which we used to construct AQO semi-honest SFE from AQO batch-OT) leaves several open

questions. For instance, is there an information-theoretic AQO reduction of zero-knowledge proofs to batch-OT? Finally, the idea of using leaky functionalities (such as batch-OT or batch-OLE) in downstream applications, which we used to construct our AQO ZK protocol, could be useful in other contexts.

2 Preliminaries

Basic notations and conventions. We denote the security parameter by λ and by the abbreviation PPT to denote probabilistic polynomial-time. We write $\widetilde{O}(\cdot)$ to suppress polylogarithmic factors. In this work we consider nonuniform adversaries that are modeled by Boolean circuits.

2.1 Asymptotic Quasi-Optimality

In this section, we define the notion of asymptotic quasi-optimality (AQO) for the cryptographic primitives we explore in this work. Recall that a major distinction between this notion and some earlier notions of asymptotic (quasi)-optimality from the literature [IKOS08, DIK10, BCG+17, BISW18] is that here, we demand (and obtain) a near-optimal tradeoff between security and efficiency for *every instance size and security level*, as opposed to sufficiently big polynomial-size (in the security parameter) instances. In contrast, previous works neglect additive terms that depend polynomially on the security parameter. For all primitives, we define a notion of *instance size* that we denote by n and a security parameter λ. Informally, asymptotic quasi-optimality demands that the algorithms for the primitives run in time $\widetilde{O}(n+\lambda)$ and provides $2^{-\lambda}$-security against adversaries of size 2^{λ}.

We now describe how such a definition manifests in the case of two-party secure function evaluation. Here, the instance size n refers to the size of a Boolean circuit implementing the underlying functionality. Such protocols are formally captured by a polynomial-time *protocol compiler* that, given inputs the security parameter 1^{λ} and a circuit C, outputs a pair of circuits (P_0, P_1) that implements the next message function of the two parties in the protocol. The AQO efficiency requirement is that the size of the circuits P_0 and P_1 output by the compiler is quasilinear in $n+\lambda$.

While the correctness requirement (when no party is corrupted) should hold irrespective of the choice of λ, C, the security requirement only considers adversaries of size at most 2^{λ}. The definition follows the standard definition of security for two-party secure function evaluation [Gol04] with the exception that we use the following "exact" notion of 2^{λ}-indistinguishability:

Definition 1. *Let $X = \{X(\lambda,a)\}_{\lambda\in\mathbb{N},a\in\{0,1\}^*}$ and $Y = \{Y(\lambda,a)\}_{\lambda\in\mathbb{N},a\in\{0,1\}^*}$ be two distribution ensembles. We say that the ensembles X and Y are 2^{λ}-*indistinguishable*, denoted $X \approx_{2^{\lambda}} Y$, if for every non-uniform circuit $\mathcal{D}$ of size at most 2^{λ}, every $a \in \{0,1\}^*$, and all sufficiently large λ,*

$$\left| \Pr\left[\mathcal{D}(X(\lambda,a), 1^{\lambda}, a) = 1\right] - \Pr\left[\mathcal{D}(Y(\lambda,a), 1^{\lambda}, a) = 1\right] \right| \leq 2^{-\lambda}.$$

The definitions for other AQO primitives considered in this paper follow as special cases of AQO secure function evaluation. We defer the formal definitions to the full version of this paper.

2.2 Ring Learning with Errors

Define the ring $\mathcal{R} := \mathbb{Z}[x]/(x^k + 1)$, where we take k to be a positive power of 2. For a modulus q, let $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^k + 1)$. Let $\mathcal{U}(\mathcal{R}_q)$ be the uniform distribution over $\mathcal{R}_q$. For $\sigma \in \mathbb{R}^+$, let χ denote the *error distribution*, which is a discrete, zero-centered Gaussian distribution with variance σ^2 over $\mathcal{R}$. A sample $e \leftarrow \chi$ is produced by sampling each coefficient from a discrete, zero-centered Gaussian with variance σ^2. We now define the decisional Ring-LWE problem [LPR10], borrowing formalisms from [BEP+20].

Definition 2 (Decisional Ring Learning with Errors Problem). *For a modulus $q \in \mathbb{N}^+$, k a power of 2, and a standard deviation $\sigma \in \mathbb{R}^+$, let $\mathcal{R}_q$ and χ be as defined above. We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the problem $\mathsf{RLWE}_{n,q,\chi}$ if the following holds:*

$$\Big|\Pr[b = 1 \mid \mathbf{a} \leftarrow \mathcal{U}(\mathcal{R}_q),\ \mathbf{s}, \mathbf{e} \leftarrow \chi,\ b \leftarrow \mathcal{A}(\mathbf{a}, \mathbf{as} + \mathbf{e})] - \Pr[b = 1 \mid \mathbf{a} \leftarrow \mathcal{U}(\mathcal{R}_q),\ \mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q)),\ b \leftarrow \mathcal{A}(\mathbf{a}, \mathbf{u})]\Big| \geq \epsilon$$

The decisional ring-LWE assumption postulates that every 2^λ-time adversary has advantage at most $2^{-\lambda}$ in the distinguishing game above.

In order to achieve the definition, one would set $k = k(\lambda)$ to be a large enough polynomial function of λ. The cryptanalytic state of the art suggests that $k(\lambda)$ can be *quasi-linear* in λ.

2.3 Ring-LWE Encryption

We describe a batched version of an encryption scheme from Lyubashevsky, Peikert and Regev [LPR10] (henceforth called batch-LPR). There are four parameters that define the scheme: $k = k(\lambda)$, the ring dimension; $q = q(\lambda)$, the ciphertext modulus; $p = p(\lambda)$, the plaintext modulus; and χ, an error distribution. There are several constraints on these parameters that we will describe in the sequel.

The scheme operates over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(f(x))$ where $f(x)$ is a degree-k polynomial that is irreducible over $\mathbb{Q}[x]$. Typically, and throughout this paper, we consider $f(x) = x^k + 1$ where k is a power of two.

We will let χ be a (truncated) discrete Gaussian distribution over $\mathbb{Z}^k$ which is interpreted as a distribution over the coefficient embedding of $\mathcal{R} = \mathbb{Z}[x]/(x^k + 1)$. Thus, a polynomial $v \in \mathcal{R}$ is sampled according to the distribution by sampling each coefficient independently from a truncated discrete Gaussian, namely a discrete Gaussian with standard deviation σ whose support is contained in an

Euclidean ball of radius $\sigma\sqrt{k}$. Note that we truncate the Gaussian distribution to have statistical distance of at most $2^{-k} \leq 2^{-\lambda}$ from the untruncated Gaussian distribution [MR04], hence truncation adds a $2^{-\lambda}$ factor in security games, which we typically ignore. Let $U(\mathcal{R}_q)$ be the uniform distribution over $\mathcal{R}_q$.

Encryption Scheme The encryption scheme proceeds as follows. The parameters k, p, q and σ are assumed to be known to all the algorithms.

- $\mathsf{LPR.KeyGen}(1^\lambda)$: Choose $a_1, \ldots, a_\ell, s \leftarrow U(\mathcal{R}_q)$ and $e_1, \ldots, e_\ell \leftarrow \chi$, where ℓ is a tunable parameter that will be set later during the rerandomization procedure. Output the secret key $sk = s$ and the rerandomization key $rk = (a_1, \ldots, a_\ell, a_1 s + e_1, \ldots, a_\ell s + e_\ell)$.
- $\mathsf{LPR.Encode}(p, m)$ and $\mathsf{LPR.Decode}(p, \widehat{m})$: The public, deterministic, encoding algorithm transforms the message into a form that will be used by the encryption algorithm, and the public decoding algorithm is its inverse operation. Both operations are linear.
  The plaintext space for batch-LPR is $\mathbb{Z}_p^k$. To encode $m \in \mathbb{Z}_p^k$, apply the number-theoretic transform (NTT) over $\mathcal{R}_p := \mathbb{Z}_p[x]/(x^k + 1)$ to convert it into $\widehat{m} \in \mathcal{R}_q$. The key property is that for every $m_1, m_2 \in \mathbb{Z}_p^k$, $m_1 \circ m_2 = \mathsf{LPR.Decode}(\widehat{m}_1 \cdot \widehat{m}_2)$ where $\cdot$ denotes multiplication of polynomials in $\mathcal{R}_p$ and $\circ$ denotes coordinate-wise multiplication of vectors in $\mathbb{Z}_p^k$.
- $\mathsf{LPR.Enc}(sk, m)$: Sample $a \leftarrow \mathcal{R}_q$ and $e \leftarrow \chi$. Let $\Delta = \lceil q/p \rfloor$ and let $\widehat{m} = \mathsf{LPR.Encode}(p, m)$. Output the ciphertext

$$ct = (a, as + e + \Delta\widehat{m}) \in \mathcal{R}_q^2$$

  (In this paper, q will be chosen as a multiple of p, so $\Delta = q/p$.)
- $\mathsf{LPR.Dec}(sk, ct)$: Parse $sk = s$. Decryption of a ciphertext $ct = (a, b)$ proceeds by computing

$$\widehat{m} = \left\lfloor \frac{b - as}{\Delta} \right\rceil$$

  Output $\mathsf{LPR.Decode}(p, \widehat{m})$.

Correctness, Security and Parameter Settings. There are several interrelated constraints on the parameters that must be balanced when instantiating the scheme. For correctness, we need that

$$||b - as||_\infty \leq \Delta/2 \approx q/2p$$

This places a lower bound on $q = \Omega(p\sigma^2\sqrt{k})$. Since we insist on full utilization of all k plaintext slots, we require $\mathbb{Z}[x]/(x^k+1)$ to split completely mod p, requiring us to have $p = 1 \pmod{2k}$. Thus, we have $q > p \geq 2k + 1$. (Additionally to support quasi-linear time operations, we will also need to support NTT mod q, so we requires the factors of q to be $1 \pmod{2k}$ as well.)

The relationship between parameters is further complicated because of homomorphic operations, which can grow the error term in the ciphertext. To

maintain correctness, this may require the ciphertext modulus q to grow. Increasing q can raise the lower bound on k since the known attacks on Ring-LWE improve in quality as the ratio between q and σ increases; to compensate for it, one needs to increase k. In turn, this increases the smallest p that can be supported.

Homomorphic Operations We now define two basic homomorphic operations on the encryption scheme that allow us to construct a batch-OLE protocol. The first operation Lin supports linear functions of the form $f_{\alpha,\beta}(m) = \alpha \circ m + \beta$. It is often desirable that homomorphic operations produce a ciphertext that does not leak the circuit computed to generate the ciphertext, even to the party that generated the input ciphertext and who knows the secret key. This property is often called *function privacy* [Gen09, GHV10] and is not satisfied by the Lin algorithm. To achieve this function-hiding property, we need a rerandomization algorithm ReRand.

- $\mathsf{LPR.Lin}(ct, \alpha, \beta)$: The homomorphic addition algorithm outputs a ciphertext ct' that decrypts to $\alpha \circ m + \beta$ if ct encrypts m. Letting $ct = (a, b)$, the algorithm outputs
$$ct' = (\widehat{\alpha}a, \widehat{\alpha}b + \Delta\widehat{\beta})$$
where $\widehat{\alpha} = \mathsf{LPR.Encode}(p, \alpha)$ and $\widehat{\beta} = \mathsf{LPR.Encode}(p, \beta)$. Note that while we think of the output of $\mathsf{LPR.Encode}$ as living in $\mathcal{R}_p$, we think of it as a polynomial over $\mathcal{R}$ when multiplying it with a and b.
Denoting $\widehat{\alpha}a$ by a', the ciphertext ct' is of the form $(a', a's + \widehat{\alpha}e + \Delta(\widehat{\alpha}\widehat{m} + \widehat{\beta}))$. Assuming that $\widehat{\alpha}e$ is small enough, decrytion recovers $\widehat{\alpha}\widehat{m} + \widehat{\beta}$ and decoding it recovers $\alpha \circ m + \beta$.
- $\mathsf{LPR.ReRand}(rk, ct, B)$: Let χ_{flood} be the uniform distribution over $[-B, \ldots, B]$. The rerandomization operation parses $rk = (a_1, a_2, \ldots, a_\ell, b_1, b_2, \ldots, b_\ell)$ and $ct = (a, b)$ and outputs
$$(a + \sum_{i=1}^{\ell} r_i a_i + r_0, b + \sum_{i=1}^{\ell} r_i b_i + f)$$
where r_i are polynomials with coefficients from a discrete Gaussian distribution, and f is a random polynomial with coefficients chosen from χ_{flood}.
Denoting the first component of the above ciphertext by a', and assuming that $b = as + e + \Delta\widehat{m}$, the second component can be written as $a's + e + \sum_{i=1}^{\ell} r_i e_i + \Delta\widehat{m}$. This is an encryption of $\widehat{m}$ as long as the error $e + \sum_{i=1}^{\ell} r_i e_i$ is small enough.

The rerandomization procedure is often used with $B > 2^\lambda \cdot p\sigma^2 k$. Indeed, $p\sigma^2 k$ is an upper bound on the ℓ_∞ norm of the noise term in the output of Lin. By the noise-flooding lemma [Gen09, GKPV10, AJL+12], this gives us $2^{-\lambda}$ statistical security of $\mathsf{LPR.ReRand}$. In this work, we will use a narrower flooding distribution.

2.4 Entropy and Extraction

The min-entropy of a random variable X is $H_\infty(X) = -\log \max_x \Pr[X = x]$. The conditional min-entropy of X given Y, defined in [DORS08], is $\widetilde{H}_\infty(X|Y) = -\log \mathbb{E}_y[\max_x \Pr[X = x|Y = y]]$. We need the following fact.

Lemma 1 (Lemma 2.2 in [DORS08]). *Let X, Y, Z be random variables where Y takes at most 2^ℓ possible values. Then,*

$$\widetilde{H}_\infty(A|B,C) \geq \widetilde{H}_\infty(A,B|C) - \ell \geq \widetilde{H}_\infty(A|C) - \ell \ .$$

We also need the following regularity lemma [Mic02, LPR13].

Lemma 2 (Corollary 7.5 in [LPR13]). *Let $a_1, \ldots, a_\ell$ be chosen at random from $\mathcal{R}_q = \mathbb{Z}[x]/(x^k + 1)$ where k is a power of two, and let $r_0, r_1, \ldots, r_\ell$ be ring elements each of whose coefficients is chosen from a discrete Gaussian with parameter $\sigma \geq 2k \cdot q^{(n+2)/n\ell}$. Then the distribution of $r_0 + \sum_{i=1}^{\ell} r_i a_i$ (given $a_1, \ldots, a_\ell$) has statistical distance at most $2^{-\Omega(k)}$ from the uniform distribution over $\mathcal{R}_q$.*

3 AQO Semi-Honest Batch-OLE and Batch-OT

We begin this section with our first technical contribution, namely a gentle noise-flooding procedure. We then use this to construct our asymptotically quasi-optimal batch-OLE and batch-OT schemes in Section 3.3.

3.1 Gentle Noise-Flooding

The noise-flooding lemma (e.g. [Gen09, GKPV10]) states that for every integer $x \in [-P, P]$, the distribution of $x + y$ where the integer y is chosen uniformly at random from the interval $[-Q, Q]$ is statistically close to the uniform distribution over the interval $[-Q, Q]$. Specifically, the statistical distance is $O(P/Q)$. Typically, this lemma is used with $Q \geq P \cdot 2^\lambda$ so as to result in exponentially small statistical distance. Our gentle noise-flooding lemma below shows a qualitatively stronger statement: the distribution of $x + y$ can be *perfectly* simulated by an algorithm that gets x with probability $2P/(2Q + 1)$ (and $\perp$ otherwise). This is a specific, simple, instance of a statistical-to-perfect lemma as in [IKO+11].

Let $\mathcal{D} = \{D_a\}_{a \in A}$ be an ensemble of distributions indexed by a variable $a \in A$. An ϵ-leaky perfect simulator for $\mathcal{D}$ is an algorithm S such that the distribution obtained by outputting $S(a)$ with probability ϵ and $S(\perp)$ with probability $1 - \epsilon$ is *identically distributed* to D_a.

Lemma 3 (Gentle Noise Flooding Lemma). *Let P, Q be integers with $P < Q$. Let the encoding of $a \in [-P, P]$, denoted $\mathsf{Encode}(a)$ be $s = a + r$ where r is chosen uniformly from $[-Q, Q]$. Then, there exists a $2P/(2Q + 1)$-leaky perfect simulation for the encoding scheme.*

Proof. We first analyze the distribution $\mathsf{Encode}(a)$. Consider two cases.

- **Case 1:** $P-Q \leq s \leq Q-P$**.** In this case, we argue that no information about a is leaked. For any s such that $P-Q \leq s \leq Q-P$, and any $a \in [-P,P]$, there is a unique $r \leftarrow [-Q,Q]$ such that $s = a+r$. This implies that for any a,
$$\Pr[\mathsf{Encode}(a) = s \mid s \in [Q-P, P-Q]] = 1/(2Q-2P).$$
Furthermore, the probability that we are in Case 1, i.e., $s \in [P-Q, Q-P]$ is exactly $(2Q-2P+1)/(2Q+1)$.
- **Case 2:** $s < P-Q$ **or** $s > Q-P$**.** In this case, s leaks something about a. As the number of r's that result in Case 1 is exactly $2Q-2P+1$, the number of *bad* r's is exactly $2P$. Therefore, the probability that this case occurs is $2P/(2Q+1)$.

We now define S to be the algorithm that works as follows. On input $\perp$, it simply outputs a uniformly random value in $[P-Q, Q-P]$; and on input a, it outputs $a+r$ conditioned on $a+r \notin [P-Q, Q-P]$ where r is chosen uniformly at random from $[-Q,Q]$. The distributions induced by $S(\perp)$ and $S(a)$ are identical to the distributions from Cases 1 and 2 respectively. Since Case 2 occurs with $2P/(2Q+1)$ probability, we achieve $2P/(2Q+1)$-leaky perfect simulation. □

Corollary 1. *Let $Q \geq kP$. Let $\boldsymbol{a} \in [-P,P]^k$ be arbitrary and let $\boldsymbol{s} = \boldsymbol{a} + \boldsymbol{r}$ where $\boldsymbol{r} \leftarrow [-Q,Q]^k$ is chosen at random. Then, there exists a simulator S that takes $O(\lambda \cdot (\log P + \log k))$ bits of information on $\boldsymbol{a}$ and simulates the distribution of $\boldsymbol{s}$ to within statistical distance $2^{-\Omega(\lambda)}$.*

The statistical-to-perfect simulator S in the proof of Lemma 3 uses $2P/(2Q+1) \cdot k < 2$ coordinates of $\boldsymbol{a}$ (their values together with their locations) in expectation. The corollary follows by a Chernoff bound.

3.2 Entropically Secure Batch-OLE Protocol

We first present a "leaky" batch-OLE protocol which guarantees that the sender's input has residual entropy given the (semi-honest) receiver's view. The receiver is guaranteed simulation security against a semi-honest sender.

The receiver starts with input $m \in \mathbb{Z}_p^k$ and the sender has input $\alpha, \beta \in \mathbb{Z}_p^k$. For convenience, one can imagine that α and β are random. At the end of the protocol, the receiver gets $\gamma = \alpha \circ m + \beta$, where all operations are component-wise. Let k, p, q and σ be the parameters of the LPR scheme.

1. The receiver generates a key pair $(sk, rk) \leftarrow \mathsf{LPR.KeyGen}(1^\lambda)$. It encrypts m into a ciphertext $ct \leftarrow \mathsf{LPR.Enc}(sk, m)$, and sends (rk, ct) to the sender.
2. The sender computes a ciphertext $ct' \leftarrow \mathsf{ReRand}(rk, \mathsf{Lin}(ct, \alpha, \beta), Q)$ where $Q = \Omega(pk^2\sigma)$ and returns ct' to the client.
3. The receiver outputs $\gamma \leftarrow \mathsf{Dec}(sk, ct')$.

Correctness follows from the properties of the LPR encryption scheme in Section 2.3. The entropic security statement is captured by the lemma below.

Lemma 4 (Entropically Secure Semi-Honest Batch OLE). *Let the parameters $k = k(\lambda), p = p(\lambda), \sigma = \sigma(\lambda)$, $Q = \widetilde{\Omega}(pk^2\sigma)$ and $q = \Omega(pQ)$. Conditioned on the receiver's view, the sender input α has residual entropy at least $H_\infty(\alpha) - O(\lambda \mathsf{log} Q)$.*

Proof. The receiver's view consists of the LPR secret key s, the public polynomials $a, a_1, \ldots, a_\ell$, the error polynomials $e_1, \ldots, e_\ell$, the input m (which we collectively denote by view_0) and the sender message ct'. The latter is

$$ct' = (a', b') = (\widehat{\alpha}a + \sum_{i=1}^{\ell} r_i a_i + r_0, \widehat{\alpha}b + \Delta\widehat{\beta} + \sum_{i=1}^{\ell} r_i b_i + f)$$

$$= \left(\widehat{\alpha}a + \sum_{i=1}^{\ell} r_i a_i + r_0, (\widehat{\alpha}a + \sum_{i=1}^{\ell} r_i a_i + r_0)s + (\widehat{\alpha}e + \sum_{i=1}^{\ell} r_i e_i + f) + \Delta(\widehat{\alpha}\widehat{m} + \widehat{\beta})\right)$$

Note that ct' can be generated given $A := \widehat{\alpha}a + \sum_{i=1}^{\ell} r_i a_i + r_0$, $E := \widehat{\alpha}e + \sum_{i=1}^{\ell} r_i e_i + f$ and $\widehat{\alpha}\widehat{m} + \widehat{\beta}$. Since $\widehat{\beta}$ is random and independent of $\widehat{\alpha}$, so is $\widehat{\alpha}\widehat{m} + \widehat{\beta}$. Since the coordinates of $\widehat{\alpha}e + \sum_{i=1}^{\ell} r_i e_i$ are bounded by $\widetilde{O}(pk\sigma)$, an application of the gentle noise-flooding lemma (Corollary 1) tells us that E can be simulated given $O(\lambda)$ of its coordinates, and therefore $O(\lambda \mathsf{log} Q)$ bits. An application of Lemma 1 tells us that $\widetilde{H}_\infty(\alpha | A, E) \geq H_\infty(\alpha | A) - O(\lambda \mathsf{log} Q)$. (We implicitly condition all entropy expressions on view_0.) Finally, the regularity lemma (Lemma 2) tells us that A is $2^{-\Omega(\lambda)}$-close to uniform. Putting this together, we get that $\widetilde{H}_\infty(\alpha | A, E) \geq H_\infty(\alpha) - O(\lambda \mathsf{log} Q)$. □

Finally, we note that for a sufficiently large value of k (as a function of λ), the residual entropy is a constant fraction of the entropy of α.

3.3 Our Batch-OLE and Batch-OT Schemes

AQO Batch-OLE. Block et al. [BGMN18], building on [IKOS09a, BMN18b], showed a compiler that converts leaky OLE to fresh OLE. Our main observation is that their compiler preserves asymptotic quasi-optimality as long as one uses an error-correcting code that permits quasi-linear time encoding and erasure-decoding, both of which are satisfied by the Reed-Solomon code. This gives us the following theorem. We defer details and concrete optimizations to the full version of this paper.

Theorem 1. *There exists an asymptotically quasi-optimal BOLE protocol under the 2^λ-hardness of the RLWE assumption.*

AQO Batch-OT. Since our Batch-OLE protocol works over polynomial-sized fields, we can get a batch-OT protocol by naïvely embedding a single-bit OT into a single instance of OLE over $\mathbb{Z}_p$. The naïve embedding loses a factor of $\mathsf{log}\ p$ in the rate. We remark that it may be possible to reclaim this and achieve a constant rate by working with an extension field of $\mathbb{F}_2$, i.e., $\mathbb{F}_{2^\ell}$ for some

ℓ, and using ideas from [CCXY18, BMN18a] to embed $\mathbb{F}_2^{\ell'}$ into $\mathbb{F}_{2^\ell}$. We leave the exploration of this avenue to a future work. An alternative approach that achieves a near-constant rate is described in Section 4.1.

4 AQO Batch-OLE: The Malicious Setting

In this section, we show how to achieve a two-round AQO leaky batch-OLE that is entropically secure against a *malicious* receiver. In particular, we will show that for every (possibly maliciously chosen) receiver message, the sender input $\mathbf{a}$ has residual entropy conditioned on his message to the receiver. The sender will be assumed to be semi-honest. We defer the exact notion of entropic security to Theorem 2 and instead start with the protocol itself.

4.1 Entropically Secure OLE against a Malicious Receiver

Our starting point is to develop an additively homomorphic encryption scheme with good "post-evaluation rate", namely one where the size of *homomorphically evaluated ciphertexts* are approximately the same as the size of the messages they encrypt. Such schemes were developed very recently in a collection of independent works [BDGM19, GH19, DGI+19]. We observe that a simple tweak on an encryption scheme due to Peikert, Vaikuntanathan and Waters [PVW08] *already* gives us good post-evaluation rate together with good concrete efficiency. In contrast, all the cited works construct somewhat more complex and concretely less efficient schemes. We do pay a price, namely, freshly encrypted ciphertexts are not rate-1; in fact, they are somewhat larger than they would be otherwise. Yet, this does not matter much for us: indeed for our application to entropically secure OLE, only the size of the homomorphically evaluated ciphertext matters.

Our scheme, denoted EntOLE, is parametrized by a dimension k, plaintext modulus p, ciphertext (Ring-LWE) modulus q and a noise parameter σ. We define two additional parameters: η that will govern the message sizes, and a compressed ciphertext modulus q'. One challenge that must be overcome to achieve a low post-evaluation rate is a reduction of the ciphertext modulus. Since our homomorphic computation must support one plaintext-ciphertext multiplication, our starting ciphertext modulus q must be greater than p^2, since we must have $\log p$ bits for the message and an additional $\log p$ bits to account for the growth of the error term. The resulting ciphertext modulus q' will only be a few bits larger than p. To switch to this modulus we employ the modulus reduction operation from Brakerski & Vaikuntanathan [BV11a].

Modulus Reduction [BV11a, BGV12]. For ciphertext moduli q, q' such that q' divides q, the modulus reduction operation $\mathsf{ModRed}_{q,q'}$ takes in an element $a \in \mathcal{R}_q$ and outputs $a' \in \mathcal{R}_{q'}$ where $a' = \left\lfloor \frac{q'}{q} a \right\rceil$.

$$\mu = \left\lfloor \frac{b' - a's \mod q'}{\Delta'} \right\rceil$$

We now describe a secret-key linearly homomorphic encryption scheme with post-evaluation rate close to 1.

- $\mathsf{EntOLE.KeyGen}(1^\lambda, \eta)$: For each $i \in [\eta]$, sample $s_i \leftarrow \chi$ and output

$$sk = (s_1, \ldots, s_\eta) \in \mathcal{R}_q^\eta$$

- $\mathsf{EntOLE.Enc}\,(sk, (\mu_1, \ldots, \mu_\eta))$: Takes as input a secret key and a message vector $(\mu_1, \ldots, \mu_\eta)$ where each μ_i is in $\mathbb{Z}_p^k$. For $i \in [\eta]$, sample $a_i \leftarrow \mathcal{U}(\mathcal{R}_q)$, where $\mathcal{U}$ denotes the uniform distribution. Let $\boldsymbol{a}$ denote the column vector of length η consisting of the a_i polynomials. Similarly, let $\boldsymbol{s}$ be a row vector of length η consisting of the secret polynomials. Define the matrix $\mathcal{M} \in \mathcal{R}_q^{\eta \times \eta}$ where $\mathcal{M}[i,j] = 0$ for $i \neq j$ and $\mathcal{M}[i,i] = \mathsf{LPR.Encode}(\mu_i)$ for each $i \in [\eta]$. Finally, sample a matrix $E \in \mathcal{R}_q^{\eta \times \eta}$ such that each $E[i,j] \leftarrow \chi$ is an independently sampled error polynomial. Output the following ciphertext:

$$ct = (\boldsymbol{a} \mid \boldsymbol{a} \cdot \boldsymbol{s} + \mathcal{M}\Delta + E) \in \mathcal{R}_q^{\eta \times (\eta+1)}$$

  (Note that $\boldsymbol{a} \cdot \boldsymbol{s}$ is an $\eta \times \eta$ matrix which is the outer product of the vectors $\boldsymbol{a}$ and $\boldsymbol{s}$.)
- $\mathsf{EntOLE.Eval}(ct, \{\alpha_i\}_{i=1}^\eta, \{\beta_i\}_{i=1}^\eta)$: Takes as input a ciphertext ct and the sender's BOLE inputs where each input is in $\mathbb{Z}_p^k$. Let $\boldsymbol{\alpha}$ be the column vector of length η such that $\boldsymbol{\alpha}[i] = \mathsf{LPR.Encode}(\alpha_i)$ for $i \in [\eta]$. Let $\boldsymbol{\beta}$ be the column vector of length $\eta + 1$ such that $\boldsymbol{\beta}[i+1] = \mathsf{LPR.Encode}(\beta_i)$ for $i \in [\eta]$, and set $\boldsymbol{\beta}[1] = 0$. Compute an encryption of the BOLE result as follows:

$$ct_{bole} = (\boldsymbol{\alpha})^T \cdot ct + \Delta\boldsymbol{\beta} \in \mathcal{R}_q^{\eta+1}$$

  This ciphertext consists of $\eta + 1$ elements in $\mathcal{R}_q$.
  To achieve a high rate for the output ciphertext, we perform the modulus switching operation [BV11a, BGV12] to reduce our modulus from q to q'. We define the output ciphertext $ct_{res} \in \mathcal{R}_{q'}^{\eta+1}$ as follows:

$$ct_{res}[i] := \mathsf{ModRed}_{q,q'}(\mathsf{ct}_{bole}[i]) \quad \text{for } i \in [\eta + 1]$$

- $\mathsf{EntOLE.Dec}(sk, ct_{res})$: Takes as input a vector of η secret keys and a ciphertext in $\mathcal{R}_{q'}^{\eta+1}$. Decryption proceeds by first computing $\overline{\mu}_i$ for $i \in [\eta]$ as follows:

$$\overline{\mu}_i := \left\lfloor \frac{ct_{res}[i+1] - ct_{res}[1] \cdot sk[i]}{\Delta'} \right\rceil$$

  where $\Delta' = \lfloor q'/p \rfloor$. Set $\mu_i := \mathsf{LPR.Decode}(\overline{\mu}_i)$, and output $\{\mu_i\}_{i=1}^\eta \in \mathcal{R}_p^\eta$.

The entropically secure OLE protocol is described in Figure 1. The following theorem states the correctness and security properties of the protocol; the proof is deferred to the full version of this paper.

Entropically Sender-Secure Batch OLE protocol

Receiver Input: $\mathbf{m} = (m_1, \ldots, m_{k\eta}) \in \mathbb{Z}_p^{k\eta}$.
Sender Input: $\mathbf{a} = (a_1, \ldots, a_{k\eta}) \in \mathbb{Z}_p^{k\eta}$ and $\mathbf{b} = (b_1, \ldots, b_{k\eta}) \in \mathbb{Z}_p^{k\eta}$.

Receiver does the following:

- Splits $\mathbf{m}$ into η vectors in $\mathbb{Z}_p^k$ to get $\{\mu_i\}_{i=1}^{\eta}$, where each μ_i is in $\mathbb{Z}_p^k$.
- Samples $sk \leftarrow \mathsf{EntOLE.KeyGen}(1^\lambda, \eta)$.
- Computes $ct \leftarrow \mathsf{EntOLE.Enc}(sk, \{\mu_i\}_{i=1}^{\eta})$ and sends ct to the receiver.

Sender does the following:

- Splits $\mathbf{a}$ and $\mathbf{b}$ each into η vectors in $\mathbb{Z}_p^k$ to get $\{\alpha_i\}_{i=1}^{\eta}$ and $\{\beta_i\}_{i=1}^{\eta}$, where each α_i and β_i is in $\mathbb{Z}_p^k$.
- Compute $ct_{res} = \mathsf{EntOLE.EvalOLE}(ct, \{\alpha_i\}_{i=1}^{\eta}, \{\beta\}_{i=1}^{\eta})$ and return ct_{res} to the receiver.

Receiver computes $\{y_i\}_{i=1}^{\eta} \leftarrow \mathsf{EntOLE.Dec}(sk, ct_{res})$ and concatenates the result vectors into a single vector $\mathbf{y} \in \mathbb{Z}_p^{k\eta}$. Outputs $\mathbf{y}$.

Fig. 1: Entropically Secure Batch-OLE Scheme

Theorem 2. *Fix $\eta \in \mathbb{N}$. For a security parameter λ, let k, σ, q be parameters that give 2^λ security of ring LWE. Let p be the plaintext modulus, and let q' be a ciphertext modulus such that p divides q' and q' divides q. In addition, let $q' > pk\sigma$ and let $q > p^2k^2\sigma$. Then, there exists a BOLE protocol with batch size $n = k\eta$ with the following properties.*

1. *The communication from the receiver to the sender is $(\eta+1)\eta \cdot k \cdot \log\, q$ bits.*
2. *The communication from the sender to the receiver is $k(\eta+1)\log\, q'$ bits.*
3. *The receiver's runtime is $\Theta(\eta^2 k \log(k)\log(q))$.*
4. *The sender's runtime is $\Theta(\eta^2 \log(k)\log(q) + \eta k \log(k)\log(q'))$.*
5. *For every malicious receiver R^* that outputs a ciphertext $ct \in \mathcal{R}_q^{\eta\times(\eta+1)}$, the entropy of the sender's first input for any distribution of $\mathbf{a}$ is at least*

$$\begin{aligned}\widetilde{H}_\infty(\mathbf{a}|ct_{res}) &\geq \widetilde{H}_\infty(\mathbf{a}) - (\log(q')k(\eta+1) - \log(p)k\eta) \\ &\geq \widetilde{H}_\infty(\mathbf{a}) - O(n\log(k) + k\log(p))\end{aligned} \tag{1}$$

The last bullet shows that with a large enough η, there is considerable residual entropy in $\mathbf{a}$ given the receiver's view. Indeed, $\mathbf{a}$ has entropy $n\log p$, and the residual entropy is (up to multiplicative constants) at least $n\log p - n\log k - k\log p$ which is a constant fraction of the entropy of $\mathbf{a}$ if $p = \mathsf{poly}(k)$ and $\eta = n/k$ is a constant, or even a $1 - 1/\log^c k$ fraction if p is superpolynomial in k and η is polylogarithmic in k. The first four bullets show that the protocol has AQO efficiency.

By relying on a simple extension of [WW06], we also obtain an entropically secure OLE w.r.t malicious sender. In slight more detail, in the "reversed" protocol, the receiver will play the role of the sender and the sender the role of the receiver in the underlying batch OLE. For each instance of the batch OLE, the receiver with input x sets its input as x, r where r is chosen uniformly at random and the sender with input a, b sets its input as a. The sender learns $z = a \cdot x + r$ and sends $w = z + b$ to the receiver which can compute its output as $w - r$. If the underlying batch OLE is entropically secure against a malicious receiver with entropy loss ϵ, then the reversed protocol will be entropically secure against a malicious sender with entropy loss ϵ.

Leaky Batch-OTs from leaky Batch-OLE. We now show how to obtain a entropically secure batch OT protocol from a batch OLE protocol.

We begin by observing that a naïve embedding of n OT instances into n OLE's does not work. From Theorem 2, we have that the entropy loss with a batch size n is $\omega(n)$. The maximum entropy of the sender's message in n OT instances is $2 \cdot n$, therefore, the sender's entire input could potentially be leaked. Next, we provide a tighter reduction from batch-OT to batch-OLE.

Let c be any integer. We will design an m-batch OT protocol that is entropically secure against a malicious receiver where the entropy loss in the sender's "a" message is at most $(1/\mathsf{log}^c(n)) \cdot m$. Our compilation proceeds as follows:

1. We compile a batch OLE with batch size n over a prime p of length $\mathsf{log}^{c+2}\lambda$ bits to n OLEs over the ring modulo $p' = p_1 \cdot p_2 \cdots p_\tau$ where $p_1, \ldots, p_\tau$ are the first $\tau = \mathsf{log}^c\lambda$ prime numbers with the guarantee that except with probability $2^{-\lambda}$ at least $n - \lambda$ OLEs over p' are secure against a malicious receiver. If the original batch OLE is only entropically secure against a malicious receiver w.r.t "a"-message, then the entropy loss in the "a" message will decrease further by at most $\lambda \cdot \mathsf{log}(p')$.
2. Next, we reduce OLE over ring modulo $p' = p_1 \cdot p_2 \cdots p_\tau$ to τ OLEs over each of the primes $p_1, \ldots, p_\tau$ using a standard application of the Chinese Remainder Theorem.
3. Finally, we employ the näive reduction of OLE modulo any prime p to a bit OT, namely, the receiver feeds its input bit b as is in the OLE protocol, while the sender maps its input s_0, s_1 to $a = (s_1 - s_0), b = s_0$.

We provide details only for the first step as the remaining steps follow standard techniques.

The sender and receiver will essentially use their inputs a, b, x modulo p' as their inputs for the OLE modulo p. Recall that p' is the product of the first $\mathsf{log}^c\lambda$ primes,[10] which implies $\mathsf{log}(p') < \mathsf{log}^{c+1}\lambda$. This in turn means the maximum value of $a \cdot x + p$ computed over integers is $O(2^{\mathsf{log}^{c+1}\lambda}) < p$. So the receiver learns $a \cdot x + p$ computed over integers. This however induces a leakage as the receiver is only supposed to learn the value $\mathrm{mod}(p')$. In order to reduce the leakage, we will have the sender modify its inputs to $a' = a, b' = b + r \cdot p'$ where r is chosen

[10] The product of the first n primes is $e^{O(n \log n)}$.

uniformly at random from $[-\lambda \cdot p', \lambda \cdot p']$. By the gentle noise-flooding lemma Lemma 3, we can conclude that the probability that the OLE leaks is at most $O(1/\lambda)$. By a Chernoff bound, we can conclude that except with probability $2^{-\lambda}$ at most λ of the OLE instances are leaky.

We now analyze the entire compilation.We instantiate the batch OLE with batch size $n = k\mathsf{log}^c\lambda$ over a prime p of length $\mathsf{log}^{c+2}\lambda$ bits (where k is the Ring LWE dimension set as λ). Recall from Theorem 2 that the entropy loss of the senders "α"-message is at most $n\mathsf{log}k + k\mathsf{log}p = n\mathsf{log}\lambda + \lambda\mathsf{log}p$. The length of p' is at most $\mathsf{log}^{c+1}\lambda$ bits. Since at most λ OLEs are leaky, the maximum entropy loss in the "α"-message can be bounded by $n\mathsf{log}\lambda + \lambda\mathsf{log}p + \lambda\mathsf{log}(p') = O(\lambda\mathsf{log}^{c+2}\lambda)$.

Finally, we obtain a batch-OT protocol with a batch size of $n \cdot \tau = \lambda\mathsf{log}^{2c}\lambda$ and entropy loss of $O(\lambda\mathsf{log}^{c+2}\lambda)$. Therefore, we have the following theorem.

Theorem 3. *For any constant c, there exists a batch-OT protocol over $n = \lambda\mathsf{log}^{2c}\lambda$ instances such that for every malicious receiver R^*, and arbitrary distribution over sender's inputs, the entropy of sender's "a" input at the end of the protocol is at least: $H_\infty(a) - O(\lambda\mathsf{log}^{c+2}\lambda)$.*

5 AQO Zero-Knowledge Arguments

In this section we construct an AQO zero-knowledge argument system. Our starting point is a stronger version of Cook's theorem that follows implicitly from the PCP literature which we compile into an (honest verifier) ZKPCP. In the next step we convert our ZKPCP into a ZK argument system. The former is achieved based on the MPC-in-the-head paradigm whereas the later transformation uses AQO batch OT protocol to emulate the query phase from the ZKPCP oracle.

We begin by recalling Cook's theorem which states that there exists a pair of algorithms (A, B) where given any Boolean circuit C of size s, A maps C to a 3CNF formula F and B maps an input w for C to an input z for F such that the following properties hold:

1. Algorithms A and B run in time $\mathsf{poly}(s)$.
2. If w satisfies C then z satisfies F.
3. If C is unsatisfiable, then F is unsatisfiable.

Next, we state a stronger version of the Cook's theorem which is implicit in constructions of near-optimal PCPs from the literature [BS08, BCGT13].

Theorem 4. *There exists a pair of algorithms (A, B) and constants $a, b, c \in \mathbb{N}$ where given any Boolean circuit C of size s, A maps C to a 3CNF formula F and B maps an input w for C to an input z for F such that:*

1. *Algorithms A and B run in time $\widetilde{O}(s)$. Let the number of clauses in F be $s \cdot \mathsf{log}^a(s)$.*
2. *If w satisfies C then z satisfies F.*
3. *If C is unsatisfiable, so is F. Furthermore, for any assignment of the variables of F, at most $(1 - 1/\mathsf{log}^b(s))$ fraction of the clauses of F are satisfied.*

4. Finally, each variable in F appears at most $\mathsf{log}^c(s)$ times.

The last property can be enforced generically by replacing each variable in F by a distinct variable and then using expander-based consistency checks [PY91].

Next, we design an honest verifier (HV) ZKPCP for the language $\mathcal{L}_C$ by relying on Theorem 4 and the MPC-in-the-head paradigm [IKOS07,IKOS09b]. Note that prior approaches for converting PCP to (HV) ZKPCP either used an ad-hoc and inefficient approach [DFK+92] or used MPC-in-the-head for achieving PCP of proximity with a focus on feasibility and did not attempt to optimize the asymptotic efficiency [IW14]. Achieving AQO based on MPC-in-the-head requires taking a different approach.

We begin by describing our MPC model and then provide our compilation.

Our MPC Model. In the original work of [IKOS07] which introduced the MPC-in-the-head paradigm, the main results implies a zero-knowledge PCP over a large alphabet for a relation $\mathcal{R}$ starting from any honest majority MPC protocol that computes a functionality related to $\mathcal{R}$. In this work, we consider a specific MPC topology and apply the MPC-in-the-head paradigm. Next, we describe our MPC model and the security.

Consider an arbitrary 3CNF formula F with m clauses and t variables $x_1, \ldots, x_t$. We specify an MPC model for the formula F. In our model, we consider a set of *input clients* $C_1, \ldots, C_d$ and d distinct parties per clause of F, servers $(S_1^i, \ldots, S_d^i)$ $(i \in [m])$, an aggregator party A and an output client o. Only the input clients will receive inputs and the final output will be output by the output client. Each input client receives a share corresponding to each variable in F. Namely, C_i receives as input $(x_i^1, \ldots, x_i^t)$ for $i \in [d]$. At the onset of the protocol, each client C_i transmits x_i^j to server S_i^k if the k^{th} clause of F contains the literal x_j. In other words, for every $k \in [m]$, the servers $S_1^k, \ldots, S_d^k$ have the shares corresponding to the (three) literals occurring in the k^{th} clause. Upon receiving inputs from the input clients, the servers $S_1^k, \ldots, S_d^k$ securely compute the functionality f specified by k^{th} clause, where the assignments to the literals are obtained by first XORing the corresponding shares from the servers. We denote the MPC protocol that realizes f by Π. Namely, if the j^{th} literal occurs in the k^{th} clause, the assignment is given by $\bigoplus_{i=1}^{d} x_i^j$. All servers learn the result, and the result of the computation is then forwarded to the aggregator party by server S_1^k for each k. Then, the aggregator party computes the AND of the k values its received and relays that to the output client. The output client finally outputs whatever it receives from the aggregator.

From the description of the MPC model, it follows that if the input clients are given as inputs (XOR) additive shares of the assignment of the variables in F, and all parties behave honestly, then the result output by the output client is the evaluation of F under the corresponding assignment.

Security model: We will require the protocol to be secure against a passive corruption of at most $\lfloor \frac{d-1}{2} \rfloor$ servers among $S_1^k, \ldots, S_d^k$ for any k. In particular, this means the input clients and aggregator cannot be corrupted. It now follows that if we instantiate Π with any honest majority MPC protocol secure against

passive adversaries we have that no adversary can learn anything beyond the result output by the output client.

Compiling to ZKPCP. We defer the proof of Theorem 5 to the full version.

Theorem 5. *Given a Boolean circuit of size s, there exists a non-adaptive $s \cdot \mathsf{log}^a(s)$-query AQO honest verifier ZKPCP over the binary alphabet. More precisely, the PCP achieves perfect simulation w.r.t an honest verifier, soundness $2^{-\frac{s}{\mathsf{log}^b(c)}}$ and a proof of size $s \cdot \mathsf{log}^c(s)$, where a, b, c are constants.*

5.1 AQO-Honest Verifier ZK from AQO-Honest Verifier ZKPCP

In this section, we transform the AQO-honest verifier ZKPCP into an (interactive) AQO honest-verifier ZK proof using the entropic-secure batch OT protocol discussed in Section 4.

We consider an interactive ZK proof where the verifier queries each bit of the PCP via the OT protocol, where the verifier sets its input as 1 if it wants to query the proof bit and 0 otherwise. The prover on the other hand sets the sender's input as the proof bit corresponding to the receiver's 1 input and a random bit corresponding to the receiver's 0 input. It now follows that the honest-verifier zero-knowledge property follows directly from the security of the OT protocol against semi-honest receivers. Furthermore, if the underlying batch OT and the honest verifier ZKPCP are AQO then the resulting ZK will also be AQO.

Ideally, we would like to conclude the soundness of the ZK protocol from the soundness of the underlying ZKPCP. However, we instantiate the underlying batch OT protocol with one that is only entropically secure. Nevertheless, since the entropic loss in sender's privacy (played by the ZK verifier) against a malicious receiver (played by the ZK prover) in the underlying OT is information-theoretic, we can argue that the soundness loss can be bounded by the loss in entropy of the OT sender's input.

In more detail, the verifier first runs the underlying PCP verifier to obtain the set of queries. It next sets its input to the OT as $x \oplus r, r$ where x is a bit vector such that $x_i = 0$ if the PCP Verifier queries that location and 1 otherwise. The vector r is set uniformly at random. The sender, on the other hand, sets each element of its input vector a uniformly at random. The prover receives $a \cdot x \oplus r$ at the end of the OT executions. It then transmits $y = a \cdot x \oplus r \oplus b$ where b is the vector incorporating the symbols of the PCP proof. Finally, the verifier will be able to retrieve the i^{th} proof bit by computing $y_i \oplus r_i$.

Zero-knowledge against an honest verifier follows directly from the computational privacy of the underlying OT protocol.

We argue soundness next. Specifically, given a (possibly unbounded) prover $\mathcal{P}^*$ for the HVZK, we construct a PCP prover $\mathcal{B}$ that internally incorporates $\mathcal{P}^*$, runs an honest interaction with $\mathcal{P}^*$, emulating the honest verifier with input x, extracts the PCP as $y \oplus a \cdot x \oplus r$ by extracting a and r (in exponential time) from the OT transcript and feeds that as the PCP oracle.

Let h be the min-entropy of the distribution of the honest PCP verifier queries. We make a simplification assumption that holds for most ZKPCPs in the literature including the one built in the previous section. Namely, we assume that there is a family of subsets Q of indices (of PCP locations) such that the verifier's query distribution is uniform over Q. In particular, this implies $h \geq \mathsf{log}(|Q|)$.[11]

Let the soundness in the real world where the prover and verifier interact using the protocol be ϵ. We now identify the success probability of $\mathcal{B}$ as a function of ϵ and bound it by the soundness of the ZKPCP to conclude the soundness of the HVZK. We consider an intermediate experiment which proceeds like the real world, where the honest verifier after the OT protocol resamples its random tape consistent with the transcript and the leakage (in possibly exponential time) and uses the new random tape to verify the proof. The soundness of this verifier must be identical to the soundness of the real verifier. By construction, in an execution of the intermediate experiment, the distribution of the verifier's queries is the conditional distribution of the honest PCP verifier query distribution conditioned on the partial transcript of the OT interaction. Let X' denote this distribution. By Theorem 3 and observing h is $\widetilde{\Omega}(n)$, we can conclude that

$$H_\infty(X') \geq \left(1 - \frac{1}{\mathsf{log}^c(n)}\right) \cdot h$$

We know that the soundness of the ZKPCP system is 2^{-s}. This implies that for any proof oracle generated by a prover, the number of queries in Q on which the verifier accepts a false proof, i.e. bad queries is at most $|Q|/2^s$. Furthemore, given an OT transcript, each of the $|Q|/2^s$ bad queries can have a probability mass of at most $1/2^{H_\infty(X')}$ in X' by the definition of min-entropy. Therefore, we have

$$\epsilon \leq \frac{|Q|}{2^s} \cdot \frac{1}{2^{H_\infty(X')}} = \frac{2^h}{2^s} \cdot \frac{1}{2^{(1-1/\mathsf{log}^c(n))\cdot h}} = \frac{1}{2^{s-h/\mathsf{log}^c(n)}}$$

We have that $|Q|$ is $2^{\widetilde{O}(s)}$ as the length of the ZKPCP is $\widetilde{O}(s)$. Therefore by setting c appropriately we have that soundness is $2^{-\widetilde{\Omega}(s)}$.

Security against Malicious Verifiers. The ZK protocol described above is insecure if the verifier acts dishonestly. In particular, it may query beyond the privacy threshold of the underlying ZKPCP and violate the zero-knowledge property. In order to enforce correct behaviour and restrict a dishonest verifier to a certain query pattern, we add another phase in which the verifier commits to its randomness used both for sampling the PCP queries and for generating the OT messages. As the verifier does not have any secret input, it can reveal (decommit) this randomness upon concluding the OT phase and the prover can check if the verifier sampled the queries and participated in the OT correctly. (In fact,

[11] For example, in the classic MPC-in-the-head based ZKPCP [IKOS07], the verifier queries a random t subset out of n. Here Q contains all t subsets of $[n]$.

to enforce correct sampling of the verifier's randomness, the parties run a coin-tossing in the well, where only the verifier learns the outcome of the coin-tossing. This protocol can be implemented using commitments schemes). Recall that the prover sends the masked proof at the end of the OT phase. The prover needs to check the verifier's randomness prior to sending this message as the verifier could cheat in the OT phase, learn the proof and abort before revealing its randomness. At the same time, if the prover sees the verifier's PCP queries before sending the masked proof it can cheat. We prevent this by requiring the prover to first commit to its masked proof at the end of the OT phase before the verifier reveals its randomness and the decommit to the proof after it checks the verifier's actions in the OT phase. Our complete protocol can be found in Figure 2. We conclude with the following theorem whose proof is deferred to the full version.

Theorem 6. *Let* $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ *be an AQO honest verifier ZKPCP system (cf. Theorem 5),* Com *be an AQO commitment scheme,* Com_h *be an AQO statistically hiding commitment scheme and* Π_{OT} *be a entropic-secure AQO batch-OT scheme (cf. Section 4). Then the interactive proof from Figure 2 is a ZK argument system with soundness error* $2^{-\Omega(s)}$.

Acknowledgements. We thank Henry Corrigan-Gibbs for helpful comments and Hemanta Maji for answering our questions on [BGMN18]. C. Hazay was supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime MinisterâĂŹs Office, and by ISF grant No. 1316/18. Y. Ishai was supported in part by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20. L. de Castro and V. Vaikuntanathan were supported by grants from MIT-IBM Watson AI Labs and Analog Devices, by a Microsoft Trustworthy AI grant, and by DARPA under Agreement No. HR00112020023.

References

AHI+17. Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In *ITCS 2017*, volume 67, pages 7:1–7:31, 2017.

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS 2017*, pages 2087–2104. ACM, 2017.

AIK08. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc0. *Comput. Complex.*, 2008.

AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT 2001*, pages 119–135, 2001.

AJL+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, 2012.

Input: Public Boolean circuit C and statement $x \in \mathcal{L}_C$ for both, and a witness $w \in \mathcal{R}_C$ for the prover $\mathcal{P}$.

Building blocks: (1) AQO honest verifier ZKPCP system $(\mathcal{P}_{\text{PCP}}, \mathcal{V}_{\text{PCP}})$ (cf. Theorem 5). (2) AQO commitment scheme Com (3) AQO statistically hiding commitment scheme Com_h (4) AQO semi-honest batch OT Π_{OT}.

The protocol:
Coin-Tossing in the Well. The parties engage in a coin-tossing protocol where only the verifier learns the outcome of the coin-tossing R which it sets as its random tape for the OT Phase.

The OT Phase: Let $|\pi| = \tau$. Then the parties engage in τ instances of OT protocol Π_{OLE} where the prover plays the role of the receiver and the sender plays the role of the sender. The verifier runs $\mathcal{V}_{\text{PCP}}$ to obtain a set of query positions. The verifier chooses its input to the i^{th} OT as $(x_i \oplus r_i, r_i)$ where x_i is set to 0 if the PCP verifier queries the i^{th} location and 1 otherwise and r_i is chosen uniformly at random from $\{0, 1\}$. The prover on the other hand chooses its input to the i^{th} OT instance a_i uniformly at random. Let the vector $\mathbf{w}$ denote the output of the prover in the τ OT instances.

Committing to the PCP: Prover $\mathcal{P}$ invokes the ZKPCP prover $\mathcal{P}_{\text{PCP}}$ on (x, w) and generates a PCP proof vector $\mathbf{b}$. The prover commits to $\mathbf{y} = \mathbf{w} \oplus \mathbf{b}$ using Com.

The Reveal Phase: The verifier decommits to R. If the verifier successfully decommits and R is consistent with an honest behavior of the verifier in the OT phase, then the prover decommits to $\mathbf{y}$.

Concluding the Output: For every index i the PCP verifier queries, the verifier identifies the proof bit as $y_i \oplus r_i$. The verifier then runs the PCP verifier on the responses and accepts iff the PCP verifier accepts.

Fig. 2: ZK Argument System

BBC+18. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *CRYPTO*, 2018.

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO*, 2019.

BCG+17. Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *ASIACRYPT*, 2017.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In *CRYPTO*, 2020.

BCGT13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In *STOC*, 2013.

BCI+13. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, 2013.

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT*, 2019.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC*, 2016.

BD18. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018, Part II*, pages 370–390, 2018.

BDGM19. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, 2019.

BDOZ11. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, 2011.

BEP+20. Carsten Baum, Daniel Escudero, Alberto Pedrouzo-Ulloa, Peter Scholl, and Juan Ramón Troncoso-Pastoriza. Efficient protocols for oblivious linear function evaluation from ring-LWE. In *SCN 2020*, 2020.

BGI+17. Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT 2017, Part III*, pages 275–303, 2017.

BGMN18. Alexander R. Block, Divya Gupta, Hemanta K. Maji, and Hai H. Nguyen. Secure computation using leaky correlations (asymptotically optimal constructions). In *TCC*, 2018.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

BIO14. Joshua Baron, Yuval Ishai, and Rafail Ostrovsky. On linear-size pseudorandom generators and hardcore functions. *Theor. Comput. Sci.*, 554:50–63, 2014.

BIP+18. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In *TCC*, 2018.

BISW18. Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal snargs via linear multi-prover interactive proofs. In *EUROCRYPT*, 2018.

BKLP15. Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS*, 2015.

BLNS20. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-pcp approach to succinct quantum-safe zero-knowledge. In *CRYPTO*, 2020.

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO*, 2019.

BMN18a. Alexander R. Block, Hemanta K. Maji, and Hai H. Nguyen. Secure computation with constant communication overhead using multiplication embeddings. In *INDOCRYPT*, 2018.

BMN18b. Alexander R. Block, Hemanta K. Maji, and Hai H. Nguyen. Secure computation with constant communication overhead using multiplication embeddings. In *INDOCRYPT*, 2018.

BPMW16. Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In *CRYPTO*, 2016.

BS08. Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

BV11a. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

BV11b. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011.

CCXY18. Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. Amortized complexity of information-theoretically secure MPC revisited. In *CRYPTO*, 2018.

CDI$^+$. Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable noninteractive secure computation. In *CRYPTO 2019*.

CY21. Alessandro Chiesa and Eylon Yogev. Subquadratic snargs in the random oracle model. In *CRYPTO 2021, Part I*, pages 711–741, 2021.

dCJV21. Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. Fast vector oblivious linear evaluation from ring learning with errors. In *WAHC*, 2021.

DFK$^+$92. Cynthia Dwork, Uriel Feige, Joe Kilian, Moni Naor, and Shmuel Safra. Low communication 2-prover zero-knowledge proofs for NP. In *CRYPTO*, 1992.

DGI$^+$19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO*, 2019.

DIK10. Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, 2010.

DORS08. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

DR02. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

DS16. Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In *EUROCRYPT*, 2016.

Gam85. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC 2009*, pages 169–178. ACM, 2009.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, 2013.

GH19. Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC*, 2019.

GHKW17. Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In *TCC*, 2017.

GHS12. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, 2012.

GHV10. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In *CRYPTO*, 2010.

GIP+14. Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *STOC*, 2014.

GKPV10. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, 2010.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, 1987.

Gol00. Oded Goldreich. Candidate one-way functions based on expander graphs. *Electron. Colloquium Comput. Complex.*, (90), 2000.

Gol04. Oded Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*, pages 321–340, 2010.

HIMV19. Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. Leviosa: Lightweight secure arithmetic computation. In *CCS 2019*, pages 327–344. ACM, 2019.

HKE13. Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *CRYPTO*, 2013.

HKL+12. Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In *FSE 2012*, 2012.

IKO07. Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *CCC 2007*, pages 278–291, 2007.

IKO+11. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and Jürg Wullschleger. Constant-rate oblivious transfer from noisy channels. In Phillip Rogaway, editor, *CRYPTO*, 2011.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.

IKOS08. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, 2008.

IKOS09a. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Extracting correlations. In *FOCS*, 2009.

IKOS09b. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, 2008.

IW14. Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In Yehuda Lindell, editor, *TCC*, 2014.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC 1992*, pages 723–732, 1992.

Lin16. Yehuda Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. *J. Cryptol.*, 29(2):456–490, 2016.

LM06. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, 2006.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, 2013.

LPS08. Yehuda Lindell, Benny Pinkas, and Nigel P. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *SCN*, 2008.

LS. Alex Lombardi and Luke Schaeffer. A note on key agreement and non-interactive commitments. Cryptology ePrint Archive, Report 2019/279.

LS18. Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT*, 2018.

Mic02. Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *FOCS*, 2002.

Moo16. Dustin Moody. Post-quantum crypto: NIST plans for the future, 2016.

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. In *(FOCS 2004*, pages 372–381. IEEE Computer Society, 2004.

MS20. Daniele Micciancio and Jessica Sorrell. Simpler statistically sender private oblivious transfer from ideals of cyclotomic integers. In *ASIACRYPT 2020, Part II*, pages 381–407, 2020.

MV15. Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. *J. ACM*, 62(6):46:1–46:29, 2015.

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *SODA*, pages 448–457, 2001.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008.

PY91. Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *JCSS*, 43(3):425–440, 1991.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

RR21. Noga Ron-Zewi and Ron Rothblum. Proving as fast as computing: Succinct arguments with constant prover overhead. *ECCC*, page 180, 2021.

WRK17. Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In *ACM CCS*, 2017.

WW06. Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In *EUROCRYPT*, 2006.

XZZ+19. Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO 2019, Part III*, pages 733–764, 2019.

ZXZS20. Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, 2020.