Non-Interactive Secure Computation of Inner-Product from LPN and LWE

Geoffroy Couteau[1] and Maryam Zarezadeh[2]

[1] CNRS, IRIF, Université Paris Cité, Paris, France
couteau@irif.fr
[2] Barkhausen Institut, Dresden, Germany
maryam.zarezadeh@barkhauseninstitut.org

Abstract. We put forth a new cryptographic primitive for securely computing inner-products in a scalable, non-interactive fashion: any party can broadcast a public (computationally hiding) encoding of its input, and store a secret state. Given their secret state and the other party's public encoding, any pair of parties can *non-interactively* compute additive shares of the inner-product between the encoded vectors.

We give constructions of this primitive from a common template, which can be instantiated under either the LPN (with non-negligible correctness error) or the LWE (with negligible correctness error) assumptions. Our construction uses a novel twist on the standard non-interactive key exchange based on the Alekhnovich cryptosystem, which upgrades it to a non-interactive inner product protocol almost for free. In addition to being non-interactive, our constructions have linear communication (with constants smaller than all known alternatives) and small computation: using LPN or LWE with quasi-cyclic codes, we estimate that encoding a length-2^{20} vector over a 32-bit field takes less that 2s on a standard laptop; decoding amounts to a single cheap inner-product.

We show how to remove the non-negligible error in our LPN instantiation using a one-time, logarithmic-communication preprocessing. Eventually, we show to to upgrade its security to the malicious model using new sublinear-communication zero-knowledge proofs for low-noise LPN samples, which might be of independent interest.

1 Introduction

In this work, we put forth a new approach for non-interactive secure computation of inner products, one of the most basic and fundamental functionalities in secure computation. Our approach can be instantiated under either the learning parity with noise (LPN) or the learning with error (LWE) assumptions, two of the most important post-quantum assumptions. It builds upon a simple but powerful observation: a well-chosen tweak of the Alekhnovich key exchange [4] turns it into a non-interactive secure protocol for approximately computing inner products. Borrowing tools from the recent line of work on pseudorandom correlation generators [15–17], we show how to turn this into full fledged secure protocols for inner product, using a small preprocessing phase with communication much smaller than the length of the vectors, both in the semi-honest and in the malicious setting.

1.1 Secure Inner-Product Made as Easy as Non-Interactive Key Exchange

To better capture the attractive efficiency features of our protocols, we introduce the notion of *non-interactive inner product* (NIIP) protocols. At a high level, a NIIP specifies a pair of algorithm, Encode and Decode, where:

- Encode takes an input vector $\mathbf{x} \in \mathbb{F}^n$ over some field $\mathbb{F}$, and produces a pair $(\mathsf{pk}_{\mathbf{x}}, \mathsf{sk}_{\mathbf{x}})$. $\mathsf{pk}_{\mathbf{x}}$ is the *public encoding*, and $\mathsf{sk}_{\mathbf{x}}$ is the secret state. All parties can publicly reveal the encodings $\mathsf{pk}_{\mathbf{x}}$, since they computationally hide their vectors $\mathbf{x}$.
- Decode takes as input a public encoding $\mathsf{pk}_{\mathbf{x}}$, and a secret state $\mathsf{sk}_{\mathbf{y}}$, and outputs a value z, such that the following holds: $z = \mathsf{Decode}(\mathsf{pk}_{\mathbf{x}}, \mathsf{sk}_{\mathbf{y}})$ and $z' = \mathsf{Decode}(\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{x}})$ form *additive shares* of the inner product $\mathbf{x}^{\intercal} \cdot \mathbf{y} = z + z'$ over $\mathbb{F}$.

An NIIP provides an appealing way to compute inner products with a minimalistic interaction pattern: multiple parties can compute and publish encodings of their input ahead of time, locally keeping a secret state. Then, whenever two parties want to securely compute the inner product between their inputs, they can locally and non-interactively decode the other party's public encoding with their own secret state, and obtain additive shares of the output. One can compare this interaction pattern to the pattern of non-interactive key exchange: after broadcasting their public keys, any two individuals from a network can locally compute a shared secret key. We achieve exactly the same interaction pattern, but for the significantly more "advanced" functionality of securely computing (shares of) inner products. We believe that this minimalistic interaction pattern makes our construction appealing in many natural scenarios, and allows them to scale more efficiently to large networks of users (which is typically a bottleneck for secure computation).

LPN-based Instantiation. Our primary instantiation of this approach relies on the learning parity with noise assumption. There, we only achieve correctness up to a vanishing (but non-negligible) error term ε, which is of the order of λ^2/n, where λ is a security parameter, and n is the vector dimension. Therefore, our protocol provides non-trivial correctness only for values of $n > \lambda^2$. We note that this is likely to be optimal: an NIIP with a much smaller correctness error would imply an LPN-based key exchange under LPN with noise rate higher than $\sqrt{n}$, which is a famous and long-standing open problem. Furthermore, we improve the protocol in two ways:

– Using an input-independent preprocessing phase with sublinear communication $O(\log n)$ (where the $O(\cdot)$ hides $\mathsf{poly}(\lambda)$ factors), the protocol can be made perfectly correct.
– By developing new types of zero-knowledge proofs with sublinear communication tailored to our protocol, we show how the security of our protocol can be enhanced from semi-honest to malicious, at a small cost. Our new zero-knowledge proofs, which demonstrate knowledge of a sparse vector in the kernel of a matrix with communication sublinear in the dimension (but linear in the sparsity), are of independent interest.

LWE-based Instantiation. Our second instantiation is based on the learning with error assumption. There, we focus on the semi-honest setting, and directly achieve a full-fledged (negligible error) NIIP, without any preprocessing. This makes our protocol highly versatile in environments where it is desirable to minimize interactions. Furthermore, our LWE-based instantiation can be shown to provide information-theoretic security for one of the two parties.

1.2 Efficiency

In addition to their optimal interaction pattern, our LPN-based protocols have linear communication $O(n)$, with concrete small constants. Specifically, the constant is always smaller than 6, and can be asymptotically reduced to $2 + \epsilon$ for arbitrarily small ϵ when n grows (approaching the optimal cost of just exchanging the two vectors in the clear). In terms of computation, using relatively standard variants of LPN (or LWE) with a quasi-cyclic matrix (the LPN assumption with quasi-cyclic codes is relatively well studied [1, 3, 16], and has been used in recent submissions to the post-quantum NIST competition [3, 5, 48]), our protocols have $O(n \cdot \log n)$ computational complexity, where the cost is dominated by that of doing a matrix-vector multiplication with a quasi-cyclic matrix (this boils down to computing FFT's in dimension n). For $n \approx 10^7$, using the library of [23], the full matrix multiplication can be executed in less than 2 seconds on a personal laptop, according to the implementation of [16]. Furthermore, when assuming instead the hardness of LPN with respect to the Druk-Ishai family of linear-time encodable code [34] (a less standard, but plausible assumption), our LPN-based protocol enjoys computational complexity $O(n)$, which is optimal up to a constant factor.

1.3 Comparison to the State of the Art

Many methods from the literature can be used to securely compute inner products. We go through the main options here, and compare them to our result.

From OT/OLE. A first option is to use generic oblivious-transfer-based secure computation for inner product. This works especially well over $\mathbb{F}_2$, since the inner product between n bit vectors can be reduced to n oblivious transfers (OT). Using recent advances in silent OT extension [15–17], this can be done with asymptotic communication approaching three bits per oblivious transfer.

However, things become significantly more complicated over larger fields. To handle multiplications over a larger field $\mathbb{F}$, the standard OT-based method [37] induces a $\log|\mathbb{F}|$ overhead in the total number of OTs, which can quickly get prohibitive. A more efficient alternative is to build on recent advances in batch oblivious linear evaluation (OLE) over general fields, since an inner-product between length-n vectors over $\mathbb{F}$ can be reduced to a batch of n OLEs over $\mathbb{F}$. To our knowledge, the most efficient protocols for generating many OLEs are the work of [18], which constructs a "silent OLE extension" protocol assuming the hardness of *ring*-LPN over a fully-splitting ring, and the result of [9]. Being silent, the protocol of [18] achieves an asymptotically optimal communication of $2n + o(n)$ elements of $\mathbb{F}$, for a computational cost of $\tilde{O}(n)$ operations.

Our protocol achieves essentially the same asymptotic communication, and our computational complexity is also essentially on par with theirs. However, we improve on three core aspects:

– *Communication Pattern.* The protocol of [18] requires running a generic, interactive secure computation protocol to generate the seeds for the silent OLE extension, before running a local expansion and "derandomizing" the pseudorandom OLEs with additional interaction. In contrast, we achieve a minimal interaction pattern, where a single encoding of the input is broadcast simultaneously by all parties.
– *Underlying Assumption.* The protocol of [18] inherently requires a new "ring-LPN with fully splitting ring" assumption. In fact, their starting point is a construction based on a standard variant of LPN (LPN with quasi-cyclic codes, which we use here), which has *superquadratic* computational complexity $\tilde{O}(n^2)$. Then, their new assumption is introduced as a way to overcome this quadratic overhead. In contrast, we directly achieve quasilinear overhead, under the standard LPN assumption over quasi-cyclic codes.
– *Concrete Efficiency.* Measuring the concrete efficiency of [18] is relatively complex, but working out the parameters in the paper, the communication complexity of setting up the correlation is around $40 \cdot n$ for $n = 2^{20}$. For lower values of n, it is much higher, and it drops quickly for higher values of n (e.g. around $3 \cdot n$ for $n = 2^{24}$). In contrast, our setup costs are minimal (e.g. around $0.7 \cdot n$ for $n = 2^{20}$) In practice, this means that this approach will start to outperform our protocol communication-wise only for $n > 2^{24}$.

As for the protocol of [9], their communication overhead is $\sim 33\%$ larger than ours for short-ish vectors (from $6n$ to $8n$ elements of $\mathbb{F}$), and up to 4 times larger asymptotically (from $(2+\varepsilon)n$ to $8n$ elements of $\mathbb{F}$). In addition, their construction requires a dedicated setup phase (while we only need a common random string). Their dedicated setup can be replaced with a PKI setup, at the cost of sacrificing further some efficiency. Other low-communication OLE protocols have been described in [50], but their concrete computational efficiency is significantly lower than that of [18].

From homomorphic encryption. Another standard solution is to rely on linearly homomorphic encryption, such as Paillier encryption [52]. In these solutions, one party encrypts its vector $\mathbf{x}$ and sends it to the other party, who homomorphically computes and sends back a rerandomized encryption of $\mathbf{x}^\intercal \cdot \mathbf{y}$, which the first party decrypts. For extremely large fields ($\log|\mathbb{F}| \gg 2048$), one can achieve the smallest communication across all known alternatives, with a communication of only $(n + o(n))\log|\mathbb{F}|$ bits (i.e., essentially the cost of sending one of the two vectors in the clear), using a rate-1 homomorphic encryption scheme such as Damgård-Jurik [28]. However, this solution is not competitive with the previous approaches for any reasonable field sizes, communication-wise and computation-wise.

Using Ring-LWE-based linearly homomorphic encryption, a recent unpublished work [21] devised a carefully optimized semi-honest OLE protocol. By tailoring their protocol to inner products, we estimate that their protocol can achieve a communication comparable to our semi-honest protocol. This comes at the cost of using PKI setup and not having a non-interactive communication pattern as we do (furthermore, our protocol can be based on LWE rather than Ring-LWE).

1.4 Applications

Inner products are a fundamental operation in many standard privacy-preserving applications. In many of these applications, the non-interactive structure of our new protocol enables a very appealing realization of these applications in a multi-party setting. This includes for example biometric authentication [51] or pattern matching [39] (computing the Hamming distance between two strings can be non-interactively reduced to computing an inner product, since the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ is $\mathsf{HW}(\mathbf{x}) + \mathsf{HW}(\mathbf{y}) - 2 \cdot \mathbf{x}^\intercal \cdot \mathbf{y}$, where HW denotes the Hamming weight). With our non-interactive protocol, each user could publish a compact encoding associated to its fingerprint, and each authority could also have a list of public encodings of authorized fingerprints. Then, users can authenticate themselves with an authority with almost zero communication: the authority and the user locally compute shares of the Hamming distance, and the user reveal his share to the authority (a single field element). If the shares reconstruct to a value below the threshold, the authentication is successful.

Other applications can include distributed data mining and machine learning applications such as finding k-nearest neighbors (KNN) [60], rule mining [32], decision trees [56], support vector machine (SVM) classification [62], or privacy preserving neural network learning [7, 24].

Inner products are also used in secure similarity measure protocols such as secure multi-keyword searchable schemes [45], secure keyword similarity [46], similar document detection for plagiarism prevention, copyright protection and duplicate submission detection (where similar documents between two entities should be detected while keeping documents confidential [41, 49]), or secure profile proximity matching in social networks (e.g. in some applications, a user profile is defined as a vector of integers where attributes correspond to an interest; social proximity is defined as dot product of two user's vectors [25]. Similar methods are used in secure protocols for friend discovery in mobile social networks [33]). In many of these applications, the non-interactive nature of our protocols can allow to design scalable, multi-user variants.

2 Preliminaries

Throughout the paper, we denote the security parameter by λ. We use upper-case letters like M to denote matrices, bold lower-case letters like $\mathbf{v}$ to denote row vectors, and for column vectors we use the transpose $\mathbf{v}^\intercal$. We write $\mathbf{u}^\intercal || \mathbf{v}^\intercal$ two denote the horizontal concatenations of (horizontal) vectors, and $\mathbf{u} /\!/ \mathbf{v}$ to denote vertical concatenation. Eventually, we write $x \overset{\$}{\leftarrow} X$ (resp. $x \overset{\$}{\leftarrow} \mathcal{D}$) to denote that x is uniformly sampled from the set X (resp. randomly sampled according to distribution $\mathcal{D}$). For a finite set S, we denote the uniform distribution on S by $\mathbb{U}(S)$. We denote by Ber_τ the Bernoulli distribution with parameter τ, i.e., $e \sim \mathsf{Ber}_\tau$ means that the random variable e evaluates to 1 with probability τ and to 0 with probability $1 - \tau$. More generally, we write $\mathsf{Ber}_\tau(\mathbb{F})$ to denote the distribution that outputs a uniformly random element of $\mathbb{F}$ with probability τ, and 0 otherwise (note that with this definition, $\mathsf{Ber}_\tau(\mathbb{F}_2) = \mathsf{Ber}_{(1+\tau)/2}$; we ignore this slight discrepancy). We write $\mathcal{D}_0 \overset{c}{\approx} \mathcal{D}_1$ to denote that two (families of) distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable. Eventually, we recall a standard lemma known in the LPN literature as the piling-up lemma:

Lemma 1 (Piling-up Lemma). *For any $0 < \tau < 1/2$ and random variables $(X_1, \cdots, X_n)$ i.i.d. to Ber_τ, it holds that* $\Pr\left[\bigoplus_{i=1}^n X_i = 0\right] = \left(1 + (1 - 2\tau)^n\right)/2$.

2.1 Learning Parity with Noise

The learning parity with noise (LPN) assumption with dimension k, m noisy samples, and noise rate τ states that it is infeasible to distinguish $(A, A \cdot \mathbf{s} + \mathbf{e})$ from random, where A is a random matrix in $\mathbb{F}_2^{m \times k}$, $\mathbf{s}$ is a random length-k vector, and $\mathbf{e}$ is a length-m vector whose entries are sampled from Ber_τ. More generally, the LPN assumption can be formulated with respect to a family of linear codes over an arbitrary field $\mathbb{F}$, in which case it states that it is hard to distinguish a noisy codeword $A \cdot \mathbf{s} + \mathbf{e}$ from random (where A is a generator matrix for a random code from the family). Formally, given a dimension k, number of samples m, and field $\mathbb{F}$, let $\mathsf{Code}(m, k, \mathbb{F})$ be a probabilistic code generation algorithm that outputs a matrix $A \in \mathbb{F}^{m \times k}$ (A is viewed as the generator matrix of a linear code). Furthermore, we let $\mathsf{Code}^\perp(m, m - k, \mathbb{F})$ be a probabilistic code generation algorithm for the dual of Code, which outputs random *parity-check matrices* $B \in \mathbb{F}^{m \times m - k}$ for a random code $A \in \mathsf{Code}(m, k, \mathbb{F})$ (i.e., a full-rank matrix B such that $B^\intercal \cdot A = 0$; B is a generator for the dual of the code generated by A). We define the LPN assumption over $\mathbb{F}$ with respect to a code Code below.

Definition 2 (Learning Parity with Noise). *Fix a field $\mathbb{F} = \mathbb{F}(\lambda)$, dimension $k = k(\lambda)$, number of samples $m = m(\lambda)$, and noise rate $\tau = \tau(\lambda)$. The $\mathsf{LPN}^m_{k,\tau}$ assumption with respect to* Code *states* *that*

$$\{(A,\mathbf{b}) \mid A \xleftarrow{\$} \mathsf{Code}(m,k,\mathbb{F}), \mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau(\mathbb{F})^m, \mathbf{s} \xleftarrow{\$} \mathbb{F}^k, \mathbf{b} \leftarrow A\cdot\mathbf{s}+\mathbf{e}\} \overset{c}{\approx}$$
$$\{(A,\mathbf{b}) \mid A \xleftarrow{\$} \mathsf{Code}(m,k,\mathbb{F}), \mathbf{b} \xleftarrow{\$} \mathbb{F}^m\}$$

The above LPN assumption has an equivalent dual formulation:

Definition 3 (Dual Learning Parity with Noise). *Fix a field $\mathbb{F} = \mathbb{F}(\lambda)$, dimension $k = k(\lambda)$, number of samples $m = m(\lambda)$, and noise rate $\tau = \tau(\lambda)$. The* $\mathsf{dual\text{-}LPN}^m_{k,\tau}$ *assumption with respect to* $\mathsf{Code}^\perp$ *states that*

$$\{(H,\mathbf{b}) \mid H \xleftarrow{\$} \mathsf{Code}^\perp(m,m-k,\mathbb{F}), \mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau(\mathbb{F})^m, \mathbf{b} \leftarrow H^\intercal\cdot\mathbf{e}\} \overset{c}{\approx}$$
$$\{(H,\mathbf{b}) \mid H \xleftarrow{\$} \mathsf{Code}^\perp(m,m-k,\mathbb{F}), \mathbf{b} \xleftarrow{\$} \mathbb{F}^{m-k}\}$$

The following is standard:

Lemma 4. *For any $\mathbb{F}, k, m, \tau$ and code generation algorithm* Code*, the* $\mathsf{LPN}^m_{k,\tau}(\mathbb{F})$ *assumption with respect to* Code *and the* $\mathsf{dual\text{-}LPN}^m_{k,\tau}(\mathbb{F})$ *assumption with respect to* $\mathsf{Code}^\perp$ *are equivalent.*

Standard codes and noise distributions. The classical LPN assumption is recovered by setting $\mathbb{F} = \mathbb{F}_2$ and Code to be the uniform distribution over $\mathbb{F}_2^{m\times k}$. However, the hardness of LPN is commonly assumed for other families of codes in the literature, such as sparse codes [4] (often called the “Alekhnovich assumption”), quasi-cyclic codes (used in several recent submissions to the NIST post-quantum competition [3,5,48]), Toeplitz matrices [36,47], Druk-Ishai codes [34], and many more. All these variants of LPN generalize naturally to larger fields (and LPN is typically believed to be at least as hard, if not harder, over larger fields).

In addition, it is also relatively common to consider alternative noise distributions beyond the Bernoulli noise. The two most standard choices are exact noise (where the noise is sampled uniformly from the set of all $\tau\cdot m$-sparse vectors of $\mathbb{F}^m$) and regular noise (where the noise is a concatenation of $\tau\cdot m$ random unit vectors of length $1/\tau$). See [15,17] for discussions about these alternative noise distributions. We will denote by $\mathsf{XN}_{\tau,m}(\mathbb{F})$ (for eXact Noise) the exact noise distribution, and by $\mathsf{RN}_{\tau,m}(\mathbb{F})$ (for Regular Noise) the regular noise distribution.

Security of LPN and its variants. Numerous attacks on LPN have been devised. Among the most standard attacks are Gaussian elimination, which solves LPN in time and sample complexity $\Theta(1/(1-\tau)^k)$ using $\Theta(k^2)$ memory, and its variants (e.g. pooled Gauss [35], and BKW [13]), and the Information Set Decoding attacks (introduced by Prange [53] and further improved in a long sequence of papers, see e.g. [11, 12]). In this work, we will be interested in variants of LPN with a very low number of samples (linear in the dimension) and a very low noise rate. This has several consequences: first, algorithms such as BKW (which require a very large number of samples) do not apply, and using a regular noise distribution has no known effect on security (in contrast, if the number of samples is at least quadratic in the dimension, attacks such as the Arora-Ge attack [6] can take advantage of the noise structure). Second, in the very low-noise regime, all improved variants of ISD become equivalent to the original (much simpler) algorithm of Prange [58].

Increasing the field size beyond 2 is not known to reduce security (and actually seems to slightly *improve* security with respect to known attacks). When using a different family of linear code, a necessary condition is to be a good code (i.e. a random code from the family has a linear minimum distance with high probability). In the case of quasi-cyclic codes, the strong structure allows for the DOOM attack [55], which slightly reduces security (but can be easily compensated by a small increase in the noise rate). We refer the reader to [15–18] for more detailed discussions on the security of LPN with various types of noise distributions and code ensembles. As a rule of thumb, in our parameter setting, all known attacks will have a complexity of the form $2^{O(\tau\cdot m)}$. Hence, fixing the noise rate τ to λ/m for some fixed security parameter λ suffices to achieve exponential security (in λ) against all known attacks.

2.2 Learning with Errors

The learning with errors (LWE) assumption is a close variant of the LPN assumption. In essence, and using our generalized definition of LPN, the LWE assumption with dimension k, and m samples, is simply the LPN assumption over $\mathbb{Z}_q$ (for some large enough prime q) with respect to a different noise distribution, which trades *sparsity* for *small magnitude* – i.e., instead of being a distribution over vectors whose entries are mostly zero, the noise distribution samples vectors whose entries are *small in magnitude.* Multiple choices of such noise distributions are standard in the literature, including discrete Gaussian noise, or noise sampled uniformly from $[-B, B]$, where $B \ll q$ is a bound on the magnitude. We call '$\mathsf{LWE}_k^m(\mathbb{Z}_q, \chi)$ with respect to Code' the LWE assumption with dimension k, m samples, over $\mathbb{Z}_q$, with noise vector sampled from χ^m and matrix sampled from Code.

Rounding lemma. Let $\lceil x \rfloor$ denotes the rounding of $x \in \mathbb{R}$ to the nearest integer. We recall the rounding lemma, from [19]:

Lemma 5 (Rounding of noisy shares). *Let (p, q) be two integers with $q/p \in \mathbb{N}$. Fix any $z \in \mathbb{Z}_p$, and (t_0, t_1) be two random elements of $\mathbb{Z}_q$ subject to $t_0 + t_1 = (q/p) \cdot z + e \bmod q$, where e is such that $q/(p \cdot |e|) \geq \lambda^{\omega(1)}$ (λ is a security parameter). Then with probability at least $1 - (|e| + 1) \cdot p/q \geq 1 - \lambda^{-\omega(1)}$, it holds that $R(t_0) + R(t_1) = z \bmod p$, where R is the deterministic rounding function $R : x \to \lceil (p/q) \cdot z \rfloor \bmod p$ and the probability is over the random choice of (t_0, t_1).*

3 Non-Interactive Approximate Inner Product from LPN and LWE

In this section, we describe a general non-interactive protocol for securely computing the inner product between two vectors over $\mathbb{F}^n$, with ε correctness error (independent of the value of the inputs). Our general protocol can be instantiated either under the LPN assumption, in which case the error will be noticeable (but arbitrarily small), or under the LWE assumption (in which case the error can be made negligible). Our protocol enjoys an attractive *key exchange structure*: consider two parties Alice and Bob with respective inputs $(\mathbf{u}, \mathbf{v}) \in \mathbb{F}^n \times \mathbb{F}^n$. The protocol has the following interaction pattern:

- First, Alice and Bob broadcast *encodings* of their respective vectors $(\mathbf{u}, \mathbf{v})$, denoted $\mathsf{pk}_\mathbf{u}$ and $\mathsf{pk}_\mathbf{v}$, and locally keep a private state, which we denote by $\mathsf{sk}_\mathbf{u}$ and $\mathsf{sk}_\mathbf{v}$ respectively. The encodings have length $O(n)$ (the $O(\cdot)$ hides a small constant) and computationally hide the vector they encode.
- Second, Alice (resp. Bob) can locally compute $\alpha \leftarrow \mathsf{Decode}(\mathsf{pk}_\mathbf{v}, \mathsf{sk}_\mathbf{u})$ (resp. $\beta \leftarrow \mathsf{Decode}(\mathsf{pk}_\mathbf{u}, \mathsf{sk}_\mathbf{v})$), where Decode is some deterministic decoding algorithm. The values α and β form additive shares of a value $w \in \mathbb{F}$, where it holds that $w = \mathbf{u}^\intercal \cdot \mathbf{v}$ with probability at least ε (over the random coins of the encoding procedure).

We call a protocol with the above interaction pattern a *non-interactive approximate inner-product protocol* (NIAIP). We formalize this notion below.

3.1 Non-Interactive Approximate Inner Product

Definition 6. *A non-interactive ε-approximate inner-product protocol (ε-NIAIP) over a field $\mathbb{F}$ is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Decode})$ *such that* Decode *is deterministic, and*

- $\mathsf{Setup}(1^\lambda)$: *on input the security parameter 1^λ in unary, outputs a common reference string (CRS)* crs.
- $\mathsf{Encode}(\mathsf{crs}, b, \mathbf{u})$: *on input the CRS* crs*, a bit b, $\mathbf{u} \in \mathbb{F}^n$, outputs a pair $(\mathsf{pk}_b, \mathsf{sk}_b)$;*
- $\mathsf{Decode}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}')$: *on input the CRS* crs*, a public encoding* pk *and a secret state* sk'*, outputs a value $\gamma \in \mathbb{F}$.*

Furthermore, an NIAIP *must satisfy two properties:*

- ***ε-Correctness.*** *For every common reference string* crs *in the domain of* $\mathsf{Setup}(1^\lambda)$ *and every pair $(\mathbf{u}_0, \mathbf{u}_1) \in \mathbb{F}^n \times \mathbb{F}^n$ of vectors, it holds that*

$$\Pr[\mathsf{Decode}(\mathsf{crs}, \mathsf{pk}_0, \mathsf{sk}_1) + \mathsf{Decode}(\mathsf{crs}, \mathsf{pk}_1, \mathsf{sk}_0) = \mathbf{u}_0^\intercal \cdot \mathbf{u}_1] \geq \varepsilon(\lambda, n),$$

where the probability is taken over the joint random coins of both instances of Encode*, $((\mathsf{pk}_b, \mathsf{sk}_b) \xleftarrow{\$} \mathsf{Encode}(\mathsf{crs}, b, \mathbf{u}_b))_{b \in \{0,1\}}$.*

– ***Indistinguishability.*** *For every $b \in \{0,1\}$, the advantage of any (stateful) probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ in distinguishing the following two experiments, parametrized by a bit σ, is negligible:*
  - *$\mathcal{A}$ receives* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ *and outputs $\mathbf{u} \in \mathbb{F}^n$.*
  - *The challenger samples a pair* $(\mathsf{pk}_b, \mathsf{sk}_b) \xleftarrow{\$} \mathsf{Encode}(\mathsf{crs}, b, \mathbf{v})$*, where $\mathbf{v}$ is 0^n if $\sigma = 0$, and $\mathbf{v} = \mathbf{u}$ otherwise. The challenger sends* pk_b *to $\mathcal{A}$.*

A note on syntax. We note that in the above definition, the parties have fixed roles. In a multiparty setting, if all pairs of parties want to compute inner products, this means that they must publish two encodings of their input, one with role 0, and one with role 1. In many applications, however, it is natural to have "type-0" and "type-1" parties (e.g. clients and servers), such that secure computations tasks are only carried between a type-0 and a type-1 party.

3.2 A $(1 - \tau^2 m)$-NIAIP from LPN

We now proceed with the construction of an ε-NIAIP, from the learning parity with noise assumption. The construction is relatively simple in hindsight: it is a natural twist on the Alekhnovich cryptosystem. The construction is parametrized by a field $\mathbb{F}$, and a vector length n. We let $k(n), m(n)$ denote respectively a dimension parameter and a number of samples, both to be specified later (but the reader can think of k and m as linear in n, e.g. $k = 2n$ and $m = 4n$), and $t = t(\lambda, n)$ denote a noise parameter (the reader can consider $t = \lambda$ to be a reasonable choice). Let Code be a probabilistic code generation algorithm. The construction is represented on Figure 1.

– $\mathsf{Setup}(1^\lambda)$: sample $H \xleftarrow{\$} \mathsf{Code}^\perp(m, k+n, \mathbb{F})$ and output $\mathsf{crs} = H$.
– $\mathsf{Encode}(\mathsf{crs}, b, \mathbf{u})$: parse crs as H and sample $\mathbf{r}_b \xleftarrow{\$} \mathsf{Ber}_\tau^m(\mathbb{F})$. If $b = 0$, output $\mathsf{pk}_0 \leftarrow (\mathbf{u} /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0$ and $\mathsf{sk}_0 \leftarrow \mathbf{r}_0$. If $b = 1$, sample $\mathbf{s} \xleftarrow{\$} \mathbb{F}^k$, and output $\mathsf{pk}_1 \leftarrow H \cdot (\mathbf{u} /\!/ \mathbf{s}) + \mathbf{r}_1$ and $\mathsf{sk}_1 \leftarrow (\mathbf{u} /\!/ \mathbf{s})$.
– $\mathsf{Decode}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}')$: output $\mathsf{pk}^\intercal \cdot \mathsf{sk}'$.

Fig. 1. A non-interactive approximate inner-product over $\mathbb{F}$ for vectors of length n

Before we state the theorem, we introduce some notation: let $\mathsf{Code}^\perp_{\mathsf{right}}$ be the code generator that samples a random matrix $H \xleftarrow{\$} \mathsf{Code}^\perp(m, k+n, \mathbb{F})$ (hence $H \in \mathbb{F}^{m \times k+n}$) and outputs the matrix $H_{\mathsf{right}} \in \mathbb{F}^{m \times k}$ which contains the last k columns of H. Furthermore, we say that $\mathsf{Code}^\perp$ is a *nice code* if given H_{right}, there is an efficient algorithm to sample a random matrix H from $\mathsf{Code}^\perp(m, k+n, \mathbb{F})$ whose last k columns are exactly H_{right}[3]. We denote $H \xleftarrow{\$} \mathsf{Code}^\perp|_{H_{\mathsf{right}}}(m, k+n, \mathbb{F})$ this process.

Theorem 7. *Let $\mathsf{Code}^\perp$ be a nice code. Assume that the $\mathsf{dual\text{-}LPN}^m_{m-(k+n),\tau}(\mathbb{F})$ assumption with respect to $\mathsf{Code}^\perp$, and the (primal) $\mathsf{LPN}^m_{k,\tau}(\mathbb{F})$ assumption with respect to $\mathsf{Code}^\perp_{\mathsf{right}}$ both hold. Then the construction* $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Decode})$ *on Figure 1 is an ε-NIAIP, with $\varepsilon \geq 1 - m\tau^2$.*

Proof. We first prove ε-correctness. Observe that for any pair of inputs $(\mathbf{u}_0, \mathbf{u}_1) \in F^n \times \mathbb{F}^n$ and every matrix $H \in \mathbb{F}^{m \times k+n}$, it holds that

$$\begin{aligned}
&\mathsf{Decode}(\mathsf{crs}, \mathsf{pk}_0, \mathsf{sk}_1) + \mathsf{Decode}(\mathsf{crs}, \mathsf{pk}_1, \mathsf{sk}_0)\\
&\quad = \mathsf{pk}_0^\intercal \cdot \mathsf{sk}_1 + \mathsf{pk}_1^\intercal \cdot \mathsf{sk}_0\\
&\quad = ((\mathbf{u}_0 /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0)^\intercal \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + (H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + \mathbf{r}_1)^\intercal \cdot \mathbf{r}_0\\
&\quad = (\mathbf{u}_0^\intercal || \mathbf{0}^\intercal) \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) - \mathbf{r}_0^\intercal \cdot H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + (\mathbf{u}_1 /\!/ \mathbf{s})^\intercal \cdot H^\intercal \cdot \mathbf{r}_0 + \mathbf{r}_1^\intercal \cdot \mathbf{r}_0\\
&\quad = \mathbf{u}_0^\intercal \cdot \mathbf{u}_1 - \mathbf{r}_0^\intercal \cdot H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + (\mathbf{r}_0^\intercal \cdot H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}))^\intercal + \mathbf{r}_1^\intercal \cdot \mathbf{r}_0\\
&\quad = \mathbf{u}_0^\intercal \cdot \mathbf{u}_1 + \mathbf{r}_1^\intercal \cdot \mathbf{r}_0 \text{ (since the transpose of a single field element is itself).}
\end{aligned}$$

Now, since $\mathbf{r}_0$ and $\mathbf{r}_1$ are random Bernoulli noise vectors with rate τ, we have

$$\Pr[\mathbf{r}_1^\intercal \cdot \mathbf{r}_0 = 0] \geq 1 - m \cdot \tau^2,$$

[3] All known LPN-friendly codes satisfy this property.

since $\Pr[\mathbf{r}_1^\intercal \cdot \mathbf{r}_0 = 0] \geq \Pr[r_0^{(i)} \cdot r_1^{(i)} = 0 \forall i \leq m]$, which equal to $1 - \Pr[\exists i, r_0^{(i)} \cdot r_1^{(i)} = 1] \geq 1 - m\tau^2$, using a straightforward union bound and the fact that $\Pr[r_0^{(i)} \cdot r_1^{(i)} = 1] = \tau^2$ for any i.

We now prove indistinguishability, for $b = 0$ and $b = 1$. We proceed in a sequence of games of the form $G^i_{b,\sigma}$:

- Game $G^0_{0,0}$ is the initial game, with bits $b = 0$ and $\sigma = 0$. The challenger samples $H \xleftarrow{\$} \mathsf{Code}^\perp(m, k{+}n, \mathbb{F})$. Upon receiving $\mathbf{u} \in \mathbb{F}^n$ from $\mathcal{A}(\mathsf{crs})$, the challenger returns $\mathsf{pk}_0 \leftarrow 0^{k+n} - H^\intercal \cdot \mathbf{r}_0$, where $\mathbf{r}_0$ is a random Bernoulli noise.
- Game $G^1_{0,0}$: the challenger first receives a challenge, denoted $(H, \mathbf{c})$, for the $\mathsf{dual\text{-}LPN}^m_{m-(k+n),\tau}$ assumption with respect to $\mathsf{Code}^\perp$, where $\mathbf{c}$ is $H^\intercal \cdot \mathbf{e}$ for some noise vector $\mathbf{e}$. Upon receiving $\mathbf{u} \in \mathbb{F}^n$ from $\mathcal{A}(\mathsf{crs})$, the challenger returns $\mathsf{pk}_0 \leftarrow 0^{k+n} - \mathbf{c}$. This game is perfectly indistinguishable from the previous one.
- Game $G^2_{0,0}$ is exactly as Game $G^1_{0,0}$, except that $\mathbf{c}$ is now a random vector from $\mathbb{F}^m$. Observe that distinguishing between $G^1_{0,0}$ and $G^2_{0,0}$ is exactly solving the $\mathsf{dual\text{-}LPN}^m_{m-(k+n),\tau}$ assumption with respect to $\mathsf{Code}^\perp$.
- Game $G^3_{0,0}$: the challenger proceeds as in Game $G^2_{0,0}$, except that it outputs $\mathsf{pk}_0 \xleftarrow{\$} (\mathbf{u}/\!/\mathbf{0}) - \mathbf{c}$. Since $\mathbf{c}$ is a uniformly random vector, this game is perfectly indistinguishable from the previous one.
- Game $G^4_{0,0}$: as the previous one, except that $\mathbf{c}$ is back to being of the form $H^\intercal \cdot \mathbf{e}$ for some noise vector $\mathbf{e}$. Distinguishing this game from $G^3_{0,0}$ is exactly solving the $\mathsf{dual\text{-}LPN}^m_{m-(k+n),\tau}$ assumption with respect to $\mathsf{Code}^\perp$.
- Game $G^0_{0,1}$: this game is simply the initial game with bits $b = 0$ and $\sigma = 1$. Game $G^0_{0,1}$ is perfectly indistinguishable from $G^4_{0,0}$.

From the above, we conclude that the advantage of any polynomial time adversary in the indistinguishability experiment with $b = 0$ is at most twice its advantage against the $\mathsf{dual\text{-}LPN}^m_{m-(k+n),\tau}(\mathbb{F})$ assumption with respect to $\mathsf{Code}^\perp$. We now address the case $b = 1$.

- Game $G^0_{1,0}$ is the initial game, with bits $b = 1$ and $\sigma = 0$. The challenger samples $H \xleftarrow{\$} \mathsf{Code}(m, k{+}n, \mathbb{F})$. Upon receiving $\mathbf{u} \in \mathbb{F}^n$ from $\mathcal{A}(\mathsf{crs})$, the challenger returns $\mathsf{pk}_1 \leftarrow H \cdot (0^n/\!/\mathbf{s}) + \mathbf{r}_1$, where $\mathbf{r}_1$ is a random Bernoulli noise and $\mathbf{s}$ is a random vector from $\mathbb{F}^k$.
- Game $G^1_{1,0}$: the challenger first receives a challenge, denoted $(H_{\mathsf{right}}, \mathbf{c})$, for the $\mathsf{LPN}^m_{k,\tau}$ assumption with respect to $\mathsf{Code}^\perp_{\mathsf{right}}$, where $\mathbf{c}$ is $H_{\mathsf{right}} \cdot \mathbf{s} + \mathbf{e}$ for some random vector $\mathbf{s}$ and some noise vector $\mathbf{e}$. The challenger samples H as $H \xleftarrow{\$} \mathsf{Code}^\perp|_{H_{\mathsf{right}}}(m, k + n, \mathbb{F})$ (which is possible by definition since $\mathsf{Code}^\perp$ is a nice code). Let H_{left} be such that $H = H_{\mathsf{left}} || H_{\mathsf{right}}$. Upon receiving $\mathbf{u} \in \mathbb{F}^n$ from $\mathcal{A}(H)$, the challenger returns $\mathsf{pk}_1 \leftarrow \mathbf{c}$. By construction of $\mathbf{c}$, since $H \cdot (0^n/\!/\mathbf{s}) = H_{\mathsf{right}} \cdot \mathbf{s}$, this game is perfectly indistinguishable from the previous one.
- Game $G^2_{1,0}$ is exactly as Game $G^1_{0,0}$, except that $\mathbf{c}$ is now a random vector from $\mathbb{F}^m$. Observe that distinguishing between $G^1_{0,0}$ and $G^2_{0,0}$ is exactly solving the $\mathsf{LPN}^m_{k,\tau}$ assumption with respect to $\mathsf{Code}^\perp_{\mathsf{right}}$.
- Game $G^3_{1,0}$: the challenger proceeds as in Game $G^2_{0,0}$, except that it outputs $\mathsf{pk}_0 \xleftarrow{\$} H_{\mathsf{left}} \cdot \mathbf{u} + \mathbf{c}$. Since $\mathbf{c}$ is a uniformly random vector, this game is perfectly indistinguishable from the previous one.
- Game $G^4_{1,0}$: as the previous one, except that $\mathbf{c}$ is back to being of the form $H_{\mathsf{right}} \cdot \mathbf{s} + \mathbf{e}$. Distinguishing this game from $G^3_{0,0}$ is exactly solving the $\mathsf{LPN}^m_{k,\tau}$ assumption with respect to Code.
- Game $G^0_{1,1}$: this game is simply the initial game with bits $b = 1$ and $\sigma = 1$. Since $H \cdot (\mathbf{u}/\!/\mathbf{s}) = H_{\mathsf{left}} \cdot \mathbf{u} + H_{\mathsf{right}} \cdot \mathbf{s}$, Game $G^0_{0,1}$ is perfectly indistinguishable from $G^4_{0,0}$.

From the above, we conclude that the advantage of any polynomial time adversary in the indistinguishability experiment with $b = 1$ is at most twice its advantage against the $\mathsf{LPN}^m_{k,\tau}(\mathbb{F})$ assumption with respect to $\mathsf{Code}^\perp_{\mathsf{right}}$. This concludes the proof.

3.3 Non-Interactive Inner Product from LWE

A simple variant of our construction of non-interactive approximate inner-product leads to a construction under the learning with error (LWE) assumption. Unlike its LPN-based counterpart, this variant can actually achieve correctness exponentially close to 1.

Let $\mathbb{F}_p$ be the prime-order field over which we want to compute a non-interactive inner-product. Fix a bound B on the magnitude of the noise. Let $\mathbb{Z}_q$ be a ring, for some multiple q of p of size $q > (m \cdot B^2 + 1) \cdot p \cdot \lambda^{\omega(1)}$. The variant is described on Figure 2. Eventually, we let χ denote a noise distribution. The exact choice of χ does not matter much, but we assume that all entries in a random sample from χ^m belong to $[-B, B]$ with overwhelming probability. Note that we follow an LPN-style description, by viewing the matrix of the LWE assumption as the generator matrix of some linear code over the ring $\mathbb{Z}_q$. While this is not so common in the LWE literature, this viewpoint allows for considerations on the choice of better codes to improve efficiency.

- $\mathsf{Setup}(1^\lambda)$: sample $H \overset{\$}{\leftarrow} \mathsf{Code}^\perp(m, k+n, \mathbb{Z}_q)$ and output $\mathsf{crs} = H$.
- $\mathsf{Encode}(\mathsf{crs}, b, \mathbf{u})$: parse crs as H and sample $\mathbf{r}_b \overset{\$}{\leftarrow} \chi^m$. If $b = 0$, output $\mathsf{pk}_0 \leftarrow (q/p) \cdot (\mathbf{u} /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0$ and $\mathsf{sk}_0 \leftarrow \mathbf{r}_0$. If $b = 1$, sample $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$, and output $\mathsf{pk}_1 \leftarrow H \cdot (\mathbf{u} /\!/ \mathbf{s}) + \mathbf{r}_1$ and $\mathsf{sk}_1 \leftarrow (\mathbf{u} /\!/ \mathbf{s})$.
- $\mathsf{Decode}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}')$: output $\lceil (p/q) \cdot \mathsf{pk}^\intercal \cdot \mathsf{sk}' \rfloor \bmod p$.

Fig. 2. An LWE-based non-interactive inner-product over $\mathbb{F}_p$ for vectors of length n

Theorem 8. *Assuming the $\mathsf{LWE}_k^m(\mathbb{Z}_q, \chi)$ with respect to* Code*, the construction of Figure 2 is an* ε-NIAIP*, with correctness ε negligibly close to* 1.

Proof. The protocol of Figure 2 is identical to the LPN-based protocol of Figure 1, up to two differences:

- $H^\intercal \cdot \mathbf{r}_0$ is used to mask $(q/p) \cdot (\mathbf{u} /\!/ \mathbf{0})$ instead of $(\mathbf{u} /\!/ \mathbf{0})$, and
- the output of Decode is fed to the *rounding procedure* R of the rounding lemma (Lemma 5) which, on input $x \in \mathbb{Z}_q$, outputs $R(x) = \lceil (p/q) \cdot x \rfloor \bmod p$.

Using the same analysis as for the correctness of the LPN-based protocol, if $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ are encodings of two inputs $(\mathbf{u}_0, \mathbf{u}_1) \in \mathbb{F}_p \times \mathbb{F}_p$, we have

$$\mathsf{pk}_0^\intercal \cdot \mathsf{sk}_1 + \mathsf{pk}_1^\intercal \cdot \mathsf{sk}_0 = (q/p) \cdot \mathbf{u}_0^\intercal \cdot \mathbf{u}_1 + \mathbf{r}_1^\intercal \cdot \mathbf{r}_0,$$

where $|\mathbf{r}_1^\intercal \cdot \mathbf{r}_0| \leq m \cdot B^2$. Let $e \leftarrow \mathbf{r}_1^\intercal \cdot \mathbf{r}_0$ denote the *output noise* and $z \leftarrow \mathbf{u}_0^\intercal \cdot \mathbf{u}_1$ denote the target output. The values $\mathsf{pk}_0^\intercal \cdot \mathsf{sk}_1$ and $\mathsf{pk}_1^\intercal \cdot \mathsf{sk}_0$ form random shares of $(q/p) \cdot z + e$ over $\mathbb{Z}_q$ with $|e| \leq m \cdot B^2$. Therefore, by the rounding lemma (Lemma 5), the outputs of Decode form additive shares of $z \in \mathbb{Z}_p$ with overwhelming probability. This concludes the proof of overwhelming correctness.

For security, the second part of the analysis is identical to the security analysis of the LPN-based protocol, and reduces to the $\mathsf{LWE}_k^m(\mathbb{Z}_q, \chi)$ assumption with respect to Code. The first part of the analysis, however, differs in a crucial way: a standard application of the leftover hash lemma shows that $H^\intercal \cdot \mathbf{r}_0$ is *statistically* close to a random vector. Therefore, the NIAIP actually enjoys statistical security for one of the two parties in the LWE setting. The rest of the game hops are identical – one must simply replace invocations of the dual LPN assumption by the statistical argument.

Like its LPN-based counterpart, this protocol leads to an NIAIP over an arbitrary prime order field (and can even be modified to give an inner product protocol over $\mathbb{Z}$); furthermore, it enjoys overwhelming correctness. However, as we will see later, it is possible to upgrade the correctness of the LPN-based NIAIP to perfect correctness, *and* its security to security against malicious adversaries, at a cost *sublinear* in n; this means that, asymptotically, the LPN-based protocol can be made perfectly correct and maliciously secure at negligible cost. In contrast, making the LWE-based protocol secure against malicious adversaries is more challenging, and we leave it to future work.

3.4 From NIAIP to Secure Computation of Inner Product

The natural usecase for NIAIP is to securely compute inner products: two parties $P_0, P1$ publish encodings of their respective inputs $\mathbf{u}$ and $\mathbf{v}$, locally compute shares of the inner product, and exchange their shares to reconstruct the output. An important technicality here is that the NIAIP indistinguishability notion does not directly imply security when revealing the share of P_0 to its opponent P_1. When correctness is overwhelming (as with our LWE-based instantiation), this is not an issue: given the output $\mathbf{u}^\intercal \cdot \mathbf{v}$ and the randomness of P_1, the simulator can compute P_1's share γ_1, and simulate the missing share as $\mathbf{u}^\intercal \cdot \mathbf{v} - \gamma_1$. Due to the overwhelming correctness, the simulation is indistinguishable from the honest protocol.

When using ε-NIAIP with non-negligible correctness error (as with our LPN-based instantiation), however, the *correctness error* translates to a *security loss* for the protocol: the simulation fails with probability $1 - \varepsilon$. Yet, this does not directly imply an attack on the protocol. In fact, for our LPN-based instantiation, we can get perfect simulation by giving the the simulator the error term $\mathbf{r}_1^\intercal \cdot \mathbf{r}_0$. Concretely, this corresponds to allowing the adversary to learn a single sparse linear equation (given by $\mathbf{r}_1$) in the LPN noise vector $\mathbf{r}_0$. In turn, this means that the security reduces to an appropriate *LPN with leakage* assumption. Such variants of LPN are relatively standard, and can in particular be reduced to the standard LPN assumption, albeit with some loss [16, 18].

In an multiparty setting, where P_0 wants to compute the inner product of $\mathbf{u}$ with many other vectors, the leakage can be accumulated across corrupted parties. This translates to a larger loss for the assumption, and the LPN parameters must be adjusted to compensate, as a function of the maximum number of corrupted parties. An alternative solution is to first remove the error instead, using the sublinear-communication preprocessing phase described in Section 4.

3.5 Choosing the Parameters and the Code

Our non-interactive inner-product communicates $k + n + m$ bits ($k + n$ for pk_0 and m for pk_1). The security of our protocol relies on a relatively unusual set of parameters: we need to assume dual LPN with dimension $m - (k + n)$, m samples, noise rate τ with respect to the matrix $H^\intercal$, as well as primal LPN with dimension k, m samples, noise rate τ with respect to the "right half" of H. We will discuss candidate choices for the underlying code afterwards. Regarding the parameters, we set $m-(k+n) = k$ to ensure that both assumptions achieve the same dimension and number of samples, in order to balance security. This implies $m = 2k + n$. From there, the choice of k induces a tradeoff between the noise rate (which must be kept low as the error probability of the protocol is $\tau^2 \cdot m$) and the communication of the protocol (which grows with k): picking a very large $k \gg n$ increase communication but achieves asymptoptically a rate $1/2$ (as m approaches $2k$).

Concrete Parameters. For concrete instantiations, we consider a reasonable middle ground and set $k = n$ (hence $m = 3k$), leading to codes of rate $1/3$. This leads to a protocol with total communication $5n$ bits, only 2.5 times more than the communication of exchanging $\mathbf{u}_0$ and $\mathbf{u}_1$ in the clear. To estimate the concrete noise rate, we rely on the analysis of [15] which provides various formulas to compute lower bounds on the bit complexity of the most standard attacks on LPN. With a rate $1/3$ and using their formulas for the cost of ISD, Gaussian elimination, and low-weight parity-check attacks, we get the following (very close) approximation of the security level: choosing $\tau = \lambda/m$ provides $\lambda - 20$ bits of security (independently of the vector length n). Hence, for example, setting $\lambda = 100$ gives 80 bits of security, and an error probability of $\lambda^2/m = 0.3\%$ for vectors of length $n = 2^{20}$ (for smaller vectors, the error probability increases rapidly: e.g. around 10% for $n = 2^{15}$).

Asymptotic Parameters. Asymptotically, letting $m = 2k + n$ as before, the code rate is k/m for both codes. Let ε be an arbitrarily small constant, and set $k = \varepsilon \cdot n$ and $\tau = \lambda/m$ for a security parameter λ. The best known attack against LPN with code rate $k/m = O(1)$ and noise rate λ/m run in time $2^{O(\lambda)}$ (where the $O(\cdot)$ hides a $1/\varepsilon$ factor). With these parameters, the protocol communicates $3k + 2n = (2 + 3\varepsilon)n$ bits, which is arbitrarily close to the optimal communication of an *insecure* NIAIP that simply reveals the inputs in the clear. Settling for subexponential security in λ can further reduce communication to $2n + o(n)$.

Choosing the Code. It remains to discuss how to choose an appropriate code to instantiate the NIAIP. While the code has no impact on communication, it represents a tradeoff between computation and security. For example, using a uniformly random code leads to a security reduction to the most standard flavor of LPN, but comes at a huge computational cost: the computation scales as $O(n^2)$.

Some variants of LPN are conjectured to be secure with respect to *linear time encodable codes*, where the mapping $\mathbf{x} \to H \cdot \mathbf{x}$ can be computed in linear time (by the transposition principle [14,40], this also implies that the mapping $\mathbf{y} \to H^\intercal \cdot \mathbf{y}$ can be computed in linear time). This is for example the case of primal LPN instantiated with a sparse matrix H, with a constant number of nonzero entry per row, which corresponds to the Alekhnovich assumption [4]. Unfortunately, for this standard choice of linear-time encodable code, the *dual assumption with respect to* $H^\intercal$ turns out to be insecure. This is equivalent to the well-known fact that LDPC codes admit an efficient decoding algorithm.

Fortunately, if we settle for *quasi-linear time* encodable codes, we can circumvent the issue. For example, quasi-cyclic codes can be encoded in time $O(n \cdot \log n)$ using Fast Fourier Transform, and given a generator matrix H for a quasi-cyclic code, LPN is widely conjectured to hold both with respect to H_{right} in its primal form, and with respect to $H^\intercal$ in its dual form. Quasi-cyclic codes have been used in numerous recent works [1,3,16] as well as in submissions to the NIST post-quantum competition [3,5,48]. We note that, when using quasi-cyclic codes, one must account for the speedup given by the DOOM attack [55], which gives a $\sqrt{k}$ speedup for the attacker. To compensate for this attack, we must therefore aim at $\lambda + \log_2 k$ "pre-DOOM" bits of security, which can be done by increasing the noise rate from $(\lambda + 20)/m$ to $(\lambda + 20 + \log_2 k)/m$ with our concrete choice of parameters.

Eventually, one can instantiate the code using the Druk-Ishai family of linear-time encodable codes [34]. While less standard, this provides a plausible candidate where the mapping $\mathbf{y} \mapsto H^\intercal \cdot \mathbf{y}$ is linear-time, and LPN can be conjectured to hold both for H in its primal form, and for $H^\intercal$ in its dual form (the work of [34] provides support for this conjecture). This yields an approximate inner-product protocol with strictly linear communication *and* computation.

4 Removing Correctness Errors via Sublinear Preprocessing

In this section, we show how to convert the LPN-based ε-NIAIP from the previous section into a two-party secure computation protocol for inner product, without correctness error. While the protocol is not an NIAIP anymore, all additional interactions take place during an input-independent preprocessing phase. Furthermore, the amount of computation and communication during this preprocessing phase is sublinear in n (more precisely, it will be of the form $\mathsf{poly}(\lambda) \cdot \log n$).

The ideal functionality $\mathcal{F}_{\mathsf{IP}}$ for secure computation of (shares of) an inner product over a field $\mathbb{F}$ is described on Figure 4 (setting $\varepsilon = 1$). The intuition behind the protocol of this section is natural: the correctness error in the protocol of Figure 1 is due to an additive term $\mathbf{r}_1^\intercal \cdot \mathbf{r}_0$ in the shares locally decoded by the parties. Since the $\mathbf{r}_b$ are sparse vectors, their inner product is zero with high probability $\approx 1 - \lambda^2/m$. To correct the error, the parties will distributively generate noise vectors $(\mathbf{r}_0, \mathbf{r}_1)$ together with additive shares of $\mathbf{r}_1^\intercal \cdot \mathbf{r}_0$. Crucially, this entire preprocessing requires communication and computation sublinear in the vector length n.

4.1 Picking the Right Noise Distribution

While the high level intuition is simple, the (asymptotic and concrete) efficiency of this approach turns out to be extremely sensitive to the noise distribution. In the previous section, we described the protocol using the standard Bernoulli noise distribution, since it allows for a reduction to the most common flavor of LPN. However, Bernoulli noise is a poor choice for allowing efficient preprocessing; using a regular noise distribution insteads allows for a considerably more efficient preprocessing, without harming security.

In a bit more details, setting $\tau = \lambda/m$, a vector $\mathbf{r}_b \xleftarrow{\$} \mathsf{Ber}_\tau(\mathbb{F})^m$ can be written as the sum of $\approx \lambda$ unit vectors. Therefore, securely computing (shares of) the inner product between two such vectors reduces to securely computing λ^2 products of elements of $\mathbb{F}$, and λ^2 secure equality tests between $\log m$-size bitstrings. This is already sublinear in $m = O(n)$, but the λ^2 overhead can incur a significant slowdown.

Instead, we sample $\mathbf{r}_0$ and $\mathbf{r}_1$ from the regular noise distribution: $\mathbf{r}_0$ and $\mathbf{r}_1$ are concatenations of λ random unit vectors. The corresponding variant of LPN, regular LPN, is not known to be any weaker than LPN in our regime of parameters. Let us introduce a few notations: we denote $\mathbf{r}_b = (\mathbf{r}_b^{(1)} /\!\!/ \cdots /\!\!/ \mathbf{r}_b^{(\lambda)})$ for $b = 0, 1$, where the $\mathbf{r}_b^{(i)}$ are unit vectors. Furthermore, we denote by $j_{b,i}$ and $r_{b,i}$ the position and the value of the nonzero entry in $\mathbf{r}_b^{(i)}$. then, we have

$$\mathbf{r}_1^\intercal \cdot \mathbf{r}_0 = \sum_{i=1}^{\lambda} (\mathbf{r}_1^{(i)})^\intercal \cdot \mathbf{r}_0^{(i)} = \sum_{i=1}^{\lambda} \mathsf{EQ}(j_{0,i}, j_{1,i}) \cdot (r_{0,i} r_{1,i}),$$

where $\mathsf{EQ}(x, y)$ returns 1 if $x = y$ and 0 otherwise. Therefore, securely distributing shares of $\mathbf{r}_1^\intercal \cdot \mathbf{r}_0$ reduces (mostly) to performing λ secure equality tests (for the $\mathsf{EQ}(j_{0,i}, j_{1,i})$ terms) between $\log(m/\lambda)$-bit strings, and secure products over $\mathbb{F}$ (for the $r_{0,i} r_{1,i}$ terms), which is quadratically reduced compared to the cost for Bernoulli noise.

4.2 A Simple Protocol with Leakage

A first solution is to let the parties sample slightly more than λ random positions $j_{0,i}, j_{1,i}$ (say, $\lambda' = 2\lambda$). Let us denote $(\mathbf{j}_0, \mathbf{j}_1)$ the respective vectors of inputs. Then, the two parties perform a *batch equality test* with inputs $(\mathbf{j}_0, \mathbf{j}_1)$, which reveals to both parties for each position i whether $j_{0,i} = j_{1,i}$. Extremely efficient batch equality-tests have been constructed in the context of private set intersection protocols, using oblivious transfers [44], or more recently vector-OLE [20,54]. Concretely, even a naive OT-based protocol would use $\lambda' \log(m/\lambda)$ invocations of a bit-OT, a very small value. Using [44], this can be further reduced to about 4λ string-OT.

Given the results of the equality tests, the parties discard all positions where $j_{0,i} = j_{1,i}$. They use a λ-sized subset of the remaining positions (a subset of this size is guaranteed to exist by a standard Chernoff bound) to define their noise vectors $\mathbf{r}_0, \mathbf{r}_1$. Note that this corresponds to letting the parties sample uniformly random regular vectors $\mathbf{r}_0, \mathbf{r}_1$ conditioned on $\mathbf{r}_0^\intercal \cdot \mathbf{r}_1 = 0$. Hence, executing the protocol with these noise vectors yields an inner product protocol without correctness error.

The downside, however, is that the noise vectors are not independent: concretely, for each block of $\mathbf{r}_0$ (recall that a regular noise is a concatenation of blocks, where each block is a unit vector), Bob learns one uniformly random position of the block which is guaranteed to *not* be noisy. Therefore, Bob gets in total a leakage of λ non-noisy positions in Alice's vector (and reciprocally, Alice gets a leakage of λ noise-free positions in Bob's noise vector). In the two-party setting, this remains manageable: the security analysis still goes through, but under a less standard LPN-with-static-leakage variant of LPN, which has been studied in the past [16, 59] (in particular, it reduces to the standard LPN assumption, albeit with a large loss). Hence, scaling up the LPN parameters to compensate for the leakage suffices to guarantee security. However, in a multiparty setting, this becomes more troublesome: if (e.g.) Alice performs an inner product protocol with many other parties, these parties can, by colluding, accumulate the leakages. The situation is similar to the issue discussed in Section 3.4, but much worse: when using the approximate inner product protocol in a multiparty setting, each execution leaks roughly one linear equation about the noise (by revealing whether $\mathbf{r}_0^\intercal \cdot \mathbf{r}_1 = 0$), while with the above error-free variant, each execution leaks λ linear equations (since $\mathbf{r}_{0,j}^\intercal \cdot \mathbf{r}_{1,j} = 0$ is guaranteed for all blocks j). Hence, the leakage accumulate significantly faster. Below, we describe a more involved preprocessing strategy which completely remove all leakage and guarantees security under the standard LPN assumption, while still achieving sublinear communication.

4.3 The Protocol

We describe below a protocol for inner product, following our previous discussion. We use the following building blocks:

- $\mathcal{F}_{\mathsf{EQ}}$ is an ideal functionality parametrized by a domain $[k]$ which, given two inputs $(x, y) \in [k]^2$, outputs random shares b_A, b_B to Alice and Bob of $\mathsf{EQ}(x, y)$;
- $\mathcal{F}_{\mathsf{OLE}}$ is an ideal functionality parametrized by a field $\mathbb{F}$ which, given two inputs $(x, y) \in \mathbb{F}^2$, outputs random shares z_A, z_B of $x \cdot y$ to Alice and Bob.

The protocol in the $(\mathcal{F}_{\mathsf{EQ}}, \mathcal{F}_{\mathsf{OLE}})$-hybrid model is given on Figure 3.

Protocol $\Pi_{\mathsf{IP}}^{\mathsf{sh}}$

- **Setup.** Let $\mathbb{F}$ be a prime order field. Sample $H \xleftarrow{\$} \mathsf{Code}^{\perp}(m, k+n, \mathbb{F})$ and output $\mathsf{crs} = H$.
- **Preprocessing.** Alice and Bob each sample random pairs $(j_{0,i}, r_{0,i}) \xleftarrow{\$} [m/\lambda] \times \mathbb{F}^*$ and $(j_{1,i}, r_{1,i}) \xleftarrow{\$} [m/\lambda] \times \mathbb{F}^*$ for $i = 1$ to λ. Let $\mathbf{r}_0, \mathbf{r}_1$ denote the corresponding regular noise vectors.
  - The parties call $\mathcal{F}_{\mathsf{EQ}}$ on inputs $(j_{0,i}, j_{1,i}) \in [m/\lambda]^2$ for $i = 1$ to λ. Let $(b_{0,i}, b_{1,i})$ denote Alice's and Bob's outputs. Let $c_{0,i} \leftarrow (-1)^{b_{0,i}}$ and $c_{1,i} \leftarrow (-1)^{b_{1,i}}$ for $i = 1$ to λ. Note that $c_{0,i} \cdot c_{1,i} = (-1)^{\mathsf{EQ}(j_{0,i}, j_{1,i})}$.
  - If $\mathbb{F} \neq \mathbb{F}_2$, the parties call $\mathcal{F}_{\mathsf{OLE}}$ over $\mathbb{F}$ twice, on inputs $(r_{0,i}, r_{1,i})$ and $(c_{0,i} \cdot r_{0,i}, c_{1,i} \cdot r_{1,i})$, for $i = 1$ to λ. Let $(\alpha_{0,i}, \alpha_{1,i})$ and $(\beta_{0,i}, \beta_{1,i})$ denote their respective outputs in each instance. Note that $\alpha_{0,i} + \alpha_{1,i} = r_{0,i} \cdot r_{1,i}$, and $\beta_{0,i} + \beta_{1,i} = c_{0,i} r_{0,i} \cdot c_{1,i} r_{1,i} = (-1)^{\mathsf{EQ}(j_{0,i}, j_{1,i})} \cdot r_{0,i} r_{1,i}$.
  - If $\mathbb{F} \neq \mathbb{F}_2$, the parties compute $z_0 = \sum_{i=1}^{\lambda} (\alpha_{0,i} - \beta_{0,i})/2$ and $z_1 = \sum_{i=1}^{\lambda} (\alpha_{1,i} - \beta_{1,i})/2$. Note that $z_0 + z_1 = \sum_{i=1}^{\lambda} r_{0,i} r_{1,i} \cdot (1 - (-1)^{\mathsf{EQ}(j_{0,i}, j_{1,i})})/2 = \sum_{i=1}^{\lambda} r_{0,i} r_{1,i} \cdot \mathsf{EQ}(j_{0,i}, j_{1,i}) = \mathbf{r}_0^{\intercal} \cdot \mathbf{r}_1$.
  - Else, if $\mathbb{F} = \mathbb{F}_2$, the parties set $(z_0, z_1) \leftarrow (\bigoplus_{i=1}^{\lambda} b_{0,i}, \bigoplus_{i=1}^{\lambda} b_{1,i})$. Note that $z_0 \oplus z_1 = \bigoplus_{i=1}^{\lambda} \mathsf{EQ}(j_{0,i}, j_{1,i}) = \mathbf{r}_0^{\intercal} \cdot \mathbf{r}_1$ (since $r_{0,i} = r_{1,i} = 1$ for all i when $\mathbb{F} = \mathbb{F}_2$).
- **Online Phase.** Let $(\mathbf{u}_0, \mathbf{u}_1)$ be the inputs of Alice and Bob.
  - Alice sends $\mathsf{pk}_0 \leftarrow (\mathbf{u}_0 /\!/ \mathbf{0}) - H^{\intercal} \cdot \mathbf{r}_0$ and sets $\mathsf{sk}_0 \leftarrow \mathbf{r}_0$, while Bob samples $\mathbf{s} \xleftarrow{\$} \mathbb{F}^k$, sends $\mathsf{pk}_1 \leftarrow H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + \mathbf{r}_1$ and sets $\mathsf{sk}_1 \leftarrow (\mathbf{u}_1 /\!/ \mathbf{s})$.
  - Alice outputs $x_0 = \mathsf{pk}_1^{\intercal} \cdot \mathsf{sk}_0 - z_0$ and Bob outputs $x_1 = \mathsf{pk}_0^{\intercal} \cdot \mathsf{sk}_1 - z_1$.

Fig. 3. A non-interactive inner-product protocol with semi-honest security $\Pi_{\mathsf{IP}}^{\mathsf{sh}}$ over $\mathbb{F}$ for vectors of length n

Theorem 9. *Let $\mathbb{F}$ be a prime order field and $\mathsf{Code}^{\perp}$ be a nice code. Assume that the regular $\mathsf{dual}\text{-}\mathsf{LPN}_{m-(k+n),\tau}^{m}(\mathbb{F})$ assumption with respect to $\mathsf{Code}^{\perp}$, and the (primal) regular $\mathsf{LPN}_{k,\tau}^{m}(\mathbb{F})$ assumption with respect to $\mathsf{Code}_{\mathsf{right}}^{\perp}$ both hold. Then protocol on Figure 3 securely realizes the inner product functionality $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$ from Figure 4 in the $(\mathcal{F}_{\mathsf{EQ}}, \mathcal{F}_{\mathsf{OLE}})$-hybrid model with semi-honest security and static corruption.*

Functionality $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$

The functionality $\mathcal{F}_{\mathsf{IP}}$ is parametrized by a field $\mathbb{F}$, and a vector length n. It interacts with two parties Alice and Bob, and an adversary $\mathcal{A}$. On input $(\mathsf{Input}, \mathbf{u} \in \mathbb{F}_q^n)$ from Alice and $(\mathsf{Input}, \mathbf{v} \in \mathbb{F}_q^n)$ from Bob, the functionality $\mathcal{F}_{\mathsf{IP}}$ proceeds as follows:

- If both parties are honest, sample $\alpha, \beta \in \mathbb{F}_q$ at random such that $\mathbf{u}^{\intercal} \cdot \mathbf{v} = \alpha + \beta$
- If Alice is corrupted, wait for a message $(\mathsf{Output}, \alpha \in \mathbb{F}_q)$ from $\mathcal{A}$ and set $\beta = \mathbf{u}^{\intercal} \cdot \mathbf{v} - \alpha$.
- If Bob is corrupted, wait for a message $(\mathsf{Output}, \beta \in \mathbb{F}_q)$ from the adversary and set $\alpha = \mathbf{u}^{\intercal} \cdot \mathbf{v} - \beta$.

The functionality outputs α to Alice and β to Bob, and then halts.

Fig. 4. Ideal functionality $\mathcal{F}_{\mathsf{IP}}$ for inner product between vectors over $\mathbb{F}^n$.

Proof. **Case 0: both parties are honest.** We first consider the case where no party is corrupted. Then, it follows by construction that $z_0 + z_1 = \mathbf{r}_0^{\intercal} \cdot \mathbf{r}_1$. Furthermore, we established previously in the proof of Theorem 7 that $\mathsf{pk}_1^{\intercal} \cdot \mathsf{sk}_0 + \mathsf{pk}_0^{\intercal} \cdot \mathsf{sk}_1 = \mathbf{u}_0^{\intercal} \cdot \mathbf{u}_1 + \mathbf{r}_0^{\intercal} \cdot \mathbf{r}_1$ (the online phase of the protocol is identical to an execution of Encode and Decode; only the distribution of $\mathbf{r}_0, \mathbf{r}_1$ changes). It follows that the outputs of Alice and Bob form additive shares of $\mathbf{u}_0^{\intercal} \cdot \mathbf{u}_1$ (with probability 1).

Case 1: Alice is corrupted. Assume now that Alice is corrupted, with input $\mathbf{u}_0$. The simulator Sim activates $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$ on behalf of Alice in the ideal world by sending $(\mathsf{Input}, \mathbf{u}_0)$. In the real world, it plays honestly the role of Bob in the preprocessing phase, emulates the answer of the functionalities $\mathcal{F}_{\mathsf{EQ}}$ and $\mathcal{F}_{\mathsf{OLE}}$ by returning either a random bit or a random element of $\mathbb{F}$, and stores the queries of Alice to the functionalities and the output z_0 that she computes from the answers to her queries. Sim extracts the $j_{0,i}$ from Alice's calls to $\mathcal{F}_{\mathsf{EQ}}$ and the $r_{0,i}$ from her calls to $\mathcal{F}_{\mathsf{OLE}}$, and reconstructs $\mathbf{r}_0 = \mathsf{sk}_0$. Sim emulates Bob in the online phase by sending $\mathsf{pk}_1 \xleftarrow{\$} \mathbb{F}^m$, and sets $x_0 \leftarrow \mathsf{pk}_1^{\intercal} \cdot \mathsf{sk}_0 - z_0$. Eventually, Sim sends (Output, x_0) to $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$.

It remains to argue why the simulation is indistinguishable from an honest execution of the protocol. Observe that the behavior of Sim is perfectly indistinguishable to that of Bob, except that it sends $\mathsf{pk}_1 \xleftarrow{\$} \mathbb{F}^m$ instead of $\mathsf{pk}_1 \leftarrow H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + \mathbf{r}_1$. Since the preprocessing phase does not leak any information about $\mathbf{r}_1$ (the answers of $\mathcal{F}_{\mathsf{EQ}}$ and $\mathcal{F}_{\mathsf{OLE}}$ to Alice being uniformly random by definition) and Sim does not need $\mathbf{r}_1$ to emulate these functionalities, the same sequence of games as in the proof of Theorem 7 shows that the advantage in distinguishing pk_1 from a uniformly random element in $\mathbb{F}^m$ is negligible under the (regular, primal) $\mathsf{LPN}^m_{k,\lambda/m}$ assumption with respect to Code.

Case 2: Bob is corrupted. Assume now that Bob is corrupted, with input $\mathbf{u}_1$. Sim plays in the preprocessing phase and interacts with $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$ in a symmetrical way, extracting the $(j_{1,i}, r_{1,i})$ and reconstructing the vector $\mathbf{r}_1$ and the value z_1. Sim emulates Alice in the online phase by sending $\mathsf{pk}_0 \xleftarrow{\$} \mathbb{F}^m$. Upon receiving pk_1 from Bob, Sim extracts $\mathsf{sk}_1 = (\mathbf{u}_1 /\!/ \mathbf{s})$ by solving $\mathsf{pk}_1 - \mathbf{r}_1 = H \cdot \mathbf{X}$ and parsing the solution $\mathbf{X}$ as $\mathsf{sk}_1 = (\mathbf{u}_1 /\!/ \mathbf{s})$ (which is guaranteed to be well-formed since Bob is semi honest). Eventally, Sim sets $x_1 \leftarrow \mathsf{pk}_0^\intercal \cdot \mathsf{sk}_1 - z_1$ and sends (Output, x_1) to $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$.

As above, proving indistinguishability from an honest execution reduces to proving that $\mathsf{pk}_0 \xleftarrow{\$} \mathbb{F}^m$ is indistinguishable from setting $\mathsf{pk}_0 \leftarrow (\mathbf{u}_0 /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0$, which can be shown (since $\mathbf{r}_0$ is perfectly hidden from Bob), using the same sequence of games as in the proof of Theorem 7, to follow from the (regular) $\mathsf{dual\text{-}LPN}^m_{m-(k+n),\lambda/m}$ assumption with respect to Code.

4.4 Variant: Replacing λ Calls to $\mathcal{F}_{\mathsf{OLE}}$ by 2λ Calls to $\mathcal{F}_{\mathsf{OT}}$

Let $\mathcal{F}_{\mathsf{OT}}(\mathbb{F})$ be the oblivious transfer functionality over $\mathbb{F}$: on input $(s_0, s_1) \in \mathbb{F}^2$ from the sender and a bit b from the receiver, it outputs s_b to the receiver and nothing to the sender.

In the protocol of Figure 3, the parties with shares $(b_{0,i}, b_{1,i})$ of $\mathsf{EQ}(j_{0,i}, j_{1,i})$ and values $(r_{0,i}, r_{1,i}) \in \mathbb{F}^2$ must compute additive shares of $(b_{0,i} \oplus b_{1,i}) \cdot r_{0,i} r_{1,i}$, which they do using two calls to $\mathcal{F}_{\mathsf{OLE}}$. We provide an alternative instantiation, which uses one call to $\mathcal{F}_{\mathsf{OLE}}$, and two additional calls to $\mathcal{F}_{\mathsf{OT}}$:

- Alice and Bob call $\mathcal{F}_{\mathsf{OLE}}$ on inputs $(r_{0,i}, r_{1,i}) \in \mathbb{F}^2$ and obtain additive shares $(\alpha_{0,i}, \alpha_{1,i})$ of their product.
- Alice and Bob perform two oblivious transfers in parallel. In the first OT, Alice plays the sender with inputs $(b_{0,i} \cdot \alpha_{0,i} + r_A, (1 - b_{0,i}) \cdot \alpha_{0,i} + r_A)$ for a random mask r_A, and Bob plays the receiver with input $b_{1,i}$. Concretely, Alice and Bob obtain this way shares of $(b_{0,i} \oplus b_{1,i}) \cdot \alpha_{0,i}$ (where Alice's share is r_A). In the other direction, Bob plays the role of the sender, using a random mask r_B, and Alice of the receiver with input $b_{0,i}$; Alice and Bob obtain additive shares of $(b_{0,i} \oplus b_{1,i}) \cdot \alpha_{1,i}$. Summing their shares, Alice and Bob do indeed obtain shares of $(b_{0,i} \oplus b_{1,i}) \cdot r_{0,i} r_{1,i}$.

4.5 Instantiating $\mathcal{F}_{\mathsf{EQ}}$ and $\mathcal{F}_{\mathsf{OLE}}$

With the above variant, the preprocessing boils down to λ invocations of $\mathcal{F}_{\mathsf{EQ}}$ on $\log(m/\lambda)$-bit strings, λ invocations of $\mathcal{F}_{\mathsf{OLE}}$ over $\mathbb{F}$, and 2λ invocations of $\mathcal{F}_{\mathsf{OT}}$ on $\log|\mathbb{F}|$-bit strings. There exists numerous options to implement the $\mathcal{F}_{\mathsf{EQ}}$ functionality. In our range of parameters, we estimate that the most efficient solution is the protocol of [26]. For equality test over ℓ-bit strings, it requires $\ell + o(\ell)$ oblivious transfers of $\log \ell$-bit strings, and $O(\log^* \ell)$ rounds of communication. Concretely, setting for example $\lambda = 120$ and $m = 3n$, for an inner product between string of length at most $n = 2^{20}$, the protocol of [26] can be instantiated either with 15 OTs of 16-bit strings and 14 OTs of bits in two rounds, or with 15 OTs of 16-bit strings, 4 OTs of 4-bit strings, and 2 OTs of bits, in three rounds (and no additional communication beyond the OTs). For $\mathcal{F}_{\mathsf{OLE}}$, the protocol of [37] requires $\log|\mathbb{F}|$ OTs per OLE over $\mathbb{F}$ (while recent OLE protocols such as [18] are much more efficient, their efficiency improvement "kicks in" only for a large enough number of OLE). With these choices of protocol, the full preprocessing boils down to $\lambda \cdot (\log(m/\lambda) + \log|\mathbb{F}| + 2)$ oblivious transfers.

Overall, setting $m = O(n)$, the communication of the preprocessing phase boils down to $O(\lambda \cdot (\log n + \log|\mathbb{F}|))$ oblivious transfer of small strings ($O(\log n)$-bit or $\log|\mathbb{F}|$-bit strings), which leads to a logarithmic communication in the vector length n. For example, using the standard instantiation for short string oblivious transfer [43], computing the inner product between two strings of length 2^{20} over a 32-bit field requires about $5 \cdot 10^5 \approx 0.5 \cdot n$ bits of communication, adding only a small overhead to the entire communication of the protocol. Using recent advances in silent OT extension [16, 27], this overhead can be further reduced by a factor four.

5 Malicious Security

In this section, we enhance our protocol from Section 4 to withstand attacks from malicious adversaries.

5.1 Guaranteeing the Success of Extraction

In the malicious model, the parties may not follow the specifications of the protocol; in particular, they may not use their prescribed input. Therefore, to make the protocol from Figure 3 secure against malicious behavior, the simulator must have a mean to extract the input of the corrupted party. When Alice is corrupted, since Sim emulates the preprocessing and stores her noise vector $\mathbf{r}_0$, the *effective input* $\mathbf{u}_0$ used by Alice can be extracted by computing $\mathsf{pk}_0 + H^\intercal \cdot \mathbf{r}_0$, and parsing it as $(\mathbf{u}_0 /\!/ \mathbf{0})$. However, the success of this extraction is only guaranteed if we can ensure that pk_0 will always be well-formed (i.e. the "bottom half" of pk_0 is of the form $M \cdot \mathbf{r}_0$ for a sparse $\mathbf{r}_0$, where M is the bottom half of $H^\intercal$). Similarly, if Bob is corrupted, Sim extracts $\mathbf{u}_1$ by solving the linear system $H \cdot \mathbf{X} = \mathsf{pk}_1 - \mathbf{r}_1$ to get $(\mathbf{u}_1 /\!/ \mathbf{s})$. However, this is an overdetermined system of equations which is not guaranteed to have a solution, and extraction will again succeed only if we can guarantee that pk_0 is well-formed (i.e., this system has a solution).

To guarantee the success of extraction, we let Alice and Bob add zero-knowledge proofs that their public keys $\mathsf{pk}_0, \mathsf{pk}_1$ are well-formed. With simple manipulations, it is easy to show that in both cases, this reduces to proving that a vector $\mathbf{v}$ is of the form $M \cdot \mathbf{e}$, where M is a public compressive matrix, and $\mathbf{e}$ is a secret sparse noise vector – i.e., this reduces to proving knowledge of a preimage in an instance of the syndrome decoding problem for the code with parity-check matrix M, which is a well-studied problem [2,22,57]. Unfortunately, existing solutions are prohibitively expensive in our setting: they require $O(\kappa \cdot m)$ communication, where κ is a statistical security parameter (which stems from parallel repetitions of an underlying zero-knowledge proof with constant soundness error, e.g. 2/3 in Stern's scheme [57]) and m is the code dimension. Since our protocol operates in the high-dimension, low-noise setting, this causes a huge blowup to the total communication and computation.

5.2 A New Almost-Zero-Knowledge Proof for Low-Noise Syndrome Decoding

As a contribution of independent interest, we therefore design a new zero-knowledge proof system for the syndrome decoding problem, which is especially suited for instances with large dimension and low noise. For a syndrome decoding instance of dimension ℓ and a noise rate of λ/ℓ, our protocol boils down essentially to $O(\lambda \cdot \log \ell)$ actively secure oblivious transfers and λ OLE. On the downside, unlike Stern's protocol, our zero-knowledge proof is not an identification scheme: it is private coin and cannot be made non-interactive using the Fiat-Shamir heuristic.

Our approach follows the intuition underlying a recent line of work [8, 10, 30, 61] on efficient zero-knowledge proofs from pseudorandom correlation generators [15–17]. However, our goal is fundamentally different, since these works target linear communication zero-knowledge proofs for general (arithmetic) circuits; on the other hand, we construct a *sublinear communication* zero-knowledge proof for a specific problem.

Intuition. A recent line of work initiated in [15] has developed pseudorandom correlation generators (PCG) for the vector-OLE (VOLE) correlation. At a high level, a PCG for a VOLE correlation allows to distributively generate additive shares of $\Delta \cdot \mathbf{v}$, where Δ is a (chosen) element of $\mathbb{F}$ known to one of the parties, and $\mathbf{v}$ is a (long) pseudorandom vector over $\mathbb{F}$, known to the other party. We do not directly build on PCG, but observe that the main component in their construction is a protocol that relies on *puncturable pseudorandom functions* (PPRF) to distributively generate, with low communication, additive shares of $\Delta \cdot \mathbf{e}$ for a *sparse, regular* noise vector $\mathbf{e}$.

We rely on this PPRF-based protocol to authenticate the regular noise vector $\mathbf{e}$ (i.e., the witness of the prover) with low communication overhead, using an information-theoretic MAC Δ known to the verifier. Due to the regular structure of $\mathbf{e}$, this boils down to distributively generating and locally concatenating shares of $\Delta \cdot \mathbf{e}_i$ for $i = 1$ to λ, where the $\mathbf{e}_i$ are unit vectors (let j_i be the index of their nonzero entry, and e_i be the corresponding value). Such a protocol is called a *single point vector OLE*. We briefly recall how such shares are generated with sublinear communication:

- The verifier samples Δ, and a PRF key K for a PRF $\{\mathsf{PRF}_K : [\ell/\lambda] \mapsto \mathbb{F}\}_K$.
- The parties execute an interactive protocol to securely generate $K\{j_i\}$ (the key K punctured at j_i). Using variants of the Doerner-shelat protocol [31] on top of the GGM puncturable PRF [38], this requires $O(\log \ell)$ invocations of an oblivious transfer protocol.
- The prover obliviously receive the value $\mathsf{PRF}_K(j_i) + \Delta \cdot e_i$, using a single OLE over $\mathbb{F}$.
- In the malicious setting, when several instances are executed, additional consistency checks are required to guarantee that Δ remains the same accross all executions. An efficient protocol for this task was given in [59], with minimal overhead compared to the semi-honest protocol.

Let $(\mathbf{q}_0, \mathbf{q}_1)$ denote the additive shares of $\Delta \cdot \mathbf{e}$ generated using the above protocol. To check that $\mathbf{v}$ is indeed of the form $M \cdot \mathbf{e}$, the verifier sends a random vector ρ to the prover, who replies with the value $\mathsf{ver}_0 = -\rho^\intercal \cdot (M \cdot \mathbf{q}_0) \in \mathbb{F}$. Then, the verifier sets $\mathsf{ver}_1 \leftarrow \rho^\intercal \cdot (M \cdot \mathbf{q}_1 - \Delta \cdot \mathbf{v})$ and check that $\mathsf{ver}_0 = \mathsf{ver}_1$.

Observe that $\mathsf{ver}_1 - \mathsf{ver}_0 = \rho^\intercal \cdot (M \cdot \mathbf{q}_1 - \Delta \cdot \mathbf{v} + M \cdot \mathbf{q}_0) = \rho^\intercal \cdot (M \cdot (\Delta \cdot \mathbf{e}) - \Delta \cdot \mathbf{v}) = 0$ if $M \cdot \mathbf{e} = \mathbf{v}$. Soundness will rely on the Schwarz-Zippel lemma to show that when $M \cdot \mathbf{e} \neq \mathbf{v}$, causing $\mathsf{ver}_0 = \mathsf{ver}_1$ is as hard as guessing Δ, which can happen only with probability $1/|\mathbb{F}|$ since Δ is perfectly hidden from the prover. This readily suffices when $\mathbb{F}$ is exponentially large. For smaller fields, we simply sample Δ from an appropriate extension field $\mathbb{F}'$ of $\mathbb{F}$ such that $|\mathbb{F}'| \geq 2^\kappa$ for some statistical security parameter κ; the rest of the protocol is identical, except that the parties must use a PRF from $[\ell/\lambda]$ to $\mathbb{F}'$, and execute the OLE's over $\mathbb{F}'$.

Zero-knowledge versus almost-zero-knowledge. The above blueprint actually leads to a true zero-knowledge proof system with sublinear communication, when instantiated with a maliciously secure sublinear protocol for single point vector OLE. While it is possible to construct such protocols, recent works [16, 59] have observed that one can achieve a much greater efficiency by slightly relaxing the single point VOLE functionality. In this relaxation, the verifier is allowed to learn roughly one bit of leakage about the noise vector $\mathbf{e}$. When instantiating our construction with the protocol of [59] (the state-of-the-art protocol of this line of work), the protocol we get is therefore not truly zero-knowledge. Nevertheless, it still suffices to construct a maliciously secure inner product protocol, which is our end goal, at the cost of relying on the *LPN with static leakage* assumption (first put forth in [16]), which states (informally) that LPN remains secure given one bit of leakage about the noise vector.

Since, for better efficiency, we do not achieve full-fledged zero-knowledge but only a relaxed version which suffices in our specific context, we do not provide here an isolated description of the zero-knowledge proof, and directly integrate it into our maliciously secure protocol. However, for the sake of completeness, we provide a description of the proof system in isolation (with and without the relaxation) in Appendix A.

5.3 Maliciously Secure Inner Product from LPN with Static Leakage

The full protocol, integrating the procedure for checking that pk_0 and pk_1 are well-formed, is described on Figure 6, in the $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}$-hybrid model. These checks require the parties to have access to authenticated versions of the noise vectors $\mathbf{r}_0, \mathbf{r}_1$; this authentication procedure is executed in a preprocessing phase. The ideal functionality $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}$ describing the preprocessing phase is represented on Figure 5. It follows closely the single-point vector-OLE functionality from [59], but enhances it to also distribute the inner product between pairs of single-point VOLEs. Similarly, our instantiation of this functionality will build upon the protocol of [59].

5.4 Security Analysis

We first recall the LPN with static leakage assumption from [16, 59]:

Definition 10 (Regular LPN with static leakage). *Fix a field* $\mathbb{F} = \mathbb{F}(\lambda)$*, dimension* $k = k(\lambda)$*, number of samples* $m = m(\lambda)$*, and noise rate* $\tau = \tau(\lambda)$*. The regular* $\mathsf{LPN}_{k,\tau}^m$ *assumption with static leakage with respect to* Code *holds if for every PPT algorithm* $\mathcal{A}$*, it holds that*

$$\left| \Pr[\mathsf{LPN\text{-}Succ}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{LPN\text{-}Succ}_{\mathcal{A}}(\lambda)$ *is defined as follows:*

Functionality $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}(\mathbb{F}, \mathbb{F}', n)$

The functionality is parametrized by a field $\mathbb{F}$ and an extension field $\mathbb{F}'$ of $\mathbb{F}$, as well as a vector length n, which is assumed to be a power of 2.

Initialize. Upon receiving Input from Alice and Bob, sample $\Delta, \Delta' \xleftarrow{\$} \mathbb{F}'$ if both Alice and Bob are honest. Otherwise, if Bob is corrupted, receive Δ' from the adversary and sample $\Delta \xleftarrow{\$} \mathbb{F}'$; if Alice is corrupted, receive Δ from the adversary and sample $\Delta' \xleftarrow{\$} \mathbb{F}'$. Output Δ to Alice, Δ' to Bob, and ignore all subsequent Input commands.

Extend. Upon receiving $(\mathsf{Extend}, \mathbf{x}')$ from Alice and $(\mathsf{Extend}, \mathbf{x})$ from Bob, where $(\mathbf{x}', \mathbf{x})$ are unit vectors over $\mathbb{F}^n$, do:

1. If Bob is honest, sample $\mathbf{y} \xleftarrow{\$} (\mathbb{F}')^n$. Otherwise, receive $\mathbf{y} \in (\mathbb{F}')^n$ from the adversary. Similarly, if Alice is honest, sample $\mathbf{y}' \xleftarrow{\$} (\mathbb{F}')^n$. Otherwise, receive $\mathbf{y}' \in (\mathbb{F}')^n$ from the adversary.
2. If Alice is honest, compute $\mathbf{z} \leftarrow \mathbf{y} + \Delta \cdot \mathbf{x}$. Otherwise, receive $\mathbf{z}$ from the adversary and recompute $\mathbf{y} \leftarrow \mathbf{z} - \Delta \cdot \mathbf{x}$. Similarly, if Bob is honest, compute $\mathbf{z}' \leftarrow \mathbf{y}' + \Delta' \cdot \mathbf{x}'$. Otherwise, receive $\mathbf{z}'$ from the adversary and recompute $\mathbf{y}' \leftarrow \mathbf{z}' - \Delta' \cdot \mathbf{x}'$.
3. If party $P \in \{A, B\}$, receive a set $I \subseteq [1, n]$ from the adversary. Let $j \in [1, n]$ be the index of the nonzero entry of $\mathbf{x}$ (if $P = B$) or $\mathbf{x}'$ (if $P = A$). If $j \in I$, send $\mathsf{success}$ to P and continue. Otherwise, send abort to both parties and abort.
4. If both parties are honest, set (w_0, w_1) to be random shares over $\mathbb{F}$ of $\mathbf{x}^\intercal \cdot \mathbf{x}'$. Otherwise, if Alice (resp. Bob) is corrupted, receive w_0 (resp. w_1) from the adversary, and set $w_1 \leftarrow \mathbf{x}^\intercal \cdot \mathbf{x}' - w_0$ (resp. $w_0 \leftarrow \mathbf{x}^\intercal \cdot \mathbf{x}' - w_1$).
5. Send $(\mathbf{z}, \mathbf{y}', w_0)$ to Alice and $(\mathbf{y}, \mathbf{z}', w_1)$ to Bob.

Global-key query. If party $P \in \{A, B\}$ is corrupted, receive $(\mathsf{guess}, \hat{\Delta})$ from the adversary with $\hat{\Delta} \in \mathbb{F}'$. If $\hat{\Delta} = \Delta$ and $P = A$, or if $\hat{\Delta} = \Delta'$ and $P = B$, send $\mathsf{success}$ to P and ignore any subsequent global-key query from P. Otherwise, send abort to both parties and abort.

Fig. 5. Ideal Functionality for the preprocessing step of maliciously secure inner product, parametrized by a field $\mathbb{F}$ with extension field $\mathbb{F}'$

1. *Sample $A \xleftarrow{\$} \mathsf{Code}(m, k, \mathbb{F}), \mathbf{s} \xleftarrow{\$} \mathbb{F}^k, \mathbf{e} \xleftarrow{\$} \mathsf{RN}_{\tau,m}(\mathbb{F})$, and let $(\alpha_1, \cdots, \alpha_{\tau m}) \in [1/\tau]^{\tau m}$ denote the location of the nonzero entries of $\mathbf{e}$. Send A to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs τm subsets $(I_1, \cdots, I_{\tau m})$ of $[1/\tau]$. If $\alpha_i \in I_i$ for every $i \leq \tau m$, output $\mathsf{success}$ to $\mathcal{A}$; otherwise, abort the experiment and set the output to 0.*
3. *If the experiment did not abort, pick a random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, set $\mathbf{u} \leftarrow A \cdot \mathbf{s} + \mathbf{e}$; else, set $\mathbf{u} \xleftarrow{\$} \mathbb{F}^m$. Send $\mathbf{u}$ to $\mathcal{A}$. Output 1 if $\mathcal{A}$ answers with b, and 0 otherwise.*

We note that LPN with static leakage reduces to standard LPN assumption [16], but the reduction is not tight. Intuitively, the assumption allows the adversary to obtain one bit of leakage on $\mathbf{e}$ on average, which should reduce bit security by one bit at most. Since the reduction to LPN induces a much larger loss, we define this assumption as an independent assumption and use it with the same concrete parameter as for LPN.

On the use of a PRF. The checks in the online phase require the parties to exchange long random strings ρ_0, ρ_1. To reduce communication, this is done by exchanging short keys, which the parties locally stretch into long pseudorandom strings by evaluating a PRF on a priori fixed inputs: $\rho_b \leftarrow (\mathsf{PRF}_{K_b}(0), \cdots, \mathsf{PRF}_{K_b}(k/\lambda))$, assuming that PRF has λ-bit outputs. It is a well-known result that any statistical test that succeeds with high probability for a random string, such as our application of the Schwarz-Zippel lemma, must succeed with comparable probability when evaluating a PRF on inputs fixed before the key was sampled, since any noticeable difference can be turned into an efficient distinguisher against the PRF.

Theorem 11. *Let $\mathsf{Code}^\perp$ be a nice code. Assume that the $\mathsf{dual}\text{-}\mathsf{LPN}^m_{m-(k+n),\tau}(\mathbb{F})$ assumption with static leakage with respect to $\mathsf{Code}^\perp$, and the (primal) $\mathsf{LPN}^m_{k,\tau}(\mathbb{F})$ assumption with static leakage with respect to $\mathsf{Code}^\perp_{\mathsf{right}}$ both hold. Then the protocol $\pi_{\mathsf{IP}}^{\mathsf{mal}}$ securely computes the inner product functionality $\mathcal{F}_{\mathsf{IP}}$ with security against malicious adversaries in the $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}$-hybrid model.*

Protocol $\Pi_{\mathsf{IP}}^{\mathsf{mal}}$

Let $\mathbb{F}'$ be the smallest extension field of $\mathbb{F}$ (possibly equal to $\mathbb{F}$) such that $|\mathbb{F}'| \geq 2^\kappa$, for a statistical security parameter κ. Fix parameters (k, m) as in the semi-honest protocol. Sample $H \xleftarrow{\$} \mathsf{Code}^\perp(m, k+n, \mathbb{F})$ and output $\mathsf{crs} = H \in \mathbb{F}^{m\times(k+n)}$.

Preprocessing. Alice and Bob send Input to $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}(\mathbb{F}, \mathbb{F}', m)$, and receive respective outputs $(\Delta, \Delta') \in \mathbb{F}' \times \mathbb{F}'$.

- Alice and Bob each sample random pairs $(j_{0,i}, r_{0,i}) \xleftarrow{\$} [m/\lambda] \times \mathbb{F}^*$ and $(j_{1,i}, r_{1,i}) \xleftarrow{\$} [m/\lambda] \times \mathbb{F}^*$ for $i = 1$ to λ. Let $\mathbf{r}_b = \mathbf{r}_b^{(1)} /\!/ \cdots /\!/ \mathbf{r}_b^{(\lambda)}$ for $b = 0, 1$ denote the corresponding regular noise vectors.
- Alice and Bob call the Extend command of $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}(\mathbb{F}, \mathbb{F}', m/\lambda)$ λ times, on respective inputs $(\mathbf{r}_0^{(i)}, \mathbf{r}_1^{(i)})$ for $i = 1$ to λ. Let $(\mathbf{z}_i, \mathbf{y}'_i, w_{0,i})$ and $(\mathbf{z}'_i, \mathbf{y}_i, w_{1,i})$ denote their outputs in the i-th instance respectively. Alice constructs $\mathbf{q}_0$ by concatenating all the $\mathbf{z}_i$, and $-\mathbf{q}'_0$ by concatenating all the $\mathbf{y}'_i$. Similarly, Bob constructs $-\mathbf{q}_1$ by concatenating all the $\mathbf{y}_i$, and $\mathbf{q}'_1$ by concatenating all the $\mathbf{z}'_i$. Eventually, Alice sets $z_0 \leftarrow \sum_i w_{0,i}$ and Bob sets $z_1 \leftarrow \sum_i w_{1,i}$. Note that by definition of $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}(\mathbb{F}, \mathbb{F}', m)$, it holds that $(\mathbf{q}_0, \mathbf{q}_1)$ form additive shares of $\Delta \cdot \mathbf{r}_1$, $(\mathbf{q}'_0, \mathbf{q}'_1)$ form additive shares of $\Delta' \cdot \mathbf{r}_0$, and (z_0, z_1) form additive shares of $\mathbf{r}_0^\intercal \mathbf{r}_1$.

Online Phase. Let $(\mathbf{u}_0, \mathbf{u}_1)$ be the inputs of Alice and Bob.

- Alice sends $\mathsf{pk}_0 \leftarrow (\mathbf{u}_0 /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0$ and sets $\mathsf{sk}_0 \leftarrow \mathbf{r}_0$, while Bob samples $\mathbf{s} \xleftarrow{\$} \mathbb{F}^k$, sends $\mathsf{pk}_1 \leftarrow H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + \mathbf{r}_1$ and sets $\mathsf{sk}_1 \leftarrow (\mathbf{u}_1 /\!/ \mathbf{s})$. The following checks are performed in parallel:
- **Checking that pk_0 is well-formed:**
  - Let $M \in \mathbb{F}^{k\times m}$ be the last k rows of $H^\intercal$. Note that the statement "there exists a vector $\mathbf{u}_0$ and a λ-regular vector $\mathbf{r}_0$ such that $\mathsf{pk}_0 = (\mathbf{u}_0 /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0$" is equivalent to "there exists a λ-regular vector $\mathbf{r}_0$ such that the last k coordinates of pk_0 are equal to $M \cdot \mathbf{r}_0$". Bob sends $K_0 \xleftarrow{\$} \{0,1\}^\lambda$ and both parties expand K_0 into $\rho_0 = (\mathsf{PRF}_{K_0}(0), \cdots, \mathsf{PRF}_{K_0}(k/\lambda)) \in (\mathbb{F}')^{1\times k}$ using a PRF $\mathsf{PRF} : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$.
  - Alice sends $\mathsf{ver}_0 \leftarrow -\rho_0 \cdot (M \cdot \mathbf{q}'_0)$. Bob aborts unless $\mathsf{ver}_0 = \rho_0 \cdot (M \cdot \mathbf{q}'_1 - \Delta' \cdot \mathsf{pk}_0)$.
- **Checking that pk_1 is well-formed:**
  - Let $G \in \mathbb{F}^{(m-k-n)\times m}$ be a parity-check matrix of H. Note that the statement "there exists a vector $\mathbf{u}_1 /\!/ \mathbf{s}$ and a λ-regular vector $\mathbf{r}_1$ such that $\mathsf{pk}_1 = H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + \mathbf{r}_1$" is equivalent to "there exists a λ-regular vector $\mathbf{r}_1$ such that $G \cdot \mathsf{pk}_1 = G \cdot \mathbf{r}_1$". Alice sends $K_1 \xleftarrow{\$} \{0,1\}^\lambda$ and both parties expand K_1 into $\rho_1 = (\mathsf{PRF}_{K_1}(0), \cdots, \mathsf{PRF}_{K_1}(k/\lambda)) \in (\mathbb{F}')^{1\times(m-k-n)}$ using a PRF.
  - Bob sends $\mathsf{ver}'_1 \leftarrow -\rho_1 \cdot (G \cdot \mathbf{q}_1)$. Alice aborts unless $\mathsf{ver}'_1 = \rho_1 \cdot (G \cdot \mathbf{q}_0 - \Delta \cdot (G \cdot \mathsf{pk}_1))$.
- Alice outputs $x_0 = \mathsf{pk}_1^\intercal \cdot \mathsf{sk}_0 - z_0$ and Bob outputs $x_1 = \mathsf{pk}_0^\intercal \cdot \mathsf{sk}_1 - z_1$.

Fig. 6. A non-interactive inner-product protocol with malicious security $\Pi_{\mathsf{IP}}^{\mathsf{map}}$ over $\mathbb{F}$ for vectors of length n

Proof. Let $\mathcal{A}$ be an adversary who corrupts Alice. We construct in a sequence of games a simulator Sim with access to $\mathcal{F}_{\mathsf{IP}}$ that simulates the view of the adversary. The analysis for a corrupted Bob will follow almost identically.

- Game 1: this is the real game, where Alice interacts with Bob, who holds an input $\mathbf{u}_1$.
- Game 2: in this game, Sim honestly simulates $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}$ and stores the vectors $(\mathbf{r}_0, \mathbf{q}_0, \mathbf{q}'_0)$ and the value Δ obtained by Alice. This game is perfectly indistinguishable from the previous one.
- Game 3: in this game, Sim simulates the output of Bob without using sk_1, as follows: using $\mathsf{sk}_0 = \mathbf{r}_0$ and z_0 (both extracted by simulating $\mathcal{F}_{\mathsf{pre}}^{\mathsf{mal}}$), Sim extracts $(\mathbf{u}_0 /\!/ \mathbf{0}) \leftarrow \mathsf{pk}_0 + H^\intercal \cdot \mathbf{r}_0$ and sends $(\mathsf{Input}, \mathbf{u}_0)$ to $\mathcal{F}_{\mathsf{IP}}(\mathbb{F}, n)$ on behalf of Alice in the ideal world. Afterwards, Sim computes $x_0 = \mathsf{pk}_1^\intercal \cdot \mathsf{sk}_0 - z_0$ and sends (Output, x_0) to $\mathcal{F}_{\mathsf{IP}}$ on behalf of Alice in the ideal world, and receive Bob's output x_1.
  This game is perfectly indistinguishable from the last one if it holds that pk_0 is indeed of the form $(\mathbf{u}_0 /\!/ \mathbf{0}) - H^\intercal \cdot \mathbf{r}_0$. This is equivalent to saying that the last k coordinates of pk_0, denoted $\mathbf{p}_0$, are equal to $M \cdot \mathbf{r}_0$, where $M \in \mathbb{F}^{k\times m}$ is the last k rows of $H^\intercal$. Assume for the sake of contradiction that $\mathbf{p}_0 \neq M \cdot \mathbf{r}_0$. Then,

$$\begin{aligned}\rho_0 \cdot (M \cdot \mathbf{q}'_1 - \Delta' \cdot \mathbf{p}_0) &= \rho_0 \cdot (M \cdot (\Delta' \cdot \mathbf{r}_0 - \mathbf{q}'_0) - \Delta' \cdot \mathbf{p}_0) \\ &= \Delta' \cdot (\rho_0 \cdot (M \cdot \mathbf{r}_0 - \mathbf{p}_0)) - \rho_0 \cdot M \cdot \mathbf{q}_0 \\ &= \Delta' \cdot w + \mathsf{ver}_0,\end{aligned}$$

where $w = (\rho_0 \cdot (M \cdot \mathbf{r}_0 - \mathbf{p}_0))$. Furthermore, by assumption, $M \cdot \mathbf{r}_0 - \mathbf{p}_0$ is not the all-zero vector. Therefore, with overwhelming probability $1 - 1/2^\kappa$ over a random choice of ρ_0, $w \neq 0$ holds as well by the Schwarz-Zippel lemma. By the PRF security, this must therefore still hold with overwhelming probability when ρ_0 is constructed as $(\mathsf{PRF}_{K_0}(0), \cdots, \mathsf{PRF}_{K_0}(k/\lambda))$ for a random key K_0. Now, recall that Δ' is uniformly random over $\mathbb{F}'$, and entirely unknown to Alice. Therefore, the probability that Alice comes up with a value ver'_0 that passes the check (i.e. $\mathsf{ver}'_0 = \Delta' \cdot w + \mathsf{ver}_0$) is upper bounded by $1/|\mathbb{F}'| \leq 1/2^\kappa$. This implies that the advantage of any adversary in distinguishing Game 2 from Game 3 is at most $2/2^\kappa + \mathsf{negl}(\lambda)$ (where $\mathsf{negl}(\lambda)$ bounds the advantage against the PRF).

– Game 4: from this game on, Sim will simulate Bob's answers to the check that pk_1 is well-formed, as follows: Sim computes $\mathsf{ver}'_0 \leftarrow \rho_1 \cdot (G \cdot \mathbf{q}_0 - \Delta \cdot (G \cdot \mathsf{pk}_1))$ using the preprocessing material of Alice, and sends $\mathsf{ver}'_1 \leftarrow \mathsf{ver}'_0$. This game is perfectly indistinguishable from the previous game. Note that Sim_1 does not use $(\mathbf{s}, \mathbf{r}_1)$ in the check of pk_1 anymore.
– Game 5: let $H_{\mathsf{left}} \in \mathbb{F}^{m\times k}, H_{\mathsf{right}} \in \mathbb{F}^{m\times n}$ denote the left and right halves of H. In this game, we assume that H_{right} was received as part of a challenge $(H_{\mathsf{right}}, \mathbf{c})$ for the $\mathsf{LPN}^m_{k,\lambda/m}$ assumption over $\mathbb{F}$ with static leakage, where $\mathbf{c}$ is of the form $H_{\mathsf{right}} \cdot \mathbf{s} + \mathbf{e}$, and H was sampled as $H \overset{\$}{\leftarrow} \mathbf{C}^\intercal|_{H_{\mathsf{right}}}(m, k+n, \mathbb{F})$. Sim plays as in the previous game, except that he constructs pk_1 as $H_{\mathsf{left}} \cdot \mathbf{u}_1 + \mathbf{c}$. Furthermore, whenever Bob makes queries of the form $I \subseteq [1, m]$ to $\mathcal{F}^{\mathsf{mal}}_{\mathsf{pre}}$, Sim makes the corresponding query to the challenger for the LPN with static leakage assumption, and forwards the answer to Bob. By construction of $\mathbf{c}$, this game is perfectly indistinguishable from the previous one.
– Game 6: same as the previous game, except that $\mathbf{c}$ is now a random vector. This game is indistinguishable from the previous one under the $\mathsf{LPN}^m_{k,\lambda/m}$ assumption over $\mathbb{F}$ with static leakage.
– Game 7: in this game, Sim now samples $\mathsf{pk}_1 \overset{\$}{\leftarrow} \mathbb{F}^m$, and plays as before otherwise. Since $\mathbf{c}$ is uniformly random, this game is perfectly indistinguishable from the previous game.

We sketch the other direction, which is almost identical. When $\mathcal{A}$ corrupts Bob, Sim emulates the preprocessing functionality and extracts $\mathbf{u}_1$ by solving for $\mathsf{pk}_1 - \mathbf{r}_1 = H \cdot (\mathbf{u}_1 /\!/ \mathbf{s})$. Then, it forwards $(\mathsf{Input}, \mathbf{u}_1)$ to $\mathcal{F}_{\mathsf{IP}}$ and programs the output with $(\mathsf{Output}, x_1 = \mathsf{pk}_0^\intercal \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) - z_1)$. This simulation succeeds if it holds that pk_1 is indeed of the form $H \cdot (\mathbf{u}_1 /\!/ \mathbf{s}) + \mathbf{r}_1$; equivalently, $G \cdot \mathsf{pk}_1$ is equal to $G \cdot \mathbf{r}_1$ (G being a parity-check matrix of H). The same analysis shows that this holds with probability $1 - 2/2^\kappa$ (where a $1/2^\kappa$ term comes from the probability that $\rho_1 \cdot \mathbf{v} = 0$ for a fixed nonzero vector $\mathbf{v}$, and the other $1/2^\kappa$ term bounds the probability that Bob guessed Δ successfully). Sim also simulates the check that pk_0 is well-formed using the preprocessing material of Bob, and simulates pk_0 by sending a uniformly random vector over $\mathbb{F}^m$, which is indistinguishable from the real pk_0 under the dual $\mathsf{LPN}^m_{m-(k+n),\lambda/m}(\mathbb{F})$ assumption with static leakage. This concludes the proof.

Efficiency. Compared to the semi-honest protocol, the online phase of Figure 6 adds two rounds of interaction to the protocol, as well as 2λ bits (for exchanging the seeds) and two elements of $\mathbb{F}'$ (hence, the overall increase in communication is essentially negligible). Regarding computation, the cost of the check that pk_0 is well-formed is dominated by a multiplication by the matrix $M \cdot \mathbb{F}^{k\times m}$, which (setting $k = n$ for concreteness) is about twice faster than a multiplication by H. The cost of checking that pk_1 is well-formed is dominated by a multiplication by the parity-check matrix G of H for Bob (resp. two multiplications by G for Alice), which is about the same cost as a multiplication by H. Therefore, the computational cost of the maliciously secure protocol is about twice that of the semi-honest protocol.

Implementing the malicious preprocessing functionality. The functionality $\mathcal{F}^{\mathsf{mal}}_{\mathsf{pre}}$ is a slight extension of the single point VOLE functionality of [59], whose work also provides a very efficient instantiation of the functionality. More specifically, our preprocessing functionality enhances theirs in two ways: (1) the single point VOLE is executed twice, with the roles of Alice and Bob reversed in one of the two instances, and (2) Alice and Bob also receive additive shares of the *inner product* between their respective unit vectors.

The first item simply boils down to executing two parallel instances of the protocol of [59]. We showed in Section 4 how to efficiently generate shares of the inner product between unit vectors, using one equality test and two OLEs. This construction can be enhanced to the malicious setting using standard techniques (working on mac-authenticated values, see e.g. SPDZ [29] or MASCOT [42]).

However, a non-trivial task remain: the protocol must guarantee that the unit vectors involved in the inner product are the same as those generated through the single-point VOLE protocol of [59].

Fortunately, we observe that the single-point VOLE protocols of [16] and [59] can be modified to output not only shares of $\varDelta \cdot \mathbf{x}$ for a unit vector $\mathbf{x}$ known to Bob and a MAC $\varDelta$ known to Alice, but also additive shares (over an appropriate field) of $\varGamma \cdot j$, where j is the position of the nonzero entry (known to Bob) and $\varGamma$ is an information-theoretic MAC known to Alice. We sketch this construction here: the key intuition behind constructions of single-point VOLE is that Alice will hold a PRF key K for the GGM puncturable pseudorandom function [38]. This PRF key K defines a complete binary tree with m/λ leaves and K at its root, where the two children of each nodes are computed by applying a length-doubling PRG to the parent node. The goal of the protocol will be to distributively generate a key $K\{j\}$, i.e. a PRF key *punctured* at the index j known by Bob of the non-zero entry of its unit vector. Then, Alice's share of the unit vector is the vector $(\mathsf{PRF}_K(i))_{i \leq m/\lambda}$, and Bob's share is the vector whose coordinates are equal to $\mathsf{PRF}_{K\{j\}}(i)$ for $i \neq j$, and $\varDelta \cdot x_j + \mathsf{PRF}_K(j)$ at $i = j$ (which is computed using a single OLE).

Bob's input j is viewed as a path from K to $K\{j\}$ in the GGM tree. To securely generate $K\{j\}$, Alice and Bob will execute $\log(m/\lambda)$ oblivious transfers in parallel, where Bob will use as selection bit the bits of j, and Alice will use as input an appropriate XOR of values on the nodes of the GGM tree (the details do not matter here). The honest behavior of the parties during this interaction is guaranteed by a checking procedure perfomed by the parties at the end. A nice feature of this protocol, which was never used to our knowledge, is that it can be used to authenticate $j = j_1 j_2 \cdots j_{\log m/\lambda}$ simultaneously, with the guarantee that the authenticated index is the same as the index of the nonzero entry in Bob's unit vector. This is done as follows: during the i-th OT, Alice will append two strings $(c_i, c_i + 2^{i-1} \cdot \varGamma)$ to her two OT inputs respectively. After the interaction, Alice defines $c \leftarrow \sum_{i=1}^{\log m/\lambda}$ as her share, and Bob sum the $c_i + j_i \cdot 2^{i-1} \cdot \varGamma$, to obtain $c_i + j \cdot \varGamma$, where the j is guaranteed to be the same as in the punctured key $K\{j\}$ obtained through the interaction. Afterwards, the authenticated position j can be used inside maliciously secure equality tests using standard techniques; we omit the details here and leave a full-fledged optimized implementation of the malicious preprocessing to future work.

Acknowledgement

The first author acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE) and of the CNRS PEPS grant SCENE. Work done while the second author was visiting IRIF and partially covered by the CNRS PEPS grant SCENE.

References

1. Aguilar, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. Cryptology ePrint Archive, Report 2016/1194 (2016), https://eprint.iacr.org/2016/1194
2. Aguilar, C., Gaborit, P., Schrek, J.: A new zero-knowledge code based identification scheme with reduced communication. In: 2011 IEEE Information Theory Workshop. pp. 648–652. IEEE (2011)
3. Aguilar-Melchor, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. IEEE Transactions on Information Theory 64(5), 3927–3943 (2018)
4. Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS. pp. 298–307. IEEE Computer Society Press (Oct 2003)
5. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: bit flipping key encapsulation (2017)
6. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: International Colloquium on Automata, Languages, and Programming. pp. 403–415. Springer (2011)
7. Bansal, A., Chen, T., Zhong, S.: Privacy preserving back-propagation neural network learning over arbitrarily partitioned data. Neural Computing and Applications 20(1), 143–150 (2011)
8. Baum, C., Braun, L., Munch-Hansen, A., Scholl, P.: Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and z2k (2021)
9. Baum, C., Escudero, D., Pedrouzo-Ulloa, A., Scholl, P., Troncoso-Pastoriza, J.R.: Efficient protocols for oblivious linear function evaluation from ring-LWE. In: SCN 20. pp. 130–149. LNCS, Springer, Heidelberg (2020)
10. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. pp. 92–122. LNCS, Springer, Heidelberg (2021)

11. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in 2 n/20: How 1+ 1= 0 improves information set decoding. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 520–536. Springer (2012)
12. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Annual Cryptology Conference. pp. 743–760. Springer (2011)
13. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM (JACM) 50(4), 506–519 (2003)
14. Bordewijk, J.L.: Inter-reciprocity applied to electrical networks. Applied Scientific Research, Section A 6(1), 1–74 (1957)
15. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 896–912. ACM Press (Oct 2018)
16. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 291–308. ACM Press (Nov 2019)
17. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 489–518. Springer, Heidelberg (Aug 2019)
18. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 387–416. Springer, Heidelberg (Aug 2020)
19. Boyle, E., Kohl, L., Scholl, P.: Homomorphic secret sharing from lattices without FHE. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 3–33. Springer, Heidelberg (May 2019)
20. Bui, D., Couteau, G.: Private set intersection from pseudorandom correlation generators. Cryptology ePrint Archive, Report 2022/334 (2022), `https://eprint.iacr.org/2022/334`
21. de Castro, L., Juvekar, C., Vaikuntanathan, V.: Fast vector oblivious linear evaluation from ring learning with errors. Cryptology ePrint Archive, Report 2020/685 (2020), `https://eprint.iacr.org/2020/685`
22. Cayrel, P.L., Véron, P., El Yousfi Alaoui, S.M.: A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 171–186. Springer, Heidelberg (Aug 2011)
23. Chen, M.S., Cheng, C.M., Kuo, P.C., Li, W.D., Yang, B.Y.: Multiplying boolean polynomials with frobenius partitions in additive fast fourier transform. arXiv preprint arXiv:1803.11301 (2018)
24. Chen, T., Zhong, S.: Privacy-preserving backpropagation neural network learning. IEEE Transactions on Neural Networks 20(10), 1554–1564 (2009)
25. Cheng, Q., Gao, C.Z.: A cloud aided privacy-preserving profile matching scheme in mobile social networks. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). vol. 2, pp. 195–198. IEEE (2017)
26. Couteau, G.: New protocols for secure equality test and comparison. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18. LNCS, vol. 10892, pp. 303–320. Springer, Heidelberg (Jul 2018)
27. Couteau, G., Rindal, P., Raghuraman, S.: Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. pp. 502–534. LNCS, Springer, Heidelberg (2021)
28. Damgård, I., Jurik, M.: A length-flexible threshold cryptosystem with applications. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 03. LNCS, vol. 2727, pp. 350–364. Springer, Heidelberg (Jul 2003)
29. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (Aug 2012)
30. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-point zero knowledge and its applications. In: 2nd Conference on Information-Theoretic Cryptography (ITC 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2021)
31. Doerner, J., shelat, a.: Scaling ORAM for secure computation. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 523–535. ACM Press (Oct / Nov 2017)
32. Dong, C., Chen, L.: A fast secure dot product protocol with application to privacy preserving association rule mining. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 606–617. Springer (2014)
33. Dong, W., Dave, V., Qiu, L., Zhang, Y.: Secure friend discovery in mobile social networks. In: 2011 proceedings ieee infocom. pp. 1647–1655. IEEE (2011)
34. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In: Naor, M. (ed.) ITCS 2014. pp. 169–182. ACM (Jan 2014)
35. Esser, A., Kübler, R., May, A.: Lpn decoded. In: Annual International Cryptology Conference. pp. 486–514. Springer (2017)
36. Gilbert, H., Robshaw, M.J.B., Seurin, Y.: Good variants of HB+ are hard to find. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 156–170. Springer, Heidelberg (Jan 2008)

37. Gilboa, N.: Two party RSA key generation. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 116–129. Springer, Heidelberg (Aug 1999)
38. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. pp. 464–479. IEEE Computer Society Press (Oct 1984)
39. Hazay, C., Toft, T.: Computationally secure pattern matching in the presence of malicious adversaries. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 195–212. Springer (2010)
40. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 433–442. ACM Press (May 2008)
41. Jiang, W., Samanthula, B.K.: N-gram based secure similar document detection. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 239–246. Springer (2011)
42. Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 830–842. ACM Press (Oct 2016)
43. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (Aug 2013)
44. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 818–829. ACM Press (Oct 2016)
45. Li, H., Li, H., Wei, K., Yin, S.L., Zhao, C.: A multi-keyword search algorithm based on polynomial function and safety inner-product method in secure cloud environment. Journal of Information Hiding and Multimedia Signal Processing 8(2), 413–422 (2017)
46. Liu, Q., Peng, Y., Pei, S., Wu, J., Peng, T., Wang, G.: Prime inner product encoding for effective wildcard-based multi-keyword fuzzy search. IEEE Transactions on Services Computing (2020)
47. Lyubashevsky, V., Masny, D.: Man-in-the-middle secure authentication schemes from LPN and weak PRFs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 308–325. Springer, Heidelberg (Aug 2013)
48. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Persichetti, E., Zémor, G., Bourges, I.: Hamming quasi-cyclic (hqc). NIST PQC Round 2, 4–13 (2018)
49. Murugesan, M., Jiang, W., Clifton, C., Si, L., Vaidya, J.: Efficient privacy-preserving similar document detection. The VLDB Journal 19(4), 457–475 (2010)
50. Orlandi, C., Scholl, P., Yakoubov, S.: The rise of paillier: Homomorphic secret sharing and public-key silent OT. pp. 678–708. LNCS, Springer, Heidelberg (2021)
51. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: Scifi-a system for secure face identification. In: 2010 IEEE Symposium on Security and Privacy. pp. 239–254. IEEE (2010)
52. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (May 1999)
53. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory 8(5), 5–9 (1962)
54. Rindal, P., Schoppmann, P.: VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. pp. 901–930. LNCS, Springer, Heidelberg (2021)
55. Sendrier, N.: Decoding one out of many. In: International Workshop on Post-Quantum Cryptography. pp. 51–67. Springer (2011)
56. Shuguo, H., Ng, W.K.: Multi-party privacy-preserving decision trees for arbitrarily partitioned data. Int J Intell Control Syst 12(4), 351–358 (2007)
57. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (Aug 1994)
58. Torres, R.C., Sendrier, N.: Analysis of information set decoding for a sub-linear error weight. In: Post-Quantum Cryptography. pp. 144–161. Springer (2016)
59. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1074–1091. IEEE (2021)
60. Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. pp. 139–152 (2009)
61. Yang, K., Sarkar, P., Weng, C., Wang, X.: Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. IACR Cryptol. ePrint Arch. 2021, 76 (2021)
62. Yu, H., Vaidya, J., Jiang, X.: Privacy-preserving svm classification on vertically partitioned data. In: Pacific-asia conference on knowledge discovery and data mining. pp. 647–656. Springer (2006)

A Zero-Knowledge Proof for Low-Noise Syndrome Decoding

The protocol. Let $M \in \mathbb{F}^{\ell' \times \ell}$ be a public compressive matrix (i.e. $\ell' < \ell$), and $\mathbf{v} \in \mathbb{F}^{\ell'}$ be a public word. We describe a zero-knowledge protocol between a prover P holding a regular vector $\mathbf{e} \in \mathbb{F}^{\ell}$ of weight λ and a verifier V, for the statement " $\exists \mathbf{e} \in \mathbb{F}^{\ell}$, $\mathbf{e}$ is λ-regular and $\mathbf{v} = M \cdot \mathbf{e}$". We let $\mathscr{L}_{\mathsf{syndec}}(M, \lambda)$ denote the corresponding language. Following the blueprint given above, we rely on the maliciously secure protocol of [59] for single-point subfield vector-OLE, whose functionality is recalled on Figure 7.

Functionality $\mathcal{F}_{\mathsf{spsVOLE}}(\mathbb{F}, \mathbb{F}', n)$

The functionality is parametrized by a field $\mathbb{F}$ and an extension field $\mathbb{F}'$ of $\mathbb{F}$, as well as a vector length n, which is assumed to be a power of 2.

Initialize. Upon receiving Input from Alice and Bob, sample $\Delta \overset{\$}{\leftarrow} \mathbb{F}'$ if Bob is honest, and receive Δ from the adversary otherwise. Output Δ to Bob and ignore all subsequent Input commands.

Extend. Upon receiving Extend from Bob and $(\mathsf{Extend}, \mathbf{x})$ from Alice, where $\mathbf{x}$ is a unit vector over $\mathbb{F}^n$, do:

1. If Bob is honest, sample $\mathbf{y} \overset{\$}{\leftarrow} (\mathbb{F}')^n$. Otherwise, receive $\mathbf{y} \in (\mathbb{F}')^n$ from the adversary.
2. If Alice is honest, compute $\mathbf{z} \leftarrow \mathbf{y} + \Delta \cdot \mathbf{x}$. Otherwise, receive $\mathbf{z}$ from the adversary and recompute $\mathbf{y} \leftarrow \mathbf{z} - \Delta \cdot \mathbf{x}$.
3. (If Bob is corrupted, receive a set $I \subseteq [0, n)$ from the adversary. Let $j \in [0, n)$ be the index of the nonzero entry of $\mathbf{x}$. If $j \in I$, send success to Bob and continue. Otherwise, send abort to both parties and abort.)
4. Send $\mathbf{z}$ to Alice and $\mathbf{y}$ to Bob.

Global-key query. If Alice is corrupted, receive $(\mathsf{guess}, \Delta')$ from the adversary with $\Delta' \in \mathbb{F}'$. If $\Delta' = \Delta$, send success to Alice and ignore any subsequent global-key query. Otherwise, send abort to both parties and abort.

Fig. 7. Ideal Functionality for single-point subfield vector OLE over a field $\mathbb{F}$ with extension field $\mathbb{F}'$

In addition to $\mathcal{F}_{\mathsf{spsVOLE}}$, we use an equivocable commitment scheme com (i.e., a simulator can construct a valid-looking commitment which it can later open to any value), and a hash function $\mathsf{H} : \{0,1\}^{\lambda + \log \ell'} \mapsto \mathbb{F}$, modelled as a random oracle (which is used solely to generate a long pseudorandom string from a short seed). The formal protocol is represented on Figure 8.

(Almost) zero-knowledge proof for the statement $\mathbf{v} \in \mathscr{L}_{\mathsf{syndec}}(M, \lambda)$

Let $\mathbb{F}'$ be the smallest extension field of $\mathbb{F}$ (possibly equal to $\mathbb{F}$) such that $|\mathbb{F}'| \geq 2^{\kappa}$, for a statistical security parameter κ. We assume that $M \in \mathbb{F}^{\ell' \times \ell}$ with $\ell' < \ell$ and ℓ/λ is a power of two. Let $\mathbf{e} = (\mathbf{e}_1 /\!/ \cdots /\!/ \mathbf{e}_\lambda)$ be the witness of the prover.

1. The prover and the verifier send Input to $\mathcal{F}_{\mathsf{spsVOLE}}(\mathbb{F}, \mathbb{F}', \ell/\lambda)$ (the prover plays the role of Alice). The verifier receives a random $\Delta \in \mathbb{F}'$.
2. The prover and the verifier call the Extend command of $\mathcal{F}_{\mathsf{spsVOLE}}(\mathbb{F}, \mathbb{F}', n)$ λ times, with inputs $(\mathbf{e}_i)_{i \leq \lambda}$ from the prover, and receive outputs $(\mathbf{z}_i, \mathbf{y}_i)$. The prover sets $\mathbf{q}_0 \leftarrow \mathbf{z}_1 /\!/ \cdots \mathbf{z}_\lambda$ and the verifier sets $-\mathbf{q}_1 \leftarrow \mathbf{y}_1 /\!/ \cdots \mathbf{y}_\lambda$.
3. The verifier sends $\mathsf{seed} \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$ and both parties set $\rho \leftarrow \mathsf{H}(\mathsf{seed}||0)|| \cdots ||\mathsf{H}(\mathsf{seed}||\ell') \in \mathbb{F}^{1 \times \ell'}$.
4. The prover sends $\mathsf{ver}_0 \leftarrow -\rho \cdot (M \cdot \mathbf{q}_0)$ to the verifier, who accepts the proof iff $\mathsf{ver}_0 = \rho^{\intercal} \cdot (M \cdot \mathbf{q}_1 - \Delta \cdot \mathbf{v})$.

Fig. 8. (Amost) zero-knowledge proof for the statement $\mathbf{v} \in \mathscr{L}_{\mathsf{syndec}}(M, \lambda)$

The soundness of the protocol reduces to the binding property of the commitment scheme. If one were to remove step 3 from the functionality $\mathcal{F}_{\mathsf{spsVOLE}}$ (let us denote $\mathcal{F}'_{\mathsf{spsVOLE}}$ this punctured functionality), the protocol of Figure 8 could be shown to be zero-knowledge under the equivocability of com, leading to a "true" zero-knowledge proof in the $\mathcal{F}'_{\mathsf{spsVOLE}}$-hybrid model. However, in order to rely on the efficient protocol of [59], we must permit the verifier to obtain the leakage allowed by step 3. Observe that since the functionality aborts given any unsuccessful guess, this allows the verifier to learn on average a single bit of information in total about $\mathbf{e}$.