Maliciously Secure Oblivious Linear Function Evaluation with Constant Overhead

Satrajit Ghosh[1], Jesper Buus Nielsen[1], and Tobias Nilges[1]

[1]Aarhus University, Denmark

Abstract

In this work we consider the problem of oblivious linear function evaluation (OLE). OLE is a special case of oblivious polynomial evaluation (OPE) and deals with the oblivious evaluation of a linear function $f(x) = ax + b$. This problem is non-trivial in the sense that the sender chooses a, b and the receiver x, but the receiver may only learn $f(x)$. We present a highly efficient and UC-secure construction of OLE in the OT-hybrid model that requires only $O(1)$ OTs per OLE. The construction is based on noisy encodings introduced by Naor and Pinkas (STOC'99). Our main technical contribution solves a problem left open in their work, namely we show in a generic way how to achieve full simulation-based security from noisy encodings. All previous constructions using noisy encodings achieve only passive security. Our result requires novel techniques that might be of independent interest.

Using our highly efficient OLE as a black box, we obtain a direct construction of an OPE protocol that simultaneously achieves UC-security and requires only $O(d)$ OTs, where d is the degree of the polynomial that shall be evaluated.

1 Introduction

The oblivious evaluation of functions is an essential building block in cryptographic protocols. The first and arguably most famous result in the area is oblivious transfer (OT), which was introduced in the seminal work of Rabin [27]. Here, a sender can specify two bits s_0, s_1, and a receiver can learn one of the bits s_c depending on his choice bit c. It is guaranteed that the sender does not learn c, while the receiver learns nothing about s_{1-c}. Kilian [23] subsequently showed that OT is in fact already complete, i.e. it allows the (oblivious) evaluation of *any* function. The work on OT spawned the field of multiparty computation (MPC), which considers the generalized case of several parties obliviously evaluating generic circuits.

While there has been tremendous progress in the area of generic MPC over the last three decades, there are certain classes of functions that can be evaluated more efficiently by direct constructions instead of taking the detour via MPC. In this context, Naor and Pinkas [25] introduced oblivious polynomial evaluation (OPE) as an useful primitive. OPE deals with the problem of evaluating a polynomial P on an input α obliviously, i.e. in such a way that the sender specifies the polynomial P but does not learn α, while the receiver learns $P(\alpha)$ but nothing else about P. OPE has many applications, ranging from secure set intersection [26, 16] over RSA key generation [14] to oblivious keyword search [12]. Due to its versatility OPE has received considerable interest in recent years [24, 13, 6, 32, 22, 17, 16, 30].

A special case of OPE, called oblivious linear function evaluation (OLE, sometimes also referred to as OLFE, or OAFE for affine functions) has been considered, in particular due to potential applications in (the preprocessing of) MPC protocols for arithmetic circuits. Instead of evaluating an arbitrary polynomial P, the receiver wants to evaluate a linear or affine function $f(x) = ax + b$. Ishai et al. [19] propose a passively secure protocol for oblivious multiplication which uses a similar approach as [25], and can be easily modified to give a passively secure OLE. Based on stateful tamper-proof hardware [20] as a setup assumption, [9] build an unconditionally UC-secure protocol for OAFE.

Currently, all of the above mentioned actively secure realizations of OPE or OLE require rather expensive computations or strong setup assumptions. In contrast, the most efficient passively secure constructions built from noisy encodings and OT require only simple field operations. However, to date a direct construction of a maliciously secure protocol in this setting has been elusive. One approach to achieve this would be to apply a compiler like [18] to the passively secure protocols, which can result in an actively secure protocol with a constant overhead compared to the passively secure protocol. But such a transformation typically incurs a large constant, resulting in efficiency only in an asymptotic sense. Thus, the most efficient realizations possibly follow from applying the techniques used for the precomputation of multiplied values in arithmetic MPC protocols such as SPDZ [7] or MASCOT [21].

1.1 Our Contribution

Our main result is a UC-secure protocol for oblivious linear function evaluation in the OT-hybrid model, based on noisy encodings. The protocol has a constant overhead (namely 4) compared to the semi-honest secure implementation of OLE by [19]. The construction is based on the efficient semi-honestly secure multiplication protocol of Ishai et al. [19], which is the most efficient protocol for passively secure OLE that we are aware of. One nice property of [19] and the main reason for its efficiency is that it directly allows to multiply a batch of values. This property is preserved in our construction, i.e., we can simultaneously evaluate several linear functions.

In order to achieve our result we solve the long standing open problem of finding an actively secure OLE/OPE protocol which can directly be reduced to the security of noisy encodings (and OT). This problem was not solved in [25] and has been touched upon in follow-up work [19]. The key technical contribution of the paper is a reduction which shows that noisy encodings are robust against leakage in a strong sense, which allows their application in a malicious setting. As a matter of fact, our robustness results are more general and extend to other noisy coding-based assumptions as well.

An immediate application of our UC-secure batch-OLE construction is a UC-secure OPE construction. The construction is very simple and has basically no overhead over the OLE construction. We follow the approach taken in [26], i.e. we use the fact that a polynomial of degree d can be decomposed into d linear functions. Such a decomposed polynomial is evaluated with the batch-OLE and then put back together. UC-security against the sender directly follows from the UC-security of the batch-OLE. In order to make the protocol secure against a cheating receiver, we only have to add one additional check that ensures that the receiver choses the same input for each linear function. Table 1 compares the efficiency of our result with existing solutions in the literature.

	Assumption	OTs/Expon.	Security
[6]	OT	$O(d\kappa)$	passive
[26]	OT & Noisy Encodings	$O(d\kappa \log m)$	passive
[19]	OT & Noisy Encodings	$O(d)$	passive
[17]	CRS & DCRP	$O(ds)$	UC
[16]	DDH	$O(d)$	active
This work	**OT & Noisy Encodings**	$O(d)$	**UC**

Table 1: Overview of OPE realizations, where d is the degree of the polynomial. We compare the number of OTs and exponentiations in the respective protocols. Also note that [16] only realizes OPE in the exponent.

1.2 Technical Overview

At the heart of our constructions are noisy encodings. These were introduced by Naor and Pinkas [25] in their paper on OPE and provide a very efficient means to obliviously compute multiplications. A noisy encoding is basically an encoding of a message via a linear code that is mixed with random values in such a way that the resulting vector hides which elements belong to the codeword and which elements are random, thereby hiding the initial message. In a little more detail, the input $\mathbf{x} \in \mathbb{F}^t$ is used as t sampling points on locations α_i of an otherwise random polynomial P of some degree $d > t$. Then the polynomial is evaluated at e.g. $4d$ positions β_i, and half of these positions are replaced by uniformly random values, resulting in the encoding $\mathbf{v}$. It is assumed that this encoding is indistinguishable from a uniformly random vector.[1]

Robustness of noisy encodings The main problem of using noisy encodings in maliciously secure protocols is that the encoding is typically used in a non-black-box way. On one hand this allows for very efficient protocols, but on the other hand a malicious party obtains knowledge that renders the assumption that is made on the indistinguishability of noisy encodings useless. In a little more detail, consider a situation where the adversary obtains the encoding and manipulates it in a way that is not specified by the protocol. The honest party only obtains part of the encoding (this is usually necessary even in the passively secure case). In order to achieve active security, a check is performed which is supposed to catch a deviating adversary. But since the check is dependent on which part of the encoding the honest party learned, this check actually leaks some non-trivial information to the adversary, typically noisy positions of the codeword.

We show that noisy encodings as defined by [26, 19] are very robust with respect to leakage. In particular, we show the following theorem that is basically a stronger version of a result previously obtained by Kiayias and Yung [22].

Theorem 1. *(informal) For appropriate choices of parameters, noisy encodings are resilient against non-adaptive leakage of $O(\log \kappa)$ noisy positions.*

In a little more detail, we show that a noisy encoding generated as described above remains indistinguishable from a random vector of field elements, even for an adversary that is allowed to

[1] The problem is related to efficient polynomial reconstruction, i.e. decoding Reed-Solomon codes. The parameters have to be chosen in such a way that all known decoding algorithms fail.

fix the position of f noisy positions. Fixing f positions is of course stronger than being able to leak f positions. The security loss incurred by the fixing of f positions is 3^f.

We then show that an adversary which is given a noisy encoding cannot identify a super-logarithmic sized set consisting of only noisy positions.

Theorem 2. *(informal) For appropriate choices of parameters, an adversary cannot identify more than $O(\log \kappa)$ noisy positions in a noisy encoding.*

These theorems together show that we can tolerate the leakage of any number of noisy positions that might be guessed. This is the basis for the security of our protocol. Note that tolerance to leakage of a set of noisy positions that might be guessed is not trivial, as we are working with an indistinguishability notion. Hence leakage of a single bit might *a priori* break the assumption.

We describe the main idea behind our reduction proving the first theorem. Assume that there are a total of ρ noisy positions. Consider an adversary that is allowed to submit a set F. Then we generate a noisy encoding as above, except that all positions $i \in F$ are fixed to be noisy. The remaining $\rho - |F|$ noisy positions are picked at random. Denote the distribution by $\vec{v}^{\rho,F}$. Let $\vec{v}^{\rho} = \vec{v}^{\rho,\emptyset}$. Let $\vec{v}^{\$}$ denote a vector with all positions being uniformly random. We start from the assumption that $\vec{v}^{\rho} \approx \vec{v}^{\$}$. Let n be the total number of positions. Then clearly $\vec{v}^{n,F} = \vec{v}^{\$}$ for all F.

We want to prove that $\vec{v}^{\rho,F} \approx \vec{v}^{\$}$ for small F. Let F be a set of size f. Assume that we managed to prove that $\vec{v}^{\rho,F'} \approx \vec{v}^{\$}$ for all sets of size $f-1$. Assume also that we have managed to prove that $\vec{v}^{\rho+1,F} \approx \vec{v}^{\$}$.

For a set F let i be the smallest index in F and let $F = F' \cup \{i\}$. Consider the reduction which is given $\vec{v}$ from $\vec{v}^{\rho,F'}$ or $\vec{v}^{\$}$ and which adds noise to position i in $\vec{v}$ and outputs the result $\vec{v}'$. If $\vec{v} \sim \vec{v}^{\$}$, then $\vec{v}' \sim \vec{v}^{\$}$. If $\vec{v} \sim \vec{v}^{\rho,F'}$, then $\vec{v}' \sim \alpha\vec{v}^{\rho+1,F} + (1-\alpha)\vec{v}^{\rho,F}$, where α is the probability that i is not already a noisy position. Putting these together we get that $\vec{v}^{\rho,F'} \approx \vec{v}^{\$}$ implies that $\alpha\vec{v}^{\rho+1,F} + (1-\alpha)\vec{v}^{\rho,F} \approx \vec{v}^{\$}$. We then use that $\vec{v}^{\rho+1,F} \approx \vec{v}^{\$}$ to get that $\alpha\vec{v}^{\$} + (1-\alpha)\vec{v}^{\rho,F} \approx \vec{v}^{\$}$, which implies that $\vec{v}^{\rho,F} \approx \vec{v}^{\$}$, when α is not too large.

We are then left with proving that $\vec{v}^{\rho,F'} \approx \vec{v}^{\$}$ and $\vec{v}^{\rho+1,F} \approx \vec{v}^{\$}$. These are proven by induction. The basis for $\vec{v}^{\rho,F'} \approx \vec{v}^{\$}$ is $\vec{v}^{\rho,\emptyset} \approx \vec{v}^{\$}$. The basis for $\vec{v}^{\rho+1,F} \approx \vec{v}^{\$}$ is $\vec{v}^{n,F} = \vec{v}^{\$}$. Controlling the security loss in these polynomially deep and nested inductions is tricky. We give the full details later.

We now give the intuition behind the proof of the second theorem. Assume that some adversary can guess a set S of s noisy positions with polynomial probability p_1 given an encoding $\vec{v}^{\rho} = \vec{v}^{\$}$ and assume that s is super-logarithmic and that ρ/n is a non-zero constant. We prove the theorem for noisy level ρ but have to start with the assumption that $\vec{v}^{\rho - c\kappa} \approx \vec{v}^{\$}$ for an appropriate constant $c \in (0,1)$ and where κ is the security parameter.

Consider the reduction which is given a sample $\vec{v}$ from $\vec{v}^{\rho-c\kappa}$ or $\vec{v}^{\$}$. It starts by adding κ random positions R to $\vec{v}$ to get $\vec{v}'$. Then it feeds $\vec{v}'$ to A to get a set S. Then it uses its knowledge of R to sample the size of the intersection between S and the noisy positions in $\vec{v}'$. If it is "large" we guess that $\vec{v} \sim \vec{v}^{\rho-c\kappa}$. Otherwise we guess that $\vec{v} \sim \vec{v}^{\$}$. We pick c such that the total number of random positions in $\vec{v}'$ is ρ with polynomial probability when $\vec{v} \sim \vec{v}^{\rho-c\kappa}$, in which case S is a subset of the noisy positions with probability p_1, which will give a large intersection. If $\vec{v} \sim \vec{v}^{\$}$, then R is uniformly hidden to the adversary, and the expected size of the intersection will be smaller by a constant factor depending on ρ and c. The calibration of c and "small" is done as to allow a formal proof using a Chernoff bound. The details are given in the following.

Efficient OLE from noisy encodings. We build a UC-secure OLE protocol inspired by the passively secure multiplication protocol of Ishai et al. [19]. Let us briefly recall the construction on an intuitive level. One party, let us call it the sender, has as input t values $a_1, \ldots, a_t \in \mathbb{F}$, while the receiver has an input $b_1, \ldots, b_t \in \mathbb{F}$. The high-level idea is as follows: both sender and receiver interpolate a degree $\frac{n}{4} - 1$ polynomial through the points (α_i, a_i) and (α_i, b_i), obtaining $A(x)$ and $B(x)$, respectively. They also agree on n points $\beta_1, \ldots, \beta_n$. Now the receiver replaces half of the points $B(\beta_i)$ with uniformly random values (actually he creates a noisy encoding) and sends these n values $\bar{B}(\beta_i)$ to the sender. The sender draws an additional random polynomial R of degree $2(\frac{n}{4} - 1)$ to mask the output. He then computes $Y(\beta_i) = A(\beta_i) \cdot \bar{B}(\beta_i) + R(\beta_1)$ and uses these points as input into a $2(\frac{n}{4} - 1)$-out-of-n OT, from which the receiver chooses the $\frac{n}{4} - 1$ values in L. He can then interpolate the obtained points of $Y(\beta_i)$ to reconstruct Y and learn $a_i \cdot b_i + r_i$ in the positions α_i.

The passive security for the sender follows from the fact that the receiver obtains only $\frac{n}{4} - 1$ values and thus R completely masks the inputs $a_1, \ldots, a_t$. Passive security of the receiver follows from the noisy encoding, i.e. the sender cannot learn B from the noisy encoding.

In order to achieve active security of the above protocol, we have to ensure several things: first of all, we need to use an actively secure k-out-of-n OT. But instead of using a black-box realization, which incurs an overhead of $n \log n$ on the number of OTs, we use n OTs and ensure that the right number of messages was picked via a secret sharing, which the receiver has to reconstruct. This protocol first appeared in [29]. It does not have active security against the sender, who can guess some choice bits. A less efficient but active secure version was later given in [8], using verifiable secret sharing. We can, however, use the more efficient but less secure original variant as we can tolerate leakage of a few choice bits in the overall protocol.

Secondly, we also need to make sure that the parties used the right inputs in the computation, i.e. valid polynomials A, B and R. In order to catch deviations, we add two checks—one in each direction—to ensure this. The check is fairly simple: one party selects a random point z and the other party sends a pair $A(z), R(z)$, or $B(z), Y(z)$ respectively. Each party can now locally verify that the values satisfy the equation $A(z) \cdot B(z) + R(z) = Y(z)$.

As it turns out, both of these additions to the protocol, while ensuring protocol compliance w.r.t. the inputs, are dependent on the encoding. But this also means that a malicious sender can do selective failure attacks, e.g., it inputs incorrect shares for the secret sharing, and gets some leakage on the "secret key" of the encoding. This problem does not occur when considering semi-honest security

2 Preliminaries

We use the standard notions of probabilistic polynomial time (PPT) algorithms, negligible and overwhelming functions. Further, we denote by $\mathbf{x} \in \mathbb{F}^n$ a vector of length n and x_i as the ith element of $\mathbf{x}$. Unless noted otherwise, $P(x)$ denotes a polynomial in $\mathbb{F}[X]$, and X denotes a distribution.

We will typically denote a value $\hat{x}$ chosen or extracted by the simulator, while x^* is chosen by the adversary $\mathcal{A}$.

2.1 Universal Composability Framework

We state and prove our results in the Universal Composability (UC) framework of Canetti [4]. Security is defined via the comparison of an *ideal model* and a *real model*. In the real model, a protocol Π between the protocol participants is executed, while in the ideal model the parties only communicate with an ideal functionality $\mathcal{F}$ that is supposed to model the ideal security guarantees of the protocol. For an adversary $\mathcal{A}$ in the real protocol who coordinates the behavior of all malicious parties, there has to exist a simulator $\mathcal{S}$ for $\mathcal{A}$ in the ideal protocol. An environment $\mathcal{Z}$, which is plugged to both the real and the ideal protocol, provides the inputs to the parties and can read the outputs. The simulator $\mathcal{S}$ has to ensure that $\mathcal{Z}$ is not able to distinguish these models. Thus, even with concurrently executed protocols (running in the environment) the security holds. Usually, we assume that $\mathcal{A}$ is a dummy adversary controlled by $\mathcal{Z}$, which means that $\mathcal{Z}$ can adaptively choose its inputs depending on protocol messages it received and send messages on behalf of a (corrupted) protocol party.

More formally, let $\mathsf{Real}_{\Pi}^{\mathcal{A}}(\mathcal{Z})$ denote the random variable describing the output of $\mathcal{Z}$ when interacting with the real model, and let $\mathsf{Ideal}_{\mathcal{F}}^{\mathcal{S}}(\mathcal{Z})$ denote the random variable describing the output of $\mathcal{Z}$ when interacting with the ideal model. A protocol Π is said to *UC-realize* a functionality $\mathcal{F}$ if for any (PPT) adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for any (PPT) environment $\mathcal{Z}$, $\mathsf{Real}_{\Pi}^{\mathcal{A}}(\mathcal{Z}) \approx \mathsf{Ideal}_{\mathcal{F}}^{\mathcal{S}}(\mathcal{Z})$.

For our constructions we assume active adversaries and static corruption. We prove security in the hybrid model access to oblivious transfer (OT). For completeness the ideal functionality for OT is given in Section 2.3.

2.2 Commitment Scheme

A commitment scheme COM consists of two algorithms $(\mathsf{COM.Commit}, \mathsf{COM.Open})$. It is a two party protocol between a sender and a receiver. In the commit phase of the protocol, when the sender wants to commit to some secret value m, it runs $\mathsf{COM.Commit}(m)$ and gets back two values $(\mathsf{com}, \mathsf{unv})$. It sends com to the receiver. Later on in the unveil phase, the sender sends the unveil information unv to the receiver, who can use $\mathsf{COM.Open}$ to verify that the commitment com contains the actual secret m.

A commitment scheme must satisfy two security properties; 1)*Hiding*: The receiver cannot learn any information about the committed secret before the unveil phase, and 2)*Binding*: The sender must not be able to change the committed secret after the commit phase. For our purpose we need efficient UC-secure commitment schemes that can be realized in $\mathcal{F}_{\mathrm{OT}}$-hybrid model and in $\mathcal{F}_{\mathrm{OLE}}$-hybrid model.

In [5] the authors proposed an UC-secure commitment scheme in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model. Their protocol gives the first UC commitment scheme with "optimal properties" of rate approaching 1 and linear time complexity, in a "amortized sense". In our UC-secure OLE protocol also we need to commit to many values at a time, so we can use their UC commitment scheme in our protocol.

2.3 UC Oblivious Transfer

1-out-of-2 oblivious transfer protocol involves two parties, a sender who inputs $\mathbf{x}_0$ and $\mathbf{x}_1$ and a receiver whose input is a single bit $b \in \{0, 1\}$. At the end of the protocol the receiver learns only

$\mathbf{x}_b$ and the sender remains oblivious about the choice bit b. The ideal functionality is described in Figure 1.

Functionality $\mathcal{F}_{\mathrm{OT}}$

1. Upon receiving a message $(\texttt{inputS}, \mathbf{x}_0, \mathbf{x}_1)$ from S, where each $\mathbf{x}_i \in \{0,1\}^\lambda$, verify that there is no stored tuple, else ignore that message. Store $(\texttt{inputS}, \mathbf{x}_0, \mathbf{x}_1)$ and send a message $(\texttt{input})$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{inputR}, b)$ from R with $b \in \{0,1\}$, verify that there is no stored tuple, else ignore that message. Store $(\texttt{inputR}, b)$ and send a message $(\texttt{input})$ to $\mathcal{A}$.
3. Upon receiving a message $(\texttt{deliver}, \mathsf{S})$ from $\mathcal{A}$, check if both $(\texttt{inputS}, \mathbf{x}_0, \mathbf{x}_1)$ and $(\texttt{inputR}, b)$ are stored, else ignore that message. Send $(\texttt{delivered})$ to S.
4. Upon receiving a message $(\texttt{deliver}, \mathsf{R})$ from $\mathcal{A}$, check if both $(\texttt{inputS}, \mathbf{x}_0, \mathbf{x}_1)$ and $(\texttt{inputR}, b)$ are stored, else ignore that message. Send $(\texttt{output}, \mathbf{x}_b)$ to R.

Figure 1: Ideal functionality for 1-out-of-2 Oblivious Transfer.

2.4 Perfectly Private Secret Sharing Scheme

In a secret sharing scheme a secret is being distributed by a dealer among n parties, such that only authorized subset of parties can reconstruct the secret. It consists of two algorithms $\mathsf{SS.Share}()$ and $\mathsf{SS.Reconstruct}$. Secret sharing schemes were first proposed by Blakley [1] and Shamir [28]. In a perfectly private secret sharing scheme every unauthorized set cannot learn anything about the secret in the information theoretic sense. Shamir's scheme is an example of a perfectly private threshold linear secret sharing scheme.

Shamir's Threshold Secret-Sharing Scheme. In Shamir secret sharing scheme the secret k and the shares are the elements from a finite field $\mathbb{F}_q$ for some prime power $q > n$. To share a secret $k \in \mathbb{F}_q$, the dealer constructs d-degree polynomial $P(x) = a_0 + \sum_{i=1}^{d} a_i x^i$, where $a_0 = k$ and $a_i \in_r \mathbb{F}_q$, $\forall i \in [1, d]$. The dealer distributes $\{P(\alpha_j), \alpha_j\}$ to the j^{th} party, where $\alpha_j \in \mathbb{F}_q$, $\forall j \in [1, n]$.

Any $d+1$ parties among n parties can reconstruct the d-degree polynomial $P(x)$ using their shares, by forming $d+1$ linearly independent linear equations with $d+1$ variables $\{a_0, \ldots, a_d\}$ and solving the system of equations. Thus they can recover the secret $P(0) = k$; However, any d parties (or less than d parties) will not be able to extract any information about $P(0)$.

Packed Secret-Sharing. In [11] Franklin and Yung introduced the concept of packed secret-sharing to reduce the communication complexity of secure multi-party computation. The packed-secret sharing scheme used in [11] is similar to Shamir secret sharing, but here a block of l different values $\{k_1, \ldots, k_l\}$ are shared using a d-degree random polynomial that evaluates to $k_1, \ldots, k_l$ in l distinct points. To guarantee privacy in case of t corrupted parties, the random polynomial must have degree $d \geq t + l - 1$. In order to reduce the overall overhead we use packed secret-sharing in our OLE protocol.

3 Noisy Encodings

The security of our protocols is based on a noisy encoding assumption. Very briefly, a noisy encoding is an encoding of a message, e.g. via a linear code, that is mixed with random field elements. It is assumed that such a codeword, and in particular the encoded message, cannot be recovered. This assumption seems reasonable due to its close relationship to decoding random linear codes or the efficient decoding of Reed-Solomon codes with a large fraction of random noise.

Noisy encodings were first introduced by Naor and Pinkas [25], specifically for the purpose of realizing OPE. Their encoding algorithm basically generates a random polynomial P of degree $k-1$ with $P(0)=x$. The polynomial is evaluated at $n>k$ locations, and then $n-k$ positions are randomized. Generalizing the approach of [26], Ishai et al. [19] proposed a more efficient encoding procedure that allows to encode several field elements at once instead of a single element, using techniques of [11]. Basically, they use Reed-Solomon codes and then artificially blind the codeword with random errors in order to mask the location of the codeword elements in the resulting string.

The encoding procedure depicted in Figure 2 is nearly identical to the procedure given in [19], apart from the fact that we do not fix the signal-to-noise ratio. We also allow to pass a set of points $\mathcal{P}$ as an argument to Encode to simplify the description of our protocol later on. This change has no impact on the assumption, since these points are made public anyway via G.

$\mathsf{Encode}_{n,\rho}(\mathbf{x},\mathcal{P})$

Generator $\mathcal{G}(n,\rho)$ for Reed-Solomon code: Let the output be (G,H,L).

- Let $\ell = n-\rho$ and $k=\frac{\ell-1}{2}+1$.
- Let $\alpha_1,\dots,\alpha_k,\beta_1,\dots,\beta_n$ be the points in $\mathcal{P}$. If $\mathcal{P}=\emptyset$, pick these points randomly from $\mathbb{F}$.
- Define the $n\times k$ matrix G such that for any $\mathbf{u}\in\mathbb{F}^k$, $(G\mathbf{u})_i = P(\beta_i)$ for $i=1,\dots,n$, where P is the unique degree $k-1$ polynomial such that $P(\alpha_i)=u_i$ for $i=1,\dots,k$.
- Pick $L\subset[n]$ with $|L|=\ell$ uniformly at random.
- Let H be the $k\times 2k-1$ matrix such that $(H\mathbf{v}_L)_i = Q(\alpha_i)$, where Q is the unique degree $2(k-1)$ polynomial such that $Q(\beta_j)=\mathbf{v}_j$ for all $j\in L$.

Encoding: Let the private output be $(G,H,L,\mathbf{v})$ and the public output be $(G,\mathbf{v})$.

- Let $(G,H,L)\leftarrow\mathcal{G}(n,\rho)$.
- Pick a random $\mathbf{u}\in\mathbb{F}^k$ conditioned on $u_i=x_i$ for $i=1,\dots,t$. Compute $G\mathbf{u}\in\mathbb{F}^n$.
- Pick a random vector $v\in\mathbb{F}^n$, conditioned on $v_i=(G\mathbf{u})_i$ for $i\in L$.

Figure 2: Encoding procedure for noisy encodings.

[25] propose two different encodings and related assumptions, tailored to their protocols. One of these assumptions was later broken by [2] and [3], and a fixed version was presented in [26]. We are only interested in the unbroken assumption. The same assumption was used by [19] and we will adopt their notation in the following.

Assumption 1. *Let κ be a security parameter and $n,\rho\in\mathsf{poly}(\kappa)$. Further let $x,y\in\mathbb{F}^{t(\kappa)}$. Then the ensembles $\{\mathsf{Encode}_{n,\rho}(x)\}_\kappa$ and $\{\mathsf{Encode}_{n,\rho}(y)\}_\kappa$ are computationally indistinguishable, for $t\le\frac{\ell}{4}$.*

In order for this assumption to hold, [19] propose $n=4\kappa,\rho=2\kappa+1$ as parameters, or $n=8\kappa,\rho=6\kappa+1$ on the conservative side. As it is, our security reductions do not hold with respect to Assumption 1, but rather a variant of Assumption 1 which was already discussed in [25]. Instead of requiring indistinguishability of two encodings, we require that an encoding is pseudorandom.

Assumption 2. *Let κ be a security parameter and $n, \rho \in \mathsf{poly}(\kappa)$. Further let $x \in \mathbb{F}^{t(\kappa)}$. Then the ensembles $\{\mathsf{Encode}_{n,\rho}(x)\}_\kappa$ and $\{G \leftarrow \mathcal{G}(n,\rho), v \leftarrow \mathbb{F}^n\}_\kappa$ are computationally indistinguishable, for $t \leq \frac{\ell}{4}$.*

Clearly, Assumption 2 implies Assumption 1, while the other direction is unclear. Apart from being a very natural assumption, Kiayias and Yung [22] provide additional evidence that Assumption 2 is valid. They show that if an adversary cannot decide for a random position i of the encoding whether it is part of the codeword or not, then the noisy codeword is indeed pseudorandom.

4 Noisy Encodings are Robust Against Leakage

In this section we show that a large class of computational assumptions can be extended to allow some leakage without loss of asymptotic security. This is one of the main technical contributions of the paper and we deem the reductions to be of independent interest. We first define the class of assumptions we consider.

Definition 3. *Let a finite field $\mathbb{F}$ be given. For a positive integer n we use U^n to denote the uniform distribution on $\mathbb{F}^n$. Let n be the length of a codeword, ρ the number of randomised positions, G a generator and F some fixed random positions with $|F| \leq \rho$. The distribution $\mathsf{Y}^{n,\rho,G,F}$ is sampled as follows.*

1. *Sample $(x_1, \dots, x_n) \leftarrow G(1^\kappa)$, where κ is the security parameter.*
2. *Sample uniform $R \subseteq [n]$ under the restriction $|R| = \rho$ and $F \subseteq R$.*
3. *Sample uniform $(e_1, \dots, e_n) \leftarrow \mathsf{U}^n$ under the restriction $e_i = 0$ for $i \notin R$.*
4. *Output $\mathbf{y} = \mathbf{x} + \mathbf{e}$.*

Clearly it holds that the above defined Encode algorithm falls under Definition 3, i.e. all of the following results hold for noisy encodings as well.

We will mostly consider the case that $n = \Theta(\kappa)$ and $\rho = \Theta(\kappa)$. Typically n and G are fixed, in which case we denote the distribution of $\mathbf{y}$ by $\mathsf{Y}^{\rho,F}$. Note that $\mathsf{Y}^{n,F} = \mathsf{U}^n$.

We are going to assume that for sufficiently large ρ it holds that $\mathsf{Y}^\rho \approx \mathsf{Y}^n$, where $\approx$ means the distributions are computationally indistinguishable. For example, this is given by Assumption 2 for noisy encodings with appropriate parameters. We will use this to prove that the same result holds given limited leakage on R and that it is hard to compute a lot of elements of R given only $\mathbf{y}$.

When we prove the first result, we are going to do an argument with two nested recursive reductions. To make it easier to track the security loss, we are going to shift to a concrete security notation for a while and then switch back to asymptotic security when we have control over the security loss.

Given two distributions A_0 and A_1 and a distinguisher D let $\mathsf{Adv}_D(A_0, A_1) = \Pr[A_1 = 1] - \Pr[A_0 = 0]$. We use $A_0 \approx^t_\epsilon A_1$ to denote that it holds for all distinguishers computable in time t that $\mathsf{Adv}_D(A_0, A_1) \leq \epsilon$.

Given two distributions A_0 and A_1 and $0 \leq \alpha_0 \leq 1$ and $\alpha_1 = 1 - \alpha_0$ we use $B = \alpha_0 A_0 + \alpha_1 A_1$ to denote the distribution generated by the following procedure. Sample $c \in \{0, 1\}$ with $\Pr[c = i] = \alpha_i$. Then sample $b \leftarrow A_c$ and output b.

We will use the following simple facts on out proofs. For completeness the proof is given in Appendix A.

Lemma 4. *Let A_0, A_1 and Z be distributions.*

$$A_0 \approx^t_\epsilon A_1 \;\Rightarrow\; \alpha_0 A_0 + \alpha_1 Z \approx^t_{\alpha_0 \epsilon} \alpha_0 A_1 + \alpha_1 Z \ .$$

$$\alpha_0 A_0 + \alpha_1 Z \approx^t_\epsilon \alpha_0 A_1 + \alpha_1 Z \;\Rightarrow\; A_0 \approx^t_{\alpha_0^{-1}\epsilon} A_1 \ .$$

In the following we will show that if $\mathsf{Y}^\rho \approx^{t+t'}_\epsilon \mathsf{Y}^n$ and F is not too large, then $\mathsf{Y}^{\rho,F} \approx^t_{\epsilon'} \mathsf{Y}^{n,F}$ for ϵ' polynomially related to ϵ. Since the reduction will be recursive and will modify ϵ multiplicatively, we will keep explicit track of ϵ to ensure the security loss is not too large. As for t, each reduction will only *add* a small t' to t, namely the time to sample a distribution. The time will therefore clearly grow by at most a polynomial term. We therefore do not keep track of t, for notational convenience.

Lemma 5. *If $\rho - |F| \geq 2n/3$ and $\mathsf{Y}^j \approx_\epsilon \mathsf{Y}^n$ for all $j \geq \rho$, then $\mathsf{Y}^{\rho,F} \approx_{\sigma_\rho} \mathsf{Y}^{n,F}$ for $\sigma_\rho = 3^{|F|}\epsilon$.*

Proof. We prove the claim by induction in the size of F. If $|F| = 0$ it follows by assumption.

Consider then the following randomised function f with inputs $(y_1, \ldots, y_n)$ and F. Let i be the largest element in F and let $F' = F \setminus \{i\}$. Sample uniformly random $y'_i \in \mathbb{F}$. For $j \neq i$, let $y'_j = y_j$. Output $\mathbf{y}'$. Consider the distribution $\mathsf{Y}^{\rho,F'}$. Let R denote the randomised positions. If $i \in R$, then $f(\mathsf{Y}^{\rho,F'}) = \mathsf{Y}^{\rho,F}$. If $i \notin R$, then $f(\mathsf{Y}^{\rho,F'}) = \mathsf{Y}^{\rho+1,F}$, as we added one more random point. The point i is a fixed point not in F'. There are $n - |F| + 1$ points not in F'. There are ρ randomised points, i.e., $|R| = \rho$. Of these $|F| - 1$ are the points of F'. The other points are uniform outside F'. So there are $\rho - |F| + 1$ such points. Therefore the probability that $i \in R$ is $p = \frac{\rho - |F| + 1}{n - |F| + 1}$. It follows that

$$f(\mathsf{Y}^{n,F'}) = \mathsf{Y}^{n,F} \ , \quad f(\mathsf{Y}^{\rho,F'}) = p\mathsf{Y}^{\rho,F} + (1-p)\mathsf{Y}^{\rho+1,F} \ .$$

It then follows from $\mathsf{Y}^{n,F'} \approx_{\epsilon'} \mathsf{Y}^{\rho,F'}$ (where $\epsilon' = 3^{|F'|}\epsilon$) that

$$\mathsf{Y}^{n,F} \approx_{\epsilon'} p\mathsf{Y}^{\rho,F} + (1-p)\mathsf{Y}^{\rho+1,F} \ .$$

We now claim that $\mathsf{Y}^{\rho,F} \approx_{3\epsilon'} \mathsf{Y}^{n,F}$. The claim is trivially true for $\rho = n$, so we can assume that $\rho < n$ and assume the claim is true for all $\rho' > \rho$. Using Lemma 4 and the induction hypothesis we get that

$$p\mathsf{Y}^{\rho,F} + (1-p)\mathsf{Y}^{\rho+1,F} \approx_{(1-p)3\epsilon'} p\mathsf{Y}^{\rho,F} + (1-p)\mathsf{Y}^{n,F} \ .$$

Clearly

$$\mathsf{Y}^{n,F} \approx_0 p\mathsf{Y}^{n,F} + (1-p)\mathsf{Y}^{n,F} \ .$$

Putting these together we get that

$$p\mathsf{Y}^{n,F} + (1-p)\mathsf{Y}^{n,F} \approx_{\epsilon' + (1-p)3\epsilon'} p\mathsf{Y}^{\rho,F} + (1-p)\mathsf{Y}^{n,F} \ .$$

Using Lemma 4 we get that

$$\mathsf{Y}^{n,F} \approx_{p^{-1}(\epsilon'+(1-p)3\epsilon')} \mathsf{Y}^{\rho,F} \ .$$

We have that $p \geq 2/3$ so

$$p^{-1}(\epsilon' + (1-p)3\epsilon') \leq \frac{3}{2}(\epsilon' + \frac{1}{3}3\epsilon') = 3\epsilon' = 3^{|F|}\epsilon \ .$$

□

In the rest of the section, assume that n and ρ are functions of a security parameter κ and that $n, \rho = \Theta(\kappa)$. Also assume that $\rho \geq 2n/3$ and that $n - \rho = \Theta(\kappa)$. We say that $\mathsf{Y}^\rho \approx \mathsf{Y}^n$ if $\mathsf{Y}^{\rho'} \approx^{q(\kappa)}_{1/p(\kappa)} \mathsf{Y}^{n(\kappa)}$ for all polynomials p and q and all sufficiently large κ and all $\rho' \geq \rho$.

From $3^{O(\log\kappa)}$ being a polynomial in κ we get that

Corollary 6. *If $\mathsf{Y}^\rho \approx \mathsf{Y}^n$ and $F \subseteq [n]$ has size $O(\log\kappa)$, then $\mathsf{Y}^{\rho,F} \approx \mathsf{Y}^n$.*

Now assume that $\mathsf{Y}^\rho \approx \mathsf{Y}^n$. Let $\mathsf{Y}^{\rho,F,\neg}$ be defined as $\mathsf{Y}^{\rho,F}$ except that R is sampled according to the restriction that $F \not\subseteq R$, i.e., F has at least one element outside R. Let $p(F)$ be the probability that $F \subseteq R$ for a uniform R. Then by definition and the law of total probability $\mathsf{Y}^\rho = p\mathsf{Y}^{\rho,F} + (1-p)\mathsf{Y}^{\rho,F,\neg}$. We have that $\mathsf{Y}^\rho \approx \mathsf{Y}^n$ and that $\mathsf{Y}^{\rho,F} \approx_{3^{|F|}} \mathsf{Y}^{n,F} = \mathsf{Y}^n$. Putting these together we have that $\mathsf{Y}^n \approx_{p3^{|F|}} p\mathsf{Y}^n + (1-p)\mathsf{Y}^{\rho,F,\neg}$. We can assume wlog that $(1-p)$ is a polynomial. We then get that $\mathsf{Y}^n \approx_{3^{|F|}} \mathsf{Y}^{\rho,F,\neg}$.

Corollary 7. *If $\mathsf{Y}^\rho \approx \mathsf{Y}^n$ and $F \subseteq [n]$ has size $O(\log\kappa)$, then $\mathsf{Y}^{\rho,F,\neg} \approx \mathsf{Y}^n$.*

We will now prove that given a small non-adaptive query on R does not break the security.

Definition 8. *Let $\mathcal{A}$ be a PPT algorithm and Y as defined in Definition 3. The game $\mathcal{G}_{\mathsf{leak}}$ is defined as follows.*

1. *Run $\mathcal{A}$ to get a subset $Q \subseteq [n]$.*
2. *Sample a uniformly random bit c.*
    - *If $c = 0$, then sample $\mathbf{y} \leftarrow \mathsf{Y}^\rho$ and let R be the subset used in the sampling*
    - *If $c = 1$, then sample $\mathbf{y} \leftarrow \mathsf{Y}^n$ and let $R \subseteq [n]$ be a uniformly random subset of size ρ.*
3. *Let $r \in \{0,1\}$ be 1 iff $Q \subseteq R$ and input $(r, \mathbf{y})$ to $\mathcal{A}$.*
4. *Run $\mathcal{A}$ to get a guess $g \in \{0,1\}$.*

The advantage of $\mathcal{A}$ is $\mathsf{Adv}_\mathcal{A} = \Pr[g = 1 \mid c = 1] - \Pr[g = 1 \mid c = 0]$.

Theorem 1. *Assume that $n, \rho = \Theta(\kappa)$ and that $\rho \geq \frac{2}{3}n$. If $\mathsf{Y}^\rho \approx \mathsf{Y}^n$, then $\mathsf{Adv}_\mathcal{A} \approx 0$.*

Proof. Let p be the probability that $Q \subseteq R$. If p is negligible, then in item 3 of the game we could send the constant $r = 0$ to $\mathcal{A}$ and it would only change the advantage by a negligible amount. But in the thus modified game $\mathsf{Adv}_\mathcal{A} \approx 0$ because $\mathsf{Y}^n \approx \mathsf{Y}^\rho$. So assume that p is a polynomial.[2] Let Y_0

[2]Formally we should consider the case where it is a polynomial for infinitely many κ, but the following argument generalises easily to this case.

be the distribution of (r, y) when $c = 0$. Let Y_1 be the distribution of $(r, \mathbf{y})$ when $c = 1$. If $c = 0$, then $(r, \mathbf{y})$ is distributed as follows

$$Y_0 = p(1, \mathsf{Y}^{\rho,Q}) + (1-p)(0, \mathsf{Y}^{\rho,Q,\neg}) \ .$$

When p is polynomial, then $|F| = O(\log \kappa)$ as $n - \rho = \Theta(\kappa)$. From this we get that

$$Y_0 \approx p(1, \mathsf{Y}^n) + (1-p)(0, \mathsf{Y}^n) = Y_1 \ ,$$

using the above asymptotic corollaries. □

We will then prove that it is hard to compute a lot of elements of R.

Definition 9. *Let $\mathcal{A}$ be a PPT algorithm and Y as defined in Definition 3. The game $\mathcal{G}_{\mathsf{ident}}$ is defined as follows.*

1. *Sample $\mathbf{y} \leftarrow \mathsf{Y}^\rho$ and let R denote the randomized positions.*
2. *Input $\mathbf{y}$ to $\mathcal{A}$.*
3. *Run $\mathcal{A}$ and denote the output by $Q \subseteq [n]$. We require that $|Q| = s$.*
4. *Let $r \in \{0, 1\}$ be 1 iff $Q \subseteq R$.*
5. *Output r.*

The advantage of $\mathcal{A}$ is $\mathsf{Adv}_{\mathcal{A}}^{\rho,s} = \Pr[r = 1]$.

Theorem 2. *Let $n = \Theta(\kappa)$.*

1. *Let $\sigma = n\frac{\rho-\kappa}{n-\kappa}$ and $s = \kappa$.*
2. *Let $\sigma = \frac{n\kappa}{n-\rho-\kappa}$ and $s \in \omega(\log \kappa)$.*

If $\mathsf{Y}^\sigma \approx \mathsf{Y}^n$, then $\mathsf{Adv}_{\mathcal{A}}^{\rho,s} \approx 0$.

Proof. Let $\mathcal{A}$ be an adversary such that when $Q \leftarrow \mathcal{A}(\mathsf{Y}^\rho)$, then $Q \subseteq R$ with non-negligible probability p. The argumentation is similar for both cases. For the first part of the theorem, consider the following adversary $\mathcal{B}(\mathbf{y})$ receiving $\mathbf{y} \in \mathbb{F}^n$. It samples a uniform $X \subset [n]$ of size κ. For $i \in X$ let y_i' be uniformly random. For $i \notin X$ let $y_i' = y_i$. Compute $Q \leftarrow \mathcal{A}(\mathbf{y})$. If $|Q \cap X| \geq \frac{\kappa^2}{\rho}$, then output 1. Otherwise output 0.

We now prove that $\Pr[\mathcal{B}(\mathsf{Y}^n) = 1] \approx 0$ and that $\Pr[\mathcal{B}(\mathsf{Y}^\sigma) = 1]$ is non-negligible, which proves the first statement of the theorem.

Let R be the positions that were randomised in $\mathbf{y}$. Let $R' = R \cup X$. Note that if $\mathbf{y} \leftarrow \mathsf{Y}^\sigma$, then

$$\mathsf{E}[|R'|] = \kappa + \sigma - \mathsf{E}[|X \cap R|] = \kappa + \sigma - \kappa\frac{\sigma}{n} = \rho \ .$$

It is straight forward to verify that $\Pr[|R'| = \rho] = 1/O(\kappa)$, which implies that $\Pr[Q \subseteq R] = p/O(\kappa)$, which is non-negligible when p is non-negligible. Let E denote the event that $Q \subseteq R$. By a simple application of linearity of expectation we have that

$$\mathsf{E}[|Q \cap X| \,|\, E] = \frac{\kappa^2}{\rho},$$

as X is a uniformly random subset $X \subseteq R$ given the view of $\mathcal{A}$. From this it follows that $\Pr[\mathcal{B}(\mathsf{Y}^\sigma) = 1]$ is non-negligible.

Then consider $\mathcal{B}(\mathsf{Y}^n)$. Note that now $R = [n]$ and again X is a uniformly random subset of R independent of the view of $\mathcal{A}$. Therefore

$$\mathsf{E}[|Q \cap X|] = \frac{\kappa^2}{n} =: \mu.$$

Then

$$\Pr[|Q \cap X| \geq \frac{s(\rho - \kappa)}{\rho}] = \Pr[|Q \cap X| \geq \frac{n}{\rho}\mu] = \Pr[|Q \cap X| \geq (1+\delta)\mu]$$

for $\delta \in (0,1)$. It follows that

$$\Pr[|Q \cap X| \geq \frac{\kappa^2}{\rho}] \leq e^{-\frac{\mu\delta^2}{3}} = e^{-\Theta(\mu)} = e^{-\Theta(\kappa)} = \mathsf{negl}(\kappa).$$

To see this let $X = \{x_1, \ldots, x_\kappa\}$ and let X_i be the indicator variable for the event that the i'the element of X ends up in Q. Then $\Pr[X_i = 1] = \frac{\kappa}{n}$ and $|X \cap Q| = \sum_i X_i$. Consider then the modified experiment called *Independent Sampling*, where we sample the κ elements for X uniformly at random from $[n]$ and independently, i.e., it may happen that two of them are identical. In that case the inequality is a simple Chernoff bound. It is easy to see that when we go back to *Dependent Sampling*, where we sample x_i uniformly at random except that they must be different from $x_1, \ldots, x_{i-1}$, then we only lower the variance of the sum $\sum_i X_i$ compared to Independent Sampling, so $\Pr[|Q \cap X| \geq (1+\delta)\mu]$ will drop. Too see this, consider the sequence $x_, x_1 + x_2, \ldots, \sum_i x_i$ as a random walk. In the Dependent Sampling case, when $\sum_i x_i$ is larger than the expectation, then x_{i+1} is less likely to be in Q compared to the Independent Sampling case, as an above expectation number of slots in Q is already taken. Similarly, when $\sum_i x_i$ is smaller than the expectation, then x_{i+1} is more likely to be in Q compared to the Independent Sampling case, as a below expectation number of slots in Q is already taken. Therefore the random walk in the Dependent Sampling case will always tend closer to average compared to the Independent Sampling random walk.

The second statement of Theorem 2 follows by setting X to be a uniform subset of size $\rho - \kappa$. As above, if $\mathcal{A}$ outputs Q such that $|Q \cap X| \geq \frac{s(\rho-\kappa)}{\rho}$, then $\mathcal{B}(\mathbf{y})$ outputs 1. Otherwise it outputs 0. Let again R be the positions that were randomised in $\mathbf{y}$. Let $R' = R \cup X$. If $\mathbf{y} \leftarrow \mathsf{Y}^\sigma$, then

$$\mathsf{E}[|R'|] = \rho - \kappa + \sigma - \mathsf{E}[|X \cap R|] = \rho - \kappa + \sigma - (\rho - \kappa)\frac{\sigma}{n} = \rho\ .$$

Let E denote the event that $Q \subseteq R$. Following the above argumentation,

$$\mathsf{E}[|Q \cap X| \mid E] = \frac{s(\rho - \kappa)}{\rho}.$$

From this it follows that $\Pr[\mathcal{B}(\mathsf{Y}^\sigma) = 1]$ is non-negligible. Then consider $\mathcal{B}(\mathsf{Y}^n)$. Note that now $R = [n]$ and again X is a uniformly random subset of R independent of the view of $\mathcal{A}$. Therefore

$$\mathsf{E}[|Q \cap X|] = \frac{s(\rho - \kappa)}{n} =: \mu.$$

It follows that

$$\Pr[|Q \cap X| \geq \frac{s(\rho - \kappa)}{\rho}] \leq e^{-\frac{\mu\delta^2}{3}} = e^{-\Theta(\mu)} = e^{-\omega(\log \kappa)} = \mathsf{negl}(\kappa).$$

□

5 Constant Overhead Oblivious Linear Function Evaluation

Oblivious linear function evaluation (OLE) is the task of computing a linear function $f(x) = ax + b$ in the following setting. One party, lets call it the sender S, provides the function, namely the values a and b. The other party, the receiver R, wants to evaluate this function on his input x. This task becomes non-trivial if the parties want to evaluate the function in such a way that the sender learns nothing about x, while the receiver learns only $f(x)$, but not a and b. OLE can be seen as a special case of oblivious polynomial evaluation (OPE) as proposed by Naor and Pinkas [25], where instead of a linear function f, the sender provides a polynomial p.

5.1 Ideal Functionality

The efficiency of our protocol follows in part from the fact that we can directly perform a batch of multiplications. This is reflected in the ideal UC-functionality for $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t}}$ (cf. Figure 3), which allows both sender and receiver to input vectors of size t.

Functionality $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t}}$

1. Upon receiving a message $(\texttt{inputS}, \mathbf{a}, \mathbf{b})$ from S with $\mathbf{a}, \mathbf{b} \in \mathbb{F}^t$, verify that there is no stored tuple, else ignore that message. Store $\mathbf{a}$ and $\mathbf{b}$ and send a message $(\texttt{input})$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{inputR}, \mathbf{x})$ from R with $\mathbf{x} \in \mathbb{F}^t$, verify that there is no stored tuple, else ignore that message. Store $\mathbf{x}$ and send a message $(\texttt{input})$ to $\mathcal{A}$.
3. Upon receiving a message $(\texttt{deliver}, \mathsf{S})$ from $\mathcal{A}$, check if both $\mathbf{a}, \mathbf{b}$ and $\mathbf{x}$ are stored, else ignore that message. Send $(\texttt{delivered})$ to S.
4. Upon receiving a message $(\texttt{deliver}, \mathsf{R})$ from $\mathcal{A}$, check if both $\mathbf{a}, \mathbf{b}$ and $\mathbf{x}$ are stored, else ignore that message. Set $y_i = a_i \cdot x_i + b_i$ for $i \in [t]$ and send $(\texttt{output}, \mathbf{y})$ to R.

Figure 3: Ideal functionality for an oblivious linear function evaluation.

5.2 Our Protocol

Our starting point is the protocol of Ishai et al. [19] for *passively* secure batch multiplication. Their protocol is based on noisy encodings, similar to our construction. We will now briefly sketch their construction (with minor modifications) and then present the high-level ideas that are necessary to make the construction actively secure.

In their protocol, the receiver first creates a noisy encoding $(G, H, L, \mathbf{v}) \leftarrow \mathsf{Encode}(\mathbf{x})$ (as described in Section 3, Figure 2) and sends $(G, \mathbf{v})$ to the sender. At this point, the locations $i \in L$ of $\mathbf{v}$ hide a degree $\frac{\ell-1}{2}$ polynomial over the points $\beta_1, \ldots, \beta_n$ which evaluates to the input $\mathbf{x} = x_1, \ldots, x_t$ in the positions $\alpha_1, \ldots, \alpha_t$. The sender picks two random polynomials A and B with the restriction that $A(\alpha_i) = a_i$ and $B(\alpha_i) = b_i$ for $i \in [t]$. The degree of A is $\frac{\ell-1}{2}$, and the degree of B is $\ell - 1$.[3] This means that B completely hides A and therefore the inputs of the sender. Now the sender simply computes $w_i = A(\beta_i) \cdot v_i + B(\beta_i)$. Sender and receiver engage in an ℓ-out-of-n OTs,

[3]The value ℓ is fixed by the encoding, but we require that ℓ is uneven due to the fact that we have to reconstruct a polynomial of even degree $\frac{\ell-1}{2} + \frac{\ell-1}{2} = \ell - 1$, which requires ℓ values.

and the receiver picks the ℓ positions in L. He applies H to the obtained values and interpolates a polynomial Y which evaluates in position α_i to $a_i \cdot x_i + b_i$.

We keep the generic structure of the protocol of [19] in our protocol. In order to ensure correct and consistent inputs, we have to add additional checks. The complete description is given in Figure 4, and we give a high-level description of the ideas in the following paragraph.

First, we need to ensure that the receiver can only learn ℓ values, otherwise he could potentially reconstruct part of the input. Instead of using an expensive ℓ-out-of-n OT, we let the sender create a (ρ, n)-secret sharing (remember that $\rho + \ell = n$) of a random value e and input a share s_i together with a random value t_i into the OT. Depending on his set L, the receiver chooses t_i or the share s_i. Then he uses the shares to reconstruct e and sends it to the sender. This in turn might leak some information on L to the sender, if he can provide an inconsistent secret sharing. We thus force the sender to commit to e and later provide an unveil. Here the sender can learn some information on L, if he cheats but is not caught, but we can use our results from the previous section to show that this leakage is tolerable. The receiver can proceed and provide the encoding $\mathbf{v}$, which allows the sender to compute $\mathbf{w}$.

Second, we have to make sure that the sender computes the correct output. In order to catch a cheating sender, we add a check to the end of the protocol. Recall that the receiver knows the output Y. He can compute another polynomial X of his input and then pick a uniformly random challenge z_{R}. He sends it to the sender, who has to answer with $A(z_{\mathsf{R}}), B(z_{\mathsf{R}})$. Now the receiver can verify that $Y(z_{\mathsf{R}}) = A(z_{\mathsf{R}})X(z_{\mathsf{R}}) + B(z_{\mathsf{R}})$, i.e. the sender did not cheat in the noiseless positions. Again this leaks some information to the sender, but with the correct choice of parameters this leakage is inconsequential.

Security against a malicious receiver basically follows from the passively secure protocol. We only have to make sure that the extraction of his input is correct and that no information about the sender's inputs is leaked if e is incorrect. We thus mask the w_i by a one-time-pad and add the following check. This time the sender chooses z_{S} and the receiver has to answer with $X(z_{\mathsf{S}}, Y(z_{\mathsf{S}})$, which enforces correct extraction.

Theorem 3. *The protocol Π_{OLE} UC-realizes $\mathcal{F}_{\text{OLE}}$ in the OT-hybrid model with computational security.*

Proof. **Corrupted sender.** In the following we present a simulator $\mathcal{S}_{\mathsf{S}}$ which provides a computationally indistinguishable simulation of Π_{OLE} to a malicious sender $\mathcal{A}_{\mathsf{S}}$ (cf. Figure 5).

The main idea behind the extraction is the following. Since $\mathcal{S}_{\mathsf{S}}$ learns all inputs into the OTs, it can use the now available noisy elements $\hat{v}_i$ with $i \notin L$ to learn the input $\mathbf{a}$. The noiseless $\hat{v}_i, i \in L$ can be extrapolated to the noisy positions via a polynomial $\hat{Y}$ ($\hat{w}_i$ values imply a degree $\ell - 1$ polynomial for $i \in L$, and the receiver always learns ℓ values).

Ignoring for the moment that $\mathcal{A}_{\mathsf{S}}$ might provide inconsistent inputs, the simulator now knows two values for each position $\beta_i, i \notin L$: $\hat{w}_i = a_i \cdot \hat{v}_i + b_i$ and $\hat{Y}(\beta_i)$. Therefore, assuming that $\mathcal{A}_{\mathsf{S}}$ is honest, by computing $\hat{w}_i - \hat{Y}(\beta_i)$ the simulator gets $a_i \cdot \hat{v}_i + b_i - a_i \cdot \hat{x}_i + b_i = a_i(\hat{v}_i - \hat{x}_i)$, where $\hat{x}_i$ is the value that his input $\hat{\mathbf{x}} \in \mathbb{F}^t$ would imply according to the encoding $\hat{\mathbf{v}}_{|L}$ on position β_i. Since the simulator knows $\hat{v}_i$ and $\hat{x}_i$, it can simply compute a_i. From $\frac{\ell-1}{2} + 1$ of these points it can then compute the degree-$\frac{\ell-1}{2}$ polynomial A. From $Y = AZ + B$, it can then compute B and therefore the b_is. For this to work we only need $\frac{\ell-1}{2} + 1$ points. Therefore, if the corrupted sender sends incorrect values in at most κ positions $i \notin L$ and $\frac{\ell-1}{2} + 2\kappa < n$ there are still enough points to at least define a *correct* A and therefore also a *correct* $B = Y - AX$.

Protocol Π_{OLE}

Let $\mathcal{P} = \{\alpha_1, \ldots, \alpha_{\frac{\ell+1}{2}}, \beta_1, \ldots, \beta_n\}$ be a set of publicly known distinct points in $\mathbb{F}$. Further, let SS be a (ρ, n) secret sharing and COM be an OT-based commitment scheme. Set $\rho = \frac{7}{8}n$ and $\ell = n - \rho$.

1. Sender (Input $\mathbf{a}, \mathbf{b} \in \mathbb{F}^t$):
   - Draw a random polynomial A of degree $\frac{\ell-1}{2}$ with $A(\alpha_i) = a_i$ and a random polynomial B of degree $\ell - 1$ with $B(\alpha_i) = b_i$ $\forall i \in \{1, \ldots, t\}$.
   - Draw a uniformly random vector $\mathbf{t} \in \mathbb{F}^n$.
   - Draw a random value $e \in \mathbb{F}$ and compute $\mathbf{s} \leftarrow \mathsf{SS.Share}(e)$. Further compute $(\mathsf{com}, \mathsf{unv}) \leftarrow \mathsf{COM.Commit}(e)$.
   - Send com to the receiver and engage in n OT instances with input (t_i, s_i) for instance i.
2. Receiver (Input $\mathbf{x} \in \mathbb{F}^t$):
   - Start the encode procedure $\mathsf{Encode}_{n,\rho}(\mathbf{x}, \mathcal{P})$ and obtain $(G, H, L, \mathbf{v})$. Interpolate a polynomial X through the points (β_i, v_i) for $i \in L$.
   - For each OT instance i, if $i \in L$, set $\texttt{choice}_i = 0$, otherwise set $\texttt{choice}_i = 1$.
   - Obtain ℓ values t_i and ρ values $\tilde{s}_i$. Compute $\tilde{e} = \mathsf{SS.Reconstruct}(\tilde{\mathbf{s}}_{|\neg L})$.
   - Send $\tilde{e}$ to the sender.
3. Sender: Check if $\tilde{e} = e$, if not abort. Send unv to the receiver.
4. Receiver: Check if $\mathsf{COM.Open}(\mathsf{com}, \mathsf{unv}, \tilde{e}) = 1$, abort if not. Send $(G, \mathbf{v})$ to the sender.
5. Sender: Compute $\tilde{w}_i = A(\beta_i) \cdot v_i + B(\beta_i) + t_i$ for $i \in \{1, \ldots, n\}$. Send $\tilde{\mathbf{w}} = (\tilde{w}_1, \ldots, \tilde{w}_n)$ to the receiver.
6. Receiver: Set $w_i = \tilde{w}_i - t_i$ for $i \in L$ and interpolate the degree $\ell - 1$ polynomial Y through the points (β_i, w_i) for $i \in L$. Draw $z_{\mathsf{R}} \in \mathbb{F} \setminus \mathcal{P}$ uniformly at random and send z_{R} to the sender.
7. Sender: Draw $z_{\mathsf{S}} \in \mathbb{F} \setminus \mathcal{P}$ uniformly at random and send $\big(A(z_{\mathsf{R}}), B(z_{\mathsf{R}}), z_{\mathsf{S}}\big)$ to the receiver.
8. Receiver:
   - Check if $A(z_{\mathsf{R}}) \cdot X(z_{\mathsf{R}}) + B(z_{\mathsf{R}}) = Y(z_{\mathsf{R}})$ and abort if not.
   - Send $(X(z_{\mathsf{S}}), Y(z_{\mathsf{S}}))$ to the sender and output $\mathbf{y} = H\mathbf{w}_{|L}$.
9. Sender: Check if $A(z_{\mathsf{S}}) \cdot X(z_{\mathsf{S}}) + B(z_{\mathsf{S}}) = Y(z_{\mathsf{S}})$ and abort if not.

Figure 4: Actively secure realization of $\mathcal{F}_{\mathrm{OLE}}$ in the OT-hybrid model.

We now show that for every PPT environment $\mathcal{Z}$, the two distributions $\mathsf{Real}^{\mathcal{A}_{\mathsf{S}}}_{\Pi_{\mathrm{OLE}}}(\mathcal{Z})$ and $\mathsf{Ideal}^{\mathcal{S}_{\mathsf{S}}}_{\mathcal{F}_{\mathrm{OLE}}}(\mathcal{Z})$ are indistinguishable. Consider the following series of hybrid experiments.

Hybrid 0: This is the real protocol.

Hybrid 1: Identical to Hybrid 0, except that $\mathcal{S}_1$ extracts all inputs (s_i, t_i) input into OT by $\mathcal{A}_{\mathsf{S}}$.

Hybrid 2: Identical to Hybrid 1, except that $\mathcal{S}_2$ extracts the values $\bar{a}$ as shown in Figure 5 and aborts if the check in Step 8 is passed, but $\bar{a}_1, \ldots, \bar{a}_\rho$ has more than κ errors.

Hybrid 3: Identical to Hybrid 2, except that $\mathcal{S}_3$ encodes a random value $\hat{\mathbf{x}}$ as its input.

Simulator $\mathcal{S}_{\mathsf{S}}$

1. Let com^* be the message from $\mathcal{A}_{\mathsf{S}}$. Upon receiving $\mathtt{input}$ from $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t}}$, select a random value $\hat{\mathbf{x}} \in \mathbb{F}^t$ and compute $(\hat{G}, \hat{H}, \hat{L}, \hat{\mathbf{v}}) \leftarrow \mathsf{Encode}(\hat{\mathbf{x}})$. Further interpolate a polynomial $\hat{X}$ such that $\hat{X}(\beta_i) = \hat{v}_i$ for $i \in \hat{L}$.
2. Learn all of $\mathcal{A}_{\mathsf{S}}$'s inputs $(t_1^*, \ldots, t_n^*)$ and $\mathbf{s}^* = (s_1^*, \ldots, s_n^*)$ sent to the n OT instances.
   - Compute $\hat{e} \leftarrow \mathsf{SS.Reconstruct}(\mathbf{s}^*_{|\neg\hat{L}})$.
   - Send $\hat{e}$ to $\mathcal{A}_{\mathsf{S}}$.
3. Upon receiving unv^*, check if $\mathsf{COM.Open}(\mathsf{com}^*, \mathsf{unv}^*, \hat{e}) = 1$, if not abort. Send $(\hat{G}, \hat{\mathbf{v}})$ to $\mathcal{A}_{\mathsf{S}}$.
4. Upon receiving $\tilde{\mathbf{w}}^*$, do the following:
   - Compute $\hat{w}_i = \tilde{w}_i^* - t_i$ for all $i \in [n]$.
   - Interpolate the degree $\ell - 1$ polynomial $\hat{Y}$ such that $\hat{Y}(\beta_i) = \hat{w}_i$ for $i \in \hat{L}$.
   - Draw a random $\hat{z}_{\mathsf{R}} \in \mathbb{F} \setminus \mathcal{P}$ and send it to $\mathcal{A}_{\mathsf{S}}$.
5. Upon receiving $(A(\hat{z}_{\mathsf{R}})^*, B(\hat{z}_{\mathsf{R}})^*, z_{\mathsf{S}}^*)$, check if $A(\hat{z}_{\mathsf{R}})^* \cdot \hat{X}(\hat{z}_{\mathsf{R}}) + B(\hat{z}_{\mathsf{R}})^* = \hat{Y}(\hat{z}_{\mathsf{R}})$ and abort if not. Proceed as follows:
   - For all $i \notin \hat{L}$, set $\bar{a}_i = \frac{\hat{Y}(\beta_i) - \hat{w}_i}{\hat{X}(\beta_i) - \hat{v}_i}$.
   - Interpret the ρ points $\bar{a}_i$ as a Reed-Solomon encoded codeword. Decode $(\bar{a}_1, \ldots, \bar{a}_\rho)$ into $(\tilde{a}_1, \ldots, \tilde{a}_\rho)$ and abort if the codeword $(\bar{a}_1, \ldots, \bar{a}_\rho)$ contains more than κ errors. Interpolate a polynomial $\hat{A}$ such that $\hat{A}(\beta_i) = \tilde{a}_i$. Obtain $\hat{a}_1, \ldots, \hat{a}_t$ by evaluating $\hat{A}$ in $\alpha_1, \ldots, \alpha_t$.
   - Compute $\hat{b}_i = \hat{Y}(\beta_i) - \hat{X}(\beta_i)\hat{A}(\beta_i)$ for $i \in \hat{L}$. Interpolate a polynomial $\hat{B}$ such that $\hat{B}(\beta_i) = \hat{b}_i$. Obtain $\hat{b}_1, \ldots, \hat{b}_t$ by evaluating $\hat{B}$ in $\alpha_1, \ldots, \alpha_t$.
6. Set $\hat{\mathbf{a}} = (\hat{a}_1, \ldots, \hat{a}_t)$ and $\hat{\mathbf{b}} = (\hat{b}_1, \ldots, \hat{b}_t)$. Send $(\mathtt{inptS}, \hat{\mathbf{a}}, \hat{\mathbf{b}})$ to $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t}}$. Proceed with the simulation according to protocol.

Figure 5: Simulator against a corrupted sender in Π_{OLE}.

The indistinguishability of Hybrids 0 and 1 is immediate. We show that Hybrid 1 and Hybrid 2 are computationally indistinguishable in Lemma 9.1, and then we prove indistinguishability of Hybrid 2 and Hybrid 3 in Lemma 9.2.

Lemma 9.1. *Hybrids 1 and 2 are computationally indistinguishable from $\mathcal{Z}$'s view given that Assumption 2 holds.*

Proof. In order to prove the lemma, we have to show the following two statements.

1. $\mathcal{S}_2$ correctly extracts the input $\hat{\mathbf{a}}, \hat{\mathbf{b}}$, if there are less than κ errors in noisy positions.
2. The probability that $\mathcal{S}_2$ aborts due to more than κ errors in noisy positions is negligible in κ.

There are two ways in which $\mathcal{A}_{\mathsf{S}}$ can cheat and prevent the correct extraction: (1) it uses an inconsistent input for a noiseless value $\hat{v}_i, i \in L$ which leads to a wrong polynomial $\hat{Y}$ (and also an incorrect $\bar{a}_i$); (2) it uses an inconsistent input for a noisy value $\hat{v}_i, i \notin L$, which leads to incorrectly extracted values $\bar{a}_i$.

In case (1), the honest party will abort due to the check in Step 8 with overwhelming probability. It has to hold that $A(z)^* \cdot \hat{X}(z) + B(z)^* = \hat{Y}(z)$ for a uniformly chosen z. From Assumption 2 it follows that $\hat{X}$ (and thus $\hat{Y}$) are unknown to $\mathcal{Z}$, as they would be unconditionally hidden by a completely random vector. By the fundamental theorem of algebra there are at most $\deg(\hat{Y}) = \ell - 1$ possible values z for which $A(z)^* \cdot \hat{X}(z) + B(z)^* = \hat{Y}(z)$ for incorrect $A(z)^*, B(z)^*$. Since z_{R} is chosen uniformly at random from $\mathbb{F}$, the probability that the check succeeds with incorrect $A(z)^*, B(z)^*$ is thus upper bounded by $\frac{\ell-1}{|\mathbb{F}|}$, which is negligible in the security parameter. This means that the check in Step 8 ensures that *all* the values $\hat{w}_i$ for $i \in L$ are correct.

For case (2), we first argue that the extraction also succeeds if $\mathcal{A}_{\mathsf{S}}$ adds less than κ errors in noisy positions (the simulator will abort if more than κ errors occur). By the choice of parameters it holds that $\rho > 3\kappa = 6\ell$, and the simulator learns ρ values a_i that are supposed to represent a degree $\frac{\ell-1}{2}$ polynomial. Applying a standard Reed-Solomon decoder then yields the correct values a_i, i.e. if less than κ errors occur, $\mathcal{S}_{\mathsf{S}}$ extracts the correct $\mathbf{a} \in \mathbb{F}^t$ (and thus also the correct $\mathbf{b} \in \mathbb{F}^t$).

This shows that as long as there are less than κ errors in noisy positions, the extracted values are correct.

We claim that a $\mathcal{Z}$ that places more than κ errors in noisy positions breaks Assumption 2. The scenario of $\mathcal{Z}$ in the simulation is identical to the game $\mathcal{G}_{\mathsf{ident}}$: $\mathcal{Z}$ gets an encoding $\mathbf{v} \leftarrow \mathsf{Encode}_{n,\rho}(\mathbf{x})$ with ρ noisy positions and has to output a set of positions $Q \subseteq [n]$ such that $Q \subseteq R$ and $|Q| \geq \kappa$.

As discussed in Section 3, we can assume that $\mathsf{Encode}_{n,n/2}$ yields encodings that are indistinguishable from $\mathsf{Encode}_{n,n}$, i.e. truly random strings. In order to meet the requirements of Theorem 2, it therefore has to hold that $\sigma = n\frac{\rho-\kappa}{n-\kappa} \geq \frac{n}{2}$. Thus, we get that ρ has to be larger than $\frac{n+\kappa}{2}$, which by our choice of parameters is the case. Thus the claim directly follows from Theorem 2. $\square$

Lemma 9.2. *Hybrids 2 and 3 are computationally indistinguishable from $\mathcal{Z}$'s view given that Assumption 2 holds and* COM *is a UC commitment scheme.*

Proof. Assume that there exists a PPT $\mathcal{Z}$ that distinguishes Hybrids 2 and 3 with non-negligible probability ϵ. We will show that $\mathcal{Z}$ breaks Assumption 2 with non-negligible probability.

We have to consider all the messages that $\mathcal{A}_{\mathsf{S}}$ receives during a protocol run. First note that $\mathcal{S}_{\mathsf{S}}$ (resp. R) outputs either e or aborts in Step 4. Assume for the sake of contradiction that $\mathcal{A}_{\mathsf{S}}$ manages to create two secret sharings $s_{1,1}, \ldots, s_{1,n}$ and $s_{2,1}, \ldots, s_{2,n}$ for values e, e' such that R outputs both of e or e' with non-negligible probability ϵ without aborting depending on the set L and L', respectively. Then we create an adversary $\mathcal{B}$ from $\mathcal{Z}$ that breaks the binding property of COM. $\mathcal{B}$ simulates the protocol and learns all values s_i^*, then draws two uniformly random sets L, L'. $\mathcal{B}$ samples via L and L' two subsets of secret sharings that reconstruct to e and e', respectively, with non-negligible probability. It must hold for both values that $\mathsf{COM.Open}(\mathsf{com}, \mathsf{unv}, e) = \mathsf{COM.Open}(\mathsf{com}, \mathsf{unv}, e') = 1$, otherwise $\mathcal{B}$ aborts as the real R would. Since $\mathcal{A}_{\mathsf{S}}$ achieves that R outputs e or e' with non-negligible probability, $\mathcal{B}$ outputs com, e, e' with non-negligible probability to the binding experiment and thereby breaks the binding property of COM.

The next message he receives is the encoding. Recall that the choice bits into the OTs are derived from the set L of the encoding, i.e. a cheating $\mathcal{A}_{\mathsf{S}}$ might try to use inconsistent inputs (e.g. incorrect s_i values in positions that are supposedly not in L) in the OT such that R aborts depending on the set L. However, $\mathcal{A}_{\mathsf{S}}$ has to cheat before knowing the encoding $\mathbf{v}$ and as shown above always learns the same e, thus he can obtain at most 1 bit of leakage, namely whether the cheating was detected or not. We will now show that the leakage in Step 4 does not help $\mathcal{Z}$ to distinguish. The situation for a malicious $\mathcal{Z}$ is identical to game $\mathcal{G}_{\mathsf{leak}}$. First, $\mathcal{A}_{\mathsf{S}}$ has to decide on a set of values which he believes are not in L. Then he is informed (via a successful check) that his guess was correct, and given the encoding. Now he has to decide whether he is given a random encoding or not. We can directly apply Theorem 1, given that $\rho \geq \frac{2}{3}n$, and get that $\mathcal{Z}$'s distinguishing advantage is negligible.

After learning $\mathbf{v}$, $\mathcal{A}_{\mathsf{S}}$ has to compute the values $\mathbf{w}$, which are checked in Step 8. By cheating in noisy positions, Step 8 will succeed, but $\mathcal{A}_{\mathsf{S}}$ learns some noisy positions by learning the bit whether the check succeeded. This case is more involved than the above step, since now $\mathcal{A}_{\mathsf{S}}$ can decide on the set Q after seeing the encoding $\mathbf{v}$. We argue that the distinguishing advantage of $\mathcal{Z}$ remains negligible. It is obvious that $\mathcal{A}_{\mathsf{S}}$ can always find $O(\log\kappa)$ noisy positions with polynomial probability simply by guessing. However, Theorem 2 guarantees that in this scenario $\mathcal{A}_{\mathsf{S}}$ cannot find more than $O(\log\kappa)$ noisy positions, if $\mathsf{Y}^{\sigma} \approx \mathsf{Y}^{n}$ for $\sigma = \frac{n\kappa}{n-\rho-\kappa}$. From Theorem 1 we know that if $Q = O(\log\kappa)$ and $\sigma > \frac{2}{3}n$, then $\mathsf{Y}^{\sigma} \approx \mathsf{Y}^{n}$. Combined, we have that for $\rho = n - \frac{\kappa}{2}$, $\mathcal{A}_{\mathsf{S}}$ cannot find more than $O(\log n)$ noisy positions and the distinguishing advantage of $\mathcal{Z}$ is negligible. This concludes the proof. $\square$

Corrupted receiver. In the following we present a simulator $\mathcal{S}_{\mathsf{R}}$ which provides a statistically indistinguishable simulation of Π_{OLE} to a malicious receiver $\mathcal{A}_{\mathsf{R}}$ (cf. Figure 6). Conceptually the simulation is straight forward. The simulator learns all choice bits and thus can reconstruct the set L, which is sufficient to decode the codeword $\mathbf{v}$. Knowing X, $\mathcal{S}_{\mathsf{R}}$ can easily derive consistent inputs A, B. Care has to be taken since $\mathcal{A}_{\mathsf{R}}$ obtains one additional pair of values related to the polynomials A and B, thus he can tamper with the extraction. In a little more detail, he obtains one more value than necessary to reconstruct Y and can therefore play both with the degree of his input as well as with the correctness of L and $\mathbf{v}$. We describe and analyze a subtle attack in Lemma 9.4, which makes the analysis a bit more complex.

We now show the indistinguishability of the simulation in a series of hybrid experiments. For every PPT environment $\mathcal{Z}$, the two distributions $\mathsf{Real}^{\mathcal{A}_{\mathsf{S}}}_{\Pi_{\mathrm{OLE}}}(\mathcal{Z})$ and $\mathsf{Ideal}^{\mathcal{S}_{\mathsf{S}}}_{\mathcal{F}_{\mathrm{OLE}}}(\mathcal{Z})$ are indistinguishable.

Hybrid 0: This is the real protocol.

Hybrid 1: Identical to Hybrid 0, except that $\mathcal{S}_1$ extracts all inputs $\mathtt{choice}_i$ input into OT by $\mathcal{A}_{\mathsf{R}}$.

Hybrid 2: Identical to Hybrid 1, except that $\mathcal{S}_2$ aborts if $\mathcal{A}_{\mathsf{R}}$ passes the check in Step 3, although he selects less than ρ values s_i.

Hybrid 3: Identical to Hybrid 2, except that $\mathcal{S}_3$ reconstructs $\hat{X}$ as shown in Figure 6 and aborts if $\hat{Y}(\hat{z}_{\mathsf{S}}) \neq Y^*(\hat{z}_{\mathsf{S}})$, $\hat{X}(\hat{z}_{\mathsf{S}}) \neq X^*(\hat{z}_{\mathsf{S}})$ or $\hat{Y} = R$.

Indistinguishability of Hybrids 0 and 1 is trivial. We show the indistinguishability of Hybrids 1 and 2 in Lemma 9.3, based on the privacy of the secret sharing and the hiding property of the commitment. In Lemma 9.4 we show that we can always extract the correct input of $\mathcal{A}_{\mathsf{R}}$ and thus Hybrid 2 and Hybrid 3 are statistically indistinguishable.

Lemma 9.3. *Hybrids 1 and 2 are statistically indistinguishable from $\mathcal{Z}$'s view given that* SS *is a perfectly private secret sharing and* COM *is a statistically hiding commitment scheme.*

Simulator $\mathcal{S}_{\mathsf{R}}$

1. Upon receiving a message $\mathtt{input}$ from $\mathcal{F}^{\mathrm{t}}_{\mathrm{OLE}}$, simulate the first part of Π_{OLE} with random inputs.
   - Draw a uniformly random vector $\hat{\mathbf{t}} \in \mathbb{F}^n$ and a random value $\hat{e} \in \mathbb{F}$.
   - Compute $\hat{\mathbf{s}} \leftarrow \mathsf{SS.Share}(\hat{e})$ and $(\widehat{\mathsf{com}}, \widehat{\mathsf{unv}}) \leftarrow \mathsf{COM.Commit}(\hat{e})$.
   - Send $\hat{\mathsf{com}}$ to $\mathcal{A}_{\mathsf{R}}$ and engage in n OT instances with input $(\hat{t_i}, \hat{s_i})$ for instance i.
2. Learn all choice bits $(\mathtt{choice}^*_1, \ldots, \mathtt{choice}^*_n)$ of $\mathcal{A}_{\mathsf{R}}$ from the n OT instances. Reconstruct $\hat{L}$ as follows: for each $i \in [n]$, if $\mathtt{choice}^*_i = 0$ then $i \in \hat{L}$.
3. Upon receiving e^*, check if $\hat{e} = e^*$, otherwise abort. Send $\hat{\mathsf{unv}}$ to $\mathcal{A}_{\mathsf{R}}$.
4. Upon receiving $(G^*, \mathbf{v}^*)$ from $\mathcal{A}_{\mathsf{R}}$, proceed as follows.
   (a) Let $\deg(P_{\hat{L}})$ denote the degree of the polynomial defined by $\mathbf{v}_{|\hat{L}}$.
      - If $|\hat{L}| = \ell - 1$, interpolate the polynomial $P_{\hat{L}}$ defined over $\mathbf{v}_{|\hat{L}}$. If $\deg(P_{\hat{L}}) \leq \frac{\ell-1}{2}$, set $\hat{X} = P_{\hat{L}}$.
      - If $|\hat{L}| = \ell$, interpolate the polynomial $P_{\hat{L}}$ defined over $\mathbf{v}_{|\hat{L}}$. If $\deg(P_{\hat{L}}) \leq \frac{\ell-1}{2}$, set $\hat{X} = P_{\hat{L}}$. If $\deg(P_{\hat{L}}) > \frac{\ell+1}{2}$, try for all $\hat{i} \in \hat{L}$ if it holds that for $\hat{L}' = \hat{L} \setminus \hat{i}$, $\deg(P_{\hat{L}'}) \leq \frac{\ell-1}{2}$. If such an $\hat{i}$ exists, set $\hat{X} = P_{\hat{L}'}$ and $\hat{L} = \hat{L}'$.

   (b) Compute $\hat{x}_i = \hat{X}(\alpha_i), i \in [t]$ and send $(\mathtt{inputR}, \hat{\mathbf{x}})$ to $\mathcal{F}^{\mathrm{t}}_{\mathrm{OLE}}$. Let $(\mathtt{output}, \hat{\mathbf{y}})$ be the result. Pick a random polynomial $\hat{Y}$ such that $\deg(\hat{Y}) = \deg(\hat{X}) + \frac{\ell-1}{2}$ and $\hat{Y}(\alpha_i) = \hat{y}_i, i \in [t]$. If no $\hat{X}$ was extracted in Step 4a, set $\hat{Y}$ to be a random degree $\ell - 1$ polynomial R.

   (c) For $i \in \hat{L}$, set $\hat{w}_i = \hat{Y}(\beta_i) + t_i$, otherwise pick a uniform $\hat{w}_i$ and send $\hat{\mathbf{w}}$ to $\mathcal{A}_{\mathsf{R}}$.
5. Upon receiving z^*_{R}, draw $\hat{z}_{\mathsf{S}} \in \mathbb{F}$ and proceed as follows:
   - If $\hat{Y} \neq R$, compute $\hat{X}(z^*_{\mathsf{R}}), \hat{Y}(z^*_{\mathsf{R}})$ and sample a random $\hat{b} \in \mathbb{F}$. Set $\hat{a} = \frac{\hat{Y}(z^*_{\mathsf{R}}) - \hat{b}}{\hat{X}(z^*_{\mathsf{R}})}$ and send $(\hat{a}, \hat{b}, \hat{z}_{\mathsf{S}})$ to $\mathcal{A}_{\mathsf{R}}$.
   - If $\hat{Y} = R$, pick random $\hat{a}, \hat{b} \in \mathbb{F}$ and send $(\hat{a}, \hat{b}, \hat{z}_{\mathsf{S}})$ to $\mathcal{A}_{\mathsf{R}}$.
6. Upon receiving $(X^*(\hat{z}_{\mathsf{S}}), Y^*(\hat{z}_{\mathsf{S}}))$, proceed as follows:
   - If $\hat{Y} \neq R$, check if $Y^*(z_{\mathsf{S}}) = \hat{Y}(z_{\mathsf{S}})$ and $X^*(z_{\mathsf{S}}) = \hat{X}(z_{\mathsf{S}})$ and abort if not.
   - If $\hat{Y} = R$, abort.

Figure 6: Simulator against a corrupted receiver in Π_{OLE}.

Proof. Assume for the sake of contradiction that there exists an environment $\mathcal{Z}$ that distinguishes the hybrids, i.e., $\mathcal{Z}$ has to make $\mathcal{S}_2$ abort with non-negligible probability ε. We will construct from $\mathcal{Z}$ an adversary $\mathcal{B}$ that breaks the hiding property of COM with non-negligible probability. $\mathcal{B}$ simulates the protocol exactly like $\mathcal{S}_2$, but creates a secret sharing of a random r and picks two random e, e', which he sends to the hiding experiment. The hiding experiment returns a commitment com on one of these values. Then $\mathcal{B}$ integrates the commitment and secret sharing into the simulation and checks whether $\mathcal{Z}$ inputs less than ρ values $\mathtt{choice}_i = 1$ into OT, otherwise $\mathcal{B}$ aborts. Since SS is a perfectly private secret sharing and $\mathcal{Z}$ obtains less than ρ values s_i, these values leak nothing about r and the simulation of $\mathcal{B}$ is indistinguishable from $\mathcal{S}_2$'s simulation. Let now e^* be $\mathcal{Z}$'s answer in the simulated protocol. $\mathcal{B}$ simply forwards e^* to the hiding experiment. Since it has to hold that

$e^* = e$ or $e^* = e'$ with non-negligible probability ε (otherwise the check in Step 3 fails), $\mathcal{B}$ breaks the hiding property of COM with the same probability. From this contradiction it follows that $\mathcal{A}_\mathsf{R}$ learns at most ℓ values t_i through OT. $\square$

Lemma 9.4. *Hybrids 2 and 3 are statistically indistinguishable from $\mathcal{Z}$'s view.*

Proof. In order to distinguish Hybrids 2 and 3, $\mathcal{Z}$ must pass the check in Step 9, even though it holds that $\mathcal{S}_3$ picked a random polynomial R (allowing to distinguish the simulation from the real protocol). First note that the result $\mathbf{w}$ always defines a polynomial of degree $\ell - 1$ if $\mathcal{A}_\mathsf{R}$'s input polynomial has degree less than $\frac{\ell-1}{2}$. As we know from Lemma 9.3, $\mathcal{A}_\mathsf{R}$ learns at most ℓ values through the OTs and then one additional pair (a, b) via the check in Step 9.

Before we look at the details of the extraction, let us first describe an generic adversarial strategy that we have to cover. The adversary gets 1 free query and might try to use this query to prevent extraction. Say he picks a polynomial of degree $\frac{\ell-1}{2}$, but only uses $\ell - 1$ values of L. In the choice phase, he selects a random index $i^* \notin L$ and sets $\mathtt{choice}_{i^*} = 0$, i.e. $\mathcal{S}_3$ will assume this index is also in L. Towards the same goal, $\mathcal{A}_\mathsf{R}$ can simply set the value v_i for a random index i to a random value. $\mathcal{S}$ will then extract a wrong polynomial (with degree greater than $\frac{\ell+1}{2}$), while $\mathcal{A}_\mathsf{R}$ can still reconstruct Y via the additional values. However, since $\mathcal{A}_\mathsf{R}$ can only add exactly 1 random element, $\mathcal{S}_3$ can identify the index by trying for each $i \in L$ whether the set $L' = L \setminus i$ defines a polynomial of degree $\frac{\ell-1}{2}$ over the v_i. Here it is essential that there are no two sets L_1, L_2 with $|L_1| = \ell - 1, |L_2| = \ell$ such that $L_1 \subset L_2$ and $\deg(P_{L_1}) = \frac{\ell-1}{2}, \deg(P_{L_2}) = \frac{\ell+1}{2}$, i.e. there is only one possible index i that can be removed. This follows from the fact that the polynomial $P = P_{L_2} - P_{L_1}$ has only $\frac{\ell+1}{2}$ roots, but L_1 and L_2 have to agree on $\ell - 1$ positions. If that scenario were possible, $\mathcal{S}_3$ would not be able to distinguish these cases.

Let in the following $\deg(P_{\hat{L}})$ denote the degree of the polynomial that is defined by the points v_i for $i \in \hat{L}$.

- $|\hat{L}| \leq \ell - 2$: $\mathcal{A}_\mathsf{R}$ obtains at most $\ell - 2 + 1$ points, but Y is of degree $\ell - 1$ and thus underspecified. Clearly $\mathcal{A}_\mathsf{R}$'s probability of answering the check in Step 9 with a correct $X^*(z_\mathsf{S}), Y^*(z_\mathsf{S})$ is negligible in $\mathbb{F}$. Since $\mathcal{S}_3$ aborts as well, Hybrids 2 and 3 are indistinguishable in this case.
- $|\hat{L}| = \ell - 1$: In this case it holds that $\hat{Y} = R$ only if $\deg(P_{\hat{L}}) \geq \frac{\ell+1}{2}$.
  - $\deg(P_{\hat{L}}) = \frac{\ell-1}{2}$: In this case $\mathcal{A}_\mathsf{R}$ can reconstruct Y and pass the check in Step 9, but $\mathcal{S}_3$ extracts the correct $\hat{X}$. From the argument above, there cannot exist another polynomial X' that fits with the set $\hat{L}$ and thus Hybrids 2 and 3 are indistinguishable.
  - $\deg(P_{\hat{L}}) = \frac{\ell+1}{2}$: In this case $\mathcal{A}_\mathsf{R}$ obtains $\ell - 1 + 1$ points, but the resulting Y is of degree $\frac{\ell-1}{2} + \frac{\ell+1}{2} = \ell$, i.e. $\mathcal{A}_\mathsf{R}$ needs $\ell + 1$ points to reconstruct Y. By the same argument as above, Hybrids 2 and 3 are indistinguishable.
  - $\deg(P_{\hat{L}}) > \frac{\ell+1}{2}$: In this case $\mathcal{A}_\mathsf{R}$ can behave as described above, i.e. add a random i to the set $\hat{L}$ and thereby artificially increase $\deg(P_{\hat{L}})$. But since $|\hat{L}| = \ell - 1$, removing an additional value from $\hat{L}$ leads to the case $|\hat{L}| \leq \ell - 2$ and thus indistinguishability of Hybrids 2 and 3.
- $|\hat{L}| = \ell$: In this case it holds that $\hat{Y} = R$ only if $\deg(P_{\hat{L}}) > \frac{\ell+1}{2}$ and no index i can be identified to reduce $\deg(P_{\hat{L}})$ to $\frac{\ell-1}{2}$.

  - $\deg(P_{\hat{L}}) = \frac{\ell-1}{2}$: In this case $\mathcal{A}_{\mathsf{R}}$ can reconstruct Y and pass the check in Step 9, but $\mathcal{S}_3$ extracts the correct $\hat{X}$.
  - $\deg(P_{\hat{L}}) = \frac{\ell+1}{2}$: In this case $\mathcal{A}_{\mathsf{R}}$ obtains $\ell + 1$ points, and the resulting Y is of degree $\frac{\ell-1}{2} + \frac{\ell+1}{2} = \ell$. Thus $\mathcal{A}_{\mathsf{R}}$ can reconstruct Y and pass the check, but $\mathcal{S}_3$ extracts the correct $\hat{X}$.
  - $\deg(P_{\hat{L}}) > \frac{\ell+1}{2}$: In this case $\mathcal{A}_{\mathsf{R}}$ can behave as described above, i.e. add a random i to the set $\hat{L}$ and thereby artificially increase $\deg(P_{\hat{L}})$. Removing an additional value from $\hat{L}$ leads to the case $|\hat{L}| = \ell - 1$, i.e. $\mathcal{S}_3$ will simulate correctly. Otherwise, $\mathcal{S}_3$ will abort, but $\mathcal{A}_{\mathsf{R}}$ cannot reconstruct Y and thus fails the check in Step 9.

- $|\hat{L}| > \ell$: $\mathcal{S}_3$ aborts, and from Lemma 9.3 it follows that Hybrids 2 and 3 are indistinguishable.

The correctness of the simulation follows from the fact that either $\mathcal{S}_3$ extracts the correct input $\hat{X}$, or the check in Step 9 fails with overwhelming probability, in which case $\hat{X} = R$. Thus, the event that $\mathcal{Z}$ can provoke an abort is negligible, i.e. Hybrids 2 and 3 are indistinguishable. □

This concludes the proof. □

6 Efficient Oblivious Polynomial Evaluation

The ideal functionality $\mathcal{F}_{\mathrm{OPE}}$ for OPE is depicted in Figure 7. It allows the sender S to input a polynomial P and the receiver R to input $\alpha \in \mathbb{F}$.

Functionality $\mathcal{F}_{\mathrm{OPE}}$

1. Upon receiving a message $(\texttt{inputS}, P)$ from S where $P \in \mathbb{F}[X]$, verify that there is no stored tuple, else ignore that message. Store P and send a message $(\texttt{input})$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{inputR}, \alpha)$ from R with $\alpha \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Store α and send a message $(\texttt{input})$ to $\mathcal{A}$.
3. Upon receiving a message $(\texttt{deliver}, \mathsf{S})$ from $\mathcal{A}$, check if both P and α are stored, else ignore that message. Send $(\texttt{delivered})$ to S.
4. Upon receiving a message $(\texttt{deliver}, \mathsf{R})$ from $\mathcal{A}$, check if both P and α are stored, else ignore that message. Send $(\texttt{output}, P(\alpha))$ to R.

Figure 7: Ideal functionality for an oblivious polynomial evaluation.

In the remainder of this section we will establish the following theorem.

Theorem 4. *There exists a (constant-round) protocol Π_{OPE} that UC-realizes $\mathcal{F}_{\mathrm{OPE}}$ with unconditional security in the $\mathcal{F}^{\mathrm{t}}_{\mathrm{OLE}}$-hybrid model. In particular, for a polynomial P of degree d, $t = d + 2$.*

Our roadmap is as follows. We first show how to reduce $\mathcal{F}_{\mathrm{OPE}}$ to an intermediate OLE-based functionality $\mathcal{F}^{\mathrm{t},1}_{\mathrm{OLE}}$. After establishing this we present an efficient reduction of $\mathcal{F}^{\mathrm{t},1}_{\mathrm{OLE}}$ to $\mathcal{F}^{\mathrm{t}}_{\mathrm{OLE}}$ (or $\mathcal{F}_{\mathrm{OLE}}$).

We follow the generic idea of Naor and Pinkas [26] of using the linearization technique from [15] to construct an oblivious polynomial evaluation protocol. They decompose a polynomial P of degree d into d *linear* functions. These functions can then be evaluated using our OLE with input

α for each of the functions, and the receiver can reconstruct the value $P(\alpha)$. We state the lemma here and defer the proof to Appendix B.

Lemma 10 ([15])**.** *For every polynomial P of degree d, there exist d linear polynomials $P_1, \dots, P_d$, such that an OPE of P can be reduced to a parallel execution of an OLE of each of $P_1, \dots, P_d$, where all the linear polynomials are evaluated at the same point.*

In the semi-honest case, this approach directly works with the $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t}}$ for $t = d$. But unlike the construction of [26], our batch-OLE does not enforce the receiver to use the same input α in all of the OLEs. Therefore we cannot use the reduction of [26] that shows malicious security against a receiver. In particular, a malicious receiver might learn some non-trivial linear combinations of the coefficients of P.

Reducing $\mathcal{F}_{\mathrm{OPE}}$ to $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t},1}$. As a first step we reduce OPE to a variant of OLE where the receiver has only one input $x \in \mathbb{F}$, while the sender inputs two vectors $\mathbf{a}, \mathbf{b}$. This is depicted in Figure 8.

Functionality $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t},1}$

1. Upon receiving a message $(\texttt{inputS}, \mathbf{a}, \mathbf{b})$ from S with $\mathbf{a}, \mathbf{b} \in \mathbb{F}^t$, verify that there is no stored tuple, else ignore that message. Store $\mathbf{a}$ and $\mathbf{b}$ and send a message $(\texttt{input})$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{inputR}, x)$ from R with $x \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Store x and send a message $(\texttt{input})$ to $\mathcal{A}$.
3. Upon receiving a message $(\texttt{deliver}, \mathsf{S})$ from $\mathcal{A}$, check if both $\mathbf{a}, \mathbf{b}$ and x are stored, else ignore that message. Send $(\texttt{delivered})$ to S.
4. Upon receiving a message $(\texttt{deliver}, \mathsf{R})$ from $\mathcal{A}$, check if both $\mathbf{a}, \mathbf{b}$ and x are stored, else ignore that message. Set $y_i = a_i \cdot x + b_i$ for $i \in [t]$ and send $(\texttt{output}, \mathbf{y})$ to R.

Figure 8: Ideal functionality for a $(t, 1)$-oblivious linear function evaluation.

The reduction of $\mathcal{F}_{\mathrm{OPE}}$ to $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t},1}$ is straightforward, given Lemma 10. The sender decomposes his polynomial P into d linear functions $f_1, \dots, f_d$ with coefficients (a_i, b_i) and inputs these into $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{d},1}$. The receiver chooses his input α and obtains d linear evaluations, from which he can reconstruct $P(\alpha)$. The number of OLEs required is only dependent on the realization of $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{d},1}$.

Lemma 11. *The protocol Π_{OPE} UC-realizes $\mathcal{F}_{\mathrm{OPE}}$ in the $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{d},1}$-hybrid model with unconditional security.*

Proof. The security of Π_{OPE} is immediate: the simulator simulates $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{d},1}$ and learns all inputs, which it simply forwards to $\mathcal{F}_{\mathrm{OPE}}$ (and reconstructs if necessary). The correctness of the decomposition of P follows from Lemma 10. $\square$

Note that by taking our approach, we also remove the need for the stronger assumption of [26], while having a comparable efficiency in the resulting protocol.

Reducing $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t},1}$ to $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t+2}}$. As a second step, we need to realize $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t},1}$ from $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t}}$. Döttling et al. [10] describe a black-box protocol that realizes $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t},1}$ from $\mathcal{F}_{\mathrm{OLE}}$ (or our batch variant) with unconditional UC-security. The protocol has a constant multiplicative overhead of $2 + \varepsilon$ in the

Protocol Π_{OPE}

1. Sender (Input $P \in \mathbb{F}[X]$ of degree d):
   - Generate d linear polynomials of the form $f_i(x) = a_i x + b_i$, $\forall i \in [d]$, where $a_i, b_i \in \mathbb{F}$ according to Lemma 10.
   - Construct $\mathbf{a}, \mathbf{b} \in \mathbb{F}^d$, such that $\mathbf{a} = \{a_1, \ldots, a_d\}$ and $\mathbf{b} = \{b_1, \ldots, b_d\}$.
   - Send $(\texttt{inputS}, (\mathbf{a}, \mathbf{b}))$ to $\mathcal{F}_{\text{OLE}}^{\text{d,1}}$.
2. Receiver (Input $\alpha \in \mathbb{F}$):
   - Send $(\texttt{inputR}, \alpha)$ into $\mathcal{F}_{\text{OLE}}^{\text{d,1}}$.
   - Obtain $(\texttt{output}, \mathbf{y})$ from $\mathcal{F}_{\text{OLE}}^{\text{d,1}}$.
   - Compute $P(\alpha)$ from $\mathbf{y} = f_1(\alpha), \ldots, f_d(\alpha)$. Output $P(\alpha)$.

Figure 9: Reduction of $\mathcal{F}_{\text{OPE}}$ to $\mathcal{F}_{\text{OLE}}^{\text{d,1}}$.

number of OLEs, and works for any field $\mathbb{F}$. While this protocol basically solves our problem, we propose a more efficient variant that makes essential use of the fact that we only consider a large field $\mathbb{F}$. Our new approach requires only two additional OLEs.

Our solution for $\mathcal{F}_{\text{OLE}}^{\text{t,1}}$ is as follows. Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^t$ be given as input to the sender. It now needs to choose one additional pair of inputs (a_{t+1}, b_{t+1}) such that $\sum_{i=1}^{t+1} a_i = 0$ and b_{t+1} is uniformly random in $\mathbb{F}$. The sender inputs $\mathbf{a}', \mathbf{b}' \in \mathbb{F}^{t+1}$ into $\mathcal{F}_{\text{OLE}}^{\text{t+1}}$, while the receiver inputs $\mathbf{x}' = (x, \ldots, x) \in \mathbb{F}^{t+1}$. Now the receiver locally computes $c = \sum_{i=1}^{t+1} y_i = \sum_{i=1}^{t+1} a_i x + \sum_{i=1}^{t+1} b_i = \sum_{i=1}^{t+1} b_i$ and sends a commitment to c to the sender. This commitment can also be based on OLE, even in such a way that we can use $\mathcal{F}_{\text{OLE}}^{\text{t+2}}$ by precomputing the commitment (a detailed description is given in Appendix C). The sender answers with $c' = \sum_{i=1}^{t+1} b_i$, which the receiver can verify. This makes sure that the sender chose $\mathbf{a}'$ correctly, while c' itself does not give the receiver any new information. Now the receiver unveils, which shows the sender whether the receiver used the same x in each invocation. There is one small problem left: if the receiver cheated, he will be caught, be he might still learn some information about the senders inputs that cannot be simulated. In order to solve this issue, we let $\mathbf{a}'$ and $\mathbf{b}'$ be uniformly random and then replace these with the inputs after the check succeeded. A detailed description of the protocol is given in Figure 10.

Lemma 12. *The protocol $\Pi_{\text{OLE}}^{\text{t,1}}$ UC-realizes $\mathcal{F}_{\text{OLE}}^{\text{t,1}}$ in the $\mathcal{F}_{\text{OLE}}^{\text{t+2}}$-hybrid model with unconditional security.*

Proof. **Corrupted sender.** The simulator $\mathcal{S}_{\mathsf{S}}$ simulates $\mathcal{F}_{\text{OLE}}^{\text{t+2}}$ for the corrupted sender $\mathcal{A}_{\mathsf{S}}$. It extracts all the inputs, namely $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$. We do not need to extract the commitment, which also uses $\mathcal{F}_{\text{OLE}}^{\text{t+2}}$. $\mathcal{S}_{\mathsf{S}}$ sends a commitment to $\hat{c} = \sum_{i=1}^{t+1} \hat{v}_i$ to the receiver. If it holds that $\sum_{i=1}^{t+1} u_i \neq 0$, but the check in Step 4 succeeds, $\mathcal{S}_{\mathsf{S}}$ aborts. Otherwise, it computes $\hat{\mathbf{a}} = \hat{\mathbf{u}}'^{*} + \hat{\mathbf{u}}$ and $\hat{\mathbf{b}} = \hat{\mathbf{v}}'^{*} + \hat{\mathbf{v}}$ and inputs the first t elements of each into $\mathcal{F}_{\text{OLE}}^{\text{t,1}}$.

First note that if $\sum_{i=1}^{t+1} u_i \neq 0$, the commitment $\widehat{\mathsf{com}}$ contains an incorrect value. As long as the receiver always aborts in this case, the hiding property of COM guarantees indistinguishability of the simulation. So the only way that a malicious environment $\mathcal{Z}$ can distinguish the simulation from the real protocol is by forcing an abort. Note that if $\sum_{i=1}^{t+1} u_i = e \neq 0$, then c depends on x

Protocol $\Pi_{\mathrm{OLE}}^{\mathrm{t,1}}$

Let COM be an OLE-based commitment scheme, e.g. $\Pi_{\mathrm{COM}}^{\mathrm{pre}}$ from Appendix C.

1. Sender (Input $\mathbf{a}, \mathbf{b} \in \mathbb{F}^t$): Choose $\mathbf{u}, \mathbf{v} \in \mathbb{F}^{t+2}$ uniformly random such that $\sum_{i=1}^{t+1} u_i = 0$ and send $(\mathtt{inputS}, (\mathbf{u}, \mathbf{v}))$ to $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t+2}}$. Store (u_{t+2}, v_{t+2}) as the auxiliary receiver inputs for COM.
2. Receiver (Input $x \in \mathbb{F}$):
   - Set $\mathbf{x} = (x, \dots, x, w) \in \mathbb{F}^{t+2}$ with w random and send $(\mathtt{inputR}, \mathbf{x})$ into $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t+2}}$.
   - Obtain $(\mathtt{output}, \mathbf{z})$ from $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t+2}}$. Let $\bar{\mathbf{z}} = (z_1, \dots, z_t)$.
   - Let (w, z_{t+2}) be the auxiliary sender input for COM. Compute $c = \sum_{i=1}^{t+1} z_i$, $(\mathsf{com}, \mathsf{unv}) \leftarrow \mathsf{COM.Commit}(c)$ and send com to the sender.
3. Sender: Send $c' = \sum_{i=1}^{t+1} v_i$ to the receiver.
4. Receiver: Check if $c' = c$ and abort if not. Send unv to the sender.
5. Sender: Check if $\mathsf{COM.Open}(\mathsf{com}, \mathsf{unv}, c') = 1$ and abort if not. Send $\mathbf{u}' = \mathbf{a} - \bar{\mathbf{u}}$ and $\mathbf{v}' = \mathbf{b} - \bar{\mathbf{v}}$ to the receiver, where $\bar{\mathbf{u}}, \bar{\mathbf{v}}$ contain the first t values of $\mathbf{u}$,$\mathbf{v}$.
6. Receiver: Compute $\mathbf{y} = \mathbf{u}'x + \mathbf{v}' + \bar{\mathbf{z}} = \mathbf{a}x + \mathbf{b}$ and output $\mathbf{y}$.

Figure 10: Reduction of $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t,1}}$ to $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t+2}}$.

and is thus uniformly distributed, since

$$c = \sum_{i=1}^{t+1} z_i = \sum_{i=1}^{t+1} u_i x + \sum_{i=1}^{t+1} v_i = ex + c'.$$

Thus, the probability that $c' = c$ is negligible.

Corrupted receiver. The simulator $\mathcal{S}_{\mathsf{R}}$ against the corrupted receiver $\mathcal{A}_{\mathsf{R}}$ simulates $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t+2}}$ and learns $\hat{\mathbf{x}}$. It chooses $\hat{\mathbf{u}}, \hat{\mathbf{v}} \in \mathbb{F}^{t+2}$ according to $\Pi_{\mathrm{OLE}}^{\mathrm{t,1}}$, and computes $\hat{\mathbf{z}} \in \mathbb{F}^{t+2}$, where $\hat{z}_i = \hat{u}_i \hat{x}_i + \hat{v}_i$ $\forall i \in [1, t+2]$. $\mathcal{S}_{\mathsf{R}}$ sends $\hat{\mathbf{z}}$ to $\mathcal{A}_{\mathsf{R}}$. After receiving the commitment, $\mathcal{S}_{\mathsf{R}}$ sends $\hat{c}' = \sum_{i=1}^{t+1} \hat{z}_i$. It aborts if the commitment unveils correctly, even though $x_i \neq x_j$ for some $i, j \in [t+1]$. If that is not the case, it inputs $\hat{x}$ into $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{t,1}}$ and obtains $\mathbf{y}$. $\mathcal{S}_{\mathsf{R}}$ picks $\hat{\mathbf{v}}' \in \mathbb{F}^t$ uniformly at random, sets $\hat{u}_i' = \frac{y_i - \hat{z}_i - \hat{v}_i'}{x} \forall i \in [t]$. It sends $\hat{\mathbf{u}}', \hat{\mathbf{v}}'$ to $\mathcal{A}_{\mathsf{R}}$.

For an honest receiver, the check in Step 5 always succeeds. A malicious $\mathcal{Z}$ can only distinguish between the simulation and the real protocol by producing a correct commitment on c, even though $x_i \neq x_j$ for some $i, j \in [t+1]$. Since the commitment is binding, $\mathcal{A}_{\mathsf{R}}$ must commit to some value c before seeing c'. Let w.l.o.g. $x_j = (x + e) \neq x$ for some j. Then we have

$$c = \sum_{i=1}^{t+1} z_i = \sum_{\substack{i=1 \\ i \neq j}}^{t+1} u_i x + u_j(x+e) + \sum_{i=1}^{t+1} v_i = \sum_{i=1}^{t+1} u_i x + u_j e + \sum_{i=1}^{t+1} v_i = c' + u_j e.$$

But this means that c' is uniformly distributed from $\mathcal{A}_{\mathsf{R}}$'s point of view, because u_j is chosen uniformly and unknown to $\mathcal{A}_{\mathsf{R}}$. As a consequence, the probability that $\mathcal{Z}$ can distinguish the simulation from the real protocol is negligible. □

Combining the results from this section we get that $\mathcal{F}_{\mathrm{OPE}}$ for a polynomial P of degree d requires $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{d},1}$, which in turn can be based on $\mathcal{F}_{\mathrm{OLE}}^{\mathrm{d}+2}$. This establishes Theorem 4.

Remark. It is possible to evaluate several polynomials in parallel with the batch-OLE functionality, given that t is chosen of appropriate size. Then, for each polynomial the above described protocol is carried out (including making sure that the receiver uses the same α in all OLEs relevant to the respective polynomial).

References

[1] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

[2] D. Bleichenbacher and P. Q. Nguyen. Noisy polynomial interpolation and noisy Chinese remaindering. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 53–69. Springer, Heidelberg, May 2000.

[3] D. Boneh. Finding smooth integers in short intervals using CRT decoding. In *32nd ACM STOC*, pages 265–272. ACM Press, May 2000.

[4] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.

[5] I. Cascudo, I. Damgård, B. David, N. Döttling, and J. B. Nielsen. Rate-1, linear time and additively homomorphic UC commitments. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 179–207. Springer, Heidelberg, Aug. 2016.

[6] Y.-C. Chang and C.-J. Lu. Oblivious polynomial evaluation and oblivious neural learning. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 369–384. Springer, Heidelberg, Dec. 2001.

[7] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, Aug. 2012.

[8] B. M. David, R. Nishimaki, S. Ranellucci, and A. Tapp. Generalizing efficient multiparty computation. In A. Lehmann and S. Wolf, editors, *ICITS 15*, volume 9063 of *LNCS*, pages 15–32. Springer, Heidelberg, May 2015.

[9] N. Döttling, D. Kraschewski, and J. Müller-Quade. David & Goliath oblivious affine function evaluation - asymptotically optimal building blocks for universally composable two-party computation from a single untrusted stateful tamper-proof hardware token. Cryptology ePrint Archive, Report 2012/135, 2012. `http://eprint.iacr.org/2012/135`.

[10] N. Döttling, D. Kraschewski, and J. Müller-Quade. Statistically secure linear-rate dimension extension for oblivious affine function evaluation. In A. Smith, editor, *ICITS 12*, volume 7412 of *LNCS*, pages 111–128. Springer, Heidelberg, Aug. 2012.

[11] M. K. Franklin and M. Yung. Communication complexity of secure computation (extended abstract). In *24th ACM STOC*, pages 699–710. ACM Press, May 1992.

[12] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 303–324. Springer, Heidelberg, Feb. 2005.

[13] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

[14] N. Gilboa. Two party RSA key generation. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 116–129. Springer, Heidelberg, Aug. 1999.

[15] N. Gilboa. *Topics in private information retrieval.* PhD thesis, Thesis (Doctoral)–Technion - Israel Institute of Technology, Faculty of Computer Science, 2001, Haifa, 2001.

[16] C. Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 90–120. Springer, Heidelberg, Mar. 2015.

[17] C. Hazay and Y. Lindell. Efficient oblivious polynomial evaluation with simulation-based security. Cryptology ePrint Archive, Report 2009/459, 2009. `http://eprint.iacr.org/2009/459`.

[18] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, Aug. 2008.

[19] Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 294–314. Springer, Heidelberg, Mar. 2009.

[20] J. Katz. Universally composable multi-party computation using tamper-proof hardware. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, Heidelberg, May 2007.

[21] M. Keller, E. Orsini, and P. Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 830–842. ACM Press, Oct. 2016.

[22] A. Kiayias and M. Yung. Cryptographic hardness based on the decoding of reed-solomon codes. *IEEE Trans. Information Theory*, 54(6):2752–2769, 2008.

[23] J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[24] Y. Lindell and B. Pinkas. Privacy preserving data mining. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 36–54. Springer, Heidelberg, Aug. 2000.

[25] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *31st ACM STOC*, pages 245–254. ACM Press, May 1999.

[26] M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.

[27] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[28] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[29] B. Shankar, K. Srinathan, and C. P. Rangan. Alternative protocols for generalized oblivious transfer. In *ICDCN 2008.*, volume 4904 of *Lecture Notes in Computer Science*, pages 304–309, 2008.

[30] R. Tonicelli, A. C. A. Nascimento, R. Dowsley, J. Müller-Quade, H. Imai, G. Hanaoka, and A. Otsuka. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *International Journal of Information Security*, 14(1):73–84, 2015.

[31] S. Wolf and J. Wullschleger. Oblivious transfer is symmetric. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 222–232. Springer, Heidelberg, May / June 2006.

[32] H. Zhu and F. Bao. Augmented oblivious polynomial evaluation protocol and its applications. In S. D. C. di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 222–230. Springer, Heidelberg, Sept. 2005.

A Proof of Lemma 4

Proof. Let $B_i = \alpha_0 A_0 + \alpha_1 Z$. We first prove the first implication. Consider a distinguisher D for B_0 and B_1. Then

$$\begin{aligned}
\mathsf{Adv}_D(B_0, B_1) &= \Pr[D(B_1) = 1] - \Pr[D(B_0) = 1] \\
&= \sum_i \alpha_i \Pr[D(B_1) = 1 \mid c = i] - \sum_i \alpha_i \Pr[D(B_0) = 1 \mid c = i] \\
&= \alpha_0 \Pr[D(B_1) = 1 \mid c = 0] - \alpha_0 \Pr[D(B_0) = 1 \mid c = 0] \\
&= \alpha_0 \Pr[D(A_1) = 1] - \alpha_0 \Pr[D(A_0) = 1] \\
&= \alpha_0 \Pr[D(A_1) = 1] - \alpha_0 \Pr[D(A_0) = 1] \ ,
\end{aligned}$$

from which it follows that

$$\mathsf{Adv}_D(B_0, B_1) = \alpha_0 \mathsf{Adv}_D(A_0, A_1) \ . \tag{1}$$

From (1) it follows that $\mathsf{Adv}_D(B_0, B_1) \leq \alpha_0 \epsilon$ for all D, which proves the claim in the lemma. Consider a distinguisher D for A_0 and A_1. It can also act as distinguisher for B_0 and B_1, so from (1) we have that

$$\mathsf{Adv}_D(A_0, A_1) = \alpha_0^{-1} \mathsf{Adv}_D(B_0, B_1) \ .$$

From this the second claim follows. □

B Proof of Lemma 10

The full proof of Lemma 10 can be found in [26]. Here we reproduce the proof of first two claims, which is sufficient for our purpose, from their proof.

Claim 1. *For every polynomial $P \in \mathbb{F}[X]$ of degree d there exists d linear polynomials $f_1, \ldots, f_d \in \mathbb{F}[X]$ such that given $f_1(\alpha), \ldots, f_d(\alpha)$, for any $\alpha \in \mathbb{F}$, it is possible to compute the value of $f(\alpha)$.*

Proof. Let $P(x) = \sum_{i=0}^{d} a_i x^i$, where $a_i \in \mathbb{F}$ $\forall i \in [0, d]$. In Horner representation the polynomial can be rewritten as:

$$P(x) = (\ldots(((a_d x + a_{d-1})x + a_{d-2})x + a_{d-3})\ldots)x + a_0.$$

Define the d^{th} linear polynomial $f_d(x) = Q_d(x) - r_d$, where $r_d \in_r \mathbb{F}$ and $Q_d(x)$ is the inner-most linear polynomial in the Horner representation. That implies $f_d(x) = a_d x + a_{d-1} - r_d$. Now from the inner-most 2-degree polynomial of the Horner representation we get, $Q_{d-1}(x) = Q_d(x)x + a_{d-2} = f_d(x)x + r_d x + a_{d-2}$. Now define the $(d-1)^{th}$ linear polynomial as $f_{d-1}(x) = r_d x + a_{d-2} - r_{d-1}$, where $r_{d-1} \in_r \mathbb{F}$. In the same manner we can define linear polynomials up to P_2 as, $f_i(x) = r_{i+1}x + a_{i-1} - r_i$, where $r_i \in_r \mathbb{F}$, $\forall i \in [2, d]$. Define $f_1(x) = r_2 x + a_0$. If the linear polynomials are defined in this way, clearly:

$$P(x) = f_d(x) \cdot x^{d-1} + f_{d-1} \cdot x^{d-2} + \ldots + f_2(x) \cdot x + f_1(x). \tag{2}$$

Hence given $f_1(\alpha), \ldots, f_d(\alpha)$, for any $\alpha \in \mathbb{F}$, it is possible to compute the value of $P(\alpha)$ using Equation (2). □

Claim 2. *The computation of the d linear polynomials can be done by d OPEs that are executed in parallel.*

Proof. The linear polynomials $f_1, \ldots, f_d$ is independent of the value α. The sender can define the linear polynomials by sampling random $r_2, \ldots, r_d$ beforehand; After that the sender and the receiver can invoke d parallel OPEs to evaluate $f_1(\alpha), \ldots, f_d(\alpha)$. Then the receiver can evaluate $P(\alpha)$ using Equation (2).

Note that here the sender and the receiver evaluate d parallel instance of OPEs with linear polynomials. Hence, we use our protocol for batch-OLE ($\Pi^{\mathrm{d}}_{\mathrm{OLE}}$) to evaluate all the d linear instances at the same time. □

C UC-secure Commitment Scheme in $\mathcal{F}_{\mathrm{OLE}}$-Hybrid Model.

We describe the protocol from [9], which uses $\mathcal{F}_{\mathrm{OLE}}$ to construct a simple UC-secure commitment scheme Π_{COM} in Figure 11. Since OLE is symmetric [31], we do not need to make any assumption on the direction of $\mathcal{F}_{\mathrm{OLE}}$ in the protocol. The hiding property of the scheme is immediate from the fact that the random value b, chosen by the sender, acts as an one time pad for $c \cdot \alpha$. On the other hand, in the unveil phase if the sender can successfully open to a different value (m', b'), that directly breaks the receiver's privacy of $\mathcal{F}_{\mathrm{OLE}}$.

Protocol Π_{COM}

Commit Phase

1. Sender (Input $m \in \mathbb{F}$): Choose a random $k \in \mathbb{F}$. Input $(\texttt{inputS}, (m, k))$ into $\mathcal{F}_{\mathrm{OLE}}$.
2. Receiver: Choose a random $r \in \mathbb{F}$. Input $(\texttt{inputR}, r)$ into $\mathcal{F}_{\mathrm{OLE}}$ to obtain q.

Unveil Phase

3. Sender: Send (m, k) to the receiver.
4. Receiver: Accept if $m \cdot r + k = q$, abort otherwise.

Figure 11: Π_{COM} in the $\mathcal{F}_{\mathrm{OLE}}$-hybrid model.

Note that the commitment can be precomputed to a random value s. In order to fully commit to m, the sender sends $m' = m - s$ to the receiver. The receiver verifies $m'r + q = mr - sr + sr + b = m + b$. In the following protocol (cf. Figure 12) we show how to combine the inversion of OLE according to [31] and the derandomization to commit using a precomputed OLE in the direction from receiver to sender.

The security of this construction can be shown analogously to the security of Π_{COM} above.

Protocol $\Pi_{\text{COM}}^{\text{pre}}$

Commit Phase The sender has two values $s \in \mathbb{F}$ and $q = rs + t$ as auxiliary input, while the receiver has $r, t \in \mathbb{F}$ as auxiliary input.

1. Sender (Input $m \in \mathbb{F}$): Draw a uniformly random $k \in \mathbb{F}$, set $q' = q + k$ and $m' = m - s$. Send (q', m') to the receiver.
2. Receiver: Compute $\mathsf{com} = q' - t + m'r = mr + k$.

Unveil Phase

3. Sender: Send $\mathsf{unv} = (m, k)$ to the receiver.
4. Receiver: Accept if $\mathsf{com} = m \cdot r + k$, abort otherwise.

Figure 12: $\Pi_{\text{COM}}^{\text{pre}}$ provides a commitment based on a precomputed random OLE from receiver to sender.