# Reusable Non-Interactive Secure Computation

Melissa Chase[*]
Microsoft Research, Redmond

Yevgeniy Dodis[†]
New York University

Yuval Ishai[‡]
Technion

Daniel Kraschewski[§]
TNG Technology Consulting

Tianren Liu[¶]
MIT

Rafail Ostrovsky[||]
UCLA

Vinod Vaikuntanathan[**]
MIT

### Abstract

We consider the problem of Non-Interactive Secure Computation (NISC), a 2-message "Sender-Receiver" secure computation protocol that retains its security even when both parties can be malicious. While such protocols are easy to construct using garbled circuits and general non-interactive zero-knowledge proofs, this approach inherently makes a non-black-box use of the underlying cryptographic primitives and is infeasible in practice.

Ishai et al. (Eurocrypt 2011) showed how to construct NISC protocols that only use parallel calls to an ideal oblivious transfer (OT) oracle, and additionally make only a black-box use of any pseudorandom generator. Combined with the efficient 2-message OT protocol of Peikert et al. (Crypto 2008), this leads to a practical approach to NISC that has been implemented in subsequent works. However, a major limitation of all known OT-based NISC protocols is that they are subject to selective failure attacks that allows a malicious sender to entirely compromise the security of the protocol when the receiver's first message is reused.

Motivated by the failure of the OT-based approach, we consider the problem of basing *reusable* NISC on parallel invocations of a standard arithmetic generalization of OT known as oblivious linear-function evaluation (OLE). We obtain the following results:

- We construct an information-theoretically secure reusable NISC protocol for arithmetic branching programs and general zero-knowledge functionalities in the OLE-hybrid model. Our zero-knowledge protocol only makes an absolute constant number of OLE calls per gate in an arithmetic circuit whose satisfiability is being proved. As a corollary, we get reusable NISC/OLE for general Boolean circuits using any one-way function.
- We complement this by a negative result, showing that reusable NISC/OT is impossible to achieve, and a more restricted negative result for the case of the zero-knowledge functionality. This provides a formal justification for the need to replace OT by OLE.

[*] E-mail: melissac@microsoft.com
[†] E-mail: dodis@cs.nyu.edu
[‡] E-mail: yuvali@cs.technion.ac.il
[§] E-mail: kraschew@ira.uka.de. Work mostly done while at Technion.
[¶] E-mail: liutr@mit.edu
[||] E-mail: rafail@cs.ucla.edu
[**] E-mail: vinodv@csail.mit.edu

- We build a universally composable 2-message OLE protocol in the CRS model that can be based on the security of Paillier encryption and requires only a constant number of modular exponentiations. This provides the first arithmetic analogue of the 2-message OT protocols of Peikert et al. (Crypto 2008).
- By combining our NISC/OLE protocol and the 2-message OLE protocol, we get protocols with new attractive asymptotic and concrete efficiency features. In particular, we get the first (designated-verifier) NIZK protocols where following a statement-independent preprocessing, both proving and verifying are entirely "non-cryptographic" and involve only a constant computational overhead.

# 1 Introduction

Non-interactive secure computation (NISC) refers to the problem where Rachel wishes to publish an encryption of her input $x$, in such a way that Sam, who holds an input $y$, can send her a single message conveying the value $f(x, y)$ and nothing more. In the semi-honest setting, there are several solutions to this problem including (i) garbled circuits [Yao86, LP09] combined with two-message oblivious transfer (OT) protocols (e.g., [NP01, AIR01, PVW08]) and (ii) fully homomorphic encryption [RAD78, Gen09]. One could compile these protocols to be secure against malicious parties by using general non-interactive zero-knowledge (NIZK) proofs in the common reference string (CRS) model [BFM88]; however, this requires making non-black-box use of the underlying cryptographic primitives, and is generally infeasible in practice. A recent line of work [IKO$^+$11, AMPR14] has come up with efficient maliciously secure NISC protocols in the OT-hybrid model (which we refer to as NISC/OT protocols) that make black-box use of efficiently implementable cryptographic primitives such as a pseudorandom generator.

In general, the paradigm of designing protocols in the OT-hybrid model that are either unconditionally secure or make use of symmetric cryptographic primitives, and plugging in fast implementations of OT, has paid great dividends in cryptography for several reasons. First, we have efficient OT implementations under standard assumptions; and secondly, OT is self-reducible, so the cryptographic cost of implementing it can be pushed to an offline phase, and it can be implemented with information-theoretic security given correlated randomness. In short, combining efficient NISC/OT protocols with efficient 2-message OT implementations, we can get efficient "public-key" non-interactive variants of secure computation, as was recently accomplished in [IKO$^+$11, AMPR14]. This approach is beneficial even in simpler special cases such as (designated-verifier) NIZK. For such cases (and more generally for functionalities computed by log-depth circuits or polynomial-size branching programs) such NISC/OT protocols can be made information-theoretic.

The starting point of this work is that this rosy picture belies a major defect of all known NISC/OT protocols. To see this, imagine that Rachel wants to publish a *reusable* encryption of her input $x$, obtain messages from anyone in the world with inputs $y_i$, conveying to her the value of $f(x, y_i)$. In the semi-honest setting, all the old NISC/OT protocols work just fine. However, in all known NISC/OT protocols (including [IKO$^+$11, AMPR14]), a malicious Sam can mount a "selective failure attack", feeding malformed OT messages to Rachel and checking whether she aborts or not, and using this information to violate both the secrecy of her input and the correctness of her output. Indeed, we show that this is inherent: our first result is that there is no information-theoretic *reusably secure* implementation of NISC/OT for general functions. We also prove a similar result for zero-knowledge functionalities, though in a more restricted "black-box" framework.

Our key observation is that this inherent limitation of OT-based protocols can be overcome if we replace OT by an *arithmetic extension* of OT known as oblivious linear function evaluation (OLE). The OLE functionality maps sender inputs $(a, b)$ and receiver input $x$ to receiver output $ax + b$, where $a, b, x$ are taken from some (typically large) field or ring. The main result of this paper is a general-purpose *reusable* non-interactive secure computation protocol that only makes *parallel OLE calls* over a large ring. We denote such a protocol by rNISC/OLE.

Our rNISC/OLE protocol is information-theoretic and does not rely on any cryptographic assumptions. Its complexity is polynomial in the size of an arithmetic branching program being computed. This is sufficient to capture arithmetic log-depth ($\mathrm{NC}^1$) circuits. However, in the important special case of zero-knowledge functionalities, where verification can always be done by a shallow circuit, our NISC/OLE protocol is still information-theoretic and only makes a constant number of OLE calls per gate in an arithmetic circuit whose satisfiability is being proved. (Minimizing this constant, which we did not attempt to optimize, is an interesting future research direction that can be motivated by practical implementations.) As a corollary, we can also get reusable NISC/OLE (beyond zero-knowledge) for general boolean circuits using any one-way function.

We complement our rNISC/OLE protocol by proposing an efficient secure implementation of the OLE oracle which is compatible with our efficiency goals. Concretely, assuming the security of Paillier's encryption scheme [Pai99], we construct a universally secure 2-message OLE protocol in the CRS model (over the ring $\mathbb{Z}_N$ for an RSA modulus $N$). The communication cost of the protocol involves a constant number of group elements and its computational cost is dominated by a constant number of exponentiations. This protocol provides the first arithmetic analogue of the 2-message OT protocol of PVW [PVW08], which is commonly used in implementations of secure two-party computation (in particular, it is used by the non-interactive ones from [AMPR14]). Our efficient OLE protocol is independently motivated by other applications of OLE in cryptography; see [ADI+17, GNN17] and references therein.

The switch from OT to OLE has some unexpected efficiency benefits. Beyond the reusability issue, OT-based protocols in the malicious security model use a "cut and choose" approach that has considerable (super-constant) overhead in communication and computation. While there are effective techniques for amortizing the communication overhead (cf. [IPS08]), these come at the expense of a super-constant computational overhead and apply only in the boolean setting. Other approaches that employ OLE and apply to the arithmetic setting, such as the ones from [GIP+14, DGN+17], are inherently interactive.

The combination of our information-theoretic rNISC/OLE and the Paillier-based OLE implementation yields NISC and designated-verifier NIZK protocols with attractive new efficiency features. As discussed above, for general NISC there was no previous approach that could offer reusable security, even for the case of boolean circuits, without applying general-purpose NIZK on top of a semi-honest secure NISC protocol.

Even for the special case of zero-knowledge, where many other competing approaches are known, our approach is quite unique. In particular, we are the first to construct any kind of (reusable-setup) NIZK protocol where one can push *all of the cryptographic operations* to an offline phase; using the self-reducibility of OLE, we can have an online phase that involves only arithmetic computations in the "plaintext domain" and its security (given the preprocessed information) is unconditional. Moreover, the online phase satisfies a strong notion of *constant computational overhead* in the sense that both the prover and verifier only need to perform a constant number of addition and multiplication operations for each gate of the arithmetic verification circuit, in the same ring $\mathbb{Z}_N$ over which the circuit is defined. As a bonus feature, the preprocessing required for implementing this highly efficient online phase consists only of a constant number of exponentiations per gate, and its security relies on a conservative, "20th century" assumption.

To summarize, the main positive results we prove about reusable NISC in the OLE-hybrid model are the following.

**Theorem 1.** *There exists a statistically secure rNIZK/OLE protocol (i.e., rNISC for zero-knowledge functionalities) with $O(1)$ OLE calls per gate.*

**Theorem 2.** *There exists a statistically secure rNISC/OLE protocol for (arithmetic or boolean) branching programs and $NC^1$ circuits.*

**Theorem 3.** *If one-way functions exist, there exists an rNISC/OLE protocol for circuits.*

We optimize the concrete efficiency of our rNIZK/OLE protocol in Section 5.6, so that proving knowledge of a satisfying assignment of an arithmetic circuit costs 7 OLE calls per addition gate and 44 OLE calls per multiplication gate. We stress that since the protocol is information-theoretic in the OLE-hybrid model, each OLE involves only a small number of field operations (without any exponentiations) in the online phase.

## 1.1 Related Work

We briefly discuss several recent works that are relevant to the asymptotic efficiency features of our protocol. As discussed above, a distinctive efficiency feature of our rNIZK/OLE protocol for arithmetic verification circuits (more generally, rNISC/OLE for constant-depth arithmetic circuits) is that, in an offline-online setting, its online phase is non-cryptographic and has a constant computational overhead. Moreover, the offline phase only requires a constant number of exponentiations per arithmetic gate.

Bootle et al. [BCG$^+$17] construct zero-knowledge protocols for arithmetic verification circuits with constant computational overhead in the plain model, i.e., without any offline phase. However, this protocol relies on constant-overhead implementations of cryptographic primitives (a plausible but non-standard assumption), it requires multiple rounds of interaction (but can be made non-interactive via the Fiat-Shamir heuristic) and, most importantly in the context of our work, the cryptographic work in this protocol cannot be preprocessed. Finally, this protocol does not directly apply in the more general setting of secure computation.

Applebaum et al. [ADI$^+$17] obtain (again, under plausible but non-standard assumptions) secure two-party protocols for evaluating arithmetic circuits that have constant computational overhead in the plain model. However, these protocols are inherently interactive (even when restricted to constant-depth circuits) and are only secure against semi-honest parties.

Finally, Chaidos and Couteau [CC18] construct an alternative Paillier-based designated-verifier (reusable) NIZK protocol with a constant number of exponentiations per arithmetic gate. The constant from [CC18] is significantly smaller than ours and the protocol can be based on more general assumptions. However, whereas for NIZK there are several other competing approaches, including succinct and publicly verifiable protocols, our NISC protocol provides the *first* reusable solution for NISC that is efficient enough to be implemented. Moreover, the NIZK protocol from [CC18] (which is based on $\Sigma$-protocols) does not have the feature of a non-cryptographic online phase that our protocol inherits from the underlying information-theoretic OLE-based protocol.

# 2 Overview of the Techniques

In this section we provide a high level overview of the proofs of our main results.

## 2.1 Impossibility of rNISC/OT

We show several negative results, which highlight the hardness of reusable secure computation. The first negative result shows that: for non-interactive two-party computation protocols, prefect security against malicious sender does not imply reusable security. In particular, previous works about NISC/OT do not immediately implies rNISC/OT.

We further show that OLE is strictly stronger than OT in the sense that there exists no information-theoretic rNISC/OT protocol for OLE functionality with composable security. Finally, assuming the existence of one-way functions, in the OT hybrid model, there are no general resettably sound, non-interactive zero-knowledge proofs with black-box simulation.

**Standard security does not imply reusability.** Reusable security is not free. A perfectly secure, 2-party computation protocol can be totally broken if the receiver uses the same randomness in two protocol runs. A counter example functionality is easy to find. Assume the sender's input domain has to special symbols $\top$, $\bot$, such that the receiver will output a random string if the sender's input is either $\top$ or $\bot$. Thus if the receiver use fresh randomness in each protocol run no information about his input will be leaked when the sender's input is either $\top$ or $\bot$.

In one possible secure implement, the receiver samples a random string $r$, and outputs $r$ if the sender's input is $\top$, outputs $x \oplus r$ if the sender's input is $\bot$. Here $x$ is the receiver's input. If the receiver uses the same input $x$ and randomness $r$ in at least two protocol runs, then a malicious sender can make him output $r$ and $x \oplus r$, which together reveal the receiver's secret input $x$.

**OT is not sufficient for reusability.** We show the impossible of reusable non-interactive two-party computation protocol under OT hybrid model for two different problems and settings, while such protocols exist if we switch to OLE hybrid model. The weakness of OT is that a malicious sender can learn the receiver's choice bits if the receiver uses the same randomness in different protocol runs. Consider the following scenario: The receiver uses the same randomness and input in two protocol runs. This means that the receiver's secret OT choice bits have been fixed in a set-up phase. In the first protocol run, a malicious sender feeds $(a[i], b[i])$ to the $i$-th OT instance. In the second protocol run, the malicious sender feeds $(a', b')$ to the OT instances such that $(a', b')$ and $(a, b)$ are identical except for one bit. Say $a[j] \neq a'[j]$ is the only index where they differs. If the receiver outputs differently in these two protocol runs, the malicious sender can deduce that in the receiver's choice bit in the $j$-th OT instance equals 1.

Moreover, if the receiver outputs differently in two protocol where the sender chooses OT-input strings $(a, b)$ and $(a', b')$ respectively. The malicious sender modifies $(a, b)$ to equal $(a', b')$ by a sequence of single-bit modifications. By observing the receiver's output when the sender feeds these intermediate OT-input strings, the malicious sender can always learn the sender's $j$-th choice bit for some $j$ such that $(a[j], b[j]) \neq (a'[j], b'[j])$.

Note that in OLE hybrid model, such attack will not work. Consider a similiar scenario: The receiver uses the same randomness and input in two protocol runs, let $x[i]$ be the receiver's input in the $i$-th OLE instance. The malicious sender feed $(a[i], b[i])$ and $(a'[i], b'[i])$ to the $i$-th OLE instance in these two protocol runs respectively. Say the $(a[j], b[j]) \neq (a'[j], b'[j])$ is the only difference between $(a, b)$ and $(a', b')$ and the receiver outpus differently in these two protocol runs. Given the above information, the malicious sender can only deduce that $x[j] \neq -\frac{b[j]-b'[j]}{a[j]-a'[j]}$. Such knowledge contains little information if $x[j]$ has large min entropy.

For example, in Section 4.2 we show the impossibility of an information-theoretic non-interactive reusable OLE/OT protocol. The intuition behind our impossibility proof relies on a commitment protocol (Figure 3). There is a statistically reusable commitment protocol in OLE hybrid model: The receiver first samples a random $x \in \mathbb{F}$ as his OLE-input. To commit $s_i \in \mathbb{F}$, the sender samples random $r_i \in \mathbb{F}$ and feed $(s_i, r_i)$ to rOLE, so that the receiver gets rOLE-output $y_i = s_i \cdot x + r_i$. To unveil the $i$-th commitment, send $(s_i, r_i)$ to the receiver. Such a protocol is statistically reusable secure in OLE hybrid model. We show that in an OT-based implementation of the rOLE-primitive, a corrupted sender can recover the receiver's secret input $x$ after polynomially many rounds. The corrupted sender repeats the following so that either he recovers $x$ or he learns more about receiver's OT choice bits. The sender samples an honest run in which the sender chooses $(s, r, a, b)$, then samples $(s', r', a', b')$ from the same distribution subject to the condition that $(a', b')$ agrees with $(a, b)$ on the known receiver's choice bits. The sender can test whether $(s', r')$ and $(s, r)$ are consistent with the same $x$, i.e. whether $s'x + r' = sx + r$, by testing whether the receiver accepts $(s', r')$ as an unveil message when the sender's OT-input stirngs are $a, b$. If so, the sender recovers $x = -\frac{r-r'}{s-s'}$ and thus finish the attack ($s \neq s'$ with high probability because $s$ is statistically hidden in the receiver's view). Otherwise, the receiver would reject $(s', r')$ as an unveil message when the sender's OT-input strings are $a, b$, while accept it when the sender's OT-input strings are $a', b'$. The sender will be able to learn at least one more receiver's choice bit from such difference.

In Section 4.3 we show that there is no UC secure rNISC/OT protocol for genearl zero-knowledge proof functionality (Figure 4). Suppose such protocol exists. This means the sender can prove statements $x \in L$ just by transforming a corresponding witness $w$ into sender's OT-input strings. By assuming the existence of one-way functions, we can define the language such that it is easy to sample a random no instance $y \notin L$ or to sample a random yes instance $x \in L$ together with a witness $w$, while it's computational hard to distinguish a random yes instance with no instance. Now how can a malicious sender (prover) find some $y \notin L$ but still convince the receiver to accept $y$? He just samples a true statement $(x, w)$ and starts off flipping bits in the corresponding OT-input strings, then checks each time if the receiver still accepts. Of course, the sender only flips the part of OT-input strings where he does not know the receiver's choice bits yet. As soon as the receiver starts rejecting, the malicious prover find out one more receiver's choice bit. This process can be repeated until the malicious prover has learned sufficiently many of the receiver's choice bits. There are so few indexes where the malicious prover doesn't know the choice bits — denote these indexes by $U$ — such that even if the OT-input strings are replaced with random bits on indexes in $U$, the receiver will still accept with high probability. Then by the UC security, if the sender instead samples $y \notin L$, generates OT-input strings using the black-box simulator, and replaces the generated OT-input strings with random bits on indexes in $U$, the receiver will also accept with high probability.

## 2.2 Construction of Information-Theoretic rNISC/OLE

**Semi-honest NISC/OLE** Our rNISC/OLE construction is a complicated object with many intermediate steps. Let us start with a warm-up question, how to construct NISC with semi-honest security? As a starting point we present a construction for the semi-honest model from [IK02].

Then we will outline how to obtain security against malicious parties.

Let $\mathbf{x}$ denote the receiver's input and let $\mathbf{y}$ denote the sender's input. We consider arithmetic functionalities, i.e. both $\mathbf{x}, \mathbf{y}$ are vectors over a given finite ring, the functionality is also arithmetic over this ring. We describe the construction for an arithmetic functionality defined by an *arithmetic branching programs*, which is described as follows (see [IK02] for a more formal description). A $t$-node arithmetic branching program is specified by affine functions $g_{1,1}, g_{1,2}, \ldots, g_{t,t}$; the branching program maps input vectors $\mathbf{x}, \mathbf{y}$ to the determinant of

$$\begin{bmatrix} \mathsf{g}_{1,1}(\mathbf{x},\mathbf{y}) & \cdots & & \mathsf{g}_{1,t}(\mathbf{x},\mathbf{y}) \\ -1 & \ddots & & \\ & \ddots & \ddots & \vdots \\ & & -1 & \mathsf{g}_{t,t}(\mathbf{x},\mathbf{y}) \end{bmatrix}.$$

Let $\mathsf{G}(\mathbf{x}, \mathbf{y})$ denote the matrix. Branching programs can efficiently simulate arithmetic formulas and arithmetic $\mathrm{NC}^1$ circuits. E.g. the formula $x_1y_1 + x_2y_2 + x_3y_3$ can be computed by branching program

$$x_1y_1 + x_2y_2 + x_3y_3 = \det \begin{bmatrix} x_1 & x_2 & x_3 & \\ -1 & & & y_1 \\ & -1 & & y_2 \\ & & -1 & y_3 \end{bmatrix}.$$

The technique for securely reducing a branching program computation to parallel OLE can be viewed as an arithmetic analogue of Yao's garbled circuit technique [AIK14]. Sample two random upper triangular matrixes $R_1, R_2$ with diagonal all one, then the matrix $R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$ is a *randomized encoding* of $\det \mathsf{G}(\mathbf{x}, \mathbf{y})$ as

- $\det \mathsf{G}(\mathbf{x}, \mathbf{y})$ can be computed from $R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$ as multiplying $R_1, R_2$ doesn't change determinant;
- the distribution of $R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$ merely depends on $\det \mathsf{G}(\mathbf{x}, \mathbf{y})$.

Therefore, if the receiver gets $R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$, he will learn no information other than $\det \mathsf{G}(\mathbf{x}, \mathbf{y})$.

In the semi-honest NISC/OLE construction, the OLE allows secure evaluation of affine functions, so the receiver chooses $\mathbf{x}$ as its input, then the sender feed the affine function $\mathbf{x} \mapsto R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$ to the OLE oracle. Let us denote this affine function by $\mathsf{G}'$, i.e. $\mathsf{G}'(\mathbf{x}) := R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$. Eventually, the receiver gets $R_1\mathsf{G}(\mathbf{x}, \mathbf{y})R_2$, which leaks $\det \mathsf{G}(\mathbf{x}, \mathbf{y})$ but perfectly hides all other information.

This NISC/OLE based on randomized encoding is actually perfectly secure against malicious receivers, as the protocol doesn't leave the receiver with any room for cheating. Thus the receiver is effectively forced to be semi-honest.

But the protocol is not secure against malicious senders. Instead, the sender can choose any affine $\mathsf{G}'$ so that the receiver will output $\det \mathsf{G}'(\mathbf{x})$. For the security against malicious senders, the sender need to prove that $\mathsf{G}'$, the affine function he feed into OLE, satisfies an arithmetic constraint: the sender knows two upper triangular matrixes $R_1, R_2$ and input vector $\mathbf{y}$ such that $\mathsf{G}'(\cdot) \equiv R_1\mathsf{G}(\cdot, \mathbf{y})R_2$.

**Intermediate Primitive: Certified OLE** Based on the above discussion about semi-honest NISC/OLE, a perfectly secure NISC can be built upon a specialized OLE. In such a specialized OLE, the sender can prove that the coefficients she chose satisfies some arithmetic conditions. We name this specialized OLE as *certified OLE* and it is exactly what we are going to construct as an intermediate primitive. More precisely, we define certified OLE as a primitive that allows

- the receiver to learn the outputs of affine functions, where the inputs in chosen by the receiver and the coefficients is chosen by the sender;
- the sender to convince the receiver that the sender-chosen coefficients satisfy arbitrary arithmetic constraints.

We implement a CertifiedOLE/OLE construction, whose security is both information-theoretic and reusable.

**Intermediate Primitive: Replicated OLE** Certified OLE allows the sender to prove that her coefficients satisfy arbitrary arithmetic constraints. In particular, the sender can prove an equality constraint, i.e., to prove two of the coefficients she choose are equal. We isolated this ability as another primitive called *replicated OLE*. More precisely, we define replicated OLE as a primitive that allows

- the receiver to learn the outputs of affine functions, where the inputs in chosen by the receiver and the coefficients is chosen by the sender;
- the sender to convince the receiver that some of the sender-chosen coefficients are equal.

Be definition, replicated OLE is not as powerful as certified OLE. In section 5, we first construct replicated OLE directly from OLE[1], then construct certified OLE from replicated OLE. For now, let us assume that we already have reusable replicated OLE, and we will construct certified OLE by using replicated OLE as a black box.

It's sufficient for certified OLE to only support the following atomic operations.

1. Reveal $ax + b$ to the receiver, where $a, b \in \mathbb{F}$ are coefficients chosen by the sender, $x \in \mathbb{F}$ is an input chosen by the receiver, $\mathbb{F}$ is a finite field.

   In this overview, all coefficients chosen by the sender will be denoted by the first a few letters in the alphabet such as $a, b, c$ and inputs chosen by the receiver will be denoted by the last a few letters in the alphabet such as $x, y, z$.

2. Allow the sender to convince the receiver that two coefficients are equal.
3. Allow the sender to convince the receiver that three coefficients $a, b, c$ satisfies $a + b = c$.
4. Allow the sender to convince the receiver that three coefficients $a, b, c$ satisfies $ab = c$.

The first two atomic operations are already offered by replicated OLE. The later two atomic operations are translated into replicated OLE calls.

The third atomic operation, i.e. proving $a+b=c$, is implemented as the following. The receiver samples an random $x \in \mathbb{F}$ and chooses it as an extra input. The sender samples random $a', b' \in \mathbb{F}$

[1] The actual roadmap is different, and will be gradually revealed in the overview.

and let $c' = a' + b'$. The replicated OLE is used to reveal $ax + a', bx + b', cx + c'$ to the receiver. Finally, the receiver is convinced if and only if $(ax + a') + (bx + b') = (cx + c')$. In case $a + b \neq c$ (or $a' + b' \neq c'$), the receiver can detect with overwhelming probability.

The last atomic operation, i.e. proving $ab = c$, is implemented using similar idea. The receiver samples random $x, y \in \mathbb{F}$, let $z = xy$ and choose $x, y, z$ as extra inputs. Notate that a malicious receiver might choose $z \neq xy$ and the sender can never detect. Therefore, we design a mechanism that can "enforce" honest receiver behavior. More precisely, our mechanism will prevent the receiver from learning any information in case he choose $z \neq xy$. The detail of such mechanism will be explained later. For now, let us simply assume the receiver chooses $z = xy$.

The sender samples random $a', b', c' \in \mathbb{F}$, chooses $d = ab', e = a'b$ and samples $d', e' \in \mathbb{F}$ such that $d' + e' = a'b' - c'$. The replicated OLE is used to reveal $ax + a', by + b', cz + c', dx + d', ey + e'$ to the receiver. The receiver is convince if and only if $(ax + a')(by + b') - (cz + c') = (dx + d') + (ey + e')$. Notice that if both sender and receiver are honest, then

$$(ax + a')(by + b') - (cz + c') = a'by + b'ax + a'b' - c' = (dx + d') + (ey + e').$$

In case the sender behaves maliciously, either $c \neq ab$ or the values of $d, e, d', e'$ deviate the protocol, the receiver can detect with overwhelming probability.

**Intermediate Primitive: Half-Replicated OLE** Our replicated OLE is constructed on top of what we called *half-replicated OLE*. In each OLE calls, the sender chooses two coefficients. We distinguish them by calling them *multiplicative coefficient* and *additive coefficient* respectively. Half-replicated OLE only supports two operations:

1. Reveal $ax + b$ to the receiver, where $x \in \mathbb{F}$ is an input chosen by the receiver, $a \in \mathbb{F}$ is a multiplicative coefficient chosen by the sender, $b \in \mathbb{F}$ is an additive coefficient chosen by the sender.

2. Allow the sender to convince the receiver that two *multiplicative* coefficients are equal.

By definition, half-replicated OLE is even weaker than replicated OLE. We construct replicated OLE on top of half-replicated OLE as the following: The receiver samples random $y$ and chooses it as an extra input. For each receiver-chosen input $x$, the receiver let $x' = xy$ and chooses it as an extra input. Notice that the sender cannot detect whether $x'$ is generated honestly, we design a mechanism that can enforce $x' = xy$ which will be explained later. For now, just assume the receiver chooses $x' = xy$ honestly. Then the sender uses the replicated OLE to reveal $ax + b$ and $ax' + by$ to the receiver. (More precisely, the sender should also samples random $c$ and reveals $ax + b, ax' + c, by - c$ to the receiver.) The receiver then uses the equation $(ax + b) \cdot y = (ax' + by)$ to check whether the sender behaves honestly.

**Intermediate Primitive: Half-Replicated OLE Allowing CDS Operations** Our replicated OLE and certified OLE require the receiver to choose three input $x, y, z$ such that $z = xy$, while there is no mean for the sender to detect whether the receiver behaves honestly. In particular, if the underly OLE is ideal (e.g. implement by a trusted third party), then the sender will learn absolutely nothing about the receiver-chosen inputs. Thus instead, we design a mechanism called *conditional disclosure of secrets (CDS)*, in which the sender can disclosure a message to the receiver if and only if the receiver-chosen inputs satisfy some arithmetic constraints. For example, in certified OLE, the sender can encrypt his messages using one-time pad, and disclose the pad if and only if the receiver chooses $z = xy$ honestly.

As a first trial, in order to disclose secret $a \in \mathbb{F}$ to the receiver if and only if $z = xy$, the sender samples random $b, c \in \mathbb{F}$ and uses the half-replicated OLE to disclose

$$\begin{bmatrix} y & z \\ 1 & x \end{bmatrix} \begin{bmatrix} b \\ c \end{bmatrix} + \begin{bmatrix} a \\ 0 \end{bmatrix}$$

to the receiver. (More precisely, this means the sender should also sample random $b', c'$ that $b' + c' = a$, and use the half-replicated OLE to disclose $by + b', cz + c', cx + b$.) If $z = xy$ is satisfied, then the receiver can recover $a$ as

$$(1, -y) \cdot \left( \begin{bmatrix} y & z \\ 1 & x \end{bmatrix} \begin{bmatrix} b \\ c \end{bmatrix} + \begin{bmatrix} a \\ 0 \end{bmatrix} \right) = a.$$

It's easy to verify the security against malicious receiver. When $z \neq xy$, matrix $\left[\begin{smallmatrix} y & z \\ 1 & x \end{smallmatrix}\right]$ is invertible, in which case all information about $a$ is erased by one-time padding. But this protocol is not secure against malicious sender: As the protocol is built on top of half-replicated OLE, the sender can deviate the protocol by changing the additive coefficients. In particular, the sender can choose a non-zero $d \in \mathbb{F}$ and uses the half-replicated OLE to disclose $\left[\begin{smallmatrix} y & z \\ 1 & x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ c \end{smallmatrix}\right] + \left[\begin{smallmatrix} a \\ d \end{smallmatrix}\right]$ to the receiver. Then the receiver will recover $(1, -y) \cdot (\left[\begin{smallmatrix} y & z \\ 1 & x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ c \end{smallmatrix}\right] + \left[\begin{smallmatrix} a \\ d \end{smallmatrix}\right]) = a - dy$, which is a function of the receiver's inputs. An easy way to solve this problem is to rely on the fact that the receiver samples $y \in \mathbb{F}$ uniformly at random[2]. The sender samples a random $a'$ as an extra coefficient and the above insecure CDS protocol to disclose $a'$ if $z = xy$. If the sender is malicious, then the receiver gets $a' - d'y$. But the receiver can detect any malicious behaviour, by sampling a random $w \in \mathbb{F}$ as an extra input and asking the sender to disclose $aw + a'$ using OLE.

**Back to Half-Replicated OLE** The last missing piece is how to construct half-replicated OLE in OLE hybrid model. The key idea of the construction is the followings.

The receiver samples a random $w \in \mathbb{F}$ and chooses $w$ as an input. For each multiplicative coefficient $a \in \mathbb{F}$, the sender has to sample a random $a' \in \mathbb{F}$ and use OLE to disclose $aw + a'$. This OLE call works essentially as a commitment of $a$, this idea can be used to build a statistically secure commitment protocol (Figure 3). For each half-replicated OLE input $x \in \mathbb{F}$, the receiver translates it into two OLE inputs $y, z \in \mathbb{F}$ such that $y$ is sampled uniformly at random, and $z = x - wy$.

For each half-replicated OLE call $ax + b$, it can be translated into three OLE calls using the idea from equation

$$\begin{aligned} ax + b &= a(wy + x - wy) + b \\ &= awy + az + b \\ &= y(aw + c) - (cy + d) + (az + b + d), \end{aligned} \tag{1}$$

where $c, d$ are arbitrary numbers. More precisely, the sender should sample random $c, d \in \mathbb{F}$ and use the OLE to disclose $aw + c$, $cy + d$ and $az + b + d$ to the receiver. Finally, the receiver computes the right output using equation 1.

Such a half-replicated OLE protocol is correct and enforces the receiver to be honest. When the sender deviates the protocol, the receiver will output a random number. The randomness comes

[2] In the main body, $y$ doesn't need to be random. Moreover, we consider the general case where the arithmetic condition doesn't have to be $z = xy$.

| Primitive | Operations Supported |
|---|---|
| OLE | 1 |
| Half-Replicated OLE | 1,2 |
| Half-Replicated OLE allowing CDS | 1,2,3 |
| Certified OLE | 1,2,3,5,6 |
| Replicated OLE | 1,2,3,4,5,6 |

Table 1: Primitives and Their Supporting Operations

from $w$ and $y$, i.e., in case the sender deviates the protocol, even conditioned on the sender's view and $x$, the receiver's output is statistically close to uniform distribution. Therefore it is not too hard embed a mechanism to detect any malicious sender with overwhelming probability. We leave the detail to Section 5.2.

**Roadmap** Starting from reusable OLE, we define and construct a sequence of increasingly more powerful primitives, the last of which eventually supports all of the following operations.

1. Reveal $ax + b$ to the receiver, where $x \in \mathbb{F}$ is an input chosen by the receiver, $a \in \mathbb{F}$ is a multiplicative coefficient chosen by the sender, $b \in \mathbb{F}$ is an additive coefficient chosen by the sender.

2. Convince the receiver that two *multiplicative* coefficients are equal.

3. Disclose a message to the receiver if receiver-chosen inputs $x, y, z$ satisfies $z = xy$.

4. Convince the receiver that two coefficients are equal.

5. Convince the receiver that three multiplicative coefficients $a, b, c$ satisfies $a + b = c$.

6. Convince the receiver that three multiplicative coefficients $a, b, c$ satisfies $ab = c$.

Such a primitive readily implies reusable NIZK and reusable NISC. The intermediate primitives are sorted in Table 1 by dependence. Each of them only supports a subset of the operations.

### 2.3 Paillier-based 2-Message OLE Protocol

Consider a simplified OLE scheme as follows: The CRS will contain an ElGamal public key $(b, B_0 = b^{sk_0})$ in a Paillier group. (Paillier allows us to get additive homomorphism, while ElGamal means that the receiver will be able to construct related key pairs.) On input $\alpha$, the receiver forms another related public key $b, B_1$, such that it knows the secret key corresponding to $(b, B_1 B_0^{\alpha})$. It sends this key pair to the sender. On input $z_0, z_1$, the sender encrypts $z_0$ under $(b, B_0)$ and $z_1$ under $(b, B_1)$, using the same randomness, and sends both ciphertexts to the receiver. The receiver can then combine the ciphertexts to obtain an encryption of $\alpha z_0 + z_1$ under $(b, B_1 B_0^{\alpha})$, which it can decrypt.

Recall that in a Paillier group for $N = (2p + 1)(2q + 1)$ all elements can be decomposed into a component in a subgroup of order $2p'q'$, and a component of order $N$, call them $G_{2p'q'}$ and $G_N$; the ElGamal encryption will encode the message in the order $N$ component. Intuitively, we can argue the scheme is secure against a corrupt receiver as follows: First the CRS is indistinguishable from one where $b$ is only in $G_{2p'q'}$, but $B_0$ has a component in $G_N$. Then suppose that the receiver chooses $B_1$ whose $G_N$ component is $(1+N)^\alpha$ (and note that a simulator can recover this $\alpha$ using the factorization of $N$). The $G_N$ components of the resulting ciphertexts can be shown information theoretically to depend only on $z_0\alpha + z_1$, while the $G_{2p'q'}$ components are independent of $z_0, z_1$.[3]

Security against a corrupt sender is more challenging, because it could send invalid ciphertexts (i.e., ciphertexts in which decryption produces an element not in $G_N$). In particular, an adversarial sender could form a pair of ciphertexts that decrypt correctly under a specific $\alpha$ and incorrectly otherwise, and thus perform a selective failure attack. To prevent this, we need a way for the receiver to identify bad ciphertext pairs that can't be predicted based on $\alpha$. Suppose the receiver runs the scheme twice, once with a random input $\gamma$, and once with input $2\alpha - \gamma$, while the sender uses inputs $z_0, w$ for random $w$ in the first instance and $z_0, z_1 - w$ in the second; combining the results of the two schemes would allow the receiver to decrypt $z_0\gamma + w + z_0(\alpha - \gamma) + z_1 - w = z_0\alpha + z_1$. This would prevent the selective failure attack: we argue that (under appropriate, indistinguishable CRS) $B_1$ information theoretically hides $\gamma$, so the probability that the resulting linear combination of two invalid ciphertexts decrypts correctly is negligible.[4] Of course, we must ensure that the malicious sender uses the same $z_0$ in both instances; thus we require that all the ciphertexts are related, using the same randomness.

# 3 Preliminaries

We consider sender-receiver functions that take inputs from a sender Sam and a receiver Rachel and deliver the output to Rachel. Two simple but useful examples for such functions are OT and OLE. In this work, we consider the *reusable* extension of such sender-receiver functions, allowing Sam to invoke the function on polynomially many inputs, where Rachel's input is fixed. In each such invocation, Rachel obtains a separate output. We will sometimes use an r-prefix (as in rOT, rOLE, or rNISC) to stress that we consider the reusable variant.

## 3.1 Sender-receiver Functions & Reusable Two-party Computation

In this section we give a generic definition of reusable non-interactive secure computation (rNISC). Our complete rNISC construction for arbitrary functions is quite complex. To make it as modular as possible, we define intermediate functionalities, namely rNISC for arithmetic circuits (see Section A) and linear functions (see Section 3.2).

**Notation 4** (Sender-receiver functions)**.** *A* sender-receiver function *is specified by three sets $R_{\text{in}}, S_{\text{in}}$, $R_{\text{out}}$ and a mapping $f : R_{\text{in}} \times S_{\text{in}} \to R_{\text{out}}$. The intuition is that we have two parties: a receiver Rachel and a sender Sam. Rachel chooses an input $x \in R_{\text{in}}$, Sam chooses an input $y \in S_{\text{in}}$, and Rachel learns the corresponding output $z := f(x, y) \in R_{\text{out}}$.*

We emphasize that it is not enforced that the receiver's input $x$ is fixed before the sender chooses an input $y$ for a corresponding send phase. Neither do we forbid that the receiver provides an input $(sid', x)$ after having learned an output $(sid, z, i)$, as long as $sid \neq sid'$. Our main application just

[3] This is because the first component of the ciphertext, $b^r$ contains no information about $r \mod N$.

[4] There is a minor subtlety here, where because $G_{2p'q'}$ has an order 2 subgroup an extra component in this subgroup might not be detected; to prevent this, we actually square all the elements during decryption to eliminate this subgroup, and then decrypt the final result divided by 2.

**Functionality $\mathcal{F}^{(F)}_{\texttt{rNISC}}$**

Parametrized by a sender-receiver function $F = (R_{\text{in}}, S_{\text{in}}, R_{\text{out}}, f)$ in the sense of Notation 4.

**Choice phase:**

- Upon receiving input $(\mathit{sid}, x)$ from Rachel where $x \in R_{\text{in}}$ and $\mathit{sid}$ is a session identifier, store $(\mathit{sid}, x)$, send $(\mathit{sid}, \texttt{initialized})$ to the adversary, and ignore any further inputs $(\mathit{sid}, \tilde{x})$ from Rachel with the same session identifier $\mathit{sid}$.

**Send phases:**

- Upon receiving input $(\mathit{sid}, y, i)$ from Sam where $(y, i) \in S_{\text{in}} \times \mathbb{N}$ and $\mathit{sid}$ is a session identifier, record $(\mathit{sid}, y, i)$, send $(\mathit{sid}, \texttt{sent}, i)$ to the adversary, and ignore any further inputs $(\mathit{sid}, \tilde{y}, i)$ from Sam with the same session identifier $\mathit{sid}$ and the same value of $i$.
- Upon receiving a message $(\mathit{sid}, \texttt{Delivery}, i)$ from the adversary, verify that there are stored inputs $(\mathit{sid}, x)$ from Rachel and $(\mathit{sid}, y, i)$ from Sam; else ignore that message. Next, compute $z := f(x, y)$, send $(\mathit{sid}, z, i)$ to Rachel, and ignore further messages $(\mathit{sid}, \texttt{Delivery}, i)$ from the adversary with the same session identifier $\mathit{sid}$ and the same value of $i$.

Figure 1: Generic ideal functionality for reusable non-interactive secure computation.

provides a setting where all receiver inputs are chosen before the sender takes any action, but this is not required for the security proofs of our protocols.

The ideal functionality for reusable NISC tailored to arithmetic circuit evaluation is in Appendix A.

### 3.2 Reusable Oblivious Linear Function Evaluation

We aim at an OLE-based implementation of $\mathcal{F}^{(\Phi)}_{\texttt{rNISC}}$ for arbitrary arithmetic circuits $\Phi$ over a given ring $\mathcal{R}$, where the ring size $|\mathcal{R}|$ is determined by a statistical security parameter. More particularly, the security parameter is $\log |\mathcal{R}|$. However, we will need to restrict ourself to circuits $\Phi$ that are given as collections of formulas (i.e., the underlying graph $G$ is a forest).

The primitive we take for granted lets Rachel pick an input $x \in \mathcal{R}$ and then Sam send her tuples $(a, b) \in \mathcal{R} \times \mathcal{R}$, such that she learns the corresponding OLE-outputs $a \cdot x + b$. In particular, Sam can send several tuples $(a, b)$ for the same receiver input $x$. In other words, the ideal functionality for oblivious linear function evaluation with reusable receiver input (see Figure 18) is another special instance of the functionality $\mathcal{F}^{(F)}_{\texttt{rNISC}}$ from Figure 1, namely with $S_{\text{in}} = \mathcal{R} \times \mathcal{R}$, $R_{\text{in}} = R_{\text{out}} = \mathcal{R}$, and $f : R_{\text{in}} \times S_{\text{in}} \to R_{\text{out}}$, $(x, (a, b)) \mapsto a \cdot x + b$.

## 4 Separations and Impossibility Results

In this section we show several negative results, which highlight the hardness of reusable secure computation. Our first negative result is motivated by the following seemingly self-evident but misleading chain of thoughts: If we have a protocol for non-interactive two-party computation with perfect security against a malicious sender, then the security does not depend on the receiver's random choices and therefore the protocol can be used for reusable secure computation. Actually,

**Functionality $\mathcal{F}_{\mathtt{rOLE}}^{(\mathcal{R})}$**

Parametrized by a finite ring $\mathcal{R}$.

**Setup/choice phase:**

- Upon receiving input $(\mathit{sid}, x)$ from Rachel where $x \in \mathcal{R}$ and $\mathit{sid}$ is a session identifier, store $(\mathit{sid}, x)$, send $(\mathit{sid}, \mathtt{initialized})$ to the adversary and ignore any further inputs from Rachel with the same session identifier $\mathit{sid}$.

**Send phases:**

- Upon receiving input $(\mathit{sid}, a, b, i)$ from Sam where $(a, b, i) \in \mathcal{R} \times \mathcal{R} \times \mathbb{N}$ and $\mathit{sid}$ is a session identifier, store $(\mathit{sid}, a, b, i)$, send $(\mathit{sid}, \mathtt{sent}, i)$ to the adversary, and ignore any further inputs from Sam with the same session identifier $\mathit{sid}$ and the same value of $i$.
- Upon receiving a message $(\mathit{sid}, \mathtt{Delivery}, i)$ from the adversary where $i \in \mathbb{N}$ and $\mathit{sid}$ is a session identifier, verify that there are stored inputs $(\mathit{sid}, x)$ from Rachel and $(\mathit{sid}, a, b, i)$ from Sam; else ignore that message. Next, compute $z := a \cdot x + b$, send $(\mathit{sid}, z, i)$ to Rachel, and ignore further messages $(\mathit{sid}, \mathtt{Delivery}, i)$ from the adversary with the same session identifier $\mathit{sid}$ and the same value of $i$.

Figure 2: Ideal functionality for reusable oblivious linear function evaluation over a ring $\mathcal{R}$.

this is not true, because a corrupted sender could still cause some relations between the receiver's outputs that reveal secret information (see Section 4.1).

We further show that OLE is strictly stronger than OT in the sense that there exists no rNISC/OT protocol for the OLE functionality with composable security (see Section 4.2). Finally, assuming that one-way functions exist, we show that in the OT hybrid model there are no general resettably sound zero-knowledge proofs with black-box simulatability of malicious verifiers (see Section 4.3). In contrast, our certified OLE construction (see Section 5.4) immediately implies such zero-knowledge proofs in the (reusable) OLE hybrid model.

## 4.1 Standard Security (even if Perfect) does not Imply Reusability

We show now that reusing the receiver's messages in a general solution for secure computation can cause severe security problems. More particularly, we show that any protocol for general secure computation can be turned into a protocol with the following property: If the receiver uses fresh randomness in each protocol run, than the transformed scheme is still secure, but if the receiver uses the same input and randomness in two protocol runs with a malicious sender, then his complete input can be efficiently reconstructed from his corresponding outputs. The transformed scheme achieves the same level of security (computational, statistical, or perfect) as the original scheme, as long as the receiver uses fresh randomness in every protocol run.

Let any protocol scheme of the following form be given. The scheme is parametrized by two input domains $X, Y$, an output domain $Z$, and a mapping $f : X \times Y \to Z$. On input $x \in X$ from the receiver and $y \in Y$ from the sender, the receiver will learn and output $z := f(x, y)$. W.l.o.g., all inputs and outputs are encoded as bit-strings, i.e. $X = Y = Z = \{0,1\}^*$. Now we turn this scheme into a scheme with the abovementioned insecurity. The basic idea is to enlarge the sender's input domain by two special symbols $\top$ and $\bot$, and let the receiver output a share of his input if the sender

**Protocol $\Pi_{\text{COM}}^{(\mathbb{F})}$**

Parametrized by a finite field $\mathbb{F}$, which is also the base field of the underlying rOLE-primitive. The security parameter is $\ell := \log|\mathbb{F}|$.

**Setup phase:**

- Receiver: Pick $x \in \mathbb{F}$ uniformly at random and input it into the underlying rOLE-primitive.

**Commit phases:**

1. Sender: Let $s_i \in \mathbb{F}$ be the sender's $i$-th input. Pick $r_i \in \mathbb{F}$ uniformly at random and input $(s_i, r_i)$ into the underlying rOLE-primitive.
2. Receiver: Record the corresponding rOLE-output $y_i = s_i \cdot x + r_i$.

**Unveil phases:**

1. Sender: To unveil the $i$-th commitment, send $(s_i, r_i)$ to the receiver.
2. Receiver: If $y_i = s_i \cdot x + r_i$, output $s_i$; otherwise reject (i.e., output a special symbol $\bot$).

Figure 3: Protocol for statistically secure commitments in the reusable OLE (rOLE) hybrid model.

uses one of these special input symbols. The resulting protocol takes input $(x, r) \in \{0,1\}^* \times \{0,1\}^*$ with $|x| = |r|$ from the receiver and $y \in \{0,1\}^* \cup \{\top, \bot\}$ from the sender, and the receiver's output $z$ is defined as:

$$z = \begin{cases} f(x,y) & \text{if } y \in \{0,1\}^* \\ r & \text{if } y = \top \\ x \oplus r & \text{if } y = \bot \end{cases}$$

For secure evaluation of $f(x, y)$, the receiver just needs to input $(x, r)$ with some uniformly random $r$ of appropriate length, and the sender is supposed to input $y$. Obviously, the modified protocol is as secure as the original scheme as long as the receiver uses fresh randomness for each protocol run. If, however, the receiver uses the same $(x, r)$ in at least two protocol runs, then a malicious sender can make him output $r$ and $x \oplus r$, which together reveals the receiver's secret input $x$.

## 4.2 Impossibility of Composable rNISC/OT for the OLE Functionality

We show that reusable OLE cannot be implemented from reusable OT, if the sender can learn certain predicates about the receiver's OLE-output. The intuition behind our impossibility proof relies on the commitment protocol of Figure 3. We want this construction to be secure even if a corrupted sender learns whether an unveil was accepted by the receiver or not. This can be formulated as the following game. The receiver first fixes his rOLE-input $x \in \mathbb{F}$ uniformly at random. The sender then picks tuples $(s_i, r_i), (\tilde{s}_i, \tilde{r}_i) \in \mathbb{F} \times \mathbb{F}$ and learns whether $s_i \cdot x + r_i = \tilde{s}_i \cdot x + \tilde{r}_i$. The intuition is here that $(s_i, r_i)$ stands for the sender's rOLE-input in a commit phase and $(\tilde{s}_i, \tilde{r}_i)$ stands for the corresponding unveil message. We show now that in an OT-based implementation of the rOLE-primitive a corrupted sender can after polynomially many rounds recover the receiver's secret input $x$ (and thereby break the binding property of $\Pi_{\text{COM}}^{(\mathbb{F})}$), which is impossible with an ideal rOLE. In particular, we assume an rOLE-protocol that is correct, provides sender privacy, and is based on polynomially many (reusable) OT-instances. For a concise presentation we denote the $i$-th element of a string $s$ by $s[i]$ and a corresponding substring indexed by $I$ as $s[I]$. A malicious sender can recover $x$ by the following attack.

1. Initialize $Q_0, R_0 := \emptyset$ and $j := 1$. The intuition is that $Q_j$ will contain indices where the receiver's choice bits are 0 and $R_j$ will contain indices where the receiver's choice bits are 1.
2. Sample an rOLE-input $(s_j, r_j) \in \mathbb{F} \times \mathbb{F}$ uniformly at random. According to the presumed rOLE-protocol, generate corresponding OT-input strings $a_j, b_j$ in the sense that $\big(a_j[i], b_j[i]\big)$ is the input for the $i$-th OT-instance. Sample $(\tilde{s}_j, \tilde{r}_j, \tilde{a}_j, \tilde{b}_j)$ with the same distribution but subject to the condition that $\tilde{a}_j[Q_{j-1}] = a_j[Q_{j-1}]$ and $\tilde{b}_j[R_{j-1}] = b_j[R_{j-1}]$.
3. Test whether $s_j \cdot x + r_j = \tilde{s}_j \cdot x + \tilde{r}_j$, i.e., the receiver accepts $(s_j, r_j)$ as well as $(\tilde{s}_j, \tilde{r}_j)$ as an unveil message for a commitment where the sender's OT-input strings are $a_j, b_j$. If so and moreover $s_j \neq \tilde{s}_j$, recover $x = -\frac{r_j - \tilde{r}_j}{s_j - \tilde{s}_j}$ and thus finish the attack.
4. Repeat the following two sub-steps until there are no more changes of $a_j$ and $b_j$.
   (a) If there exists an index $i \notin Q_{j-1}$ such that $a_j[i] \neq \tilde{a}_j[i]$ and the receiver still accepts $(s_j, r_j)$ as an unveil message for a commitment where the sender's OT-input strings are $a_j, b_j$ with $a_j[i]$ replaced by $\tilde{a}_j[i]$, change the value of $a_j[i]$ to the value of $\tilde{a}_j[i]$.
   (b) If there exists an index $i \notin R_{j-1}$ such that $b_j[i] \neq \tilde{b}_j[i]$ and the receiver still accepts $(s_j, r_j)$ as an unveil message for a commitment where the sender's OT-input strings are $a_j, b_j$ with $b_j[i]$ replaced by $\tilde{b}_j[i]$, change the value of $b_j[i]$ to the value of $\tilde{b}_j[i]$.
5. Set $Q_j := Q_{j-1} \cup \big\{ i \bigm| a_j[i] \neq \tilde{a}_j[i] \big\}$ and $R_j := R_{j-1} \cup \big\{ i \bigm| b_j[i] \neq \tilde{b}_j[i] \big\}$, increase $j$ by 1, and go to step 2.

We just have to show now that the attack finishes (on average) after polynomially many iterations over $j$. For every possible tuple $(s_j, r_j, a_j, b_j)$ let $\mathsf{V}_{a_j,b_j}(s_j, r_j) = \top$ denote that the verifier would accept $(s_j, r_j)$ as an unveil message for a commitment where the sender's OT-input strings are $a_j, b_j$ and let $\mathsf{V}_{a_j,b_j}(s_j, r_j) = \bot$ denote that the verifier would reject. Our estimation of the expected number of iterations over $j$ is based on the following two observations.

- Due to the completeness of the commitment scheme (i.e., correctness of the underlying rOLE-protocol) it happens in each iteration over $j$ only with small probability that $\mathsf{V}_{a_j,b_j}(s_j, r_j) = \bot$. Thus we can condition the whole attack to the event that always $\mathsf{V}_{a_j,b_j}(s_j, r_j) = \top$, as this changes the expected number of iterations over $j$ only by a small factor.

- In each iteration over $j$ we have only a small probability $\rho_j := \Pr[s_j = \tilde{s}_j]$. The reason for this is that $-\log_2(\rho_j)$ is an upper bound for the collision entropy of $s_j$ conditioned on the receiver's view in the corresponding commit phase of $\Pi^{(\mathbb{F})}_{\mathrm{COM}}$. If $\rho_j$ was not small, the commitment would not be hiding (or in other words, the underlying rOLE-protocol had no sender privacy). Thus we can condition the whole attack to the event that always $s_j \neq \tilde{s}_j$, as this changes the expected number of iterations over $j$ only by a small factor.

We are now ready to estimate the probability that the $j$-th iteration of the attack yields nothing, meaning that the attack is not finished in this iteration and afterwards it still holds that $Q_j = Q_{j-1}$ and $R_j = R_{j-1}$. Note that $\mathsf{V}_{a_j,b_j}(s_j, r_j) = \top$ and $s_j \neq \tilde{s}_j$ by assumption. Hence, the event that the attack is not finished means that $\mathsf{V}_{a_j,b_j}(\tilde{s}_j, \tilde{r}_j) = \bot$. The event that $Q_j = Q_{j-1}$ and $R_j = R_{j-1}$ means that $\mathsf{V}_{\tilde{a}_j,\tilde{b}_j}(s_j, r_j) = \top$. However, $(s_j, r_j, a_j, b_j)$ and $(\tilde{s}_j, \tilde{r}_j, \tilde{a}_j, \tilde{b}_j)$ are identically distributed by construction. Hence, $\Pr\big[\mathsf{V}_{a_j,b_j}(\tilde{s}_j, \tilde{r}_j) = \bot \wedge \mathsf{V}_{\tilde{a}_j,\tilde{b}_j}(s_j, r_j) = \top\big] \leq \frac{1}{2}$.

Putting things together, it finally follows that the expected number of iterations over $j$ is upper bounded polynomially in the number of OT-instances used in the presumed rOLE-protocol.

### 4.3 Impossibility of rNISC/OT for General Black-Box Zero-Knowledge

The impossibility result presented in this section consists of two parts. We first show that certain probabilistically checkable proofs (PCPs) do not exist. Then we derive from this the aimed at impossibility result for the OT hybrid model. We give only a very weak definition for zero-knowledge PCPs in this section, because this yields the strongest possible impossibility result we can come up with. In particular, we require soundness, completeness, and the zero-knowledge property only to hold with respect to PPT-adversaries (i.e., adversaries that have probabilistic polynomial time complexity).

**Notation 5.** *Given any domain $D$ and any subset $S \subseteq D$, we set $S^{\mathrm{C}} := D \backslash S$. By $x \leftarrow D$ we denote that $x$ is sampled uniformly at random from the domain $D$. Given some probabilistic algorithm $\mathsf{A}$, we denote by $x \leftarrow \mathsf{A}(y)$ that $x$ is generated by running $\mathsf{A}$ on input $y$ with fresh internal randomness; the distribution of $x$ in this case is denoted $\{\mathsf{A}(y)\}$. Given any bit-string $s$, we denote its $i$-th bit as $s[i]$ and its substring indexed by some index set $I$ as $s[I]$.*

**Definition 6** (Problems that are hard on average)**.** *The membership problem of any given language $\mathcal{L} \subseteq \{0,1\}^n$ is* hard on average*, if $\emptyset \neq \mathcal{L} \neq \{0,1\}^n$ and for every PPT-distinguisher $\mathsf{D}$ it holds:*

$$\Pr\big[\mathsf{D}(x) = \mathtt{accept} \,\big|\, x \leftarrow \mathcal{L}\big] \;\leq\; \Pr\big[\mathsf{D}(x) = \mathtt{accept} \,\big|\, x \leftarrow \mathcal{L}^{\mathrm{C}}\big] + n^{-\omega(1)}$$

*Or in other words, the uniform distributions over $\mathcal{L}$ and $\mathcal{L}^{\mathrm{C}}$ are computationally indistinguishable.*

**Definition 7** (Zero-knowledge PCPs)**.** *A* zero-knowledge PCP *(ZK-PCP) scheme for an NP-language $\mathcal{L} \subseteq \{0,1\}^n$ consists of a prover $\mathsf{P}$ and a verifier $\mathsf{V}$, both PPT-algorithms. Given any word $x \in \mathcal{L}$ and a corresponding witness $w$, the prover generates a proof $\pi \in \{0,1\}^m$, which we denote $\pi \leftarrow \mathsf{P}(x,w)$. The proof length $m$ is a parameter of the scheme. The verifier takes as input any $x \in \{0,1\}^n$ and $\pi \in \{0,1\}^m$ and either accepts or rejects, which we denote as $\mathsf{V}(x,\pi) = \mathtt{accept}$ or $\mathsf{V}(x,\pi) = \mathtt{reject}$ respectively. The following conditions must hold.*

*Completeness: For every PPT-computable tuple $(x,w)$ where $x \in \mathcal{L}$ and $w$ is a corresponding witness, we have that $\Pr\big[\mathsf{V}(x,\pi) = \mathtt{accept} \,\big|\, \pi \leftarrow \mathsf{P}(x,w)\big] \geq 1 - n^{-\omega(1)}$.*

*Soundness: For all PPT-computable $(x,\pi) \in \mathcal{L}^{\mathrm{C}} \times \{0,1\}^m$ we have $\Pr\big[\mathsf{V}(x,\pi) = \mathtt{accept}\big] \leq n^{-\omega(1)}$.*

*PCP property: The verifier $\mathsf{V}$ reads only some part of $\pi$: Once $x$ and $\mathsf{V}$'s internal randomness are fixed, we have a sequence of queries $q_1, \ldots, q_t \in \{1, \ldots, n\}$ such that each $q_i$ deterministically depends on $(x, \pi[q_1, \ldots, q_{i-1}])$ and it holds that $\mathsf{V}(x,\pi) = \mathsf{V}(x,\pi')$ for all $\pi, \pi' \in \{0,1\}^m$ with $\pi[q_1, \ldots, q_t] = \pi'[q_1, \ldots, q_t]$. The number $t$ of queries is a parameter of the PCP scheme.*
*If the query set $Q := \{q_1, \ldots, q_t\}$ is independent of $\pi$, we say that the scheme is* nonadaptive*. If $Q$ is independent of $x$, we say that the scheme is* input-oblivious*.*

*Zero-knowledge property: For every corrupted PPT-algorithm $\mathsf{V}^*$ there exists a PPT-simulator $\mathsf{S}$ such that for every PPT-computable tuple $(x,w)$ where $x \in \mathcal{L}$ and $w$ is a corresponding witness, the distributions specified by $\mathsf{S}(x)$ and $\mathsf{V}^*(x, \mathsf{P}(x,w))$ are computationally indistinguishable.*
*If there exists a universal PPT-simulator $\mathsf{S}$ that works for all corrupted receivers $\mathsf{V}^*$ and just needs black-box access to $\mathsf{V}^*$ in the sense that $\mathsf{V}^*$ successively outputs queries $q_i$ which*

*are answered by* $\mathsf{S}$ *with proof bits* $\tilde{\pi}[q_i]$ *computed from* $(x, q_1, \ldots, q_{i-1})$*, then we say that the scheme has* black-box simulatability*.*

*If it is assumed that a corrupted verifier* $\mathsf{V}^*$ *still respects the aforementioned query bound* $t$ *(i.e.,* $\mathsf{V}^*$ *reads at most* $t$ *bits of* $\pi := \mathsf{P}(x, w)$*), we speak of* bounded-query zero-knowledge*.*

**Definition 8** (Resettable soundness)**.** *A ZK-PCP scheme is* resettably sound*, if no PPT-adversary* $\mathsf{P}^*$ *with oracle access to the verifier functionality* $\mathsf{V}$ *(with randomly fixed internal randomness) can compute with non-negligible success probability some* $\hat{x} \in \mathcal{L}^{\mathrm{C}}$ *and* $\hat{\pi} \in \{0,1\}^m$ *such that* $\mathsf{V}(\hat{x}, \hat{\pi}) = \mathtt{accept}$*.*

**Theorem 9.** *Let* $\mathcal{L} \subseteq \{0,1\}^n$ *be an NP-language such that one can PPT-sample from* $\mathcal{L}$ *as well as from* $\mathcal{L}^{\mathrm{C}}$ *with computationally close to uniform distribution. Moreover, let the sampling in the former case be constructive in the sense that it also provides a witness. If the membership problem of* $\mathcal{L}$ *is hard on average, then* $\mathcal{L}$ *has no resettably sound input-oblivious nonadaptive bounded-query ZK-PCP scheme with black-box simulatability.*

*Proof.* Assume that $\mathcal{L}$ is hard on average and has an input-oblivious nonadaptive bounded-query ZK-PCP scheme with black-box simulatability. We show that this scheme cannot be resettably sound. Before we can present an attack, we have to construct a basic building block for it. Let $\mathsf{S}^*$ denote the following functionality, which takes as input any $x \in \mathcal{L}$ and $Q \subseteq \{1, \ldots, m\}$ with $|Q| \leq t$ and outputs some $\pi \in \{0,1\}^m$.

1. Construct the black-box simulator $\mathsf{S}$ for a corrupted verifier that queries the proof bits indexed by $Q$ and then just outputs $\pi[Q]$.
2. Sample $\pi[Q] \leftarrow \mathsf{S}(x)$, fill $\pi[Q^{\mathrm{C}}]$ with uniform randomness, and output $\pi$.

We argue that due to the hardness of $\mathcal{L}$ the algorithm $\mathsf{S}^*$ must still have polynomial runtime for input $x \leftarrow \mathcal{L}^{\mathrm{C}}$ and the distributions $\{\mathsf{S}^*(x, Q)\}_{x \leftarrow \mathcal{L}}$ and $\{\mathsf{S}^*(x, Q)\}_{x \leftarrow \mathcal{L}^{\mathrm{C}}}$ must be computationally indistinguishable for any PPT-computable $Q \subseteq \{1, \ldots, m\}$ of size at most $t$. We further argue that $\{\mathsf{S}^*(x, Q)[Q \cap Q']\}_{x \leftarrow \mathcal{L}}$ and $\{\mathsf{S}^*(x, Q')[Q \cap Q']\}_{x \leftarrow \mathcal{L}}$ must be computationally indistinguishable for any PPT-computable $Q, Q' \subseteq \{1, \ldots, m\}$ of size at most $t$, as they are both indistinguishable from $\{\mathsf{P}(x, w)[Q \cap Q']\}_{x \leftarrow \mathcal{L}}$ where $w$ is an arbitrary witness for $x \in \mathcal{L}$.

We are now ready to present our efficient (in the sense of PPT) attack, by which a malicious prover $\mathsf{P}^*$ can break resettable soundness. It works as follows.

1. Initialize $Q_0 := \emptyset$ and $j := 1$.
2. Sample $\hat{x}_j \leftarrow \mathcal{L}^{\mathrm{C}}$ and $\hat{\pi}_j \leftarrow \mathsf{S}^*(\hat{x}_j, Q_j)$. If $\mathsf{V}(\hat{x}_j, \hat{\pi}_j) = \mathtt{accept}$, output $(\hat{x}_j, \hat{\pi}_j)$ and terminate.
3. Sample $x_j \leftarrow \mathcal{L}$ together with a corresponding witness $w_j$ and sample $\pi_j \leftarrow \mathsf{P}(x_j, w_j)$ and $\tilde{\pi}_j \leftarrow \{0,1\}^m$. If $\mathsf{V}(x_j, \pi_j) = \mathtt{reject}$, give up.
4. As long as there exists some index $i \notin Q_{j-1}$ such that $\pi_j[i] \neq \tilde{\pi}_j[i]$ and $\mathsf{V}(x_j, \cdot\,)$ still accepts $\pi_j$ with the $i$-th bit flipped, flip $\pi_j[i]$. Then set $Q_j := Q_{j-1} \cup \{i \in \{1, \ldots, m\} \mid \pi_j[i] \neq \tilde{\pi}_j[i]\}$, increase $j$ by 1, and go to step 2.

Due to the the completeness property, we have for each iteration over $j$ that $\mathsf{P}^*$ gives up in Step 3 only with negligible probability. Therefore it suffices to show that the attack needs only polynomially many iterations over $j$ before it terminates. Consider the following sequence of experiments.

**Experiment 1:** This is the attack as stated above.

**Experiment 2:** The same as Experiment 1, except that the sampling of $\hat{x}_j$ in Step 2 is replaced by $\hat{x}_j \leftarrow \mathcal{L}$. It follows by the hardness of $\mathcal{L}$ that this changes the runtime complexity of the attack only up to some negligible statistical distance, at least if either Experiment 1 or Experiment 2 has polynomial runtime complexity.

**Functionality $\mathcal{F}_{\text{ZK}}$**

Implicitly parametrized by an *NP*-language $\mathcal{L} \subseteq \{0,1\}^n$, where $n$ is a security parameter.

1. Await an input $(x, w)$ from the sender. Store $(x, w)$ and send $(\texttt{sent})$ to the adversary.
2. Await a message $(\texttt{verify})$ from the adversary. Then, if $w$ is a witness for $x \in \mathcal{L}$, send $(x, \texttt{accept})$ to the verifier; else send $(x, \texttt{reject})$.

Figure 4: Ideal functionality for zero-knowledge proofs.

**Experiment 3:** The same as Experiment 2, except that the sampling of $\pi_j$ in Step 3 is replaced by $\pi_j \leftarrow \mathsf{S}^*(x_j, Q)$, where $Q$ denotes the actual set of $\mathsf{V}$'s queries. Note that $Q$ is well-defined, because the assumed ZK-PCP scheme is input-oblivious and nonadaptive. Since $\mathsf{V}$ reads only bits indexed by $Q$ and the new distribution of these bits is computationally indistinguishable from their original distribution, this changes the runtime complexity of the attack only up to some negligible statistical distance, at least if either Experiment 2 or Experiment 3 has polynomial runtime complexity.

We show now that Experiment 3 actually has polynomial runtime complexity, which concludes the proof. In Experiment 3 we have the following situation:

$$\begin{aligned} \hat{x}_j &\leftarrow \mathcal{L} & \qquad\qquad \hat{\pi}_j &\leftarrow \mathsf{S}^*(\hat{x}_j, Q_j) \\ x_j &\leftarrow \mathcal{L} & \qquad\qquad \pi_j &\leftarrow \mathsf{S}^*(x_j, Q) \end{aligned}$$

Recall that $\hat{\pi}_j[Q_j]$ and $\pi_j[Q_j]$ are computationally indistinguishable by the properties of $\mathsf{S}^*$. Moreover, $\hat{\pi}_j[Q_j^{\text{C}}]$ is uniformly random by construction of $\mathsf{S}^*$ and in Step 4 of our attack we replace the bits of $\pi_j[Q_j^{\text{C}}]$ successively with uniform randomness. Let $\bar{\pi}_j$ denote the bit-string that equals $\pi_j$ on all entries indexed by $Q_j$ and equals $\tilde{\pi}_j$ on all other entries, i.e., $\bar{\pi}_j[Q_j] = \pi_j[Q_j]$ and $\bar{\pi}_j[Q_j^{\text{C}}] = \tilde{\pi}_j[Q_j^{\text{C}}]$. Hence, the event that the $j$-th iteration of the attack yields nothing (meaning, $\mathsf{V}(\hat{x}_j, \tilde{\pi}_j) = \texttt{reject}$ and $Q_j = Q_{j-1}$) implies that $\mathsf{V}(\hat{x}_j, \hat{\pi}_j) = \texttt{reject}$ and $\mathsf{V}(x_j, \bar{\pi}_j) = \texttt{accept}$. Since $(\hat{x}_j, \hat{\pi}_j)$ and $(x_j, \bar{\pi}_j)$ are computationally indistinguishable, this may happen at most with probability $\frac{1}{2} + o(1)$. It follows that Experiment 3 terminates with overwhelming probability after polynomially many iterations. □

**Corollary 10.** If one-way functions exist, then there are instantiations of the functionality $\mathcal{F}_{\text{ZK}}$ (see Figure 4) that have no rNISC/OT protocol with black-box simulation of a corrupted receiver.

*Proof.* Since one-way functions imply pseudorandom number generators (PRGs), we can define $\mathcal{L}$ as the set of all length-$n$ pseudorandom strings that can be generated from length-$\lfloor \frac{n}{2} \rfloor$ seeds (with respect to some given family of PRGs). One can close to uniformly sample from $\mathcal{L}^{\text{C}}$ just by sampling uniformly from $\{0,1\}^n$ and the average hardness of $\mathcal{L}$'s membership problem is just a reformulation of the defining property of a PRG. Now, if we had an rNISC/OT protocol for the corresponding $\mathcal{F}_{\text{ZK}}$-functionality with black-box simulation of a corrupted receiver, then this would be a contradiction to Theorem 9. The receiver's OT-queries would correspond to the queried bits of a proof $\pi$. The resulting PCP scheme would be resettably sound, input-oblivious, and non-adaptive by construction. A caveat is in place with respect to the bounded-query zero-knowledge property, because in the presumed rNISC/OT protocol even a corrupted receiver cannot request both sender inputs in one of the underlying OT instances. However, Theorem 9 directly carries over to ZK-PCPs where such a limit is put on malicious verifiers. □

# 5 Reusable NISC via Parallel Reusable OLE

In this section, we describe our main constructions, namely protocols for reusable (designated verifier) NIZK and NISC for branching programs, both information-theoretically secure in the reusable OLE hybrid model.We also show how to obtain a reusable secure computation protocol for arbitrary functions in the reusable OLE hybrid model, in addition making black-box use of a pseudorandom generator.

Our construction proceeds bottom-up, starting from reusable OLE and building stronger and stronger primitives, leading up to reusable non-interactive zero-knowledge first (in Theorem 5.6) and reusable secure computation right after (in Theorems 5.8 and 5.9). A general theme that runs through our constructions is that each functionality builds on the previous one by restricting the "cheating space" of either the sender or the receiver; in doing so, it introduces some room for cheating by the other party. The next one reverses the role, and so forth. We converge in a small number of such steps and first construct our reusable NIZK protocol. Secure computation follows suit with some more work.

We first proceed to describe our functionalities, and realizing them shortly after.

## 5.1 Our Ideal Functionalities

We describe, in a high level, the sequence of our functionalities. The full functionality is somewhat complex and is described in Figure 15.

**Half-replicated OLE $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.** The reusable OLE functionality allows the receiver to commit to a number $x \in \mathbb{F}$ and allow a sender to transmit values $a_i x + b_i$ to the receiver for arbitrary $a_i, b_i \in \mathbb{F}$. By the definition of the functionality, the sender is assured that the receiver uses the same $x$ in all these iterations (just because he used the same receiver message to compute his response); however, there are no restrictions on the sender side.

Our final stop is zero-knowledge where the sender has to prove some relation among a set of number he chooses. There are two major gaps in getting there: (1) forcing consistency on the sender as opposed to the receiver; and (2) allowing for complex predicates on the sender input, instead of just replication constraints (which say that certain subsets of the sender's numbers are the same).

Our first functionality, $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$, takes one step in this direction. Namely, it allows the receiver to commit to a set of numbers $\{x_i\}$ and also the sender to commit to a set $\{a_j\}$. In each invocation subsequently, the sender can "point to" two of these numbers $(i, j)$ and produce a new number $b \in \mathbb{F}$ and transmit to the receiver $a_i x_j + b$. "Half replicated" because there are constraints on the sender's choice of "a" but no constraints on the "b".

**Half-replicated OLE with CDS $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$.** It is intuitively clear that it is easier to check the consistency of the receiver's input than the sender's. Roughly speaking, the receiver sends her input along without even having to see anything from the sender; however, the sender necessarily has to see the receiver's commitment to compute his response.

Our next functionality ups the ante by additionally allowing the sender to disclose a secret $s \in \mathbb{F}$ to the receiver if and only if the receiver's input $\{x_j\}$ satisfies a condition (thus, conditional disclosure of secrets or CDS) encoded as an arithmetic branching program. Recall that our final goal is to check the sender's input, not the receiver's, but this is a step along the way. Thus, the consistency condition is more than mere replication, but it is the receiver's input whose consistency is being checked.

**Half-replicated OLE with CDS and Proof $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$.** Our next functionality flips this around and shows what is necessary for zero-knowledge. Namely, it gives a way for the sender to prove that his chosen input satisfies a given predicate encoded as an arithmetic branching program. Realizing this functionality already proves Theorem 5.6. Throughout this process, we keep track of the number of (reusable) OLE calls made and show that at the end, the overhead is an absolute constant for every gate of the verification circuit. This is the type of guarantee you get for NISC/OT except we additionally achieve reusability.

**Replicated OLE (+CDS+Proof) $\mathcal{F}_{\texttt{repOLE},\texttt{CDS},\texttt{proof}}$.** Our final step closes the loop and constructs a fully replicated OLE where the receiver can commit to $\{a_i\}$ and at any time, point to $(i, i', j)$ and transmit to the receiver $a_i x_j + a_{i'} \in \mathbb{F}$. This closes the last bit cheating space the sender has, and combined with randomized encodings for arithmetic branching programs, allows us to achieve reusable NISC/OLE for arithmetic NC$^1$. Additionally, using garbled circuits, we can also obtain such a protocol for all of $P$ at the cost of a computational assumption, namely making black-box use of a PRG.

A final remark is in order. Our implementation of reusable OLE in the next section will work over $\mathbb{Z}_N^*$ which is not a field. Fortunately, it is a *pseudo-field*, in the sense that either (a) any protocol will work just as well over $\mathbb{Z}_N^*$ as over a field; or (b) the protocol will stumble upon the factorization of $N$. Since (b) is computationally hard, we obtain protocols that work well over the ring.

**Functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$**

**Choice phase:** Upon receiving input $(sid, x)$ from the receiver where $x \in \mathbb{F}$ and $sid \in \mathbb{N}$ is a session identifier, store $(sid, x)$, send $(sid, \texttt{input-init})$ to the adversary and ignore any further input from the receiver with the same $sid$.

**Send phase (choose coefficients):** Upon receiving input $(cid, a)$ from the sender where $a \in \mathbb{F}$ and $cid \in \mathbb{N}$ is a coefficient identifier, store $(cid, a)$, send $(cid, \texttt{coeff-init})$ to the adversary and ignore any further input from Sam with the same $cid$.

**Send phase (half replicated OLE):**

- Upon receiving input $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$ from Sam, record $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$, send $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(sid, cid, i)$.
- Upon receiving a message $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{Delivery}, i)$ from the adversary, verify that there are stored inputs $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i), (cid, a)$ from Sam and $(sid, x)$ from Rachel; else ignore that message. Next, compute $z = ax + b$, send $(\frac{1}{2}\texttt{repOLE}, sid, cid, z, i)$ to Rachel, and ignore further messages $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{Delivery}, i)$ from the adversary with the same $(sid, cid, i)$.

Figure 5: ideal functionality of half-replicated OLE

### 5.2 Constructing Half-Replicated OLE from Reusable OLE

**Functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$**

**Choice phase:** Upon receiving input $(sid, x)$ from the receiver where $x \in \mathbb{F}$ and $sid \in \mathbb{N}$ is a session identifier, store $(sid, x)$, send $(sid, \texttt{input-init})$ to the adversary and ignore any further input from the receiver with the same *sid*.

**Send phase (choose coefficients):** Upon receiving input $(cid, a)$ from the sender where $a \in \mathbb{F}$ and $cid \in \mathbb{N}$ is a coefficient identifier, store $(cid, a)$, send $(cid, \texttt{coeff-init})$ to the adversary and ignore any further input from Sam with the same *cid*.

**Send phase (half replicated OLE):**

- Upon receiving input $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$ from Sam, record $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$, send $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(sid, cid, i)$.
- Upon receiving a message $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{Delivery}, i)$ from the adversary, verify that there are stored inputs $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i), (cid, a)$ from Sam and $(sid, x)$ from Rachel; else ignore that message. Next, compute $z = ax + b$, send $(\frac{1}{2}\texttt{repOLE}, sid, cid, z, i)$ to Rachel, and ignore further messages $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{Delivery}, i)$ from the adversary with the same $(sid, cid, i)$.

Figure 6: ideal functionality of half-replicated OLE

We start with the definition of the half-replicated OLE functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in Figure 6. In reusable OLE, the receiver commits to values $x_i \in \mathbb{F}$, and the sender can transfer to the receiver a tuple $(i, a_j x_i + b_j)$ for any $a_j, b_j \in \mathbb{F}$ of his choice. In other words, all the transmissions of the sender with the same $i$ are guaranteed to be consistent with a single $x_i$, but the sender's coefficients $a_j$ and $b_j$ are completely unconstrained.

Roughly speaking, half-replicated OLE allows the receiver to commit to values $x_i \in \mathbb{F}$ (just as in reusable OLE) but in addition, also has a mechanism for the sender to commit to values $a_j \in \mathbb{F}$. The sender can later transfer to the receiver a tuple $(i, j, a_j x_i + b)$ for an arbitrary choice of $b \in \mathbb{F}$, simply by "pointing to" an appropriate $i$ and $j$. In other words, when the receiver obtains two tuples $(i_1, j, z_1)$ and $(i_2, j, z_2)$, he will be sure that both executions used the same multiplicative factor $a_j$.

We construct a protocol that achieves this functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in the $\mathcal{F}_{\texttt{rOLE}}$-hybrid model. Our construction proceeds in two steps.

**A First Try: The Protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$.** Protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ (Figure 7) is a protocol attempting to implement half replicated OLE (Figure 6). As a first trial, $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ is only secure against malicious receiver. Note that reusable OLE can also be viewed as an implementation of $\Pi_{\frac{1}{2}\texttt{repOLE}}$ that is secure against malicious receiver, and secure against semi-honest sender if we simply believe that the sender use the same multiplicative factors! What makes $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ useful is that it also limits the cheating space of a malicious sender. In each evaluation, either the sender is honest such that a extractor can extracts $(sid, cid, b)$ from the sender's messagese such that the receiver will outputs $a_{cid} x_{sid} + b$. Otherwise, the sender deviates from the protocol, and the receiver will outputs a uniform random value. The randomness should comes from the honest receiver so that it stays random even conditioned on sender's view and environment's view.

In protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, the receiver inputs to the underlying reusable OLE an extra number $\alpha$ that is sampled uniformly random from $\mathbb{F}$. Whenever the sender want to choose a new coefficient $a$, a rOLE evaluation of $a\alpha + r$ is required, where $r$ is a fresh random number chosen by the sender. Such an evaluation serves as a commitment on the coefficient chosen by the sender.

**Implementing $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ (First Try): Protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$**

### Choice phase:

The receiver, at the outset, samples $\alpha \in \mathbb{F}$ and send the pair $(sid_{\mathsf{alpha}}, \alpha)$ to $\mathcal{F}_{\texttt{rOLE}}$. Then, upon receiving input $(sid, x)$ from the environment, it samples $h \in \mathbb{F}$ and sends $(sid||0, h), (sid||1, x - \alpha h)$ to $\mathcal{F}_{\texttt{rOLE}}$.

### Send phase (choose coefficients):

Upon receiving input $(cid, a)$ from the environment, the sender samples a uniformly random $r \in \mathbb{F}$, and sends $(sid_{\mathsf{alpha}}, a, r, cid)$ to $\mathcal{F}_{\texttt{rOLE}}$.

Upon receiving a message $(sid_{\mathsf{alpha}}, \hat{a}, cid)$ from $\mathcal{F}_{\texttt{rOLE}}$, the receiver stores $(sid_{\mathsf{alpha}}, \hat{a}, cid)$.

### Send phase (half replicated OLE):

Upon receiving input $(sid, cid, b, i)$ from the environment, the sender verifies that $(sid_{\mathsf{alpha}}, a, r, cid)$ was sent to $\mathcal{F}_{\texttt{rOLE}}$. If yes, send $(sid, cid, i)$ to the receiver, and send both $(sid||0, r, w, i)$ and $(sid||1, a, b + w, i)$ to $\mathcal{F}_{\texttt{rOLE}}$.

Upon receiving $(sid, cid, i)$ from the sender and receiving $(sid||0, z_0, i), (sid||1, z_1, i)$ from $\mathcal{F}_{\texttt{rOLE}}$, the receiver verifies that there is a stored message $(sid_{\mathsf{alpha}}, \hat{a}, cid)$ from $\mathcal{F}_{\texttt{rOLE}}$. Next, compute $z = \hat{a}h - z_0 + z_1$ and output $(sid, cid, z, i)$ to the environment.

Figure 7: Protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ for half-replicated OLE in the $\mathcal{F}_{\texttt{rOLE}}$-hybrid model, secure against malicious receiver and "half-secure" against malicious sender (see Lemma 5.1, Lemma 5.2 for the formal definition of "half-security")

**Lemma 5.1.** *The protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ implements the functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in the $\mathcal{F}_{\texttt{rOLE}}$-hybrid model against a malicious receiver and a semi-honest sender.*

*Proof.* Correctness follows directly from the fact that

$$\begin{aligned} ax + b &= a(\alpha h + x - \alpha h) + b \\ &= a\alpha h + a(x - \alpha h) + b \\ &= (a\alpha + r)h - rh + a(x - \alpha h) + b \\ &= \underbrace{(a\alpha + r)}_{\hat{a}} h - \underbrace{(rh + w)}_{z_0} + \underbrace{a(x - \alpha h) + w + b}_{z_1}. \end{aligned}$$

There is a straight-line simulator for any (potentially malicious) receiver $\widehat{R}$:

- Store $\alpha$ when the receiver outputs $(sid_{\mathsf{alpha}}, \alpha)$. Whenever the receiver has outputted $(sid||0, h_{sid})$ and $(sid||1, y_{sid})$, store $h_{sid}$ and send $(sid, \alpha h_{sid} + y_{sid})$ to ideal functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$. Note that the receiver has no malicious strategy: any sequence of output can comes from honest strategy for appropriate input and randomness.

- Whenever the simulator saw a new coefficient identifier *cid*, sample random $\hat{a}_{cid}$ and send $(sid_{\mathsf{alpha}}, \hat{a}_{cid}, cid)$ to $\widehat{R}$. In real world, $\hat{a}_{cid}$ is uniform random because it's one-time padded by fresh randomness sampled by sender.

- For any evaluation indexed by $(sid, cid, i)$, upon receiving $(sid, cid, z, i)$ from functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$, send $(sid\|0, z_0, i), (sid\|1, z_1, i)$ to $\widehat{R}$ such that $z_0$ is uniform random and $z_1 = z - \hat{a}_{cid}h_{sid} + z_0$. In real world, $z_0$ is uniform random because it's one-time padded by fresh randomness sampled by sender, and $z_1$ can be determined from $z_0$, $\hat{a}_{cid}$, receiver's randomness $h_{sid}$ and functionality's output. □

We now define the notion of a well-formed sender, towards describing a nice property that our first attempt achieves. For a given (potentially malicious) sender, say a coefficient identifier *cid* is valid if the sender inputs $(sid_{\mathsf{alpha}}, a, r, cid)$ for some $a, r$ to $\mathcal{F}_{\texttt{rOLE}}$; say a tuple $(sid, cid, i)$ is valid if *cid* is valid and the sender's messages include $(sid, cid, i), (sid\|0, r', b_0, i), (sid\|1, a', b_1, i)$ for some $a', r', b_0, b_1$. It's clear that the honest receiver will output tuple $(sid, cid, z_{sid,cid,i}, i)$ for some $z_{sid,cid,i}$ for each valid $(sid, cid, i)$.

**Lemma 5.2.** *The protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ implements the functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in the $\mathcal{F}_{rOLE}$-hybrid model such that for every (potentially malicious) sender $\widehat{S}$ there is a p.p.t. straight-line extractor $E$ which outputs a sequence of tuples the form $(cid, \hat{a}_{cid})$ where $\hat{a}_{cid} \in \mathbb{F}$ for each valid cid and a sequence of tuples of the form $(sid, cid, \hat{b}, i)$ where $\hat{b} \in \mathbb{F} \cup \{\perp\}$ for each valid $(sid, cid, i)$, such that for any sequence of tuples of the form $(sid, x_{sid})$ that the receiver chooses as its input, the following two properties hold:*

- Completeness: *For every extracted tuple $(sid, cid, \hat{b}, i)$ where $\hat{b} \in \mathbb{F}$, the receiver output contains $(sid, cid, \hat{a}_{cid} \cdot x_{sid} + \hat{b}, i)$.*

- Soundness: *For every extracted tuple $(sid, cid, \hat{b}, i)$ where $\hat{b} = \perp$, the receiver output contains $(sid, cid, z_{sid,cid,i}, i)$ such that conditioned on the sender's view, $\hat{z}_{sid,cid,i}$ is $\frac{1}{|\mathbb{F}|}$-close to the uniform distribution over $\mathbb{F}$. Moreover, for any $k \in \mathbb{N}$ and any sequence of tuples $(sid_1, cid_1, \hat{b}_1, i_1), \ldots, (sid_k, cid_k, \hat{b}_k, i_k)$ such that $\hat{b}_j = \perp$ and the $sid_j$ are distinct, the joint distribution of $z_{sid_1,cid_1,i_1}, \ldots, z_{sid_k,cid_k,i_k}$ is $\frac{k}{|\mathbb{F}|}$-close to uniform conditioned on the sender's view.*

*Proof.* The straight-line extractor $E$ does the following

- When the sender sends $(sid_{\mathsf{alpha}}, a_{cid}, r_{cid}, cid)$ to $\mathcal{F}_{\texttt{rOLE}}$, store $(a_{cid}, r_{cid})$ and output $(cid, a_{cid})$.

- When the sender sends $(sid, cid, i)$ to receiver and inputs $(sid\|0, r', b_0, i)$, $(sid\|1, a', b_1, i)$ to $\mathcal{F}_{\texttt{rOLE}}$, the extractor $E$ outputs $(sid, cid, \perp, i)$ if $a' \neq a_{cid}$ or $r' \neq r_{cid}$, outputs $(sid, cid, b_1 - b_0, i)$ otherwise.

For any $(sid, cid, i)$ that index an evaluation, the sender sends $(sid, cid, i)$ to receiver and inputs $(sid\|0, r', b_0, i)$, $(sid\|1, a', b_1, i)$ to $\mathcal{F}_{\texttt{rOLE}}$, the sender also sends $(sid_{\mathsf{alpha}}, a_{cid}, r_{cid}, cid)$ to $\mathcal{F}_{\texttt{rOLE}}$. The honest receiver has sampled random $\alpha, h_{sid}$ and inputs $(sid_{\mathsf{alpha}}, \alpha)$, $(sid\|0, h_{sid})$, $(sid\|1, x - \alpha h_{sid})$ to $\mathcal{F}_{\texttt{rOLE}}$, thus the honest receiver will gets $(sid_{\mathsf{alpha}}, \hat{a}, cid)$, $(sid\|0, z_0, i)$, $(sid\|1, z_1, i)$ from $\mathcal{F}_{\texttt{rOLE}}$ such that

$$\hat{a} = a_{cid}\alpha + r_{cid}, \quad z_0 = r'h_{sid} + b_0, \quad z_1 = a'(x - \alpha h_{sid}) + b_1.$$

The honest receiver then compute $z_{sid,cid,i} = \hat{a}h_{sid} - z_0 + z_1$ and output $(sid, cid, z_{sid,cid,i}, i)$, such that

$$\begin{aligned} z_{sid,cid,i} &= \hat{a}h_{sid} - z_0 + z_1 \\ &= (a_{cid}\alpha + r_{cid})h_{sid} - (r'h_{sid} + b_0) + a'(x - \alpha h_{sid}) + b_1 \\ &= a'x + b_1 - b_0 + ((a_{cid} - a')\alpha + r_{cid} - r')h_{sid}. \end{aligned}$$

- Completeness: When $a' = a_{cid}$ and $r' = r_{cid}$, the extractor will extract $(sid, cid, b_1 - b_0, i)$ for this evaluation and $z_{sid,cid,i} = a_{cid}x + b_1 - b_0$.

- Soundness: When $a' \neq a_{cid}$ or $r' \neq r_{cid}$, the extractor will extract $(sid, cid, \perp, i)$ for this evaluation. Condition on very likely event $(a_{cid} - a')\alpha + r_{cid} - r' \neq 0$, whose probability is at least $1 - \frac{1}{|\mathbb{F}|}$, the distribution of $z_{sid,cid,i}$ is uniform as it's randomized by $h_{sid}$.

  Moreover, consider the sequence $z_{sid_1,cid_1,i_1}, \ldots, z_{sid_k,cid_k,i_k}$ where for each $j \in [k]$, the extractor outputs $(sid, cid, \perp, i)$ and the honest receiver outputs $(sid, cid, z_{sid_1,cid_1,i_1}, i)$. If all $sid_j$ are distinct, then the joint distribution of $z_{sid_1,cid_1,i_1}, \ldots, z_{sid_k,cid_k,i_k}$ is close to uniform random as it's randomized by $(h_{sid_1}, \ldots, h_{sid_k})$. □

Protocol $\Pi_{\frac{1}{2}\texttt{repOLE}}$ (Figure 8) implements $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ and it is built on top of $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. Note that $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ is close to a secure implementation of half-replicated OLE $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$: it's secure against malicious receiver, and the sender has limited cheating room — in any single evaluation, the honest receiver will output a random value if the sender misbehaves.

To detect whenever the sender misbehaves, the receiver duplicates its input and sends the same value to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ under different session identifier. Say for each $(sid, x_{sid})$ picked by the environment, the receiver feeds both $(sid\|0, x_{sid})$ and $(sid\|1, x_{sid})$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. Later when each $(sid, cid, b, i)$ picked by the environment, the sender is asked to send both $(sid\|0, cid, b, i)$ and $(sid\|1, cid, b, i)$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. The honest receiver expects to receive same evaluation values. With a redundant evaluation, misbehaved sender will be caught immediately.

The counter effect of a redundant evaluation is that the receiver is able to learn extra information. A malicious receiver can choose $(sid\|0, x_{sid}), (sid\|1, x'_{sid})$ such that $x_{sid} \neq x'_{sid}$. To prevent such attack, the sender samples a random $s_1 \in \mathbb{F}$, encrypts one of the evaluation resulting using $s_1$ as a one-time pad, and leaks $s_1$ to the receiver if and only if $x_{sid} = x'_{sid}$. The general way to disclose a secret in $\mathbb{F}$ to the receiver if the receiver's chosen input satisfies some condition is discussed in Section 5.3. In this case, sender disclose $s$ to the receiver by sample random $r \in \mathbb{F}$ as a new coefficient, and sends $r(x_{sid} - x'_{sid}) + s_1$ to the receiver via 2 $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ evaluations. So that $s_1$ is hidden from the receiver if $x_{sid} \neq x'_{sid}$.

So far, it seems that we fall into a deadloop. As the sender is again able to behave maliciously. We jump out from this deadloop using a hash-and-check trick. The sender samples a larger randomness $\mathbf{s} = (s_1, s_2) \in \mathbb{F}^2$ and sends $\mathbf{s}$ to the receiver by sample random $\mathbf{r} \in \mathbb{F}^2$ and sends $(x_{sid} - x'_{sid})\mathbf{r} + \mathbf{s}$ to the receiver via 4 $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ evaluations. In addition, sends $s_1\gamma + s_2$ to the receiver, where random $\gamma \in \mathbb{F}$ is sampled by the receiver, via 1 $\mathcal{F}_{\texttt{rOLE}}$ evaluations. Then from a malicious receiver's view, if the malicious receiver sets $x_{sid} \neq x'_{sid}$, then $(x_{sid} - x'_{sid})\mathbf{r} + \mathbf{s}$ leaks no information about $\mathbf{s}$ and $s_1\gamma + s_2$ doesn't leaks enough information to recover $s_1$. From a malicious sender's view, the random $\gamma$ essentially picks a hash function $\mathbf{s} \mapsto s_1\gamma + s_2$. If the sender misbehave, then it's very likely caught by an honest receiver because the hash values don't match.

As for the concrete efficiency of $\Pi_{\frac{1}{2}\texttt{repOLE}}$: the receiver chooses 2 extra $\mathcal{F}_{\texttt{rOLE}}$ inputs at the outset; each chosen input corresponds to 2 inputs in $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, which correspond to 4 $\mathcal{F}_{\texttt{rOLE}}$ inputs; each

**Implementing $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ (Second and Final Try): Protocol $\Pi_{\frac{1}{2}\texttt{repOLE}}$**

**Choice phase:**
The receiver, at the outset, samples $\gamma \in \mathbb{F}$ and sends the pair $(sid_{\mathsf{gamma}}, \gamma)$ to $\mathcal{F}_{\texttt{rOLE}}$. Then, upon receiving input $(sid, x)$ from the environment, inputs $(sid\|0, x), (sid\|1, x)$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$.

**Send phase (choose coefficients):** Upon receiving input $(cid, a)$ from environment, the sender forwards $(cid, a)$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$.

**Send phase (half replicated OLE):**
Sender: Upon receiving input $(sid, cid, b, i)$ from the environment, sample $s_1, r_1, w_1, s_2, r_2, w_2 \in \mathbb{F}$, pick unused coefficient identifier $cid_1, cid_2$, send $(sid, cid, i, cid_1, cid_2)$ to receiver and send the queries $(cid_1, r_1)$ and $(cid_2, r_2)$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, send

$$\begin{array}{ll} (sid\|0, cid, b, i), & (sid\|1, cid, b + s_1, i), \\ (sid\|0, cid_1, s_1 + w_1), & (sid\|1, cid_1, w_1), \\ (sid\|0, cid_2, s_2 + w_2), & (sid\|1, cid_2, w_2) \end{array}$$

to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, and send $(sid_{\mathsf{gamma}}, s_1, s_2, i)$ to $\mathcal{F}_{\texttt{rOLE}}$.

Receiver: Upon receiving $(sid, cid, i, cid_1, cid_2)$ from sender, receiving $(sid_{\mathsf{gamma}}, h, i)$ from $\mathcal{F}_{\texttt{rOLE}}$ and receiving $(sid\|0, cid, z, i)$, $(sid\|1, cid, \hat{z}, i)$, $(sid\|0, cid_1, v_1)$, $(sid\|1, cid_1, u_1)$, $(sid\|0, cid_2, v_2)$, $(sid\|1, cid_2, u_2)$ from $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. Abort if either $\gamma(v_1 - u_1) + (v_2 - u_2) \neq h$ or $z \neq \hat{z} - (v_1 - u_1)$. Otherwise, output $(sid, cid, z, i)$ to environment.

Figure 8: Protocol $\Pi_{\frac{1}{2}\texttt{repOLE}}$ for half-replicated OLE in the $\mathcal{F}_{\texttt{rOLE}}$-hybrid model, using $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ as a sub-protocol

chosen coefficient corresponds to 1 coefficient in $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, which corresponds to 1 $\mathcal{F}_{\texttt{rOLE}}$ evaluation call; each evaluation corresponds to 6 $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ evaluations + 1 $\mathcal{F}_{\texttt{rOLE}}$ evaluation, which correspond to 13 $\mathcal{F}_{\texttt{rOLE}}$ evaluation calls.

**Theorem 5.3.** *The protocol $\Pi_{\frac{1}{2}\texttt{repOLE}}$ (Figure 8) implements the functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in the $\mathcal{F}_{rOLE}$-hybrid model.*

*Proof.* Correctness: Consider one evaluation indexed by $(sid, cid, i)$, assume $(sid, cid, b, i), (sid, x), (cid, a)$ are the corresponding messages from environment. When sender and receiver are honest, receiver will receive $(sid\|0, cid, z, i)$, $(sid\|1, cid, \hat{z}, i)$, $(sid\|0, cid_1, v_1)$, $(sid\|1, cid_1, u_1)$, $(sid\|0, cid_2, v_2)$, $(sid\|1, cid_2, u_2)$ from $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ and $(sid_{\mathsf{gamma}}, h, i)$ from $\mathcal{F}_{\texttt{rOLE}}$, such that

$$\begin{array}{ll} z = ax + b, & \hat{z} = ax + b + s_1, \\ v_1 = r_1 x + s_1 + w_1, & u_1 = r_1 x + w_1, \\ v_2 = r_2 x + s_2 + w_2, & u_2 = r_2 x + w_2, \\ h = \gamma s_1 + s_2, & \end{array}$$

where $s_1, s_2, r_1, r_2, w_1, w_2, \gamma$ are random values sampled by either the sender or the receiver. The receiver won't abort as $\gamma(v_1 - u_1) + (v_2 - u_2) = \gamma s_1 + s_2 = h$ and $\hat{z} - (v_1 - u_1) = \hat{z} - s_1 = z$. The receiver will output $(sid, cid, z, i)$ for $z = ax + b$ which agrees with the ideal functionality.

Security against malicious receiver is ensured by a straight-line simulator. For any (potentially malicious) receiver $\hat{R}$, the simulator does the following

- In choice phase, store $\gamma$ when the receiver outputs $(\mathit{sid}_{\textsf{gamma}}, \gamma)$ to $\mathcal{F}_{\texttt{rOLE}}$. When the receiver has outputted $(\mathit{sid}\|0, x_{\mathit{sid}}), (\mathit{sid}\|1, x'_{\mathit{sid}})$ to underlying $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ protocol, store $(\mathit{sid}, x_{\mathit{sid}}, x'_{\mathit{sid}})$ and sends $(\mathit{sid}, x_{\mathit{sid}})$ to ideal functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

- Consider one evaluation in send phase, upon receiving $(\mathit{sid}, \mathit{cid}, z, i)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in ideal world, receiver $\hat{R}$ should receives corresponding message $(\mathit{sid}\|0, \mathit{cid}, z, i)$, $(\mathit{sid}\|1, \mathit{cid}, \hat{z}, i)$, $(\mathit{sid}\|0, \mathit{cid}_1, v_1)$, $(\mathit{sid}\|1, \mathit{cid}_1, u_1)$, $(\mathit{sid}\|0, \mathit{cid}_2, v_2)$, $(\mathit{sid}\|1, \mathit{cid}_2, u_2)$ in the real world such that

$$\begin{aligned} z &= ax_{\mathit{sid}} + b, & \hat{z} &= ax'_{\mathit{sid}} + b + s_1, \\ v_1 &= r_1 x_{\mathit{sid}} + s_1 + w_1, & u_1 &= r_1 x'_{\mathit{sid}} + w_1, \\ v_2 &= r_2 x_{\mathit{sid}} + s_2 + w_2, & u_2 &= r_2 x'_{\mathit{sid}} + w_2, \\ h &= \gamma s_1 + s_2, \end{aligned}$$

  where $a, b$ are coefficients hidden from the simulator, and $r_1, s_1, w_1, r_2, s_2, w_2$ are fresh randomness sampled by the sender. Then the receiver's view can be simulated by considering whether $x_{\mathit{sid}} = x'_{\mathit{sid}}$. In the case when $x_{\mathit{sid}} = x'_{\mathit{sid}}$,

  - $z$ is given by ideal functionality;
  - the joint distribution of $v_1, u_1, v_2, u_2$ is uniform random as it's one-time padded by $(s_1, w_1, s_2, w_2)$;
  - given $z, v_1, u_1, v_2, u_2$ and the receiver's view, the value of $\hat{z}, h$ are uniquely determined by $h = \gamma(v_1 - u_1) + (v_1 - u_2)$, $\hat{z} = z + v_1 - u_1$.

  Otherwise, in the case when $x_{\mathit{sid}} \neq x'_{\mathit{sid}}$,

  - $z$ is given by ideal functionality;
  - the joint distribution of $v_1, u_1, v_2, u_2$ is uniform random as it's one-time padded by $(r_1, w_1, r_2, w_2)$;
  - given $z, v_1, u_1, v_2, u_2$ and the receiver's view, the joint distribution of $\hat{z}, h$ is uniformly random as it's one-time padded by $(s_1, s_2)$.

Security against malicious sender is ensured by a straight-line simulator which can simulate any (potentially malicious) sender $\widehat{S}$. Notice that protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ is the secure against malicious sender, thus the simulator is built on top of the straight-line extractor for protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. For each evaluation labeled by $(\mathit{sid}, \mathit{cid}, i)$, the sender outputs $(\mathit{sid}_{\textsf{gamma}}, s_1, s_2, i)$ to $\mathcal{F}_{\texttt{rOLE}}$ for some $s_1, s_2 \in \mathbb{F}$, and from the sender's outputs to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, a straight-line extractor extracts $(\mathit{cid}, a), (\mathit{cid}_1, r_1), (\mathit{cid}_2, r_2)$ for some $a, r_1, r_2 \in \mathbb{F}$ and $(\mathit{sid}\|0, \mathit{cid}, b, i), (\mathit{sid}\|1, \mathit{cid}, \hat{b}, i)$, $(\mathit{sid}\|0, \mathit{cid}_1, \hat{w}_1)$, $(\mathit{sid}\|1, \mathit{cid}_1, w_1)$, $(\mathit{sid}\|0, \mathit{cid}_2, \hat{w}_2)$, $(\mathit{sid}\|1, \mathit{cid}_2, w_2)$ for some $b, \hat{b}, \hat{w}_1, w_1, \hat{w}_2, w_2 \in \mathbb{F} \cup \{\perp\}$.

In one case, the sender might follows the protocol in this evaluation. The sender is considered honest if none of $b, \hat{b}, \hat{w}_1, w_1, \hat{w}_2, w_2$ equals $\perp$ and $\hat{b} = b + s_1, \hat{w}_1 = s_1 + w_1, \hat{w}_2 = s_2 + w_2$. In such case, it's simulatable by sending $(\mathit{cid}, a), (\mathit{sid}, \mathit{cid}, b, i)$ to ideal functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

In the other case where the sender deviates from the protocol, the honest receiver will abort with overwhelming probability $1 - O(\frac{1}{|\mathbb{F}|})$. In such case, the receiver will receives message $(\mathit{sid}, \mathit{cid}, z, i)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ and messages $(\mathit{sid}\|0, \mathit{cid}, z, i)$, $(\mathit{sid}\|1, \mathit{cid}, \hat{z}, i)$, $(\mathit{sid}\|0, \mathit{cid}_1, v_1)$, $(\mathit{sid}\|1, \mathit{cid}_1, u_1)$, $(\mathit{sid}\|0, \mathit{cid}_2, v_2)$,

$(sid\|1, cid_2, u_2)$ from underlying $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ such that

$$\begin{array}{llll} z = ax + b & \text{if } b \neq \perp, & \hat{z} = ax + \hat{b} & \text{if } \hat{b} \neq \perp, \\ v_1 = r_1 x + \hat{w}_1 & \text{if } \hat{w}_1 \neq \perp, & u_1 = r_1 x + w_1 & \text{if } w_1 \neq \perp, \\ v_2 = r_2 x + \hat{w}_2 & \text{if } \hat{w}_2 \neq \perp, & u_2 = r_2 x + w_2 & \text{if } w_2 \neq \perp, \\ h = \gamma s_1 + s_2, & & & \end{array}$$

where $x$ is hidden from the simulator, $\gamma$ is a random number sampled by the receiver.

- *Case I, if $\hat{w}_1 = \perp$ or $w_1 = \perp$ or $\hat{w}_1 \neq w_1 + s_1$:* When $\hat{w}_1 = w_1 = \perp$, the joint distribution of $v_1, u_1$ is $\frac{2}{|\mathbb{F}|}$-close to uniform, thus $s_1 = (v_1 - u_1)$ with probability at most $\frac{3}{|\mathbb{F}|}$. Otherwise, the probability that $s_1 = (v_1 - u_1)$ is even smaller. Condition on $s_1 \neq (v_1 - u_1)$, the probability $\gamma(v_1 - u_1) - (v_2 - u_2) = h$ is no more than $\frac{1}{|\mathbb{F}|}$ as $\gamma$ is uniform random. In summary, the honest receiver will abort with probability at least $(1 - \frac{4}{|\mathbb{F}|})$ in such case.

- *Case II, if $\hat{w}_2 = \perp$ or $w_2 = \perp$ or $\hat{w}_2 \neq w_2 + s_2$:* Similarly, the honest receiver will abort with probability at least $(1 - \frac{4}{|\mathbb{F}|})$ in such case.

- *Case III, otherwise if $b = \perp$ or $\hat{b} = \perp$ or $\hat{b} \neq b + s_1$:* We can assume $s_1 = \hat{w}_1 - w_1$ as it doesn't fall into case I. When $b = \hat{b} = \perp$, the joint distribution of $z, \hat{z}$ is $\frac{2}{|\mathbb{F}|}$-close to uniform, thus $z = \hat{z} - (v_1 - u_1)$ with probability at most $\frac{3}{|\mathbb{F}|}$. Otherwise the probability that $z = \hat{z} - (v_1 - u_1)$ is even smaller. In summary, the honest receiver will abort with probability at least $(1 - \frac{3}{|\mathbb{F}|})$ in such case. □

### 5.2.1 Syntactic sugar: Oblivious *Multi-variant* Linear Function Evaluation

To simplify the proofs, it's worth bringing some syntactic sugar into $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ functionality. Right now, in each evaluation, the sender can pick $(sid, cid, b)$ so that the receiver learns $(sid, cid, a_{cid}x_{sid} + b)$. This is a powerful functionality that almost immediately supports the oblivious evaluation of a *multi-variant* linear function, where the sender picks $(sid_1, cid_1, c_1, \ldots, sid_\ell, cid_\ell, c_\ell, b)$ for some $c_1, \ldots, c_\ell, b \in \mathbb{F}$ and receiver gets $(sid_1, cid_1, c_1, \ldots, sid_\ell, cid_\ell, c_\ell, z)$ that $z = \sum_{j=1}^{\ell} a_{cid_j} c_j x_{sid_j} + b$ (as formalized in Figure 9).

There is a simple protocol implementing this enhanced functionality in $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$-hybrid model and it's UC-secure. W.l.o.g. we can assume $c_j \neq 0$ for all $j \in [\ell]$.

- Upon receiving $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, b, i)$ from the environment, the sender samples $b_1, \ldots, b_n \in \mathbb{F}$ such that $\sum_{j=1}^{\ell} c_j b_j = b$. The sender picks appropriate $i_1, \ldots, i_\ell$, sends $((sid_j, cid_j, c_j)_{j=1}^{\ell}, i, (i_j)_{j=1}^{\ell})$ to the receiver, and then inputs $(\frac{1}{2}\texttt{repOLE}, sid_j, cid_j, b_j, i_j)$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ for all $j \in [\ell]$.

- Upon receiving $((sid_j, cid_j, c_j)_{j=1}^{\ell}, i, (i_j)_{j=1}^{\ell})$ from the sender and receiving $(\frac{1}{2}\texttt{repOLE}, sid_j, cid_j, z_j, i_j)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ for all $j \in [\ell]$, the receiver computes $z = \sum_{j=1}^{\ell} c_j z_j$ outputs $((sid_j, cid_j, c_j)_{j=1}^{\ell}, z, i)$.

Correctness follows directly from the fact that

$$\sum_{j=1}^{\ell} c_j z_j = \sum_{j=1}^{\ell} c_j (a_{cid_j} x_{sid_j} + b_j) = \sum_{j=1}^{\ell} c_j a_{cid_j} x_{sid_j} + b.$$

**Functionality $\mathcal{F}_{\texttt{sugar-}\frac{1}{2}\texttt{repOLE}}$**

**Choice phase (the same as $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in Figure 6):** Upon receiving input $(sid, x)$ from the receiver where $x \in \mathbb{F}$ and $sid \in \mathbb{N}$ is a session identifier, store $(sid, x)$, send $(sid, \texttt{input-init})$ to the adversary and ignore any further input from the receiver with the same $sid$.

**Send phase (choose coefficients, the same as $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ in Figure 6):** Upon receiving input $(cid, a)$ from the sender where $a \in \mathbb{F}$ and $cid \in \mathbb{N}$ is a coefficient identifier, store $(cid, a)$, send $(cid, \texttt{coeff-init})$ to the adversary and ignore any further input from Sam with the same $cid$.

**Send phase (half replicated OLE):**

- Upon receiving input $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, b, i)$ from Sam, record it, send $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, i)$.
- Upon receiving a message $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary, verify that there are stored inputs $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, b, i)$ and $(cid_j, a_j)$ for $j \in [\ell]$ from Sam and $(sid_j, x_j)$ for $j \in [\ell]$ from Rachel; else ignore that message. Next, compute $z = \sum_{j=1}^{\ell} a_{cid_j} c_j x_{sid_j} + b$, send $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, z, i)$ to Rachel, and ignore further messages $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary with the same $((sid_j, cid_j, c_j)_{j=1}^{\ell}, i)$.

Figure 9: ideal functionality of half replicated OLE with syntactic sugar

The (potentially malicious) receiver's view is a additive secret sharing of the functionality output, thus it is easily simulatable: get $((sid_j, cid_j, c_j)_{j=1}^{\ell}, z, i)$ from ideal functionality, sample $(z_j)_{j=1}^{\ell}$ such that $z = \sum_{j=1}^{\ell} c_j z_j$ and sends $(\frac{1}{2}\texttt{repOLE}, sid_j, cid_j, z_j, i_j)$ for all $j \in [\ell]$ to the receiver.

If the (potentially malicious) sender feeds $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ with tuples $(\frac{1}{2}\texttt{repOLE}, sid_j, cid_j, b_j, i_j)$ for $j \in [\ell]$, it can be simulated as sending $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid_j, c_j)_{j=1}^{\ell}, b, i)$ for $b = \sum_{j=1}^{\ell} c_j b_j$ to the ideal functionality.

## 5.3 Allowing CDS operations in Half-Replicated OLE

In Section 5.2, the construction of $\Pi_{\frac{1}{2}\texttt{repOLE}}$ implicitly contains a CDS protocol that disclose a secret to the receiver if and only if the receiver has chosen same value under two different session identifiers. In this section, we allow the sender to disclose a secret to the receiver under more complex conditions (formally defined as $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$ in Figure 10). Denote the receiver's chosen input by $x_{sid}$ if the receiver has sent $(sid, x_{sid})$.

First consider how to disclose a secret condition on an affine constraint. An affine constraint on $\ell$ variables is specified by pair $(f, \{sid_j\}_{j=1}^{\ell})$, where $f$ is an affine function defined by a sequence of non-zero coefficients $c_0, c_1, \ldots, c_\ell$ such that $f(v_1, \ldots, v_\ell) = c_0 + \sum_{j=1}^{\ell} c_j v_j$. The sender would like to disclose a secret $s \in \mathbb{F}$ to the receiver if and only if $f(x_{sid_1}, \ldots, x_{sid_\ell}) = 0$. To do so, it samples random $r \in \mathbb{F}$ sends $(c_0 + \sum_{j=1}^{\ell} c_j x_{sid_j})r + s$ to the receiver via $\ell$ $\frac{1}{2}\texttt{repOLE}$ evaluations.

Then consider more general arithmetic constraints, which can be modeled by branching programs.

**Functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$**

Parametrized by a finite field $\mathbb{F}$, and $\mathfrak{C}$ a class of functions.

**Choice phase, part I (the same as Figure 6):** Upon receiving input $(sid, x)$ from the receiver where $x \in \mathbb{F}$ and $sid \in \mathbb{N}$ is a session identifier, store $(sid, x)$, send $(sid, \texttt{input-init})$ to the adversary and ignore any further input from the receiver with the same $sid$.

**Choice phase, part II (CDS):** Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from Rachel where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(sid_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, record $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$, send $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \texttt{sent})$ to the adversary, and ignore any further inputs from Rachel with the same $(f, \{sid_j\}_{j=1}^{\ell})$.

**Send phase (choose coefficients, the same as Figure 6):** Upon receiving input $(cid, a)$ from the sender where $a \in \mathbb{F}$ and $cid \in \mathbb{N}$ is a coefficient identifier, store $(cid, a)$, send $(cid, \texttt{coeff-init})$ to the adversary and ignore any further input from Sam with the same $cid$.

**Send phase, part I (half replicated OLE, the same as Figure 6):**

- Upon receiving input $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$ from Sam, record $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$, send $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(sid, cid, i)$.
- Upon receiving a message $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{Delivery}, i)$ from the adversary, verify that there are stored inputs $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i), (cid, a)$ from Sam and $(sid, x)$ from Rachel; else ignore that message. Next, compute $z = ax + b$, send $(\frac{1}{2}\texttt{repOLE}, sid, cid, z, i)$ to Rachel, and ignore further messages $(\frac{1}{2}\texttt{repOLE}, sid, cid, \texttt{Delivery}, i)$ from the adversary with the same $(sid, cid, i)$.

**Send phase, part II (CDS):**

- Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from Sam where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from Rachel; else ignore that message. Next, record $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$, send $(f, \{sid_j\}_{j=1}^{\ell}, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(f, \{sid_j\}_{j=1}^{\ell}, i)$.
- Upon receiving a message $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary, verify that there are stored input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from Sam and $(sid_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, send to Rachel $(f, \{sid_j\}_{j=1}^{\ell}, s, i)$ if $f(x_1, \ldots, x_{\ell}) = 0$ otherwise $(f, \{sid_j\}_{j=1}^{\ell}, \perp, i)$, and ignore further messages $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary with the same $(f, \{sid_j\}_{j=1}^{\ell}, i)$.

Figure 10: ideal functionality of half replicated OLE + CDS

**Branching programs** Before formalizing branching programs, for convenience, define affine function *labeled with session identifiers*, or labeled affine function in short, as pairs of the form $\mathsf{f} = (f, \{sid_j\}_{j=1}^{\ell})$, where $f$ is an affine function on $\mathbb{F}^{\ell}$. Let $\mathbf{x} = (x_{sid})_{sid}$ denotes all the inputs chosen by receiver and use convenient notation $\mathsf{f}(\mathbf{x})$ to denote $\mathsf{f}(\mathbf{x}) := f(x_{sid_1}, \dots, x_{sid_\ell})$.

A $(t+1)$-node branching program can be formalized as a mapping $\mathsf{G}$ from the input vector to matrix in $\mathbb{F}^{t\times t}$ such that

$$\mathsf{G}(\mathbf{x}) := \begin{bmatrix} \mathsf{g}_{1,1}(\mathbf{x}) & \cdots & & \mathsf{g}_{1,t}(\mathbf{x}) \\ -1 & \ddots & & \\ & \ddots & \ddots & \vdots \\ & & -1 & \mathsf{g}_{t,t}(\mathbf{x}) \end{bmatrix}$$

where each of $\mathsf{g}_{1,1}, \dots, \mathsf{g}_{t,t}$ is a labeled affine function. Then $\mathsf{G}$ is explicitly labeled with a sequence of session identifiers. The value of the branching program on input $\mathbf{x}$ is defined as $\det \mathsf{G}(\mathbf{x})$.

Define a square matrix to be *canonical* if and only if it is of the form

$$\begin{bmatrix} v_{1,1} & \cdots & & v_{1,t} \\ -1 & \ddots & & \\ & \ddots & \ddots & \vdots \\ & & -1 & v_{t,t} \end{bmatrix}$$

Clearly, $\mathsf{G}(\mathbf{x})$ is canonical for all branching program $\mathsf{G}$ and vector $\mathbf{x}$. For any canonical matrix $M$, define its top row as its head, denoted by $\mathsf{head}_M$; define the rest part of the matrix as its body, denoted by $\mathsf{Body}_M$. The body of a canonical matrix is always full row-rank. Therefore, a canonical matrix is not full rank if and only if its head can be spanned by its body. More precisely, for a canonical matrix $M \in \mathbb{F}^{t\times t}$, matrix $M$ is non-invertible if and only if there exists an (unique) vector $\mathsf{key}_M \in \mathbb{F}^{t-1}$ such that $\mathsf{head}_M = \mathsf{key}_M^T \cdot \mathsf{Body}_M$.

As a first trial to disclose a secret $s \in \mathbb{F}$ to the receiver iff $\det \mathsf{G}(\mathbf{x}) = 0$ for some public branching program $\mathsf{G}$, the sender samples random $\mathbf{r} \in \mathbb{F}^t$ and sends $\mathbf{z} := \mathsf{G}(\mathbf{x})\mathbf{r} + (-s, 0, \dots, 0)$ to the receiver via $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE}}$. From a (potentially malicious) receiver's view, if $\det \mathsf{G}(\mathbf{x}) \neq 0$, then $\mathbf{z}$ leaks no information about $s$ as it's one-time padded by $\mathbf{r}$; if $\det \mathsf{G}(\mathbf{x}) = 0$, then the receiver can recover secret $s$ by

$$\langle (-1, \mathsf{key}_{\mathsf{G}(\mathbf{x})}), \mathbf{z} \rangle = \langle (-1, \mathsf{key}_{\mathsf{G}(\mathbf{x})}), \mathsf{G}(\mathbf{x})\mathbf{r} \rangle + \langle (-1, \mathsf{key}_{\mathsf{G}(\mathbf{x})}), (-s, 0, \dots, 0) \rangle = s.$$

Such a protocol is secure against malicious receiver, but insecure against malicious sender.

In order to protect against malicious sender, upon receiving $(\mathtt{CDS}, \mathsf{G})$ in choice phase, the receiver samples $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{F}^{t-1}$ satisfying $\mathbf{k}_1 + \mathbf{k}_2 = \mathsf{key}_{\mathsf{G}(\mathbf{x})}$. Then the receiver computes canonical matrix $M_1, M_2$ such that

$$\mathsf{Body}_{M_1} = \mathsf{Body}_{M_2} = \mathsf{Body}_{\mathsf{G}(\mathbf{x})}, \quad \mathsf{head}_{M_1} = \mathbf{k}_1^T \cdot \mathsf{Body}_{\mathsf{G}(\mathbf{x})}, \quad \mathsf{head}_{M_2} = \mathbf{k}_2^T \cdot \mathsf{Body}_{\mathsf{G}(\mathbf{x})},$$

and inputs all the values in $M_1, M_2$ into underlying $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE}}$. Then when the sample is asked to disclose a secret $s \in \mathbb{F}$ conditioned on $\det \mathsf{G}(\mathbf{x}) = 0$, the sampler additively shares the secret as $s = s_1 + s_2 + s_3$, discloses $s_3$ conditioned on affine constraints

$$\mathsf{Body}_{M_1} = \mathsf{Body}_{M_2} = \mathsf{Body}_{\mathsf{G}(\mathbf{x})}, \quad \mathsf{head}_{M_1} + \mathsf{head}_{M_2} = \mathsf{head}_{\mathsf{G}(\mathbf{x})}.$$

Next, samples $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{F}^t$, and sends $\mathbf{z}_1 := M_1\mathbf{r}_1 + (-s_1, 0, \ldots, 0), \mathbf{z}_2 := M_2\mathbf{r}_2 + (-s_2, 0, \ldots, 0)$ to receiver via $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$. Then the honest receiver recovers $s_1, s_2$ by $\langle(-1, \mathbf{k}_j), \mathbf{z}_j\rangle = s_j$. So far, the protocol is still insecure against malicious sender. A malicious sender can pick arbitrary $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{F}^t$ and sends $\mathbf{z}_j^* := M_j\mathbf{r}_j + \mathbf{b}_j$ to the receiver via $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$. In such case, the honest receiver will recovers $s_1^*, s_2^*$ such that $\langle(-1, \mathbf{k}_j), \mathbf{b}_j\rangle = s_j^*$. Notice that either the sender picked $\mathbf{b}_j = (-s_j, 0, \ldots, 0)$ for some $s_j \in \mathbb{F}$ so that $s_j^* = s_j$, or $s_j^*$ is uniformly random due to the randomness of $\mathbf{k}_j$. We've resolved a similar situation using hash-and-check trick, and we will apply the trick once more.

**Theorem 5.4.** *The protocol $\Pi_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ (Figure 11) implements the functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}(\mathcal{C})$ in the $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$-hybrid model, when $\mathcal{C}$ is the class of all arithmetic branching programs.*

*Proof.* First consider affine constraints. Whenever the CDS is conditioning on affine constraint $f, \{sid_j\}_{j=1}^{\ell}$, let $c_0, c_1, \ldots, c_\ell$ be the coefficients of $f$. When the sender is semi-honest, the (potentially malicious) receiver gets $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid, c_j)_{j=1}^{\ell}, z, i)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ such that

$$z = \sum_{j=1}^{\ell} c_{cid} c_j x_{sid_j} + (rc_0 + s) = \sum_{j=1}^{\ell} rc_j x_{sid_j} + rc_0 + s = s.$$

This ensures both correctness and security against malicious receiver.

Assume a (potentially malicious) sender outputs $(cid, r)$ and $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid, c_j)_{j=1}^{\ell}, b, i)$. Then a honest receiver will output $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, z, i)$ such that

$$z = \sum_{j=1}^{\ell} c_{cid} c_j x_{sid_j} + b = \sum_{j=1}^{\ell} rc_j x_{sid_j} + b = b - rc_0.$$

Therefore, the sender can be simulated by inputing $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, b - rc_0, i)$ to the ideal functionality.

Then consider more general arithmetic constraints modeled by branching programs. Let $\mathsf{G}$ be the corresponding $(t+1)$-node branching program.

For completeness, assume $\det \mathsf{G}(\mathbf{x}) = 0$ and both sender and receiver are honest. The receiver chooses canonical matrixs $M_1, M_2$ such that

$$\begin{aligned} \mathsf{Body}_{M_1} = \mathsf{Body}_{M_2} &= \mathsf{Body}_{\mathsf{G}(\mathbf{x})}, && (2) \\ \mathsf{head}_{M_1} + \mathsf{head}_{M_2} &= \mathsf{head}_{\mathsf{G}(\mathbf{x})}, && (3) \\ \det(M_1) = \det(M_2) &= 0. && (4) \end{aligned}$$

As both (2) and (3) are satisfied, the receiver gets $(s_{h,w})_{1 \le h \le w \le t}$ from recursive CDS calls. The receiver also receives $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, u_1, u_2$ such that

$$\begin{aligned} &\mathbf{z}_1 = M_1\mathbf{r}_1 + (-s_1, 0, \ldots, 0), \quad &&\mathbf{z}_2 = M_2\mathbf{r}_2 + (-s_2, 0, \ldots, 0), \\ &\mathbf{z}_3 = M_1\mathbf{r}_3 + (-s_3, 0, \ldots, 0), \quad &&\mathbf{z}_4 = M_2\mathbf{r}_4 + (-s_4, 0, \ldots, 0), \\ &u_1 = s_1\gamma + s_3, &&u_2 = s_2\gamma + s_4. \end{aligned} \qquad (5)$$

where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4 \in \mathbb{F}^t$, $s_3, s_4 \in \mathbb{F}$ are sampled by sender, $s_1, s_2$ are parts of the additive secret sharing of $s$ and $\gamma$ is sampled by receiver. The receiver then computes

$$\begin{aligned} &\hat{s}_1 := \langle(-1, \mathbf{k}_1), \mathbf{z}_1\rangle = \langle(-1, \mathsf{key}_{M_1}), \mathbf{z}_1\rangle = s_1, \quad &&\hat{s}_2 := \langle(-1, \mathbf{k}_2), \mathbf{z}_2\rangle = \langle(-1, \mathsf{key}_{M_2}), \mathbf{z}_2\rangle = s_2, \\ &\hat{s}_3 := \langle(-1, \mathbf{k}_1), \mathbf{z}_3\rangle = \langle(-1, \mathsf{key}_{M_1}), \mathbf{z}_3\rangle = s_3, \quad &&\hat{s}_4 := \langle(-1, \mathbf{k}_2), \mathbf{z}_4\rangle = \langle(-1, \mathsf{key}_{M_2}), \mathbf{z}_4\rangle = s_4. \end{aligned}$$

**Implementing $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$: Protocol $\Pi_{\frac{1}{2}\texttt{repOLE,CDS}}$**

When $\mathfrak{C}$ is the class of all arithmetic branching programs.

## Choice phase:

The receiver, at the outset, samples $\gamma \in \mathbb{F}$ and sends the pair $(sid_{\mathsf{gamma}}, \gamma)$ to $\mathcal{F}_{\texttt{rOLE}}$. (If the underlaying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ is implemented by $\Pi_{\frac{1}{2}\texttt{repOLE}}$, it's fine to reuse $sid_{\mathsf{gamma}}, \gamma$ sampled by $\Pi_{\frac{1}{2}\texttt{repOLE}}$.) Then, upon receiving any $(sid, x)$ from the environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

## Choice phase, part (CDS):

Receiver: Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from environment,

- If $f$ is an affine function, ignore this input.
- Otherwise, let $\mathsf{G}$ be the corresponding $(t+1)$-node branching program. Compute canonical matrix $M = \mathsf{G}(\mathbf{x})$, i.e. if the $(h, w)$ location of $\mathsf{G}$ is $\mathsf{g}_{h,w} = (g, (sid_j)_{j=1}^{\ell'})$, let $M[h, w] = \mathsf{g}_{h,w}(\mathbf{x}) = g(x_{sid_1}, \ldots, x_{sid_{\ell'}})$. Additively share $\mathsf{key}_M$ as $\mathsf{key}_M = \mathbf{k}_1 + \mathbf{k}_2$, and store $(\texttt{CDS}, \mathsf{G}, \mathbf{k}_1, \mathbf{k}_2)$. Compute matrixes $M_1, M_2$ such that $\mathsf{Body}_{M_1} = \mathsf{Body}_{M_2} = \mathsf{Body}_M$, $\mathsf{head}_{M_1} = \mathbf{k}_1^T \cdot \mathsf{Body}_M$, $\mathsf{head}_{M_2} = \mathbf{k}_2^T \cdot \mathsf{Body}_M$. Send $(\mathsf{G}\|0\|h\|w, M[h, w])$ for all $2 \le h \le w \le t$, $(\mathsf{G}\|1\|1\|w, M_1[1, w])$ for all $1 \le w \le t$, $(\mathsf{G}\|2\|1\|w, M_2[1, w])$ for all $1 \le w \le t$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

## Send phase (choose coefficients, half replicated OLE):

Sender: Upon receiving any $(cid, c)$ from the environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

Sender: Upon receiving $(\frac{1}{2}\texttt{repOLE}, sid, cid, b, i)$ from environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

Receiver: Upon receiving $(\frac{1}{2}\texttt{repOLE}, sid, cid, z, i)$ from underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$, outputs it.

## Send phase, part (CDS):

Sender: Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from environment,

- If $f$ is an affine function: Let $c_0, c_1, \ldots, c_\ell$ be the coefficients that $f(v_1, \ldots, v_\ell) = c_0 + \sum_j c_j v_j$. The sender picks a new coefficient identifier $cid$ and sends $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, i, cid)$ to receiver, samples random $r \in \mathbb{F}$, and inputs $(cid, r)$, $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid, c_j)_{j=1}^{\ell}, rc_0 + s, i)$ to underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.
- (outlined) Otherwise, let $\mathsf{G}$ be the corresponding $(t+1)$-node branching program. Additively share $s$ as $s = s_1 + s_2 + \sum_{1 \le h \le w \le t} s_{h,w}$. For $2 \le h \le w \le t$, disclose $s_{h,w}$ to the receiver conditioned on $\mathsf{g}_{h,w}(\mathbf{x}) = M[h, w]$ via a recursive call to the CDS functionality for affine constraints. For $1 \le w \le t$, disclose $s_{1,w}$ to the receiver conditioned on $\mathsf{g}_{1,w}(\mathbf{x}) = M_1[1, w] + M_2[1, w]$ via a recursive call to the CDS functionality for affine constraints. Sample random $s_3, s_4 \in \mathbb{F}$, input $(sid_{\mathsf{gamma}}, s_1, s_3, \mathsf{G}\|i\|1)$, $(sid_{\mathsf{gamma}}, s_2, s_4, \mathsf{G}\|i\|2)$ to $\mathcal{F}_{\texttt{rOLE}}$. Sample random $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4 \in \mathbb{F}^t$, sends $\mathbf{z}_1 := M_1\mathbf{r}_1 + (-s_1, 0, \ldots, 0)$, $\mathbf{z}_2 := M_2\mathbf{r}_2 + (-s_2, 0, \ldots, 0)$, $\mathbf{z}_3 := M_1\mathbf{r}_3 + (-s_3, 0, \ldots, 0)$, $\mathbf{z}_4 := M_2\mathbf{r}_4 + (-s_4, 0, \ldots, 0)$ to the receiver via $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$.

Receiver: Upon receiving messages from sender corresponds to CDS operation indexed by $(f, \{sid_j\}_{j=1}^{\ell}, i)$,

- If $f(x_{sid_1}, \ldots, x_{sid_\ell}) \neq 0$, outputs $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \perp, i)$.
- If $f$ is an affine function, when $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, i, cid)$, $(\frac{1}{2}\texttt{repOLE}, (sid_j, cid, c_j)_{j=1}^{\ell}, z, i)$ was received from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$, outputs $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, z, i)$.
- (outlined) Otherwise, let $\mathsf{G}$ be the corresponding $(t+1)$-node branching program, verify that there is stored $(\texttt{CDS}, \mathsf{G}, \mathbf{k}_1, \mathbf{k}_2)$. $(s_{h,w})_{1 \le h \le w \le t}$ is recovered from recursive calls. $(sid_{\mathsf{gamma}}, u_1, \mathsf{G}\|i\|2)$, $(sid_{\mathsf{gamma}}, u_2, \mathsf{G}\|i\|1)$ are received from $\mathcal{F}_{\texttt{rOLE}}$, and $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$. Compute $\hat{s}_1 = \langle(-1, \mathbf{k}_1), \mathbf{z}_1\rangle$, $\hat{s}_2 = \langle(-1, \mathbf{k}_2), \mathbf{z}_2\rangle$, $\hat{s}_3 = \langle(-1, \mathbf{k}_1), \mathbf{z}_3\rangle$, $\hat{s}_4 = \langle(-1, \mathbf{k}_2), \mathbf{z}_4\rangle$, and abort if either $\hat{s}_1\gamma + \hat{s}_3 \neq u_1$ or $\hat{s}_2\gamma + \hat{s}_4 \neq u_2$. Otherwise, output $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \hat{s}_1 + \hat{s}_2 + \sum_{1 \le h \le w \le t} s_{h,w}, i)$.

Figure 11: Protocol $\Pi_{\frac{1}{2}\texttt{repOLE,CDS}}$ for half-replicated OLE allowing CDS in the $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$-hybrid model

Thus the receiver will not abort as $\hat{s}_1\gamma + \hat{s}_3 = s_1\gamma + s_3 = u_1$, $\hat{s}_2\gamma + \hat{s}_4 = s_2\gamma + s_4 = u_2$. The receiver recovers the secret $s$ by $\hat{s}_1 + \hat{s}_2 + \sum_{1\le h\le w\le t} s_{h,w} = s_1 + s_2 + \sum_{1\le h\le w\le t} s_{h,w} = s$.

For security against (potentially malicious) receiver, assume $\mathsf{G}(\mathbf{x}) \neq 0$ i.e. the condition is not satisfied. Because otherwise the simulator gets the secret $s$ from ideal functionality and can simulate by emulating an honest sender. The malicious receiver chooses canonical matrixs $M_1, M_2$ and outputs them as extra input.

- If either (2) or (3) is not satisfied by $M_1, M_2$, let $(h, w)$ be one location where $M_1, M_2$ don't satisfy (2), (3). Then receiver's view leaks no information about $s$ because $s$ is hidden by one-time pad $s_{h,w}$ and $s_{h,w}$ is hidden by a recursive CDS call. For this reason, the receiver's view is simluatable by emulating the honest sender on any $s$.

- Otherwise $M_1, M_2$ satisfy (2) and (3), then $\det M_1 + \det M_2 = \det \mathsf{G}(\mathbf{x}) \neq 0$. Thus one of $\det M_1$, $\det M_2$ is non-zero. W.l.o.g. assume $\det M_1 \neq 0$. In real world, receiver receives $(s_{h,w})_{1\le h\le w\le t}$ from recursive CDS calls, also receives $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, u_1, u_2$ such that (5) holds, where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4 \in \mathbb{F}^t$, $s_3, s_4 \in \mathbb{F}$ are sampled by sender and $s_1, s_2$ are parts of the additive secret sharing of $s$. Then $(\mathbf{z}_1, \mathbf{z}_3, u_1)$ hides $s_1$ as it's one-time padded by $(\mathbf{r}_1, \mathbf{r}_2, s_3)$. Thus receiver's view, together with hidden variable $s_2, s_4$, can be simulated as the following:
  - The joint distribution $s_2, (s_{h,w})_{1\le h\le w\le t}$ is uniform random because they are parts of a additive secret sharing.
  - The joint distribution $\mathbf{z}_1, \mathbf{z}_3, u_1$ is uniform random as it's one-time padded by $(\mathbf{r}_1, \mathbf{r}_2, s_3)$.
  - Given, $s_2, (s_{h,w})_{1\le h\le w\le t}, \mathbf{z}_1, \mathbf{z}_3, u_1$, the remaining $\mathbf{z}_2, \mathbf{z}_4, u_2$ are determined as in (5).

For security against (potentially malicious) sender, assume $\det \mathsf{G}(\mathbf{x}) = 0$, as otherwise the honest receiver will always output $\perp$. The sender can choose numbers $s_1, s_2, s_3, s_4, (s_{h,w})_{1\le h\le w\le t} \in \mathbb{F}$ and vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4 \in \mathbb{F}^t$ so that the honest receiver gets $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, u_1, u_2$ and $(s_{h,w})_{1\le h\le w\le t}$ where

$$\begin{aligned} \mathbf{z}_1 &= M_1\mathbf{r}_1 + \mathbf{b}_1, & \mathbf{z}_2 &= M_2\mathbf{r}_2 + \mathbf{b}_2, \\ \mathbf{z}_3 &= M_1\mathbf{r}_3 + \mathbf{b}_3, & \mathbf{z}_4 &= M_2\mathbf{r}_4 + \mathbf{b}_4, \\ u_1 &= s_1\gamma + s_3, & u_2 &= s_2\gamma + s_4, \end{aligned} \tag{6}$$

$\gamma \in \mathbb{F}$ is sampled by the receiver from uniform and canonical matrix $M_1, M_2$ are sampled by the receiver such that each of $\mathbf{k}_1 := \mathsf{key}_{M_1}, \mathbf{k}_2 := \mathsf{key}_{M_2}$ is unifromly random. (The joint distribution of $\mathbf{k}_1, \mathbf{k}_2$ forms a additive secure sharing of $\mathsf{key}_{\mathsf{G}(\mathbf{x})}$.)

$$\begin{aligned} \hat{s}_1 &:= \langle(-1, \mathbf{k}_1), \mathbf{z}_1\rangle = \langle(-1, \mathbf{k}_1), \mathbf{b}_1\rangle, & \hat{s}_2 &:= \langle(-1, \mathbf{k}_2), \mathbf{z}_2\rangle = \langle(-1, \mathbf{k}_2), \mathbf{b}_2\rangle, \\ \hat{s}_3 &:= \langle(-1, \mathbf{k}_1), \mathbf{z}_3\rangle = \langle(-1, \mathbf{k}_1), \mathbf{b}_3\rangle, & \hat{s}_4 &:= \langle(-1, \mathbf{k}_2), \mathbf{z}_4\rangle = \langle(-1, \mathbf{k}_2), \mathbf{b}_4\rangle. \end{aligned}$$

- If there exists $y \in \{1, 2, 3, 4\}$ such that $\mathbf{z}_y \neq (-s_y, 0, \ldots, 0)$. W.o.l.g. assume one of such $y$ lays in $\{1, 3\}$. Then $\hat{s}_y \neq s_y$ with probabiliy at least $1 - \frac{1}{|\mathbb{F}|}$, due to the randomness of $\mathbf{k}_1$. Conditioned on $\hat{s}_y \neq s_y$, the receiver will found $\hat{s}_1\gamma + \hat{s}_3 \neq s_1\gamma + s_3 = u_1$ with probabiliy at least $1 - \frac{1}{|\mathbb{F}|}$, due to the randomness of $\gamma$. In summary, the receiver will abort with probabiliy at least $1 - \frac{2}{|\mathbb{F}|}$.

- Otherwise, $\mathbf{z}_y = (-s_y, 0, \ldots, 0)$ for all $y \in \{1, 2, 3, 4\}$. Then $\hat{s}_y = s_y$ for all $y \in \{1, 2, 3, 4\}$. The receiver will not abort and will output reconstructed secret $\hat{s}_1 + \hat{s}_2 + \sum_{1\le h\le w\le t} s_{h,w} = s_1 + s_2 + \sum_{1\le h\le w\le t} s_{h,w}$. Thus this sender can be simulated as sending the condition together with secret $s = s_1 + s_2 + \sum_{1\le h\le w\le t} s_{h,w}$ to the ideal functionality. □

### 5.3.1 Syntactic sugar: Conditional Disclosure of Coefficients

To simplify the construction of rNIZK protocol, it's useful to redefine $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ slightly by adding some syntactic sugar. Right now, the sender can disclose a secret in $\mathbb{F}$ to the receiver if the receiver's input satisfies a chosen predicate. An easy extension allow the sender to disclose a coefficient which the sender have committed to, instead of an arbitrarily chosen field element (Figure 12).

**Functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{sugar-CDS}}$**

Parametrized by a finite field $\mathbb{F}$, and $\mathfrak{C}$ a class of functions.

**Choice phase:** the same as $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ in Figure 10.

**Send phase (choose coefficients, half replicated OLE):** the same as $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ in Figure 10.

**Send phase (CDS):**

- Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, cid)$ from Sam where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from Rachel and stored input $(cid, a)$ from Sam; else ignore that message. Next, record $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, cid)$, send $(f, \{sid_j\}_{j=1}^{\ell}, cid, \texttt{send})$ to the adversary, and ignore any further inputs from Sam with the same $(f, \{sid_j\}_{j=1}^{\ell}, cid)$.
- Upon receiving a message $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, cid, \texttt{Delivery})$ from the adversary, verify that there are stored input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, cid)$, $(cid, a)$ from Sam and $(sid_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, send to Rachel $(f, \{sid_j\}_{j=1}^{\ell}, cid, a)$ if $f(x_1, \ldots, x_\ell) = 0$ otherwise $(f, \{sid_j\}_{j=1}^{\ell}, cid, \perp)$, and ignore further messages $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, cid, \texttt{Delivery})$ from the adversary with the same $(f, \{sid_j\}_{j=1}^{\ell}, cid)$.

Figure 12: ideal functionality of $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ with syntactic sugar

There is a simple protocol that implements such an enhanced functionality in $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$-hybrid model and it's UC-secure. The main idea is to disclose a pair $(a_{cid}, b)$ conditionally, where $b \in \mathbb{F}$ is sampled by the sender. Then convince the receiver that $a_{cid}$ is sent by evaluating $a_{cid}\gamma + b$ obliviously using half-replicated OLE.

- In choice phase: The receiver, at the outset, samples $\gamma \in \mathbb{F}$ and sends the pair $(sid_{\textsf{gamma}}, \gamma)$ to $\mathcal{F}_{\texttt{rOLE}}$. Then forward all message from environment to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$. If the underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ is implemented by $\Pi_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$, the sender can reuse the $(sid_{\textsf{gamma}}, \gamma)$ picked by $\Pi_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$.
- Upon receiving $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, cid)$ from the environment, the sender samples random $b \in \mathbb{F}$, chooses a new index $i$ and inputs $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, a_{cid}, i)$, $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, b, i+1)$ and $(sid_{\textsf{gamma}}, cid, b, i)$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$.
- Upon receiving $(sid_{\textsf{gamma}}, cid, z, i)$, $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \hat{s}_1, i)$, $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \hat{s}_2, i+1)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$, the receiver outputs $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, cid, \hat{s}_1)$ if $\hat{s}_1\gamma + \hat{s}_1 = z$; and aborts otherwise.

**Functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$**

Parametrized by a finite field $\mathbb{F}$, and $\mathfrak{C}, \mathfrak{F}$ classes of functions.

**Choice phase, part I (the same as Figure 6):** Upon receiving input $(\mathit{sid}, x)$ from the receiver where $x \in \mathbb{F}$ and $\mathit{sid} \in \mathbb{N}$ is a session identifier, store $(\mathit{sid}, x)$, send $(\mathit{sid}, \texttt{input-init})$ to the adversary and ignore any further input from the receiver with the same $\mathit{sid}$.

**Choice phase, part II (CDS, the same as Figure 10):** Upon receiving input $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell})$ from Rachel where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(\mathit{sid}_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, record $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell})$, send $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, \texttt{sent})$ to the adversary, and ignore any further inputs from Rachel with the same $(f, \{\mathit{sid}_j\}_{j=1}^{\ell})$.

**Send phase (choose coefficients, the same as Figure 6):** Upon receiving input $(\mathit{cid}, a)$ from the sender where $a \in \mathbb{F}$ and $\mathit{cid} \in \mathbb{N}$ is a coefficient identifier, store $(\mathit{cid}, a)$, send $(\mathit{cid}, \texttt{coeff-init})$ to the adversary and ignore any further input from Sam with the same $\mathit{cid}$.

**Send phase, part I (half replicated OLE, the same as Figure 6):**

- Upon receiving input $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b)$ from Sam, record $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b)$, send $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b, \texttt{sent})$ to the adversary, and ignore any further inputs from Sam with the same session identifier $\mathit{sid}$ and same coefficient identifiers $\mathit{cid}_a, \mathit{cid}_b$.
- Upon receiving a message $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b, \texttt{Delivery})$ from the adversary, verify that there are stored inputs $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_b, \mathit{cid}_b), (\mathit{cid}_a, a), (\mathit{cid}_b, b)$ from Sam and $(\mathit{sid}, x)$ from Rachel; else ignore that message. Next, compute $z = ax + b$, send $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b, z)$ to Rachel, and ignore further messages $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b, \texttt{Delivery})$ from the adversary with the same session identifier $\mathit{sid}$ and same coefficient identifiers $\mathit{cid}_a, \mathit{cid}_b$.

**Send phase, part II (CDS, the same as Figure 10):**

- Upon receiving input $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, s, i)$ from Sam where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell})$ from Rachel; else ignore that message. Next, record $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, s, i)$, send $(f, \{\mathit{sid}_j\}_{j=1}^{\ell}, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(f, \{\mathit{sid}_j\}_{j=1}^{\ell}, i)$.
- Upon receiving a message $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary, verify that there are stored input $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, s, i)$ from Sam and $(\mathit{sid}_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, send to Rachel $(f, \{\mathit{sid}_j\}_{j=1}^{\ell}, s, i)$ if $f(x_1, \ldots, x_\ell) = 0$ otherwise $(f, \{\mathit{sid}_j\}_{j=1}^{\ell}, \perp, i)$, and ignore further messages $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary with the same $(f, \{\mathit{sid}_j\}_{j=1}^{\ell}, i)$.

**Send phase, part III (certified OLE):**

- Upon receiving input $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell})$ from Sam where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{F}$, record $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell})$, send $(f, \{\mathit{cid}_j\}_{j=1}^{\ell}, \texttt{send})$ to the adversary, and ignore any further inputs from Sam with the same function $(f, \{\mathit{cid}_j\}_{j=1}^{\ell})$.
- Upon receiving a message $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell}, \texttt{Delivery})$ from the adversary, verify that there are stored inputs $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell})$ from Sam and $(\mathit{cid}_j, a_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, compute $v = f(a_1, \ldots, a_\ell)$, send to Rachel $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell}, v)$, and ignore further messages $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell}, \texttt{Delivery})$ from the adversary with the same function $(f, \{\mathit{cid}_j\}_{j=1}^{\ell})$.

Figure 13: ideal functionality of half replicated, certified OLE

### 5.4 Embedding a Zero-Knowledge Proof System

The new feature introduce in certified OLE (Figure 13) is to allow the sender to prove that the coefficients it chose satisfy certain equations. For technical reason, the functionality is defined as: sender inputs $(f, \{cid_j\}_{j=1}^{\ell})$ to ideal functionality, where $f$ is an arithmetic function and $\{cid_j\}$ are coefficient identifiers corresponding to coefficients that the sender has already chosen; receiver gets $(f, \{cid_j\}_{j=1}^{\ell}, f(a_{cid_1}, \ldots, a_{cid_\ell}))$. But it's actually constructed and used as a proof system: the sender transforms the argument to prove into an equation $y = f(a_{cid_1}, \ldots, a_{cid_\ell})$ (or a system of equations), then send $y$ and convinces the receiver that the coefficients satisfies $f(a_{cid_1}, \ldots, a_{cid_\ell}) = y$.

By adding certificates as extra coefficients, it's sufficient that the sender can convince the receiver of two atomic types of arguments:

- Affine arguments $c_0 + \sum_{j=1}^{\ell} c_j a_{cid_j} = 0$, specified by constants $c_0, c_1, \ldots, c_\ell \in \mathbb{F}$ and coefficient identifiers $cid_1, \ldots, cid_\ell$.
- Multiplicative arguments: $a_{cid_1} a_{cid_2} = a_{cid_3}$, specified by three coefficient identifiers $cid_1, cid_2, cid_3$.

In order to convince the receiver an satisfying affine argument $c_0 + \sum_{j=1}^{\ell} c_j a_{cid_j} = 0$, the receiver should have sampled a random input $\mu$. The sender sends $(\sum_{j=1}^{\ell} c_j a_{cid_j})\mu$ to the receiver using (the syntactic sugar of) half-replicated OLE functionality. The receiver expects to receive $-c_0\mu$.

In order to convince the receiver a multiplicative argument $a_{cid_1} a_{cid_2} = a_{cid_3}$, the receiver should have sampled random input $\mu, \nu, \xi$ satisfying $\mu\nu = \xi$. The sender then sends $a_{cid_1}\mu + b_1, a_{cid_2}\nu + b_2, a_{cid_3}\xi + b_3, b_2\mu + b_4, a_1\nu + b_5$ using half-replicated OLE functionality for appropriate $b_1, \ldots, b_5$ such that

$$(a_{cid_1}\mu + b_1)(a_{cid_2}\nu + b_2) = (a_{cid_3}\xi + b_3) + (b_2\mu + b_4) + (a_1\nu + b_5)$$

for any $\mu, \nu, \xi$ that $\mu\nu = \xi$. And to protect against malicious receiver, sender encrypts these evaluations using one-time pad and discloses the pad conditioning on $\mu\nu = \xi$.

**Theorem 5.5.** *The protocol $\Pi_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ (Figure 14) implements the functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ in the $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$-hybrid model.*

*Proof.* It's sufficient to construct proof system proving arguments of the two atomic operations. First consider the protocol verifying an affine argument $c_0 + \sum_{j=1}^{\ell} c_j a_{cid_j} = 0$:

When the sender is honest and the argument holds, the receiver will receive $(\frac{1}{2}\texttt{repOLE}, (sid_{\mathsf{mu}}, cid_j, c_j)_{j=1}^{\ell}, z)$ from underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ such that $z = \sum_{j=1}^{\ell} c_j a_{cid_j}\mu = -c_0\mu$. For completeness, $z = -c_0\mu$ ensures an honest receiver will accept. For security against malicious receiver, the view of the (potentially malicious) receiver can be simulated by public knowledge and the receiver's randomness $\mu$.

When the receiver is honest, a (potentially malicious) sender outputs $(\frac{1}{2}\texttt{repOLE}, (sid_{\mathsf{mu}}, cid_j, c_j)_{j=1}^{\ell}, b)$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ for some $b \in \mathbb{F}$. Then the honest receiver gets $(\frac{1}{2}\texttt{repOLE}, (sid_{\mathsf{mu}}, cid_j, c_j)_{j=1}^{\ell}, z)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS}}$ such that $z = \sum_{j=1}^{\ell} c_j a_{cid_j}\mu + b$. Unless the argument $c_0 + \sum_{j=1}^{\ell} c_j a_{cid_j} = 0$ is satisfied and $b = 0$, the receiver would find $z \neq -c_0\mu$ with probability at least $1 - \frac{1}{|\mathbb{F}|}$ as

$$c_0\mu + z = (c_0 + \sum_{j=1}^{\ell} c_j a_{cid_j})\mu + b.$$

**Implementing $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS},\mathtt{proof}}$: Protocol $\Pi_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS},\mathtt{proof}}$**

### Choice phase:

The receiver, at the outset, samples $\mu, \nu \in \mathbb{F}$ compute $\xi = \mu\nu$ and sends the pairs $(sid_{\mathsf{mu}}, \mu)$, $(sid_{\mathsf{nu}}, \nu)$, $(sid_{\mathsf{xi}}, \xi)$ to $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$, let function $f_{\mathrm{mult}}(v_1, v_2, v_3) := v_1 v_2 - v_3$ and send tuple $(\mathtt{CDS}, f_{\mathrm{mult}}, (sid_{\mathsf{mu}}, sid_{\mathsf{nu}}, sid_{\mathsf{xi}}))$ to $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$. Then, upon receiving any $(sid, x)$ or $(\mathtt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from the environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$.

### Send phase (choose coefficients, half replicated OLE, CDS):

Sender: Upon receiving any $(cid, c)$ or $(\frac{1}{2}\mathtt{repOLE}, sid, cid, b, i)$ or $(\mathtt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from the environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$.

Receiver: Upon receiving $(\frac{1}{2}\mathtt{repOLE}, sid, cid, z, i)$, $(\mathtt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from underlying $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$, output it.

**Send phase (certified OLE):** All operations are decomposed into proving affine arguments and multiplicative arguments.

*For any affine argument $c_0 + \sum_{j=1}^{\ell} c_j a_{cid_j} = 0$:* The sender inputs $(\frac{1}{2}\mathtt{repOLE}, (sid_{\mathsf{mu}}, cid_j, c_j)_{j=1}^{\ell}, 0)$ to underlying $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$.

The receiver, upon receiving $(\frac{1}{2}\mathtt{repOLE}, (sid_{\mathsf{mu}}, cid_j, c_j)_{j=1}^{\ell}, z)$ from underlying $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$, accepts the proof if $z = -c_0\mu$, otherwise rejects (aborts).

*For any multiplicative argument $a_{cid_1} a_{cid_2} = a_{cid_3}$:* The sender samples random $b_1, b_2, b_3, b_4, b_5 \in \mathbb{F}$, picks two new coefficient identifiers $cid_4 = cid_1 \| cid_2 \| cid_3 \| 2$and $cid_5 = cid_1 \| cid_2 \| cid_3 \| 1$,computes $s = b_1 b_2 - b_3 - b_4 - b_5$, and inputs $(cid_4, b_2 a_{cid_1}), (cid_5, b_1 a_{cid_2})$,

$$(\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{mu}}, cid_1, b_1), (\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{nu}}, cid_2, b_2), (\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{xi}}, cid_3, b_3),$$
$$(\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{mu}}, cid_4, b_4), (\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{nu}}, cid_5, b_5), (\mathtt{CDS}, f_{\mathrm{mult}}, (sid_{\mathsf{mu}}, sid_{\mathsf{nu}}, sid_{\mathsf{xi}}), s)$$

to $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$.
The receiver, upon receiving

$$(\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{mu}}, cid_1, z_1), (\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{nu}}, cid_2, z_2), (\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{xi}}, cid_3, z_3),$$
$$(\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{mu}}, cid_4, z_4), (\tfrac{1}{2}\mathtt{repOLE}, sid_{\mathsf{nu}}, cid_5, z_5), (\mathtt{CDS}, f_{\mathrm{mult}}, (sid_{\mathsf{mu}}, sid_{\mathsf{nu}}, sid_{\mathsf{xi}}), s)$$

from $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$, accepts the proof if $s = z_1 z_2 - z_3 - z_4 - z_5$ otherwise rejects (aborts).

Figure 14: Protocol $\Pi_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS},\mathtt{proof}}$ for half-replicated, certified OLE in the $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE},\mathtt{CDS}}$-hybrid model

This ensures completeness and security against malicious sender.

Then consider the protocol verifying any multiplicative arguments $a_{cid_1}a_{cid_2} = a_{cid_3}$:

When sender is honest, the (potential malicious) receiver gets $z_1, \dots, z_5$ from messages from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$ that

$$z_1 = a_{cid_1}\mu + b_1, \quad z_2 = a_{cid_2}\nu + b_2, \quad z_3 = a_{cid_3}\xi + b_3, \quad z_4 = b_2 a_{cid_1}\mu + b_4, \quad z_5 = b_1 a_{cid_2}\nu + b_5,$$

where $\mu, \nu, \xi$ are chosen by the receiver and $b_1, \dots, b_5 \in \mathbb{F}$ are random numbers sampled by the sender. Moreover, the receiver receives $s$ from message from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$ that $s = b_1b_2 - b_3 - b_4 - b_5$ if $\mu\nu = \xi$ and $s = \perp$ otherwise. Thus we have

$$\begin{aligned}
&z_1z_2 - z_3 - z_4 - z_5 \\
=& (a_{cid_1}\mu + b_1)(a_{cid_2}\nu + b_2) - (a_{cid_3}\xi + b_3) - (b_2\mu + b_4) - (b_1\nu + b_5) \\
=& a_{cid_3}(\mu\nu - \xi) + b_1b_2 - b_3 - b_4 - b_5,
\end{aligned}$$

from which correctness directly follows. For security against malicious receiver, the (potential malicious) receiver's view can be simulated as

- the joint distribution of $z_1, \dots, z_5$ is uniformly random as it's one-time padded by $(b_1, \dots, b_5)$;
- $s$ is determined, as $s = z_1z_2 - z_3 - z_4 - z_5$ if $\mu\nu = \xi$ and $s = \perp$ otherwise.

When receiver is honest, it gets $z_1, \dots, z_5, s$ from messages received from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$ that

$$z_1 = a_{cid_1}\mu + b_1, \quad z_2 = a_{cid_2}\nu + b_2, \quad z_3 = a_{cid_3}\xi + b_3, \quad z_4 = b_2'\mu + b_4, \quad z_5 = b_1'\nu + b_5,$$

where $\mu, \nu$ are random numbers sampled by the receiver, $\xi = \mu\nu$ and $b_1, \dots, b_5, b_1', b_2' \in \mathbb{F}$ are chosen by the (potential malicious) sender. The honest receiver will accept if and only $z_1z_2 - z_3 - z_4 - z_5 = s$.

$$\begin{aligned}
&z_1z_2 - z_3 - z_4 - z_5 - s \\
=& (a_{cid_1}\mu + b_1)(a_{cid_2}\nu + b_2) - (a_{cid_3}\xi + b_3) - (b_2'\mu + b_4) - (b_1'\nu + b_5) - s \\
=& (a_{cid_1}a_{cid_2} - a_{cid_3})\mu\nu + (b_2a_{cid_1} - b_2')\mu + (b_1a_{cid_2} - b_1')\nu + (b_1b_2 - b_3 - b_4 - b_5 - s)
\end{aligned}$$

Unless $a_{cid_3} = a_{cid_1}a_{cid_2}$ is satisfied and the sender picks coefficients such that $b_2' = b_2a_{cid_1}$, $b_1' = b_1a_{cid_2}$, $s = b_1b_2 - b_3 - b_4 - b_5$, the receiver would find $z_1z_2 - z_3 - z_4 - z_5 \neq s$ with probability at least $1 - \frac{2}{|\mathbb{F}|}$ due the randomness of $\mu, \nu$. □

### 5.4.1 Constructing Reusable NIZK from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$

A reusable non-interactive zero-knowledge proof already lays inside of $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$. Thus constructing a rNIZK in $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$-hybrid model is simply wiring.

**Theorem 5.6.** *There exists a statistically secure rNIZK protocol in $\mathcal{F}_{rOLE}^{(\mathbb{F})}$-hybrid model. When the argument is presented as a boolean circuits or arithmetic circuits, proving the prover's knowledge of a satisfying assignment need $O(1)$ OLE queries per gate.*

Any 2-message implementation of $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$ can be used as a rNIZK proof system. The receiver is the verifier and the sender is the prover. The sender (prover) simply feeds the certificate into $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$ as coefficients, then $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$ will convince the receiver (verifier) that the sender knows one certificate that satisfies the predicate they care about.

*Proof of Theorem 5.6.* rNIZK protocol can be easily built in $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$-hybrid model. Then replace $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ with its secure implementation in $\mathcal{F}_{\texttt{rOLE}}$-hybrid model.

**Setup:** The receiver (verifier) initialize $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ without feeding any input.

**Prove:** In order to prove knowledge of a satisfying assignment $(y_1, \ldots, y_\ell)$ of $C(y_1, \ldots, y_\ell) = 0$, where $C$ is an arithmetic circuits with $m$ gates. Let $y_\ell, \ldots, y_{\ell+m}$ be the intermediate wires' values when evaluating circuit $C(y_1, \ldots, y_\ell)$. Let $G_t(y_1, \ldots, y_{\ell+m}) = 0$ be the equation representing if the $t$-th gate is correctly evaluated, i.e. if the $t$-th gate is an addition gate, then $G_t$ is an affine function; if the $t$-th gate is a multiplication gate, then $G_t(y_1, \ldots, y_{\ell+m})$ is of the form $y_{cid_1} y_{cid_2} - y_{cid_3}$. The sender (prover) commit $y_1, \ldots, y_{\ell+m}$ by choosing $y_1, \ldots, y_{\ell+m}$ as coefficients in $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$. Then sends $(\texttt{proof}, G_1), \ldots, (\texttt{proof}, G_m)$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$. (Coefficient identifiers are omitted in this description.)

**Verify:** Upon receiving $(\texttt{proof}, G_1, z_1), \ldots, (\texttt{proof}, G_m, z_m)$ from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$, the receiving (verifier) accepts if $z_1 = \ldots = z_m = 0$.

If the argument to prove is specified by a boolean circuit $C$. Note that $C$ can be transformed into an arithmetic cirucuit $C^*$ of proportionately the same size, such that $C^*$ computed the same function as $C$ in any field. Then proving the knowledge of binary $(y_1, \ldots, y_\ell)$ such that $C(y_1, \ldots, y_\ell) = 0$, can be transformed into proving the knowledge of $(y_1, \ldots, y_\ell) \in \mathbb{F}$ such that $C^*(y_1, \ldots, y_\ell) = y_1(y_1 - 1) = \ldots = y_\ell(y_\ell - 1) = 0$. □

## 5.5 Constructing Replicated OLE

Assume the receiver has chosen $x$ as one of its input by sending $(sid, x)$ to the $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$, and the sender has chosen coefficients $a, b$ by sending $(cid_1, a)$, $(cid_b, b)$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$. Now they are looking for a secure protocol that leaks $ax + b$ to the receiver. An naive trial is a bare half-replicated OLE, i.e. the sender sends $(\frac{1}{2}\texttt{repOLE}, sid, cid_a, b, i)$ to underlying functionality so that the receiver gets $(\frac{1}{2}\texttt{repOLE}, sid, cid_a, ax + b, i)$. But a malicious sender might deviates from the protocol by choosing some additive factor other than $b$.

In order to catch such misbehavior, the receiver samples random $\beta$ and sends $\beta$, $\beta x$ as two extra input. The sender is asked to send

$$ax + b \qquad \text{and} \qquad a(\beta x) + b\beta$$

via $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ so that the receiver can check whether the latter is $\beta$ times the former. If the sender deviates from the protocol, the receiver will get

$$ax + b_1 \qquad \text{and} \qquad a(\beta x) + b\beta + b_2$$

for some $b_1, b_2$ picked by the sender that $(b_1, b_2) \neq (b, 0)$. In such case, the latter doesn't equal $\beta$ times the former with high probability.

Then, the receiver can try to cheat, by choosing $\beta$ as the first extra input, and something other than $\beta x$ as the second extra input. Such cheating space is eliminated by a CDS protocol: sender encrypts the transmissions using one-time pad, and discloses the pad if and only if the receiver is honest, i.e. if the two extra input are $\beta$ and $\beta x$.

**Theorem 5.7.** *The protocol* $\Pi_{\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ *(Figure 16) implements the functionality* $\mathcal{F}_{\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ *in the* $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$*-hybrid model.*

**Functionality $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$**

Parametrized by a finite field $\mathbb{F}$, and $\mathfrak{C}, \mathfrak{F}$ classes of functions.

**Choice phase, part I (the same as Figure 6):** Upon receiving input $(sid, x)$ from the receiver where $x \in \mathbb{F}$ and $sid \in \mathbb{N}$ is a session identifier, store $(sid, x)$, send $(sid, \texttt{input-init})$ to the adversary and ignore any further input from the receiver with the same *sid*.

**Choice phase, part II (CDS, the same as Figure 10):** Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from Rachel where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(sid_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, record $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$, send $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \texttt{sent})$ to the adversary, and ignore any further inputs from Rachel with the same $(f, \{sid_j\}_{j=1}^{\ell})$.

**Send phase (choose coefficients, the same as Figure 6):** Upon receiving input $(cid, a)$ from the sender where $a \in \mathbb{F}$ and $cid \in \mathbb{N}$ is a coefficient identifier, store $(cid, a)$, send $(cid, \texttt{coeff-init})$ to the adversary and ignore any further input from Sam with the same *cid*.

**Send phase, part I (half replicated OLE):** Skipped because it's dominated by replicated OLE.

**Send phase, part II (CDS, the same as Figure 10):**

- Upon receiving input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from Sam where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{C}$, verify that there are stored input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell})$ from Rachel; else ignore that message. Next, record $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$, send $(f, \{sid_j\}_{j=1}^{\ell}, \texttt{send}, i)$ to the adversary, and ignore any further inputs from Sam with the same $(f, \{sid_j\}_{j=1}^{\ell}, i)$.
- Upon receiving a message $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary, verify that there are stored input $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, s, i)$ from Sam and $(sid_j, x_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, send to Rachel $(f, \{sid_j\}_{j=1}^{\ell}, s, i)$ if $f(x_1, \ldots, x_\ell) = 0$ otherwise $(f, \{sid_j\}_{j=1}^{\ell}, \perp, i)$, and ignore further messages $(\texttt{CDS}, f, \{sid_j\}_{j=1}^{\ell}, \texttt{Delivery}, i)$ from the adversary with the same $(f, \{sid_j\}_{j=1}^{\ell}, i)$.

**Send phase, part III (certified OLE, the same as Figure 13):**

- Upon receiving input $(\texttt{proof}, f, \{cid_j\}_{j=1}^{\ell})$ from Sam where $f : \mathbb{F}^{\ell} \to \mathbb{F}$ is a function in $\mathfrak{F}$, record $(\texttt{proof}, f, \{cid_j\}_{j=1}^{\ell})$, send $(f, \{cid_j\}_{j=1}^{\ell}, \texttt{send})$ to the adversary, and ignore any further inputs from Sam with the same function $(f, \{cid_j\}_{j=1}^{\ell})$.
- Upon receiving a message $(\texttt{proof}, f, \{cid_j\}_{j=1}^{\ell}, \texttt{Delivery})$ from the adversary, verify that there are stored inputs $(\texttt{proof}, f, \{cid_j\}_{j=1}^{\ell})$ from Sam and $(cid_j, a_j)$ from Rachel for all $j \in [\ell]$; else ignore that message. Next, compute $v = f(a_1, \ldots, a_\ell)$, send to Rachel $(\texttt{proof}, f, \{cid_j\}_{j=1}^{\ell}, v)$, and ignore further messages $(\texttt{proof}, f, \{cid_j\}_{j=1}^{\ell}, \texttt{Delivery})$ from the adversary with the same function $(f, \{cid_j\}_{j=1}^{\ell})$.

**Send phase, part IV (replicated OLE):**

- Upon receiving input $(\texttt{repOLE}, sid, cid_a, cid_b)$ from Sam, record $(\texttt{repOLE}, sid, cid_a, cid_b)$, send $(\texttt{repOLE}, sid, cid_a, cid_b, \texttt{sent})$ to the adversary, and ignore any further inputs from Sam with the same session identifier *sid* and same coefficient identifiers $cid_a, cid_b$.
- Upon receiving a message $(\texttt{repOLE}, sid, cid_a, cid_b, \texttt{Delivery})$ from the adversary, verify that there are stored inputs $(\texttt{repOLE}, sid, cid_b, cid_b), (cid_a, a), (cid_b, b)$ from Sam and $(sid, x)$ from Rachel; else ignore that message. Next, compute $z = ax + b$, send $(\texttt{repOLE}, sid, cid_a, cid_b, z)$ to Rachel, and ignore further messages $(\texttt{repOLE}, sid, cid_a, cid_b, \texttt{Delivery})$ from the adversary with the same session identifier *sid* and same coefficient identifiers $cid_a, cid_b$.

Figure 15: ideal functionality of replicated, certified OLE

**Implementing $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$: Protocol $\Pi_{\texttt{repOLE,CDS,proof}}$**

**Choice phase:** The receiver does to following:
At the outset, samples $\beta \in \mathbb{F}$ and sends the pair $(\mathit{sid}_{\texttt{beta}}, \beta)$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$.

Upon receiving $(\mathit{sid}, x)$ from the environment, $(\mathit{sid}, x)$, $(\mathit{sid}\|\texttt{beta}, \beta x)$, $(\texttt{CDS}, f_{\text{mult}}, (\mathit{sid}, \mathit{sid}_{\texttt{beta}}, \mathit{sid}\|\texttt{beta}))$ to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$, where $f_{\text{mult}}$ denotes the muliplication condition $f_{\text{mult}}(v_1, v_2, v_3) := v_1 v_2 - v_3$.

Upon receiving any $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell})$ from the environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$.

**Send phase (choose coefficients, CDS, certified OLE):**
Sender: Upon receiving any $(\mathit{cid}, c)$ or $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, s, i)$ or $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell})$ from the environment, forward it to underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$.

Receiver: Upon receiving $(\texttt{CDS}, f, \{\mathit{sid}_j\}_{j=1}^{\ell}, s, i)$, $(\texttt{proof}, f, \{\mathit{cid}_j\}_{j=1}^{\ell}, v)$ from underlying $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$, output it.

**Send phase (replicated OLE):**
Sender: Upon receiving input $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b)$ from environment, verify that there are stored $(\mathit{cid}_a, a), (\mathit{cid}_b, b)$ from environment. Pick a new index $i$, sample random $s \in \mathbb{F}$ and input

$$\begin{gathered}(\tfrac{1}{2}\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, b, i),\\ (\tfrac{1}{2}\texttt{repOLE}, ((\mathit{sid}\|\texttt{beta}, \mathit{cid}_a, 1), (\mathit{sid}_{\texttt{beta}}, \mathit{cid}_b, 1)), s, i),\\ (\texttt{CDS}, f_{\text{mult}}, (\mathit{sid}, \mathit{sid}_{\texttt{beta}}, \mathit{sid}\|\texttt{beta}), s, i)\end{gathered}$$

to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$.

Receiver: Upon receiving messages

$$\begin{gathered}(\tfrac{1}{2}\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, z, i),\\ (\tfrac{1}{2}\texttt{repOLE}, ((\mathit{sid}\|\texttt{beta}, \mathit{cid}_a, 1), (\mathit{sid}_{\texttt{beta}}, \mathit{cid}_b, 1)), \hat{z}, i),\\ (\texttt{CDS}, f_{\text{mult}}, (\mathit{sid}, \mathit{sid}_{\texttt{beta}}, \mathit{sid}\|\texttt{beta}), \hat{s}, i)\end{gathered}$$

from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$, abort if $\beta z \neq \hat{z} - \hat{s}$. Otherwise, output $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b, z)$.

Figure 16: Protocol $\Pi_{\texttt{repOLE,CDS,proof}}$ for replicated, certified OLE in the $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$-hybrid model

*Proof.* Consider an evaluation indexed by $(\mathit{sid}, \mathit{cid}_a, \mathit{cid}_b)$. Say the receiver gets

$$(\tfrac{1}{2}\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, z, i),$$
$$(\tfrac{1}{2}\texttt{repOLE}, ((\mathit{sid}\|\texttt{beta}, \mathit{cid}_a, 1), (\mathit{sid}_{\texttt{beta}}, \mathit{cid}_b, 1)), \hat{z}, i),$$
$$(\texttt{CDS}, f_{\text{mult}}, (\mathit{sid}, \mathit{sid}_{\texttt{beta}}, \mathit{sid}\|\texttt{beta}), \hat{s}, i)$$

from $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$.

For correctness, assume both sender and receiver are honest. The receiver gets $z, \hat{z}, \hat{s}$ such that $z = ax_{\mathit{sid}} + b, \hat{z} = a\beta x_{\mathit{sid}} + b\beta + s, \hat{s} = s$, where $s$ is sample by the sender. Then the receiver will not abort as $\hat{z} - \hat{s} = a\beta x_{\mathit{sid}} + b\beta = z\beta$. And the receiver outputs $(\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, \mathit{cid}_b, z)$ for $z = ax_{\mathit{sid}} + b$.

For security against receiver, assume the (potentially malicious) receiver choose $\beta, \hat{x}$ and outputs $(\mathit{sid}_{\texttt{beta}}, \beta)$, $(\mathit{sid}\|\texttt{beta}, \hat{x})$, to functionality $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$. The when the sender is honest, the receiver will get $z, \hat{z}, \hat{s}$ such that $z = ax_{\mathit{sid}} + b, \hat{z} = a\hat{x} + b\beta + s$, and $\hat{s} = \begin{cases} s, & \text{if } \hat{x} = \beta x_{\mathit{sid}} \\ \perp, & \text{otherwise} \end{cases}$, where $s \in \mathbb{F}$ is sample by the sender. The simulator gets $z = ax_{\mathit{sid}} + b$ from ideal functionality. If $\hat{x} = \beta x_{\mathit{sid}}$, the receiver's view $(z, \hat{z}, \hat{s})$ is simulated as $\hat{s}$ is uniform random and $\hat{z}$ is determined by $\hat{z} = \beta z + \hat{s}$. Otherwise $\hat{x} \neq \beta x_{\mathit{sid}}$, the receiver's view $(z, \hat{z}, \hat{s})$ is simulated as $\hat{z}$ is uniform random and $\hat{s} = \perp$.

For security against sender, assume the (potentially malicious) sender chooses $b', s_1, s_2$ and sends

$$(\tfrac{1}{2}\texttt{repOLE}, \mathit{sid}, \mathit{cid}_a, b', i),$$
$$(\tfrac{1}{2}\texttt{repOLE}, ((\mathit{sid}\|\texttt{beta}, \mathit{cid}_a, 1), (\mathit{sid}_{\texttt{beta}}, \mathit{cid}_b, 1)), s_1, i),$$
$$(\texttt{CDS}, f_{\text{mult}}, (\mathit{sid}, \mathit{sid}_{\texttt{beta}}, \mathit{sid}\|\texttt{beta}), s_2, i)$$

to $\mathcal{F}_{\frac{1}{2}\texttt{repOLE},\texttt{CDS},\texttt{proof}}$. Then the honest receiver gets $z, \hat{z}, \hat{s}$ such that $z = ax_{\mathit{sid}} + b'$, $\hat{z} = a\beta x_{\mathit{sid}} + b\beta + s_1$, $\hat{s} = s_2$, where $\beta \in \mathbb{F}$ is sample by the receiver. Then receiver will abort when $\beta z \neq \hat{z} - \hat{s}$ and

$$\beta z - (\hat{z} - \hat{s}) = \beta(ax_{\mathit{sid}} + b') - (a\beta x_{\mathit{sid}} + b\beta + s_1) + s_2 = \beta(b' - b) - (s_1 - s_2).$$

Therefore, due to the randomness of $\beta$, the receiver will abort which probability $1 - \frac{1}{|\mathbb{F}|}$ unless $b' = b$, $s_1 = s_2$. This enforces the sender to behave honestly. □

### 5.5.1 Constructing Reusable NISC from $\mathcal{F}_{\texttt{repOLE},\texttt{CDS},\texttt{proof}}$ based on Randomized Encodings

**Theorem 5.8.** *There exists a statistically secure rNISC protocol (Figure 17) for arbitrary $NC^1$ circuits in $\mathcal{F}_{rOLE}^{(\mathbb{F})}$-hybrid model. Securely evaluating an $NC^1$ circuit over $n$ input variables with $t$ gates need $O(nt^3)$ OLE queries. The protocol works for boolean circuits as well as arithmetic circuits.*

In order to construct rNISC from $\mathcal{F}_{\texttt{repOLE},\texttt{CDS},\texttt{proof}}$, we requires randomizing encoding techniques that transforms the evaluation of an arbitrary (arithmetic) function to a affine function [IK02, AIK14]. In a high level, for any arithmetic functionality $\Phi : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^l$, there is random encoder $\mathsf{enc}_\Phi$: $\mathsf{enc}_\Phi$ is a randomized algorithm, given any sender's input $\mathbf{y}$, it outputs $(\mathbf{M}, \mathbf{b}) \leftarrow \mathsf{enc}_\Phi(\mathbf{y})$ such that the distribution of $\mathbf{Mx} + \mathbf{b}$ only depends on $\Phi(\mathbf{x}, \mathbf{y})$, and leaks no information about $\mathbf{y}$.

**Implementing $\mathcal{F}^{(\Phi)}_{\texttt{rNISC}}$: Protocol $\Pi^{(\Phi)}_{\texttt{rNISC}}$**

Parametrized by an arithmetic circuit $\Phi$ over a finite field $\mathbb{F}$, with input variables divided into two disjoint sets $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$, and output variables $Z_1, \ldots, Z_l$.

**Setup/choice phases:**

Upon receiving input $(\mathit{sid}, \mathbf{x})$ from environment, the receiver sends $(\mathit{sid} \| j, x_j)$ to underlying $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$ for all $1 \leq j \leq n$.

**Send phase (for arithmetic function, outlined):**

Sender: Upon receiving $(\mathit{sid}, \mathbf{y})$ from environment, compute $(\mathbf{M}, \mathbf{b}) \leftarrow \mathsf{enc}_{\Phi}(\mathbf{y})$ using randomness $\mathbf{R}_1, \mathbf{R}_2$. Commits $\mathbf{M}, \mathbf{b}, \mathbf{y}, \mathbf{R}_1, \mathbf{R}_2$ to $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$ as coefficients. Convince the receiver that $(\mathbf{M}, \mathbf{b}) = \mathsf{enc}_{\Phi}(\mathbf{y}; \mathbf{R}_1, \mathbf{R}_2)$.

Receiver: Once received $\mathbf{z} = \mathbf{M}\mathbf{x} + \mathbf{b}$ from $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$ and was convince that the sender's coefficients are honestly generated, output $(\mathit{sid}, \mathsf{dec}_{\Phi}(\mathbf{z}), i)$

**Send phase (for boolean function, outlined):**

Sender: Upon receiving $(\mathit{sid}, \mathbf{y})$ from environment, compute $(\mathbf{M}, \mathbf{b}) \leftarrow \mathsf{enc}_{\Phi}(\mathbf{y})$ using randomness $\mathbf{R}_1, \mathbf{R}_2$, samples random $\mathbf{s}$ and compute $\hat{\mathbf{b}} = \mathbf{b} + \mathbf{s}$. Commits $\mathbf{M}, \mathbf{b}, \hat{\mathbf{b}}, \vec{s}, \mathbf{y}, \mathbf{R}_1, \mathbf{R}_2$ to $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$ as coefficients. Convince the receiver that $(\mathbf{M}, \mathbf{b}) = \mathsf{enc}_{\Phi}(\mathbf{y}; \mathbf{R}_1, \mathbf{R}_2)$ and $\hat{\mathbf{b}} = \mathbf{b} + \mathbf{s}$ and $\mathbf{y} \in \{0,1\}^m$. Disclose $\mathbf{s}$ to the receiver if $\mathbf{x} \in \{0,1\}^n$.

Receiver: Once received $\mathbf{z} = \mathbf{M}\mathbf{x} + \hat{\mathbf{b}}, \mathbf{s}$ from $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$ and was convince that the sender's coefficients are honestly generated, output $(\mathit{sid}, \mathsf{dec}_{\Phi}(\mathbf{z} - \mathbf{s}), i)$

Figure 17: Protocol for NISC in the $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$-hybrid model

Moreover, there is also and decoder $\mathsf{dec}_\Phi$, such that $\mathsf{dec}_\Phi(\mathbf{M}\mathbf{x} + \mathbf{b}) = \Phi(\mathbf{x}, \mathbf{y})$ as long as $\mathbf{M}, \mathbf{b}$ is generated by $\mathsf{enc}_\Phi(\mathbf{y})$.

Randomized encoding is already sufficient for rNISC in $\mathcal{F}_{\mathtt{rOLE}}$-hybrid model, when only requires one-side security against malicious receiver. The sender generates $(\mathbf{M}, \mathbf{b}) \leftarrow \mathsf{enc}_\Phi(\mathbf{y})$ using randomized encoding, and obliviously evaluates $\mathbf{M}\mathbf{x} + \mathbf{b}$ using underlying rOLE. Then receiver can recover $\Phi(\mathbf{x}, \mathbf{y})$ using the decoder, but no more information is leaked. To make the protocol secure against a malicious sender, the receiver needs a guarantee that $\mathbf{M}, \mathbf{b}$ are honestly generated. Such "proof of honest behavior" is captured by $\mathcal{F}_{\mathtt{repOLE,CDS,proof}}$.

For concreteness, consider an explicit randomized encoding scheme. Given any $\mathrm{NC}^1$ arithmetic circuit $\Phi$ of $t$ gates, there exists an $t$-node branching program $\mathsf{G}$ such that $\Phi(\mathbf{x}, \mathbf{y}) := \det \mathsf{G}(\mathbf{x}, \mathbf{y})$. The encoder samples random triangular matrixes $R_1, R_2$ with diagonal all one, i.e. $\mathbf{R}_1, \mathbf{R}_2$ are of the form

$$\begin{bmatrix} 1 & \$ & \cdots & \$ \\ & 1 & \ddots & \vdots \\ & & \ddots & \$ \\ & & & 1 \end{bmatrix},$$

then outputs the factor matrix and the shift vector of the affine mapping

$$\mathbf{x} \mapsto \mathbf{R}_1 \cdot \mathsf{G}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{R}_2.$$

Correctness for this randomized encoder follows directly from $\det(\mathbf{R}_1 \cdot \mathsf{G}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{R}_2) = \det \mathsf{G}(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}, \mathbf{y})$. This randomized encoding is also private, because the distribution of $\mathbf{R}_1 \cdot \mathsf{G}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{R}_2$ only depends on $\det \mathsf{G}(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}, \mathbf{y})$. For efficiency, note $\mathbf{M}, \mathbf{b}$ is determined by $\mathbf{y}, \mathbf{R}_1, \mathbf{R}_2$, with computation complexity $O(nt^3 + mt^2)$. Thus argument that $(\mathbf{M}, \mathbf{b}, \mathbf{y}, \mathbf{R}_1, \mathbf{R}_2)$ is consistent can be written as arithmetic circuits of $O(nt^3 + mt^2)$ gates in total.

Finally, in the case of evaluating boolean circuits. Convert the circuit as a arithmetic circuit, and each party should feed input in $\{0, 1\}^n$. The functionality $\mathcal{F}_{\mathtt{repOLE,CDS,proof}}$ allows the enforcement of such constraints. These leads us to Theorem 5.8.

### 5.5.2 Constructing Reusable NISC from $\mathcal{F}_{\mathtt{repOLE,CDS,proof}}$ based on Garbled Circuits

**Theorem 5.9.** *If one-way functions exist, there is a computationally secure rNISC protocol in $\mathcal{F}^{(\mathbb{F})}_{rOLE}$-hybrid model for arbitrary polynomial size circuits. Such that securely evaluating an $NC^1$ circuit over $n$ input variables with $t$ gates need* $\mathrm{poly}(\lambda) \cdot O(n + t)$ *OLE queries, where $\lambda$ is the security parameter.*

Using one-way function, we can embed garbled circuit in rNISC for better efficience. As a high level view of the protocol:

- In the choice phase, the receiver sends its input $(sid, \mathbf{x})$ to $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE,CDS,proof}}$. As ensuring (using CDS) that $\mathbf{x} \in \{0, 1\}^n$.
- In the send phase, upon receiving $(sid, \mathbf{y}, i)$ from environment. The sender prepares the garbled circuit computing $\mathbf{x} \to \Phi(\mathbf{x}, \mathbf{y})$, then sends its input, the garbled circuit together with all intermediate variable to $\mathcal{F}_{\mathtt{repOLE,CDS,proof}}$. And convince the receiver that the garbled circuit is honestly generated using $\mathcal{F}_{\mathtt{repOLE,CDS,proof}}$.

- For each input wire, discloses appropriate tag is disclosed according to receiver's input. E.g. for input $x_i$, the receiver gets $\mathbf{t}_{i,0} + x_i(\mathbf{t}_{t,1} - \mathbf{t}_{t,0})$ where $\mathbf{t}_{i,0}, \mathbf{t}_{i,1}$ are the tags for this input wire. Moreover, use CDS functionality to ensure the receiver gets no information if he feeds $x_i \notin \{0,1\}$ to $\mathcal{F}_{\texttt{repOLE,CDS,proof}}$.

### 5.6 Efficiency Optimization

So far, we've implemented $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$, which directly implies a rNIZK protocol in $\mathcal{F}_{\texttt{rOLE}}$-hybrid model. To order to prove knowledge of a satisfying assignment of an arithmetic circuit, the cost is no more than 300 $\mathcal{F}_{\texttt{rOLE}}$ evaluations gate. In this section, we point out how to reduce the cost.

Along this chain of functionalities $\mathcal{F}_{\texttt{rOLE}}$, $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$, $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS}}$, $\mathcal{F}_{\frac{1}{2}\texttt{repOLE,CDS,proof}}$, we present a black-box implementation of the next functionality in the hybrid model of the previous functionality. This approach ensures that the final result is combination of several well-separated steps. If we instead allow more entanglement, the efficiency can be dramatically improved.

*Trick 1: Using* $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. Protocol $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ implements $\mathcal{F}_{\frac{1}{2}\texttt{repOLE}}$ but it is not fully secure against a malicious sender. So we wrap it in the final construction $\Pi_{\frac{1}{2}\texttt{repOLE}}$, which eliminate the cheating space of a malicious sender but also increase the cost by more than 6 times. Therefore if we replace the occurrence of $\Pi_{\frac{1}{2}\texttt{repOLE}}$ in later constructions by $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$, efficiency would be improved. We notice that in many cases, this actually does not hurt the security.

*Trick 2: Amortization.* In $\Pi_{\frac{1}{2}\texttt{repOLE,CDS}}$, the sender discloses secret to the receiver if the input chosen by the receiver satisfies some constraints. The current constraint only consider the case when the secret is a single field element. The amortized cost can be improved if we consider longer secret. This amortization is particularly useful, because all the secret are actually disclosed conditioning on the same constraint.

For example in $\Pi_{\frac{1}{2}\texttt{repOLE,CDS}}$, when the sender discloses secrets $s_1, \ldots, s_k \in \mathbb{F}$ conditioning on an affine constraint $c_0 + \sum_{j=1}^{\ell} c_j x_{sid_j} = 0$, the following protocol combines trick 1, trick 2 and has a cheaper cost:

- In the chosen phase, the receiver samples $\gamma_1, \ldots, \gamma_k \in \mathbb{F}$, and chooses $\gamma_t$ as input under session id $sid_{\mathsf{gamma}\text{-}t}$.

- The sender samples $r_1, \ldots, r_{k+1} \in \mathbb{F}$, chooses new coefficient id $cid_1, \ldots, cid_{k+1}$ and sends $(cid_1, r_1), \ldots, (cid_{k+1}, r_{k+1})$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$. Then samples $s_{k+1}, b_1, \ldots, b_k$ satisfying $s_{k+1} = \sum_t b_t$, and sends $(sid_{\mathsf{gamma}\text{-}t}, s_t, b_t, cid_t)$ to $\mathcal{F}_{\texttt{rOLE}}$for $t \in [k]$. Then samples $w_{1,1}, \ldots, w_{k+1,\ell} \in \mathbb{F}$ satisfying $\sum_j c_j w_{t,j} = r_t c_0 + s_t$ for all $t \in [k+1]$ and sends $(sid_j, cid_t, w_{t,j}, t)$ to $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ for $t \in [k+1]$, $j \in [\ell]$

- The receiver, upon receiving $(sid_{\mathsf{gamma}\text{-}t}, s_t, u_t, cid_t)$ from $\mathcal{F}_{\texttt{rOLE}}$for $t \in [k]$, and $(sid_j, cid_t, z_{t,j}, t)$ from $\Pi_{\alpha\text{-}\frac{1}{2}\texttt{repOLE}}$ for $t \in [k+1]$, $j \in [\ell]$, computes $\hat{s}_t = \sum_j c_j z_{t,j}$, then aborts if $\sum_{t=1}^{k} \gamma_t \hat{s}_t + \hat{s}_{k+1} \neq \sum_t u_t$, outputs $\hat{s}_1, \ldots, \hat{s}_k$ otherwise.

The amortized cost for disclosing one field element is $(2 + \frac{1}{k})\ell + 2$ $\mathcal{F}_{\texttt{rOLE}}$ evaluation calls. In comparison, the cost for disclosing one field element in the unoptimized construction (Figure 11) is $13\ell$ $\mathcal{F}_{\texttt{rOLE}}$ evaluation calls.

*Trick 3: Relaying on external randomness.* In our construction, the CDS functionality is only used when the sender need to disclose secrets conditioning on $\mu\nu = \xi$, where $\mu, \nu, \xi \in \mathbb{F}$ are inputs sampled by the receiver that $(\mu,\nu)$ is uniformly random. The protocol $\Pi_{\frac{1}{2}\mathtt{repOLE,CDS}}$ can be simplified by making use of this external randomness.

Let $M := \left[\begin{smallmatrix}\mu & \xi\\ -1 & \nu\end{smallmatrix}\right]$, then the constraint $\mu\nu = \xi$ is equivalent to $\det M = 0$. In section 5.3, we first generate matrices $M_1, M_2$ from $M$ so that each of $\mathsf{key}_{M_1}, \mathsf{key}_{M_2}$ is uniformly random. These randomness helps protect the protocol against malicious senders. But by the external randomness, we already have $\mathsf{key}_M = (-\mu)$ is uniformly random. Therefore the process of generating $M_1, M_2$ and "proving" $\mathsf{head}_{M_1} + \mathsf{head}_{M_2} = \mathsf{head}_M$ can be skipped, which at least halves the cost.

By combining with trick 1 and trick 2, the amortized cost for disclosing one field element conditioning on $\mu\nu = \xi$, is $9 + O(\frac{1}{k})$ $\mathcal{F}_{\mathtt{rOLE}}$ evaluation calls. In comparison, the unoptimized construction (Figure 11) requires 245 $\mathcal{F}_{\mathtt{rOLE}}$ evaluations for the same thing.

In $\Pi_{\frac{1}{2}\mathtt{repOLE,CDS,proof}}$, trick 1 can be used when proving that the coefficients satisfy an affine constraint. The cost of proving an $\ell$-variant affine argument is $2\ell$ $\mathcal{F}_{\mathtt{rOLE}}$ evaluation calls.

To improve the cost of proving a multiplicative argument, notice that 3 of out the 5 $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE}}$ evaluation calls can be safely handled using $\Pi_{\alpha\text{-}\frac{1}{2}\mathtt{repOLE}}$. The cost of proving such a multiplicative argument is $43 + O(\frac{1}{k})$ $\mathcal{F}_{\mathtt{rOLE}}$ evaluation calls.

As for the concrete efficiency of our rNIZK protocol: each fan-in-2 addition gate corresponds to a $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE,CDS,proof}}$ proof of 3-variant affine argument, which costs 6 $\mathcal{F}_{\mathtt{rOLE}}$ evaluation calls; each fan-in-2 multiplication gate corresponds to a $\mathcal{F}_{\frac{1}{2}\mathtt{repOLE,CDS,proof}}$ proof of some $a_{cid_1}a_{cid_2} = a_{cid_3}$, which costs $43 + O(\frac{1}{k})$ $\mathcal{F}_{\mathtt{rOLE}}$ evaluation calls; in addition, each input wire and each intermediate wire corresponds to a coefficient chosen by the sender, which costs 1 $\mathcal{F}_{\mathtt{rOLE}}$ evaluation call.

# 6 A Reusable OLE Construction based on Paillier

In this section, we show a reusable OLE construction $\Pi_{\mathrm{rOLE}}$ based on the Paillier assumption. Our construction proceeds as follows.

- $\mathsf{CRSSetup}(1^k)$: Let $N = pq$ be such that $p = 2p' + 1$ and $q = 2q' + 1$ for primes $p', q'$ of the appropriate length for security parameter $k$. All operations will be mod $N^2$ unless otherwise specified. Choose $b, B_0 \leftarrow \mathbb{Z}^*_{N^2}$, and let $h = 1 + N$. Let $T = 2^k N^2$.

  Output $crs = (N, h, b, B_0, T)$

- $\mathsf{ReceiverRequest}(crs, \alpha)$: Parse $crs = (N, h, b, B_0, T)$. Choose random $sk, sk', \gamma$ from $[T]$. Let $B = b^{sk}$ and $B' = b^{sk'}$. Send $B_1 = BB_0^{-\gamma}$ and $B_1' = B'B_0^{-\alpha+\gamma}$ to the sender, and output state $sk, sk', \alpha, \gamma$.

- $\mathsf{SenderResponse}(crs, (B_1, B_1'), z_0, z_1)$: Parse $crs = (N, h, b, B_0, T)$. Chooose random $r, w$ from $[T]$. Compute $c = b^r$, $C_0 = {B_0}^r h^{z_0}$, $C_1 = {B_1}^r h^w$, and $C_1' = {B_1'}^r h^{z_1 - w}$. Send $c, C_0, C_1, C_1'$ to the receiver.

- $\mathsf{ReceiverReceive}(crs, (c, C_0, C_1, C_1'), (sk, sk', \alpha, \gamma))$: Compute $X = {C_0}^\gamma C_1 / c^{sk}$ and $X' = {C_0}^{\alpha-\gamma} C_1' / c^{sk'}$. If it is not the case that $X^2$ and ${X'}^2$ are of the form $1 + xN$ and $1 + x'N$ for some $x, x'$, then output $\perp$. Otherwise output $z = (x + x')/2 \mod N$.

We first show correctness. If both parties are honest, then the response is computed as follows:

$$\begin{aligned}
X &= C_0{}^{\gamma} C_1 / c^{sk} \text{ (from } \mathsf{ReceiverReceive}) \\
&= (B_0{}^{r} h^{z_0})^{\gamma} B_1^{r} h^{w} / (b^{r})^{sk} \text{ (from } \mathsf{SenderResponse}) \\
&= (B_0{}^{\gamma} B_1 / b^{sk})^{r} h^{z_0\gamma + w} \\
&= (B_0{}^{\gamma} B B_0^{-\gamma} / b^{sk})^{r} h^{z_0\gamma + w} \text{ (from } \mathsf{ReceiverRequest}) \\
&= (B / b^{sk})^{r} h^{z_0\gamma + w} \\
&= (b^{sk} / b^{sk})^{r} h^{z_0\gamma + w} \text{ (from } \mathsf{ReceiverRequest}) \\
&= h^{z_0\gamma + w}
\end{aligned}$$

and similarly, $X' = h^{z_0(\alpha-\gamma)+z_1-w}$

Next, note that $(h^{z_0\gamma+w})^2 = (1+N)^{2(z_0\gamma+w)} = 1+2(z_0\gamma+w)N \mod N^2$, and $(h^{z_0(\alpha-\gamma)+z_1-w})^2 = 1 + 2(z_0(\alpha-\gamma) + z_1 - w)N \mod N^2$ so the receiver will output $z = (x + x')/2 = (z_0\gamma + w) + (z_0(\alpha-\gamma) + z_1 - w) = z_0\alpha + z_1 \mod N$.

**Theorem 6.1.** $\Pi_{\mathrm{rOLE}}$ *is a UC-secure realization of the reusable OLE functionality* $\mathcal{F}_{\mathrm{rOLE}}$ *over the ring* $\mathbb{Z}_N^*$.

## 6.1 Security Against Dishonest Receiver

Let $G_{2p'q'}$ be the subgroup of $\mathbb{Z}_{N^2}^*$ of elements of the form $x = x'^N$ for $x' \in \mathbb{Z}_{N^2}^*$. Consider the following simulator $\mathcal{S}$.

To generate the CRS, the simulator generates $N, h, T$ as above, but stores the factorization of $N$. It then chooses random $b, Y_0 \leftarrow G_{2p'q'}$, computes $B_0 = Y_0 h$, and outputs $crs = (N, h, b, B_0, T)$.

When the adversary sends $(B_1, B_1')$, the simulator proceeds as follows: First, use the factorization of $N$ to compute $Y_1, Y_1' \in G_{2p'q'}$ and $\delta, \delta' \in \mathbb{Z}_N$ such that $B_1 = Y_1 h^{\delta}$ and $B_1' = Y_1' h^{\delta'}$. Send $-(\delta + \delta')$ to $\mathcal{F}$.

When $\mathcal{F}$ sends $z$: Choose random $u \leftarrow \mathbb{Z}_{2p'q'}$, $v_0, v_1 \leftarrow \mathbb{Z}_N$, and compute

$$\begin{aligned}
c &= b^u \\
C_0 &= Y_0^u h^{v_0} \\
C_1 &= Y_1^u h^{v_1} \\
C_1' &= Y_1^u h^{z + v_0(\delta+\delta') - v_1}
\end{aligned}$$

Send $(c, C_0, C_1, C_1')$ to $\mathsf{A}$.

**Claim 1.** The CRS generated above is indistinguishable from the CRS generated by $\mathsf{CRSSetup}$ under the decisional composite residuosity assumption (DCRA).

*Proof.* First consider an intermediate CRS generated by sampling $b \leftarrow G_{2p'q'}$ and $B_0 \leftarrow \mathbb{Z}_{N^2}$. This is indistinguishable from the real CRS by DCRA. Then, note that we obtain an idential distribution if we choose $Y_0 \leftarrow \mathbb{Z}_{N^2}$ and output $B_0 = Y_0 h$. Finally, sampling $Y_0 \leftarrow \mathbb{Z}_{N^2}$ is indistinguishable from sampling $Y_0 \leftarrow G_{2p'q'}$ again by DCRA. □

**Claim 2.** Consider an intermediate game in which the environment and $\mathsf{A}$ interact with the honest sender, but where the CRS is generated as above. The environment's view in this game is statistically close to it's view when interacting with $\mathcal{S}$ and $\mathcal{F}$ as defined above.

*Proof.* To see this, first consider a game where the CRS is generated as described above, but the sender's response is generated according to $\mathsf{SenderResponse}$ with the exception that $r, w$ are chosen at random from $\mathbb{Z}_{2p'q'N}$ rather than $[T]$. Note that since $T/2p'q'N$ is exponential in the security parameter, this will be statistically close to the original distribution.

Now we will argue that the environment's view in this game is identical to its view when interacting with $\mathcal{S}$ and $\mathcal{F}$. To see this, consider an experiment where the CRS is generated as described above, and when running $\mathsf{SenderResponse}$, we choose $u, w' \leftarrow \mathbb{Z}_{2p'q'}$ and $v_0, v_1 \leftarrow \mathbb{Z}_N$, and set $r, w \in \mathbb{Z}_{2p'q'N}$ such that $r = u \mod 2p'q'$ and $r = v_0 - z_0 \mod N$, and $w = w' \mod 2p'q'$ and $w = v_1 - \delta(v_0 - z_0) \mod N$. Note that the resulting $r, w$ are uniform over $\mathbb{Z}_{2p'q'N}$, so this is identical to the previous game. Given $B_1, B_1'$, let $Y_1, Y_1' \in G_{2p'q'}$ and $\delta, \delta' \in \mathbb{Z}_N$ such that $B_1 = Y_1 h^{\delta}$ and $B_1' = Y_1' h^{\delta'}$.

The resulting $c, C_0, C_1, C_1'$ are as follows:

$$\begin{aligned}
c &= b^r = b^u \\
C_0 &= {B_0}^r h^{z_0} = (Y_0 h)^r h^{z_0} = {Y_0}^u h^{v_0 - z_0} h^{z_0} = {Y_0}^u h^{v_0} \\
C_1 &= {B_1}^r h^w = (Y_1 h^{\delta})^r h^w = {Y_1}^u h^{\delta(v_0 - z_0)} h^{v_1 - \delta(v_0 - z_0)} = {Y_1}^u h^{v_1} \\
C_1' &= {B_1'}^r h^{z_1 - w} = (Y_1' h^{\delta'})^r h^{z_1 - w} = {Y_1'}^u h^{\delta'(v_0 - z_0)} h^{z_1 - v_1 + \delta(v_0 - z_0)} \\
&= {Y_1'}^u h^{-(\delta+\delta')z_0 + z_1 + (\delta+\delta')v_0 - v_1} = {Y_1'}^u h^{z + (\delta+\delta')v_0 - v_1} \quad \text{for } z = -(\delta+\delta')z_0 + z_1.
\end{aligned}$$

Finally, note that this is exactly the distribution that would be produced by $\mathcal{S}$ and $\mathcal{F}$, since $\mathcal{S}$ would send $-(\delta + \delta')$ to $\mathcal{F}$ and $\mathcal{F}$ would return $z = -(\delta + \delta')z_0 + z_1$.

□

## 6.2 Security Against Dishonest Sender

Let $G_{p'q'}$ be the subgroup of $\mathbb{Z}_{N^2}^*$ of the form $x = x'^{2N}$ for $x' \in \mathbb{Z}_{N^2}^*$.

Consider the following simulator $\mathcal{S}$.

To generate the CRS, the simulator generates $N, h, T$ as above. It then chooses random $b \leftarrow G_{p'q'}$ and $sk_0 \leftarrow [T]$, computes $B_0 = b^{sk_0}$, and outputs $crs = (N, h, b, B_0, T)$.

The simulator then generates $B_1, B_1'$ as follows: choose random $sk_1, sk_1' \leftarrow [T]$ and send $B_1 = b^{sk_1}$ and $B_1' = b^{sk_1'}$.

When the adversary sends $(c, C_0, C_1, C_1')$, proceed as follows: Compute $X_0 = C_0/c^{sk_0}$, $X_1 = C_1/c^{sk_1}$, and $X_1' = C_1'/c^{sk_1'}$. If it is not the case that $X_0^2$, ${X_1}^2$, and ${X_1'}^2$ are of the form $1 + x_0 N$, $1 + x_1 N$ and $1 + x_1' N$ for some $x_0, x_1, x_1'$, then send $\perp$ to $\mathcal{F}$. Otherwise send $x_0/2$ and $(x_1 + x_1')/2$ to $\mathcal{F}$.

**Claim 3.** The CRS generated above is indistinguishable from the CRS generated by $\mathsf{CRSSetup}$ as long as the decisional composite residuosity assumption (DCRA) and the quadratic residuosity assumption (QRA) hold in $\mathbb{Z}_{N^2}$.

*Proof.* Recall that in the real game $b, B_0$ are sampled uniformly from $\mathbb{Z}_{N^2}$. We first consider a modification where we compute $b = x'^N$ for random $x'$; this is indistinguishable by DCRA. Then we change to computing $b = (x'^2)^N$; this will be indistinguishable by QRA. Note that this is equivalent to sampling $b \leftarrow G_{p'q'}$. Next, we argue similarly that by DCRA, QRA, we can choose $B_0 \leftarrow G_{p'q'}$ instead of $B_0 \leftarrow \mathbb{Z}_{N^2}$. Then observe that if we instead choose $sk_0 \leftarrow \mathbb{Z}_{p'q'}$ and set $B_0 = b^{sk_0}$, the resulting distribution will be statistically close (because with all but negligible probability $b \leftarrow G_{p'q'}$ is a generator). Finally, we note that choosing $sk_0 \leftarrow T$ and then computing $B_0 = b^{sk_0}$ is statistically close to choosing $sk_0 \leftarrow \mathbb{Z}_{p'q'}$, because $T/p'q'$ is exponential. □

**Claim 4.** Consider an intermediate game in which the environment and A interact with the honest receiver, but where the CRS is generated as above. The environment's view in this game is statistically close to it's view when interacting with $\mathcal{S}$ and $\mathcal{F}$ as defined above.

*Proof.* To see this, first consider a variation on the intermediate game, where $sk_0$ and the random values $sk, sk', \gamma$ used in ReceiverRequest are chosen at random from $\mathbb{Z}_{2p'q'N}$ rather than $[T]$. Note that since $T/2p'q'N$ is exponential in the security parameter, this will be statistically close to the original distribution.

Next, we modify the game slightly, so that instead of choosing $sk, sk'$ directly, we choose random $sk_1, sk'_1 \leftarrow \mathbb{Z}_{2p'q'N}$ and set $sk = sk_1 + \gamma sk_0$ and $sk' = sk'_1 + (\alpha - \gamma)sk_0$. Note that the resulting distribution will be identical.

Now, we modify the conditions under which the receiver outputs $\perp$, as follows: First, suppose in addition to running ReceiverReceive, we also compute $X_0, X_1, X'_1$ as described above, and output $\perp$ if $X^2, X'^2, {X_0}^2, {X_1}^2, {X'_1}^2$ are not of the form $1 + vN$ for some values of $v$. Now we argue that for any environment, this game will be statistically indistinguishable from the previous one. To see this, note that the only way that these games will be distinguishable is if $X^2, X'^2$ are of the form $1 + vN$, but at least one of ${X_0}^2, {X_1}^2, {X'_1}^2$ is not. (If we are considering the re-usable functionality, then this requires that on at least one of the sender's messages, this is true - consider the first such message.) We will argue that this happens with negligible probability. First, by our computation of $X_0, X_1, X'_1, X, X'$, we have

$$\begin{aligned} X &= {C_0}^{\gamma} C_1 / c^{sk} \\ &= {C_0}^{\gamma} C_1 / c^{sk_1 + \gamma sk_0} \qquad \text{by our choice of } sk, sk' \\ &= (C_0/c^{sk_0})^{\gamma} C_1 / c^{sk_1} \\ &= {X_0}^{\gamma} X_1 \qquad \text{by definition of } X_0, X_1. \end{aligned}$$

and similarly $X' = X_0^{\alpha-\gamma} X'_1$.

Now, let $Y_0, Y_1, Y'_1 \in G_{p'q'}$ and $\delta_0, \delta_1, \delta'_1 \in \mathbb{Z}_N$ be such that ${X_0}^2 = Y_0 h^{\delta_0}, {X_1}^2 = Y_1 h^{\delta_1}$, and $X'_1 = Y_1 h^{\delta'_1}$. If these values allow the environment to distinguish, then at least one of $Y_0, Y_1, Y'_1$ is not 1. Because $X = X_0^{\gamma} X_1$ and $X' = X_0^{\alpha-\gamma} X'_1$, and both $X^2, X'^2$ have no $G_{p'q'}$ component, it must be the case that ${Y_0}^{\gamma} Y_1 = 1$ and ${Y_0}^{\alpha-\gamma} Y'_1 = 1$. However, we observe that $\gamma$ is completely independent of the environment's view so far. Now, suppose $Y_0 \neq 1$; then the probability that ${Y_0}^{\gamma} Y_1 = 1$ is $1/p'q'$, which is negligible. If $Y_0 = 1$, then ${Y_0}^{\gamma} Y_1 = 1$ and ${Y_0}^{(\alpha-\gamma)} Y'_1 = 1$ implies that $Y_1$, $Y'_1$ are both 1 as well. Thus, the probability that $X, X'$ do not cause $\perp$, but $X_0, X_1, X'_1$ do is negligible.

Next, consider the honest receiver's output when it is not $\perp$. In the above game, the receiver gets $c, C_0, C_1, C'_1$ from the sender, computes $X, X$ such that $X^2 = (1 + xN)$ and $X'^2 = (1 + x'N)$, and outputs $(x+x')/2$. However, as we have argued above, by the way we have defined $sk, sk'$, $X = {X_0}^{\gamma} X_1$ and $X' = {X_0}^{\alpha-\gamma} X_1'$, so we could instead compute $X_0, X_1, X_1'$ which assuming we don't abort will be such that ${X_0}^2 = 1 + x_0 N$, ${X_1}^2 = 1 + x_1 N$, and ${X_1'}^2 = 1 + x_1' N$, Then $X^2 = {X_0}^2 X_1 = (1+x_0N)^{\gamma}(1+x_1N) = 1 + (x_0\gamma + x_1)N$ and similarly ${X'}^2 = 1 + (x_0(\alpha-\gamma) + x_1')N$, so we could equivalently compute $x = x_0\gamma + x_1 \mod N$ and $x' = x_0(\alpha - \gamma) + x_1' \mod N$ and output $(x+x')/2 = (x_0\gamma + x_1 + x_0(\alpha-\gamma) + x_1')/2 = (x_0/2)\alpha + (x_1 + x_1')/2 \mod N$.

Finally, we change to a game where we output $\perp$ only if $X_0, X_1$, or $X_1'$ is not of the correct form. As we argued above, we will have $X = {X_0}^{\gamma} X_1$ and $X' = {X_0}^{\alpha-\gamma} X_1'$, so if $X^2$ or ${X'}^2$ has a non-trivial component in $G_{p'q'}$, then it must be the case that at least one of ${X_0}^2, {X_1}^2, {X_1'}^2$ also has a non-trivial component in $G_{p'q'}$, so this is identical to the previous game. We then observe that this game is identical to an interaction with $\mathcal{S}$ and $\mathcal{F}$, which concludes the proof.

□

# References

[ADI+17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 223–254. Springer, 2017.

[AIK14] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM J. Comput.*, 43(2):905–929, 2014.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2001.

[AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 387–404, 2014.

[BCG+17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 336–365. Springer, 2017.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Janos Simon, editor, *Proceedings of the 20th*

*Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112. ACM, 1988.

[CC18] Pyrros Chaidos and Geoffroy Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, pages 193–221, 2018.

[DGN$^+$17] Nico Döttling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation. In *CCS 2017*, pages 2263–2276, 2017.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[GIP$^+$14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *STOC 2014*, pages 495–504, 2014.

[GNN17] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 629–659. Springer, 2017.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology, Proceedings of EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425. Springer, 2011.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 448–457, 2001.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 554–571, 2008.

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 165–179. Academic Press, 1978.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *Proceedings of FOCS 1986*, pages 162–167. IEEE Computer Society, 1986.

# A Reusable Secure Arithmetic Circuit Evaluation

**Definition 11** (Arithmetic Circuits and Formulas)**.** *An* arithmetic circuit $\Phi$ *over a given ring* $\mathcal{R}$ *with input variables* $X_1, \ldots, X_n$ *and output variables* $Z_1, \ldots, Z_l$ *is defined by:*

*(1) a directed acyclic graph* $G = (V, E)$ *that has exactly* $l$ *vertices with out-degree* 0 *and*

*(2) a labeling which assigns*

- *to every vertex with in-degree* 0 *a label from* $\{X_1, \ldots, X_n\} \cup \mathcal{R}$*,*
- *to every other vertex a label from* $\{+, \times\}$*,*
- *and additionally to every vertex with out-degree* 0 *a unique label from* $\{Z_1, \ldots, Z_l\}$*.*

*If* $G$ *is a tree (and thus* $m = 1$*), then* $\Phi$ *is called an* arithmetic formula*.*

*The circuit* $\Phi$ *computes a function* $\varphi : \mathcal{R}^n \to \mathcal{R}^l$ *in a natural way. Each* input gate *(vertex with some label* $X_i$*) and each* constant gate *(vertex with label in* $\mathcal{R}$*) computes the polynomial it is labeled with. Each* addition gate *(vertex labeled* $+$*) computes the sum of its children, and each* multiplication gate *(vertex labeled* $\times$*) computes the product of its children. The function* $\varphi$ *is then defined by the polynomials computed by the* output gates *(vertices with additional label* $Z_i$*).*

*We refer to the number of gates* $|\Phi| := |V|$ *as the* size *of* $\Phi$*, and to the length of the longest directed path in* $G$ *as the* depth *of* $\Phi$*. We restrict ourself to the case that all gates have in-degree* 2*, so that any formula of depth* $d$ *has size* $O(2^d)$*.*

**Remark A.1.** Every arithmetic circuit with $l$ output variables can be written as a collection of $l$ arithmetic formulas of the same depth. The size of these formulas can grow exponentially in their depth even for linear size circuits. Still, $\mathrm{NC}^1$ circuits can be written as polynomial size formulas, since we require that all gates have in-degree 2.

The ideal functionality for secure arithmetic circuit evaluation with reusable receiver inputs (see Figure 18) is a special instance of $\mathcal{F}^{(F)}_{\texttt{rNISC}}$ from Figure 1.

**Functionality $\mathcal{F}^{(\Phi)}_{\texttt{rNISC}}$**

Parametrized by an arithmetic circuit $\Phi$ over a finite commutative ring $\mathcal{R}$, with input variables divided into two disjoint sets $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$, and output variables $Z_1, \ldots, Z_l$.

**Setup/choice phases:**

- Upon receiving input $(sid, x_1, \ldots, x_n)$ from Rachel where $(x_1, \ldots, x_n) \in \mathcal{R}^n$ and *sid* is a session identifier, store $(sid, x_1, \ldots, x_n)$, send $(sid, \texttt{initialized})$ to the adversary and ignore any further inputs from Rachel with the same session identifier *sid*.

**Send phases:**

- Upon receiving input $(sid, y_1, \ldots, y_m, i)$ from Sam where $(y_1, \ldots, y_m, i) \in \mathcal{R}^m \times \mathbb{N}$ and *sid* is a session identifier, record $(sid, y_1, \ldots, y_m, i)$, send $(sid, \texttt{sent}, i)$ to the adversary, and ignore any further inputs from Sam with the same session identifier *sid* and the same value of $i$.
- Upon receiving a message $(sid, \texttt{Delivery}, i)$ from the adversary, verify that there are stored inputs $(sid, x_1, \ldots, x_n)$ from Rachel and $(sid, y_1, \ldots, y_m, i)$ from Sam; else ignore that message. Next, compute $(z_1, \ldots, z_l)$ according to $\Phi$'s functionality from $(x_1, \ldots, x_n, y_1, \ldots, y_m)$, send $(sid, z_1, \ldots, z_l, i)$ to Rachel, and ignore further messages $(sid, \texttt{Delivery}, i)$ from the adversary with the same session identifier *sid* and the same value of $i$.

Figure 18: Ideal functionality for secure arithmetic circuit evaluation with reusable receiver inputs.

We will henceforth omit the superscripts $F$ and $\Phi$ when they are clear from the context. Other functionalities such as the zero knowledge functionality can be defined as a special case of the rNISC functionality.