Reusable Non-Interactive Secure Computation

Melissa Chase[1], Yevgeniy Dodis[2], Yuval Ishai[3(✉)], Daniel Kraschewski[4], Tianren Liu[5(✉)], Rafail Ostrovsky[6], and Vinod Vaikuntanathan[5]

[1] Microsoft Research, Redmond, USA
melissac@microsoft.com
[2] New York University, New York, USA
dodis@cs.nyu.edu
[3] Technion, Haifa, Israel
yuvali@cs.technion.ac.il
[4] TNG Technology Consulting GmbH, Unterföhring, Germany
daniel.kraschewski@tngtech.com
[5] MIT, Cambridge, USA
{liutr,vinodv}@mit.edu
[6] UCLA, Los Angeles, USA
rafail@cs.ucla.edu

Abstract. We consider the problem of Non-Interactive Two-Party Secure Computation (NISC), where Rachel wishes to publish an encryption of her input x, in such a way that any other party, who holds an input y, can send her a single message which conveys to her the value $f(x, y)$, and nothing more. We demand security against malicious parties. 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. We also get

A. Boldyreva and D. Micciancio (Eds.): CRYPTO 2019, LNCS 11694, pp. 462–488, 2019.
https://doi.org/10.1007/978-3-030-26954-8_15

reusable NISC in the OLE-hybrid model for general Boolean circuits using any one-way function.
- We complement this by a negative result, showing that reusable NISC is impossible to achieve in the OT-hybrid model. This provides a formal justification for the need to replace OT by OLE.
- We build a universally composable 2-message *reusable* 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 protocol in the OLE-hybrid model 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 for NP where following a statement-independent preprocessing, both proving and verifying are entirely "non-cryptographic" and involve only a constant computational overhead. Furthermore, we get the first *statistical* designated-verifier NIZK argument for NP under an assumption related to factoring.

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 any other party, who holds an input y, can send her a single message which conveys to her the value $f(x, y)$, and nothing more. In the semi-honest setting, there are several solutions to this problem including (i) garbled circuits [23,29] combined with two-message oblivious transfer (OT) protocols (e.g., [2,24,26]) and (ii) fully homomorphic encryption [9,16,27].

In reality, we care about security against potentially malicious parties and indeed, we have tools to achieve this level of security. For example, 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 [6]. However, this requires making *non-black-box* use of the underlying cryptographic primitives, and is generally infeasible in practice. A recent line of work [1,19] has come up with *efficient* maliciously secure NISC protocols that make oracle calls to an oblivious transfer primitive. This model is referred to as the OT-hybrid model, and we henceforth refer to NISC protocols in the OT-hybrid model succinctly as NISC/OT protocols.

The paradigm of designing protocols in the OT-hybrid model that are either information-theoretically secure or make use of symmetric cryptographic primitives such as a pseudorandom generator, and plugging in fast implementations of OT, has paid great dividends in cryptography for several reasons. First, we have fast OT implementations under standard assumptions. Secondly, OT is self-reducible, so the cryptographic cost of implementing it can be pushed to an offline phase. OT can itself also 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 [1,19]. This approach is beneficial even in simpler special cases such as constructing (designated-verifier) NIZK. For these cases, and more generally for functionalities computed by log-depth circuits or polynomial-size branching programs, such NISC/OT protocols can be made information-theoretically secure (in particular, there is no need for the pseudorandom generator).

Selective Failure Attacks and (Non-)Reusability. 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, any NISC/OT protocol is guaranteed to be reusable, in the sense that if we fix Rachel's OT inputs and let Sam choose fresh OT inputs in each invocation, security still holds. However, in all known NISC/OT protocols (e.g., [1,19]), a malicious Sam can mount a "selective failure attack", feeding malformed OT messages to Rachel, checking whether she aborts or not, and using this information to violate both the secrecy of her input and the correctness of her output. The same holds for the special case of zero-knowledge functionalities, namely for NIZK/OT protocols (e.g., [5,20]). Such attacks have been previously considered in other contexts, such as in the setting of verifiable outsourcing of computation [15], and seem notoriously difficult to eliminate.

1.1 Our Contribution

We start by showing that the above limitation of OT-based protocols is inherent. That is, we show:

Theorem 1.1 (Informal). *There is no information-theoretic and* reusably secure *implementation of NISC/OT for general functions, or even for* NC^1 *functions.*

We also prove a similar result for zero-knowledge functionalities, though in a more restricted "black-box" framework that is still broad enough to capture all existing NIZK/OT protocols.

Achieving Reusability with OLE. Towards bypassing this negative result, we make the following key observation: the 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 high level intuition is that by making the domain size of the receiver's selections super-polynomial, we can effectively eliminate the correlation between the receiver's OT inputs and the event of the receiver detecting failure. We note that OLE enjoys many of the

useful features of OT, including a self-reducibility property that enables shifting almost all of the implementation cost to an offline phase.

Our main result shows that this relaxation from OT to OLE is indeed helpful. We present a general-purpose *reusable* non-interactive secure computation protocol that only makes *parallel OLE calls* over a large field.[1] Here reusability means that the same receiver OLE inputs x_i can be used for polynomially many function evaluations while still ensuring full simulation-based security. This implies in particular that even if the sender learns full or partial information about the receiver's outputs (such as whether the receiver accepts or rejects), the sender cannot obtain more influence on the receiver's outputs than in an ideal function evaluation. We denote such a reusable NISC protocol by rNISC/OLE.

Theorem 1.2. *There exists a statistically secure rNISC/OLE protocol for (arithmetic or boolean) branching programs and* NC^1 *circuits. Evaluating an t-node n-variant branching program costs* $O(nt^3)$ *OLE calls. It comes with an efficient black-box straight-line simulator, the statistical simulation error is* $O(nt^3)/|\mathbb{F}|$.

Our rNISC/OLE protocol for branching programs 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 (NC^1) circuits. In the case of general polynomial-size circuits, we obtain a similar result, except that we also make use of a pseudorandom generator.

Theorem 1.3. *If one-way functions exist, there exists an rNISC/OLE protocol for circuits.*

In the important special case of zero-knowledge functionalities, where verification can always be done by a shallow circuit, our rNISC/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. Even in the single-shot case (without a reusability requirement), and even in the special case of zero-knowledge functionalities, similar NIZK/OLE protocols were not known prior to our work.

Theorem 1.4. *There exists a statistically secure rNIZK/OLE protocol (i.e., rNISC for zero-knowledge functionalities), such that proving the satisfiability of an arithmetic or boolean circuit costs* $O(1)$ *OLE calls per gate. It comes with an efficient black-box straight-line simulator, the statistical simulation error is* $O(\textit{circuit size})/|\mathbb{F}|$.

We optimize the concrete efficiency of our rNIZK/OLE protocol in the full version, so that proving knowledge of a satisfying assignment of an arithmetic

[1] Alternatively, settling for computational security, one can replace a field by a "pseudo-field," namely a ring whose description hides all zero-divisors. A useful example is the ring $\mathbb{Z}_N$ for an RSA modulus N with an unknown factorization.

circuit costs 7 OLE calls per addition gate and 44 OLE calls per multiplication gate. (Minimizing this constant is an interesting future research direction that can be motivated by practical implementations.) 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.

Constructing Reusable OLE. We complement our rNISC/OLE protocol by proposing an efficient secure implementation of the reusable OLE oracle which is compatible with our efficiency goals. Concretely, assuming the security of Paillier's encryption scheme [25], we construct a universally secure 2-message reusable 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 [26], which is commonly used in implementations of secure two-party computation (in particular, it is used by the non-interactive ones from [1]). Our efficient OLE protocol is independently motivated by other applications of OLE in cryptography; see [3,17] and references therein.

Theorem 1.5 (Informal). *Under the Paillier assumption, there is a reusable OLE scheme in the common reference string (CRS) model with a communication complexity of $O(1)$ group elements, and a computational cost of $O(1)$ group exponentiations. Moreover, depending on the CRS distribution, the OLE can be statistically secure against either the sender or the receiver.*

Combining our statistical NIZK/OLE with the OLE protocol in the statistical sender security mode, we get the first statistical NIZK argument for NP under an assumption related to factoring.

Theorem 1.6 (Informal). *Under the Paillier assumption, there exists a* statistical *designated-verifier NIZK argument for NP.*

On the Efficiency Benefits of Switching to OLE. 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. [21]), 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 [12,14], 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 additional bonus features, 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.

In addition, even in the stand-alone (non-reusable) world, our approach has its benefits. For example, as a corollary of our approach, we show that the satisfiability of an arithmetic circuit of size s can be proved in zero knowledge with soundness error $O(s)/|\mathbb{F}|$ via $O(s)$ parallel calls to (regular, non-reusable) OLE over $\mathbb{F}$. We also get an analogous result for NISC where the number of OLE calls depends polynomially on the arithmetic branching programs size.

1.2 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. [7] 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. [3] 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 [11] construct an alternative Paillier-based designated-verifier (reusable) NIZK protocol with a constant number of exponentiations per arithmetic gate. The constant from [11] 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 [11] (which is based on Σ-protocols) does not have the feature of a non-cryptographic online phase that our protocol inherits from the underlying information-theoretic OLE-based protocol.

There has been recent interest in the goal of constructing reusable NIZK protocols with different forms of setup from alternative assumptions such as LWE [10,22,28]. Our work provides a new avenue for constructing such protocols by reducing this goal to the construction of reusable 2-message OLE. The usefulness of our approach has been demonstrated in the recent work of Boyle et al. [8], which constructs (bounded) reusable NIZK with a correlated randomness setup from the Learning Parity with Noise (LPN) assumption over large fields. This construction can rely on a simplified honest-verifier variant of our rNIZK/OLE protocol, as the correlated randomness setup effectively restricts the verifier to be honest.

There are two main qualitative differences between our work and all of the recent NIZK-related works mentioned above. First, we obtain non-trivial (positive and negative) results on NIZK in a natural and well-motivated *information-theoretic* setting, whereas all of the above works are inherently cast in a computational setting. This information-theoretic aspect of our positive results is crucial for obtaining reusable NIZK protocols that have a non-cryptographic online phase given offline preprocessing. Second, our positive results apply to NISC, which is more general than NIZK in a useful way. While many different techniques for constructing NIZK protocols from different assumptions are known, including black-box constructions from cryptographically hard groups, our work provides the first black-box constructions of reusable NISC protocols of any kind.

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, even perfect security against malicious senders *does not* imply reusable security. In particular, previous works that construct NISC/OT protocols do not immediately imply rNISC/OT.

The second negative result shows that OLE is strictly stronger than OT in the sense that there exists *no* information-theoretic rNISC/OT protocol for the OLE functionality with composable security. Third and finally, assuming the existence of one-way functions, in the OT-hybrid model, we show that there are no general resettably sound, non-interactive zero-knowledge proofs with black-box simulation. We describe below an outline of the second and third impossibility results. For the technical details, we refer the reader to the full version.

OT is Not Sufficient for Reusability. The basic intuition behind the weakness ("non-reusability") of OT is the following: a malicious sender can learn the receiver's choice bits if the receiver uses the same randomness and input in different protocol runs. In particular, suppose the receiver's private OT choice bit has been fixed in a set-up phase. In the first protocol run, a malicious sender feeds (a, b) to the OT. In the second protocol run, the malicious sender feeds (a', b') to the OT where either $a \neq a'$ or $b \neq b'$ (but not both). If the receiver output is different in these two protocol runs, the malicious sender can deduce the receiver's choice bit.

Moreover, imagine implementing a functionality, such as reusable OLE or reusable NIZK, in the OT hybrid model. Such an implementation will involve the receiver calling the OT functionality using a vector of choice bits. Suppose now that the receiver has different outputs in two protocol runs where the sender chooses OT-input strings $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{a}', \mathbf{b}')$ respectively. The malicious sender can now modify $(\mathbf{a}, \mathbf{b})$ to equal $(\mathbf{a}', \mathbf{b}')$ 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 $(\mathbf{a}[j], \mathbf{b}[j]) \neq (\mathbf{a}'[j], \mathbf{b}'[j])$.

In the *OLE*-hybrid model, this intuition does not work. Consider a similar scenario: The receiver uses the same randomness and input in two protocol runs, let $\mathbf{x}[i] \in \mathbb{F}$ be the receiver's input in the i-th OLE instance. The malicious sender feeds $(\mathbf{a}[i], \mathbf{b}[i])$ and $(\mathbf{a}'[i], \mathbf{b}'[i])$ to the i-th OLE instance in these two protocol runs respectively. Say the $(\mathbf{a}[j], \mathbf{b}[j]) \neq (\mathbf{a}'[j], \mathbf{b}'[j])$ is the only difference between $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{a}', \mathbf{b}')$ and the receiver outputs differently in these two protocol runs. Given the above information, the malicious sender can only deduce that $\mathbf{x}[j] \neq -\frac{\mathbf{b}[j]-\mathbf{b}'[j]}{\mathbf{a}[j]-\mathbf{a}'[j]}$. Such knowledge contains little information if $\mathbf{x}[j]$ has large min entropy.

We now outline how to translate this intuition into concrete impossibility results for the OLE functionality first, and then the zero-knowledge functionality. Details can be seen in the full version.

First, we outline the impossibility of a statistically reusable non-interactive OLE/OT protocol. The intuition behind our impossibility proof relies on a commitment protocol. There is a statistically reusable commitment protocol in the 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 feeds (s_i, r_i) to the OLE oracle, so that the receiver gets OLE-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 has statistical

reusable security in the OLE-hybrid model. We show that in an OT-based implementation of the OLE primitive, a corrupted sender can recover the receiver's private 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 $(s, r, \mathbf{a}, \mathbf{b})$, then samples $(s', r', \mathbf{a}', \mathbf{b}')$ from the same distribution subject to the condition that $(\mathbf{a}', \mathbf{b}')$ agrees with $(\mathbf{a}, \mathbf{b})$ on the *known* receiver 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 strings are $\mathbf{a}, \mathbf{b}$. 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 $\mathbf{a}, \mathbf{b}$, while accept it when the sender's OT-input strings are $\mathbf{a}', \mathbf{b}'$. The sender will be able to learn at least one more receiver's choice bit from such a difference. At the end of this process, the sender learns all the *relevant* choice bits of the receiver, upon which he can sample an (s, r) and (s', r') pair that results in the same commitment $sx + r = s'x + r'$. This, by the calculation above, allows him to extract x.

Also in the full version we show that there is no UC secure rNISC/OT protocol for general zero-knowledge proof functionality. 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 computationally hard to distinguish a random yes instance from a 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 indices 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, breaking soundness. The details of this impossibility result are in the full version.

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, namely,

how to construct NISC with semi-honest security. As a starting point, we present a construction for the semi-honest model from the work of Ishai and Kushilevitz [18]. Then we will outline the main contribution of our work, namely, how to obtain (reusable) security against malicious parties.

Let $\mathbf{x}$ denote the receiver Rachel's input and let $\mathbf{y}$ denote the sender Sam's input. We consider arithmetic functionalities. Namely, both $\mathbf{x}, \mathbf{y}$ are vectors over a given finite field $\mathbb{F}$. The functionality is computed by an *arithmetic branching program*, defined as follows (see [18] 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 the matrix

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

Branching programs can efficiently simulate arithmetic formulas and arithmetic NC^1 circuits. For example, the formula $x_1y_1 + x_2y_2 + x_3y_3$ can be computed by the 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 calls can be viewed as an arithmetic analogue of Yao's garbled circuit technique [4,29]. In a nutshell, the construction of the two-party protocol works as follows: The sender Sam samples two random upper triangular matrixes R_1, R_2 with an all-one diagonal. Observe that the matrix $R_1\mathsf{G}(\mathbf{x},\mathbf{y})R_2$ is a *randomized encoding* of $\det \mathsf{G}(\mathbf{x},\mathbf{y})$ since:

- $\det \mathsf{G}(\mathbf{x},\mathbf{y})$ can be computed from $R_1\mathsf{G}(\mathbf{x},\mathbf{y})R_2$ because multiplying by R_1 and R_2 preserves the determinant; and
- the distribution of $R_1\mathsf{G}(\mathbf{x},\mathbf{y})R_2$ merely depends on $\det \mathsf{G}(\mathbf{x},\mathbf{y})$ and not on $\mathbf{x}$ and $\mathbf{y}$. (This depends crucially on the structure of G, in particular the fact that the one-off diagonal of $\mathbf{G}$ consists of -1; we refer the reader to [18] for more details.)

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

Now to construct a semi-honest NISC/OLE protocol, we use the fact that the OLE functionality allows secure evaluation of affine functions. Therefore, the receiver chooses $\mathbf{x}$ as its input, and the sender feeds 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 G', i.e. $\mathsf{G}'(\mathbf{x}) = \mathsf{G}'_{R_1,R_2,\mathbf{y}}(\mathbf{x}) := R_1\mathsf{G}(\mathbf{x},\mathbf{y})R_2$. 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.

Additionally, this NISC/OLE protocol is perfectly secure against *malicious receivers*, *if the underlying OLE protocol is reusable*. Since this is the first time reusability rears its head, let us explain in a bit more detail why this is the case. The affine function in question can be thought of as

$$\mathbf{x} \mapsto R_1 \mathsf{G}(\mathbf{x}, \mathbf{y}) R_2 := \mathbf{v}_0 + \sum_{i \in [n]} x_i \mathbf{v}_i$$

for some vectors $\mathbf{v}_0, \ldots, \mathbf{v}_n$ chosen by the sender (as functions of $\mathbf{y}$, R_1 and R_2.) Now, it turns out to be easy to create a functionality out of (non-reusable) OLE where the sender inputs $(\mathbf{v}_0[j], \mathbf{v}_1[j], \ldots, \mathbf{v}_n[j])$, the receiver inputs $\mathbf{x} := (x_1, \ldots, x_n)$, and the receiver obtains $\mathbf{v}_0[j] + \sum_{i \in [n]} x_i \mathbf{v}_i[j]$. That is, each coordinate of the computation above can be realized using (non-reusable) OLE. However, using non-reusable OLE to compute the entire output by repeating this process once per co-ordinate runs into a serious issue when the receiver Rachel is malicious: she can feed the different instances of OLE with different values of $\mathbf{x}$. On the other hand, if the underlying OLE functionality is *reusable*, it permits the receiver to send a single message $\mathbf{x}$ that can be used for multiple invocations of OLE, *ipso facto* forcing Rachel to be semi-honest.

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

This is *the* key problem that remains to be solved. We now describe how to achieve this goal in a number of steps that make this task successively simpler, eventually reducing everything to reusable OLE.

An Intermediate Primitive: Certified OLE. Certified OLE is a specialized OLE wherein the sender can prove that the coefficients he chose satisfies some arithmetic conditions. More precisely, we define certified OLE as a primitive that allows:

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

We will implement a CertifiedOLE/OLE construction in the reusable world, whose security is information-theoretic.

An Intermediate Primitive: Replicated OLE. Certified OLE allows the sender to prove that his coefficients satisfy arbitrary arithmetic constraints. In particular, the sender can prove an equality constraint, i.e., prove that two of the coefficients she chose are equal. We isolate this ability into another (weaker) 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, as before, the inputs are chosen by the receiver and the coefficients by the sender;
– the sender to convince the receiver that some of the sender-chosen coefficients are equal.

Replicated OLE is not as powerful as certified OLE, yet it is an important stepping stone to our eventual construction. In the corresponding section in the full version, we first construct replicated OLE directly from OLE, then construct certified OLE from replicated OLE[2]. For now, let us assume that we already have reusable replicated OLE, and we will construct (reusable) certified OLE using replicated OLE as a black box.

To begin with, note that to construct certified OLE, it is sufficient to 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 few letters in the alphabet such as a, b, c and inputs chosen by the receiver will be denoted by the last few 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 supported by replicated OLE. We will implement latter two using calls to replicated OLE.

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 uses it as an input to OLE. The sender samples random $a', b' \in \mathbb{F}$ and sets $c' = a' + b'$. The replicated OLE is used to reveal $ax + a', bx + b', cx + c'$ to the receiver. Clearly, the receiver cannot cheat; no matter which x be picks, he will receive three values, the first two of which are random and the third is the sum of the first two. How about a cheating sender? Note that the receiver is convinced if and only if $(ax + a') + (bx + b') = (cx + c')$. Since x is randomly chosen by the receiver and hidden from the sender, in case the sender sets $a + b \neq c$ (or $a' + b' \neq c'$), the receiver can detect this with overwhelming probability.

The last atomic operation, i.e., proving $ab = c$, is implemented using a similar idea. The receiver samples random $x, y \in \mathbb{F}$, sets $z = xy$ and uses x, y, z as inputs to an OLE. Note that a malicious receiver might choose $z \neq xy$ and the sender can never detect this. Therefore, we have to design a mechanism that can "enforce" honest receiver behavior. More precisely, our mechanism should prevent the receiver from learning any information in case he chooses $z \neq xy$. We will explain the details of this mechanism later; for now, let us simply assume the receiver chooses $z = xy$.

[2] The actual roadmap is somewhat different, and will be gradually revealed in this overview. An impatient reader is referred to Fig. 1 at the end of the overview.

The sender samples random $a', b', c' \in \mathbb{F}$, sets $d = ab', e = a'b$ 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 convinced if and only if $(ax + a')(by + b') - (cz + c') - (dx + d') - (ey + e') = 0$. 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, the receiver will detect this with overwhelming probability. To prove this, we consider $(ax + a')(by + b') - (cxy + c') - (dx + d') - (ey + e')$ as a polynomial in variables x and y, which equals

$$(ab - c)xy + (ab' - d)x + (a'b - e)y + a'b' - c' - d' - e',$$

This turns out to be a non-zero polynomial as long as the sender deviates from the protocol. Thus by sampling random $x, y \in \mathbb{F}$, the receiver will detect cheating with overwhelming probability.

Now, there are two outstanding issues we have not handled: (1) in the description above, we assumed that the receiver chooses x, y, z such that $z = xy$ honestly; and (2) we haven't yet shown how to construct replicated OLE, starting from (reusable) OLE. We will both of these in turn.

An Intermediate Primitive: Half-Replicated OLE. Our replicated OLE is constructed on top of what we call *half-replicated OLE*. In each OLE call, the sender chooses two coefficients. We separate them, and call them the *multiplicative coefficient* and the *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 (as before); and
2. Allow the sender to convince the receiver that two *multiplicative* coefficients are equal.

Half-replicated OLE is even weaker than replicated OLE. We first construct replicated OLE on top of half-replicated OLE by the following protocol: The receiver samples random y and sets it as an input to OLE. For each receiver-chosen input x, the receiver let $x' = xy$ and sets it as an extra input to OLE. Notice that as before, the sender cannot detect whether the receiver generated the tuple (x, y, x') honestly. Therefore, we have to design a mechanism that can enforce $x' = xy$. Again, analogous to the construction of certified OLE from replicated OLE, we will defer this to later; for now, just assume the receiver chooses $x' = xy$ honestly.

The sender uses the replicated OLE to reveal $ax + b$ and $ax' + by$ to the receiver. (More precisely, the sender does this by sampling a random c and revealing $ax + b, ax' + c, by - c$ to the receiver.) The receiver then uses the identity $(ax + b) \cdot y = (ax' + c) + (by - c)$ to check whether the sender behaves honestly.

Using a completely analogous argument as before, we can show that the receiver catches a cheating sender with high probability. (For this argument to work, we need the fact that the sender uses the same a in the first two invocations, but we already have this by the half-replicated guarantee since a is a *multiplicative* coefficient.)

At this point, there are *three* outstanding issues we have not handled: (1) in the construction of certified OLE from replicated OLE, we assumed that the receiver chooses x, y, z such that $z = xy$ honestly; (2) in the construction of replicated OLE from half-replicated OLE, we assumed that the receiver chooses x, y, x' such that $x' = xy$ honestly; and (3) we haven't yet shown how to construct half-replicated OLE, starting from (reusable) OLE. (1) and (3) are issues we already saw, but we just added (2) to our list. (In fact, as the reader might observe, (1) and (2) are really the same issue.)

We will first solve issues (1) and (2) by introducing the primitive of half-replicated OLE allowing CDS operations.

An Intermediate Primitive: Half-Replicated OLE Allowing CDS Operations. Our replicated OLE and certified OLE require the receiver to choose three inputs x, y, z such that $z = xy$. Unfortunately, there was no means for the sender to detect whether the receiver behaves honestly, and we left this problem open.

We design a mechanism called *conditional disclosure of secrets (CDS)*, in which the sender can disclose 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.

We now show how to design a half-replicated OLE allowing CDS operations starting from any half-replicated OLE.

As a first try, 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 is not hard to verify security against malicious receiver. When $z \neq xy$, the 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 a malicious sender: As the protocol is built on top of half-replicated OLE, the sender can deviate from 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, and constitutes a deviation from the protocol.

An easy way to solve this problem is to rely on the fact that the (honest) receiver samples $y \in \mathbb{F}$ uniformly at random, and we can use this ability to fight against the malicious sender.[3] The sender samples a random a' as an extra coefficient and uses the above insecure CDS protocol (with freshly sampled b' and c' in the place of b and c) to disclose a' if $z = xy$. If the sender is malicious, then the receiver gets $a' - d'y$.

Finally, the receiver can now detect malicious behaviour, by running a third subprotocol: sample a random $w \in \mathbb{F}$ as an extra input and ask the sender to disclose $aw + a'$ using OLE.

In summary, there are three sub-protocols going on here:

1. In sub-protocol 1, the sender inputs (an arbitrary) a and uniformly random b, c and the receiver inputs x, y, z and the receiver gets

$$\left[\begin{smallmatrix} y & z \\ 1 & x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ c \end{smallmatrix}\right] + \left[\begin{smallmatrix} a \\ 0 \end{smallmatrix}\right]$$

2. In sub-protocol 2, the sender inputs uniformly random a', b', c' and the receiver inputs (the same) x, y, z and the receiver gets

$$\left[\begin{smallmatrix} y & z \\ 1 & x \end{smallmatrix}\right]\left[\begin{smallmatrix} b' \\ c' \end{smallmatrix}\right] + \left[\begin{smallmatrix} a' \\ 0 \end{smallmatrix}\right]$$

3. In sub-protocol 3, the sender inputs a and a', the receiver inputs a random w and gets $a' + wa$.

The receiver has no cheating room here. The reusability of the underlying (half-replicated) OLE forces her to use the same x, y and z in the subprotocols. Furthermore, if she chooses $z \neq xy$, she gets nothing, as we argued above. Finally, choosing w arbitrarily in the third subprotocol doesn't help her either due to the randomness of a'.

As for a cheating sender, the details of the argument are somewhat more complex but it is very similar in spirit to earlier arguments of the same flavor.

In turn, it is not hard to see that fortifying half-replicated OLE with CDS operations (as we just did) solves both problems (1) and (2) discussed above. It remains to solve (3), namely constructing a (reusable) half-replicated OLE protocol starting from any reusable OLE.

Revisiting Half-Replicated OLE. The last missing piece is to construct half-replicated OLE in the (reusable) OLE-hybrid model. The key idea of the construction is the following.

The receiver samples a random $w \in \mathbb{F}$ and sets w as an input to the OLE. 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

[3] In the main body, we do not need to assume that y is random. Moreover, we will consider more general arithmetic conditions beyond $z = xy$.

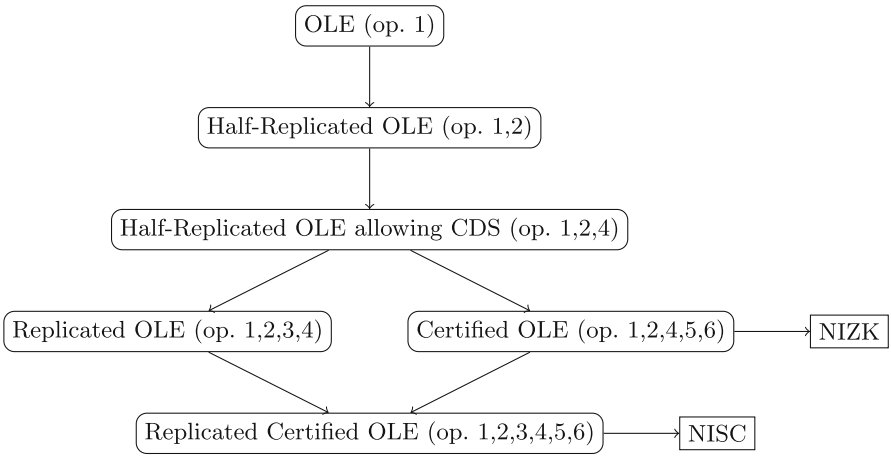

Fig. 1. Primitives (and supported operations – described below in text – in the bracket). We remark that "Replicated OLE" in this figure is only defined and used in the overview. In the main body, our proof follows the other path, directly constructing "Replicated Certified OLE".

a commitment of a. 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 $z = x - wy$.

Each half-replicated OLE call $ax + b$ can be translated into three OLE calls using the 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 Eq. 1.

We refer to this half-replicated OLE protocol as $\Pi_{\alpha\text{-}\frac{1}{2}\mathtt{repOLE}}$ in the main body. The correctness of such a half-replicated OLE protocol is straight-forward. In this protocol, the sender can cheat without being detected by the receiver. Instead, when the sender deviates from the protocol, the receiver will output a random number. Moreover, as the randomness comes from w and y which are sampled by the receiver, the receiver's output is statistically close to the uniform distribution, even conditioned on the sender's view and x. Therefore it is not hard to embed another mechanism which detects any malicious sender behaviour with overwhelming probability. We leave the details to the full version.

Roadmap. We defer the detailed presentation of the results in this subsection to the full version. There, 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. Convince the receiver that two coefficients are equal.
4. Disclose a message to the receiver if receiver-chosen inputs x, y, z satisfies $z = xy$.
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 Fig. 1 by dependence. Each of them only supports a subset of the operations.

A Corollary: Single-Shot (Non-reusable) NISC/OLE and NIZK/OLE. As a corollary of our techniques, we get a (non-reusable) NISC/OLE protocol with interesting features. In particular, we get a single-shot NISC/OLE protocol where the number of OLE calls depends polynomially on the arithmetic branching program size, and the simulation is statistical with an error of poly(branching program size)$/|\mathbb{F}|$. In the special case of the zero-knowledge functionality, we get a single-shot NIZK/OLE protocol which (a) uses $O(1)$ (non-reusable) OLE calls per gate of the verification circuit; and (b) is entirely non-cryptographic in its online phase.

These results are proved by combining the following two facts:

- Our reusable NISC/OLE protocol immediately implies a single-shot NISC protocol in the (non-reusable) *vector-OLE* hybrid model. Vector OLE is a generalization of OLE where the receiver inputs a scalar $x \in \mathbb{F}$ and a number $k \in \mathbb{N}$, the sender gets k and inputs a pair of vectors $(\mathbf{a}, \mathbf{b}) \in (\mathbb{F}^k)^2$, and the receiver obtains $\mathbf{a}x + \mathbf{b} \in \mathbb{F}^k$. Vector OLE can be viewed as reusable OLE under the constraints that the number of OLE calls is known in the choice phase, and all OLE calls are non-adaptive. Our simulator also fits this (non-reusable) protocol.
- The result of Döttling, Kraschewski and Müller-Quade [13] shows an efficient equivalence between OLE and vector OLE. In particular, they show a constant rate statistical vector-OLE protocol in OLE hybrid model. It also comes with an efficient straight-line simulator achieving O(communication)$/|\mathbb{F}|$ statistical soundness error.

Putting these together, we get our single-shot NISC/OLE and NIZK/OLE protocols.

2.3 Paillier-Based 2-Message OLE Protocol

In this subsection, we provide a quick overview of our Paillier-based instantiation of reusable OLE. For more details, we refer the reader to Sect. 4.

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 α, 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 α 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.[4]

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 α 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 α. Suppose the receiver runs the scheme twice, once with a random input γ, 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 γ, so the probability that the resulting linear combination of two invalid ciphertexts decrypts correctly is negligible.[5] 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.

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

[5] 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.

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 and 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 the full version) and linear functions (see Sect. 3.2).

Notation 1 (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* $z := f(x, y) \in R_{\text{out}}$*.*

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

Parametrized by a sender-receiver function $F = (R_{\text{in}}, S_{\text{in}}, R_{\text{out}}, f)$ in the sense of Notation 1.

Choice phase:

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

Send phases:

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

Fig. 2. Generic ideal functionality for reusable non-interactive secure computation.

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 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 formally defined in the full version.

3.2 Reusable Oblivious Linear Function Evaluation

We aim at an OLE-based implementation of $\mathcal{F}_{\mathtt{rNISC}}^{(\Phi)}$ for arbitrary arithmetic circuits Φ 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 Φ 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 his 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 is another special instance of the functionality $\mathcal{F}_{\mathtt{rNISC}}^{(F)}$ from Fig. 2, namely with $S_{\text{in}} = \mathcal{R} \times \mathcal{R}$, $R_{\text{in}} = R_{\text{out}} = \mathcal{R}$, and $f : R_{\text{in}} \times S_{\text{in}} \to R_{\text{out}}$, $(x, (a, b)) \mapsto a \cdot x + b$ (Fig. 3).

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

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

Setup/choice phase:

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

Send phases:

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

Fig. 3. Ideal functionality for reusable oblivious linear function evaluation over a ring $\mathcal{R}$.

4 A Reusable OLE Construction Based on Paillier

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

- $\mathsf{CRSSetup}(1^\lambda)$: Sample primes p', q' of the appropriate length for security parameter k such that $p = 2p' + 1, q = 2q' + 1$ are also primes. Let $N = pq$, $h = N + 1$ and $T = 2^\lambda N^2$. All operations will be in $\mathbb{Z}_{N^2}$ unless otherwise specified. Sample $w', W_0' \leftarrow \mathbb{Z}_{N^2}^*$ and let $w = (w')^{2N}, W_0 = (W_0')^{2N} h$.
  Output $crs = (N, h, w, W_0, T)$.
- $\mathsf{CRSDualSetup}(1^\lambda)$: Sample N, h, w, T the same way as in $\mathsf{CRSSetup}(1^\lambda)$. Sample $W_0' \leftarrow \mathbb{Z}_{N^2}^*$ and let $W_0 = (W_0')^{2N}$.
  Output $crs = (N, h, w, W_0, T)$.
- $\mathsf{ReceiverRequest}(crs, x)$: Parse $crs = (N, h, w, W_0, T)$. Sample $sk_1, sk_2, x_1 \leftarrow [T]$, let $x_2 := x - x_1$. Send $W_1 = w^{sk_1} W_0^{-x_1}$ and $W_2 = w^{sk_2} W_0^{-x_2}$ to the sender. Output state (sk_1, sk_2, x_1, x_2).
- $\mathsf{SenderResponse}(crs, (W_1, W_2), a, b)$: Parse $crs = (N, h, w, W_0, T)$. Sample $r \leftarrow [T]$, $b_1 \leftarrow \mathbb{Z}_N$. Let $b_2 := b - b_1$. Send $v = w^r$, $V_0 = {W_0}^r h^a$, $V_1 = {W_1}^r h^{b_1}$ and $V_2 = {W_2}^r h^{b_2}$ to the receiver.
- $\mathsf{ReceiverReceive}(crs, (v, V_0, V_1, V_2), (sk_1, sk_2, x_1, x_2))$: Compute $Z_1 = {V_0}^{x_1} V_1 / v^{sk_1}$ and $Z_2 = {V_0}^{x_2} V_1 / v^{sk_2}$. If it is not the case that Z_1^2 and Z_2^2 are of the form $1 + z_1 N$ and $1 + z_2 N$ for some $z_1, z_2 \in \mathbb{Z}_N$, then output $\perp$. Otherwise output $z = (z_1 + z_2)/2$.

We first show correctness when both parties are honest. Then the response is computed as follows, for $i \in \{0, 1\}$

$$
\begin{aligned}
Z_i &= {V_0}^{x_i} V_i / v^{sk_i} && \text{(from ReceiverReceive)} \\
&= ({W_0}^r h^a)^{x_i} W_i^r h^{b_i} / (w^r)^{sk_i} && \text{(from SenderResponse)} \\
&= ({W_0}^{x_i} W_i / w^{sk_i})^r h^{ax_i + b_i} \\
&= (w^{sk_i} / w^{sk_i})^r h^{ax_i + b_i} && \text{(from ReceiverRequest)} \\
&= h^{ax_i + b_i}.
\end{aligned}
$$

Note that $(Z_i)^2 = h^{2(ax_i + b_i)} = 1 + 2(ax_i + b_i)N$. So the $z_i = 2(ax_i + b_i)$ and the receiver will output $z = (z_1 + z_2)/2 = a(x_1 + x_2) + (b_1 + b_2) = ax + b \mod N$.

Theorem 4.1. *$\Pi_{\texttt{rOLE}}$ is a UC-secure realization of the reusable OLE functionality $\mathcal{F}_{\texttt{rOLE}}$ over the ring $\mathbb{Z}_N$. Moreover, the statistical simulation error against malicious receiver is negligible when the CRS is generated by* $\mathsf{CRSSetup}$*; the statistical simulation error against malicious sender is negligible when the CRS is generated by* $\mathsf{CRSDualSetup}$*.*

4.1 Indistinguishability of CRS

Lemma 4.2. *The CRS generated from* $\mathsf{CRSDualSetup}$ *is indistinguishable from the CRS generated by* $\mathsf{CRSSetup}$ *as long as the decisional composite residuosity assumption (DCRA) holds.*

Proof. Let N, h, w, T be generated as in $\mathsf{CRSSetup}$, and W_0' be sampled uniformly from $\mathbb{Z}_{N^2}^*$. By DCRA, W_0' and $(W_0')^N$ are indistinguishable even given N, h, w, T. Therefore, $(W_0')^2$ and $(W_0')^{2N}$ are indistinguishable, and $(W_0')^2 h$ and $(W_0')^{2N} h$ are also indistinguishable. Moreover, $(W_0')^2$ equals $(W_0')^2 h$ in distribution as $h = h^{N+1} = (h^{\frac{N+1}{2}})^2$ is also a quadratic residue.

In a nutshell,

$$\Big[\underbrace{W_0 = (W_0')^{2N}}_{\text{generated by } \mathsf{CRSDualSetup}} \Big] \approx_C \underbrace{\Big[W_0 = (W_0')^2\Big] \stackrel{d}{=} \Big[W_0 = (W_0')^2 h\Big]}_{\text{a random quadratic residue}} \approx_C \Big[\underbrace{W_0 = (W_0')^{2N} h}_{\text{generated by } \mathsf{CRSSetup}} \Big].$$

The CRS distributions produced by $\mathsf{CRSSetup}$ and $\mathsf{CRSDualSetup}$ are indistinguishable. Thus the statistical UC-security against malicious sender in dual mode implies the computational version of the same security in primal mode; and vice versa, the statistical UC-security against malicious receiver in primal mode implies the computational version of the same security in dual mode. Moreover, the computational UC-security would be preserved if the CRS is sampled from any other computationally indistinguishable distribution.

4.2 Statistical Security Against Malicious Receiver

Let $G_{4p'q'}$ be the subgroup of $\mathbb{Z}_{N^2}^*$ consisting of elements of the form $w = (w')^N$ for $w' \in \mathbb{Z}_{N^2}^*$. $G_{4p'q'}$ is isomorphic to $\mathbb{Z}_N^*$ and $\mathbb{Z}_{p'} \times \mathbb{Z}_2 \times \mathbb{Z}_{q'} \times \mathbb{Z}_2$. Consider the following simulator $\mathcal{S}$.

- CRS is generated as in $\mathsf{CRSSetup}$, but stores the factorization of N.
- When the adversary sends (w, W_1, W_2), the simulator proceeds as follows: Use the factorization of N to compute $U_0, U_1, U_2 \in G_{4p'q'}$ and $\hat{x}_1, \hat{x}_2 \in \mathbb{Z}_N$ such that $W_0 = U_0 h$, $W_1 = U_1 h^{-\hat{x}_1}$ and $W_2 = U_2 h^{-\hat{x}_2}$. Send $\hat{x}_1 + \hat{x}_2$ to $\mathcal{F}$.
- When receive z from $\mathcal{F}$: Sample random $r \leftarrow \mathbb{Z}_{2p'q'}$, $s_0, s_1 \leftarrow \mathbb{Z}_N$, and compute

$$\begin{aligned} v &= w^r, \\ V_0 &= U_0^r h^{s_0}, \\ V_1 &= U_1^r h^{s_1}, \\ V_2 &= U_2^r h^{z - s_0(\hat{x}_1 + \hat{x}_2) - s_1}. \end{aligned}$$

  Send (v, V_0, V_1, V_2) to A.

Lemma 4.3. *The environment's view in the real world is statistically close to its view when interacting with simulator $\mathcal{S}$ and functionality $\mathcal{F}$ as defined above.*

Proof. The sender samples $r \leftarrow [T]$ in the real game. Let r' be its mod-$2p'q'$ component and r'' be its mod-N component. Since $T/2p'q'N$ is exponential in the security parameter, the joint distribution of (r', r'') is statistically close to uniform distribution over $\mathbb{Z}_{2p'q'} \times \mathbb{Z}_N$. In the real game, the sender will response

$$\begin{aligned} v &= w^r & &= w^{r'}, \\ V_0 &= {W_0}^r h^a = {U_0}^r h^{r+a} & &= {U_0}^{r'} h^{r''+a}, \\ V_1 &= {W_1}^r h^{b_1} = {U_1}^r h^{-r\hat{x}_1+b_1} & &= {U_1}^{r'} h^{-r''\hat{x}_1+b_1}, \\ V_2 &= {W_2}^r h^{b_2} = {U_2}^r h^{-r\hat{x}_2+b_2} & &= {U_2}^{r'} h^{-r''\hat{x}_2+b_2}. \end{aligned}$$

Now we will argue that the environment's view here is identical to its view when interacting with simulator $\mathcal{S}$ and functionality $\mathcal{F}$. Set $s_0 := r'' + a$ and $s_1 := -r''\hat{x}_1 + b_1$. Then (s_0, s_1) is uniformly random in $\mathbb{Z}_N \times \mathbb{Z}_N$ due to the (uniform) randomness of r'' and b_1. The resulting (v, V_0, V_1, V_2) is as follows:

$$\begin{aligned} v &= w^{r'}, \\ V_0 &= {U_0}^{r'} h^{r''+a} = {U_0}^{r'} h^{s_0}, \\ V_1 &= {U_1}^{r'} h^{-r''\hat{x}_1+b_1} = {U_1}^{r'} h^{s_1}, \\ V_2 &= {U_2}^{r'} h^{-r''\hat{x}_2+b_2} = {U_2}^{r'} h^{-r''\hat{x}_1+b-b_1} = {U_2}^{r'} h^{-r''(\hat{x}_1+\hat{x}_2)+b-s_1} \\ &= {U_2}^{r'} h^{(a-s_0)(\hat{x}_1+\hat{x}_2)+b-s_1} = {U_2}^{r'} h^{z-s_0(\hat{x}_1+\hat{x}_2)-s_1} \end{aligned}$$

for $z := a(\hat{x}_1 + \hat{x}_2) + b$. Finally, note that this is exactly the distribution that would be produced by the simulator.

4.3 Statistical Security Against Malicious Sender in Dual Mode

The group $\mathbb{Z}^*_{N^2}$ is isomorphic to $\mathbb{Z}_N \times (\mathbb{Z}_2)^2 \times \mathbb{Z}_{p'q'}$. Thus it can be decomposed into the following three groups. Let G_N be the subgroup of $\mathbb{Z}^*_{N^2}$ expended by h, which is isomorphic to $\mathbb{Z}_N$. Let G_4 be the subgroup of $\mathbb{Z}^*_{N^2}$ consisting of elements of the form $x = x'^{p'q'N}$ for $x' \in \mathbb{Z}^*_{N^2}$, which is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2$. Let $G_{p'q'}$ be the subgroup of $\mathbb{Z}^*_{N^2}$ consisting of elements of the form $x = x'^{2N}$ for $x' \in \mathbb{Z}^*_{N^2}$, which is isomorphic to $\mathbb{Z}_{p'q'}$. For every element $x \in \mathbb{Z}^*_{N^2}$, there exists an unique decomposition $(a, b, c) \in G_{p'q'} \times G_4 \times \mathbb{Z}_N$ such that $x = ab(1 + cN)$.

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

- To generate the CRS, the simulator generates N, h, w, T as in CRSDualSetup. It then sample random $sk_0 \leftarrow \mathbb{Z}^*_{p'q'}$, let $W_0 = w^{sk_0}$, and outputs $crs = (N, h, w, W_0, T)$.
- The simulator generates W_1, W_2 as follows: sample random $sk'_1, sk'_2 \leftarrow \mathbb{Z}_{p'q'}$ and send $W_1 = w^{sk'_1}$ and $W_2 = w^{sk'_2}$.
- When the adversary responses (v, V_0, V_1, V_2), proceed as follows: Compute $C_0 = V_0/v^{sk_0}$, $C_1 = V_1/v^{sk'_1}$, and $C_2 = V_2/v^{sk'_2}$. If it is not the case that C_0^2, C_1^2, and C_2^2 are of the form $1 + c_0N$, $1 + c_1N$ and $1 + c_2N$ for some $c_0, c_1, c_2 \in \mathbb{Z}_N$, then send $\perp$ to $\mathcal{F}$. Otherwise send $c_0/2$ and $(c_1 + c_2)/2$ to $\mathcal{F}$.

Lemma 4.4. *The environment's view when interacting with simulator $\mathcal{S}$ and $\mathcal{F}$ as defined above, is statistically close to its view in the real game when the CRS is generated by* CRSDualSetup.

Proof. The CRS produced by the simulator $\mathcal{S}$ is statistically close to the CRS produced by CRSDualSetup. CRSDualSetup generates W_0 as a fresh sample from uniform distribution over $G_{p'q'}$. When w is a generator of $G_{p'q'}$, which happens with overwhelming probability $1 - \frac{1}{p'} - \frac{1}{q'}$, the simulator will also sample W_0 uniformly from $G_{p'q'}$.

We consider a variation of the real game where the receiver samples (sk_1, sk_2, x_1) from $\mathbb{Z}_{p'q'} \times \mathbb{Z}_{p'q'}$ In the real game, the receiver will sends $W_1 = w^{sk_1} W_0^{-x_1} = w^{sk_1 - sk_0 x_1}$ and $W_2 = w^{sk_2} W_0^{-x_2} = w^{sk_2 - sk_0 x_2}$. Let $sk_1' := sk_1 - sk_0 x_1 \mod p'q'$, $sk_2' := sk_2 - sk_0 x_2 \mod p'q'$, then (sk_1', sk_2', x_1) is statistically close to the uniform distribution over $\mathbb{Z}_{p'q'} \times \mathbb{Z}_{p'q'} \times [T]$, and the receiver will send $W_1 = w^{sk_1'}$ and $W_2 = w^{sk_2'}$, which is the same as what the simulator $\mathcal{S}$ will send.

Finally, the environment responses the receiver with message (v, V_0, V_1, V_2). We show how the receiver's behavior is simulated with negligible statistical error. The simulator defines intermediate variables $C_0 = V_0 / v^{sk_0}, C_1 = V_1 / v^{sk_1'}, C_2 = V_2 / v^{sk_2'}$. The intermediate variables used by the receiver can be expressed as

$$Z_i = V_0^{x_i} V_i / v^{sk_i} = V_0^{x_i} V_i / v^{sk_0 x_i + sk_i'} = (C_0)^{x_i} C_i.$$

The receiver in the real game will abort unless both $(Z_1)^2$ and $(Z_2)^2$ lay inside G_N. We decompose $(C_0)^2, (C_1)^2, (C_2)^2$ into their components in $G_{p'q'}, G_4, G_N$. Let $D_0, D_1, D_2 \in G_{p'q'}$, $c_0, c_1, c_2 \in \mathbb{Z}_N$ be such that $(C_i)^2 = D_i(1 + c_i N)$, note that squaring erases the component in G_4.

In the case when $D_0 = D_1 = D_2 = 1$, the receiver will compute $(Z_1)^2 = 1 + (c_0 x_1 + c_1)N$ and $(Z_2)^2 = 1 + (c_0 x_2 + c_2)N$, then output

$$z = (z_1 + z_2)/2 = \frac{c_0 x_1 + c_1 + c_0 x_2 + c_2}{2} = c_0 x + (c_1 + c_2)/2.$$

In the ideal world, the simulator will feed c_0 and $(c_1 + c_2)/2$ to the functionality $\mathcal{F}$, and the functionality will produce the same output.

In the case when at least one of D_0, D_1, D_2 is not the identity element, the simulator $\mathcal{S}$ will output $\perp$. We show the receiver will also abort with overwhelming probability. W.l.o.g. assume $(D_0, D_1) \neq (1, 1)$, the receiver will abort if $(Z_1)^2 \notin G_N$ where

$$(Z_1)^2 = (C_0)^{2x_i} C_1 = (D_0)^{x_1} D_1 h^{c_0 x_1 + c_1}.$$

This condition is satisfied with overwhelming probability as x_1 is uniformly random in $[T]$.

Acknowledgements. Y. Dodis was supported in part by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546. Y. Ishai was supported by ERC Project NTSC (742754), ISF grant 1709/14, NSF-BSF grant 2015782, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India. D. Kraschewski Supported by the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement

no. 259426 - ERC Cryptography and Complexity. Work mostly done while at the Technion. T. Liu was supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, an MIT-IBM grant and a DARPA Young Faculty Award. R. Ostrovsky was supported by NSF grant 1619348, BSF grant 2015782, DARPA SafeWare subcontract to Galois Inc., DARPA SPAWAR contract N66001-15-C-4065, JP Morgan Faculty Research Award, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government. V. Vaikuntanathan was supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, an MIT-IBM grant and a DARPA Young Faculty Award.

References

1. Afshar, A., Mohassel, P., Pinkas, B., Riva, B.: Non-interactive secure computation based on cut-and-choose. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 387–404. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_22
2. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_8
3. Applebaum, B., Damgård, I., Ishai, Y., Nielsen, M., Zichron, L.: Secure arithmetic computation with constant computational overhead. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 223–254. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_8
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. SIAM J. Comput. **43**(2), 905–929 (2014)
5. Bellare, M., Micali, S., Ostrovsky, R.: The (true) complexity of statistical zero knowledge. In: Ortiz, H. (ed.) Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, Baltimore, Maryland, USA, 13–17 May 1990 (1990)
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA, 2–4 May 1988 (1988)
7. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 336–365. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70700-6_12
8. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, 15–19 October 2018 (2018)
9. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011 (2011)
10. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D.: Fiat-Shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004 (2018)

11. Chaidos, P., Couteau, G.: Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 193–221. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_7
12. Döttling, N., Ghosh, S., Nielsen, J.B., Nilges, T., Trifiletti, R.: TinyOLE: efficient actively secure two-party computation from oblivious linear function evaluation. In: CCS 2017 (2017)
13. Döttling, N., Kraschewski, D., Müller-Quade, J.: Statistically secure linear-rate dimension extension for oblivious affine function evaluation. In: Smith, A. (ed.) ICITS 2012. LNCS, vol. 7412, pp. 111–128. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32284-6_7
14. Genkin, D., Ishai, Y., Prabhakaran, M., Sahai, A., Tromer, E.: Circuits resilient to additive attacks with applications to secure computation. In: STOC 2014 (2014)
15. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_25
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May–2 June 2009 (2009)
17. Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 629–659. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_22
18. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45465-9_22
19. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_23
20. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. **39**(3), 1121–1152 (2009)
21. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D.A. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32
22. Kim, S., Wu, D.J.: Multi-theorem preprocessing NIZKs from lattices. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 733–765. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_25
23. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. J. Cryptol. **22**(2), 161–188 (2009)
24. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, Washington, DC, USA, 7–9 January 2001 (2001)
25. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16

26. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_31
27. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In: DeMillo, R.A., Dobkin, D.P., Jones, A.K., Lipton, R.J. (eds.) Foundations of Secure Computation (1978)
28. Rothblum, R.D., Sealfon, A., Sotiraki, K.: Towards non-interactive zero-knowledge for NP from LWE. IACR Cryptology ePrint Archive 2018, 240 (2018)
29. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: Proceedings of FOCS 1986 (1986)