Actively Secure Garbled Circuits with Constant Communication Overhead in the Plain Model*

Carmit Hazay† Yuval Ishai‡ Muthuramakrishnan Venkitasubramaniam§

Abstract

We consider the problem of constant-round secure two-party computation in the presence of active (malicious) adversaries. We present the first protocol that has only a *constant* multiplicative communication overhead compared to Yao's protocol for passive adversaries, and can be implemented in the *plain model* by only making a black-box use of (parallel) oblivious transfer and a pseudo-random generator. This improves over the polylogarithmic overhead of the previous best protocol. A similar result could previously be obtained only in an amortized setting, using preprocessing, or by assuming bit-oblivious-transfer as an ideal primitive that has a constant cost.

We present two variants of this result, one which is aimed at minimizing the number of oblivious transfers and another which is aimed at optimizing concrete efficiency. Our protocols are based on a novel combination of previous techniques together with a new efficient protocol to certify that pairs of strings transmitted via oblivious transfer satisfy a global relation. Settling for "security with correlated abort", the concrete communication complexity of the second variant of our protocol can beat the best previous protocols with the same kind of security even for realistic values of the circuit size and the security parameter. This variant is particularly attractive in the offline-online setting, where the online cost is dominated by a single evaluation of an authenticated garbled circuit, and can also be made non-interactive using the Fiat-Shamir heuristic.

Keywords: Secure two-party computation, constant-round protocols, garbled circuits, low-complexity cryptography

*This is a preliminary full version of the conference paper [HIV17], which fixes some mistakes and includes additional results. See Appendix A for a summary of the differences.

†Bar-Ilan University. Email: carmit.hazay@cs.biu.ac.il.

‡Technion and UCLA. Email: yuvali@cs.technion.ac.il

§University of Rochester. Email: muthuv@cs.rochester.edu.

Contents

1 Introduction

Secure multiparty computation allows two or more parties to perform a distributed computation while protecting to the extent possible the secrecy of the inputs and the correctness of the outputs. The focus of this work is on *constant-round* secure *two-party* computation. The most practical approach to this problem is Yao's garbling paradigm [Yao86]. It is convenient to describe Yao's original protocol for the case of computing deterministic two-party functionalities, represented by Boolean circuits, that deliver output to only one party. (The general case can be easily reduced to this case.) We will refer to the party who gets an output as the *receiver* and to the other party as the *sender*. The protocol proceeds by having the sender randomly generate an encoded version of the circuit, referred to as a *garbled circuit,* together with pairs of *input keys*, a pair for each input bit. It sends the garbled circuit to the receiver along with the input keys corresponding to the sender's inputs, and allows the receiver to select its own input keys using oblivious transfer (OT). From the garbled circuit and the selected input keys, the receiver can compute the output.

This simple version of Yao's protocol is only secure in the presence of passive (semi-honest) corruptions, since it allows a malicious sender to freely manipulate the honest receiver's output by sending a badly formed garbled circuit. Nevertheless, being the simplest protocol of its type, it serves as a benchmark for the efficiency of secure two-party computation. Given a length-doubling pseudo-random generator[1] (PRG) $G : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$, this protocol can be used to evaluate a Boolean circuit C using $O(\kappa|\mathrm{C}|)$ bits of communication, $O(|\mathrm{C}|)$ PRG invocations, and n OTs on pairs of κ-bit strings, where n is the length of the receiver's input. Obtaining security in the presence of active (malicious) adversaries is much more challenging. To rule out practically inefficient solutions that rely on general zero-knowledge proofs [GMW87] or alternatively require public-key operations for every gate in the circuit [JS07, Gen09], it is useful to restrict the attention to protocols that make a *black-box* use of a PRG, as well as a constant-round parallel oblivious transfer (OT) protocol. The latter is in a sense necessary, since parallel OT is an instance of secure computation. It is convenient to abstract away from the use of an actual OT sub-protocol by casting protocols in the *OT-hybrid* model, where the parties can invoke an ideal OT oracle. This is justified by the fact that the cost of implementing the OTs is typically not an efficiency bottleneck.[2] In the following, we will refer to a protocol that only makes a black-box use of a PRG (or a stronger "symmetric" primitive)[3] and OT (either a parallel OT protocol or an ideal OT oracle) as a *black-box protocol*. Yao's passively secure protocol is black-box in this sense.

Lindell and Pinkas [LP07] (following [MF06]) presented the first constant-round black-box protocol that achieves simulation-based security against active adversaries. Their protocol replaces expensive (and non-black-box) zero-knowledge proofs by an ad-hoc use of a "cut-and-choose" technique. Since then, a large body of works attempted to improve the efficiency of such protocols both asymptotically and concretely (see, e.g., [IKL+13, NST16, NO16, WMK17, WRK17] and references therein). The main goal of the present work is to minimize the asymptotic communication complexity of this type of protocols.

In protocols that rely on the "cut-and-choose" technique, the sender sends $O(s)$ independent copies of the garbled circuit, for some statistical parameter s, following which a subset chosen by the receiver

[1] Garbled circuits are often described and implemented using a pseudo-random function (PRF) $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ instead of a length doubling PRG G. Since G can be implemented via two calls to F but the converse direction is not known, formulating positive (asymptotic) results in terms of the number of PRG calls makes them stronger.

[2] The number of OTs used by such protocols is typically smaller than the circuit size. Moreover, the cost of a large number of OTs can be amortized via efficient OT extension [IKNP03, KOS15].

[3] It is sometimes helpful to replace the PRG by a stronger symmetric primitive, such as symmetric encryption, a correlation-robust hash function [IKNP03], or even a random oracle. While the main question we consider was open even when the use of such stronger symmetric primitives is allowed, our main asymptotic results only require a PRG.

is opened to demonstrate correctness. The parameters in this approach have been sharpened, and the best protocols can achieve sender simulation error of 2^{-s} using only s copies [HKE13, Lin16]. However, the multiplicative communication overhead[4] in all these protocols over Yao's protocol is at least s, and similarly the cryptographic cost involves at least $\Omega(s)$ PRG calls per gate, compared to $O(1)$ in Yao's protocol. Using a different technique (see Section 1.2), the asymptotic communication overhead has been improved to $\mathsf{polylog}(s)$ [IKO+11, IKL+13], at the cost of relying on heavy "non-cryptographic" machinery (that includes linear-time encodable error-correcting codes and routing networks) and poor concrete efficiency.

Towards minimizing the overhead of active security, another line of works analyzed the *amortized* setting, where multiple evaluations of the same circuit are conducted by the two parties [LR14, HKK+14, RR16, NO16]. In this setting, a recent work of Nielsen and Orlandi [NO16] shows how to protect Yao's protocol against active adversaries with only a constant (amortized) multiplicative communication overhead. Besides relying on a collision-resistant hash-function (or even private information retrieval schemes for functions with large inputs), the main caveat is that this approach only applies in the case of multiple circuit evaluations, and moreover the number of evaluations must be bigger than the size of the circuit.

Finally, a recent work of Wang, Ranellucci, and Katz [WRK17] obtains an actively secure version of Yao's protocol that can be instantiated to have constant communication overhead in the OT-hybrid model. Unfortunately, this protocol requires $\Omega(\kappa)$ separate *bit-OT* invocations for each gate of the circuit. As a result, its black-box implementation in the plain model has $\tilde{\Omega}(\kappa)$ communication overhead over Yao's protocol.[5] A similar overhead applies to the computational cost in the plain model. We note that even in the bit-OT hybrid, the constant-overhead variant of [WRK17] inherits from [IPS08, CC06] the use of heavy tools such as algebraic geometric codes, and has poor concrete efficiency.

To conclude, prior to the present work there was no constant-round actively secure protocol that makes a black-box use of oblivious transfer and symmetric primitives and has a constant communication overhead over Yao's protocol in plain model. This state of affairs leaves the following question open:

> *What is the best achievable communication overhead of constant-round "black-box" actively secure two-party protocols* in the plain model *compared to Yao's passively secure protocol? In particular, is constant multiplicative overhead achievable?*

As discussed above, it will be convenient to consider this question in the OT-hybrid model. To ensure relevance to the plain model we will only consider a "κ-bit string-OT" hybrid, where each ideal OT is used to transfer a string whose length is at least κ (a computational security parameter) and the communication cost also includes the communication with the OT oracle.

1.1 Our Results

Our main result is an affirmative answer to the question of actively secure garbled circuits with constant communication overhead. This is captured by the following theorem.

[4] Following the common convention in the secure computation literature, the multiplicative overhead considers the typical case where the circuit is (polynomially) bigger than the input length and the security parameter, and ignores low-order additive terms that are asymptotically dominated by the circuit size when the circuit size is a sufficiently large polynomial in the other parameters. Concretely, when we say that the asymptotic multiplicative overhead is $c(s)$, we mean that the communication complexity can be bounded by $c(s) \cdot O(\kappa|C|) + |C|^{\epsilon} \cdot \mathsf{poly}(n, s, \kappa)$ for every constant $\epsilon > 0$.

[5] There are no known protocols for realizing many instances of bit-OT in the plain model with less than $\tilde{\Omega}(\kappa)$ bits per instance, except via a heavy use of public-key cryptography for each OT instance [IKOS09, BGI17] or polynomial-stretch local PRGs [IKOS08]. This is true even for passively secure bit-OT, even in the random oracle model, and even when using the best known OT extension techniques [IKNP03, KK13].

Theorem 1.1 (Informal.) *Let κ denote a computational security parameter, s a statistical security parameter, and n the length of the shorter input. Then, for any constant $\epsilon > 0$, there exists an actively secure constant-round two-party protocol Π_{C} for evaluating a Boolean circuit C with the following efficiency features (ignoring lower order additive terms):*

- *It uses $O(\kappa \cdot |\mathrm{C}|)$ bits of communication;*
- *It makes $O(|\mathrm{C}|)$ black-box calls to a length-doubling PRG of seed length κ;*
- *It makes $n + O(s \cdot |\mathrm{C}|^{\epsilon})$ calls to κ-bit string OT oracle, or alternatively $(n + |\mathrm{C}|^{\epsilon}) \cdot \mathsf{poly}(\kappa)$ calls to any (parallel, constant-round) bit-OT protocol in the plain model, assuming explicit constant-degree polynomially unbalanced unique-neighbor expanders.*[6]

Concrete efficiency. The above result is focused on optimizing the asymptotic communication complexity while using a small number of OTs. We also present a second variant of the main result which is geared towards concrete efficiency. Our second variant avoids the heavy machinery of linear-time encodable error-correcting codes, algebraic geometric codes, and expander graphs that are used in the first variant at the cost of settling for a weaker notion of security. Specifically, this variant satisfies "security with correlated abort," which relaxes the ideal model execution by allowing a malicious sender to specify a predicate of the honest receiver's input that would make the receiver abort. In particular, if the malicious sender can learn whether the receiver aborts, this leaks one bit of information about the receiver's input (but requires the sender to take a risk of being caught cheating). A similar notion of security is realized by protocols based on the dual execution paradigm [MF06, HKE12]. Security with correlated abort is arguably sufficient in many cases, as the leakage legitimately allowed by the ideal functionality is often more significant than the very limited type of leakage allowed by correlated abort. For instance, secure computation with one bit of leakage has been adopted by Calctopia [cal] to perform secure computation on data stored in local private spreadsheets of the parties. The notion of security with correlated abort realized by the second variant of our protocol is in fact strictly stronger than the notion of security achieved by dual execution protocols in that we achieve full security against one of the parties (the receiver).

Optimizing our second variant, we get better concrete communication complexity than the best previous protocols in this security model. For instance, for computational security parameter $\kappa = 128$ and statistical security parameter $s = 40$, our asymptotic multiplicative communication overhead over the best passively secure protocols (in the random oracle model) is 1.625 $(= 1 + 2s/\kappa)$ and concretely our overhead is 1.78 for circuits with more than 30,000 AND gates. This should be compared to: an overhead of 40 in optimized cut-and-choose and of roughly 24 in [WRK17, KRRW18] (where these two approaches achieve full security), and an overhead of 2 for dual-execution [MF06, HKE12]. While we have not fully implemented our protocol, its computational cost seems comparable to that of other protocols from the literature. In contrast to dual execution protocols, our protocol can be made non-interactive via the Fiat-Shamir heuristic, and moreover its offline-online version has a much more efficient online phase. See Sections 1.3 and 9.2 for a detailed concrete analysis of the second variant of our protocol and comparison with the concrete efficiency of other recent protocols.

An optimized version of [WRK17] has recently appeared in [KRRW18] making the former compatible with half-gates optimization while decoupling the information theoretic MAC from the garbled circuit.

[6] This assumption is needed for the existence of polynomial-stretch local s-wise PRGs [MST03]. It is a mild assumption (arguably more so than standard cryptographic assumptions) that can be instantiated heuristically (see, e.g., [IKOS08, ADI+17]). One can dispense with this assumption by allowing $O(|\mathrm{C}|)$ OTs of κ-bit strings, or replacing the PRG by a stronger symmetric primitive such as a correlation-robust hash function or a random oracle.

Non-interactive implementation. Another useful feature of our protocol is that, following a function-independent preprocessing, it can be made *non-interactive* in the sense of [IKO+11] by using the Fiat-Shamir heuristic. In the non-interactive variant, the receiver can post an "encryption" of its input and go offline, allowing the sender to evaluate a circuit C on the inputs by sending a single message to the receiver.

1.2 Our Techniques

At a high level, our results combine the following main techniques. First, to break the cut-and-choose barrier we apply an authenticated variant of the garbled circuit construction, as was previously done in [IKO+11, WRK17]. To eliminate some of the effects of selective failure attacks by a malicious sender, we apply the multiparty circuit garbling technique of Beaver, Micali, and Rogaway (BMR) [BMR90], which was used for a similar purpose in the two-party protocols of [LPSY15, WRK17]. Finally, we crucially rely on a new "certified oblivious transfer" protocol to prove in zero-knowledge that pairs of strings transmitted via OT satisfy a global relation, providing a more efficient alternative to a similar protocol from [IKO+11].

We now give a more detailed account of our techniques. Our starting point is the work of Ishai, Kushilevitz, Ostrovsky, Prabhakaran, and Sahai [IKO+11] (IKOPS), which obtained a "non-interactive" black-box protocol with polylogarithmic communication overhead. More concretely, the IKOPS protocol only makes use of parallel OTs and a single additional message from the sender to the receiver, and its communication complexity is $\mathsf{polylog}(s) \cdot \kappa$ bits per gate. On a high-level, the IKOPS protocol for securely computing a functionality $\mathcal{F}$ can be broken into three non-interactive reductions. We begin with explaining the IKOPS reductions and then discuss our modified steps.

1. **Reducing $\mathcal{F}$ to an NC^0 functionality $\widehat{\mathcal{F}}$.** The first step is to securely reduce the computation of $\mathcal{F}$ to a single invocation of a related NC^0 functionality $\widehat{\mathcal{F}}$ whose output length is $O(\kappa \cdot |\mathcal{F}|)$. The functionality $\widehat{\mathcal{F}}$ takes from the sender a pair of keys for each wire and the purported PRG outputs on these keys. It also takes from the receiver a secret key that is used to authenticate the information provided by the sender. Note that $\widehat{\mathcal{F}}$ is non-cryptographic and cannot check that the given PRG outputs are consistent with the inputs. However, the authentication ensures that the output of $\widehat{\mathcal{F}}$ obtained by the receiver commits the sender to unique values. If the receiver detects an inconsistency with the authentication information during the garbled circuit evaluation, it aborts. The protocol for $\mathcal{F}$ only invokes $\widehat{\mathcal{F}}$ once, and only makes a black-box use of the given PRG. In fact, two variants of this reduction are suggested in [IKO+11]: one where $\widehat{\mathcal{F}}$ authenticates every PRG output provided by the sender, and one where only the color bits are authenticated.

2. **Reducing $\widehat{\mathcal{F}}$ to certified OT.** The second step is an information-theoretic protocol for $\widehat{\mathcal{F}}$ using an ideal *certified oblivious transfer* oracle, namely a parallel OT oracle in which the receiver is additionally assured that the pairs of strings transmitted (which also includes strings it does not receive) satisfy a global consistency predicate. Such a protocol is obtained in two steps: (1) Start with a non-interactive protocol for $\widehat{\mathcal{F}}$ using a standard parallel OT, where the protocol is only secure in the presence of a passive sender and an active receiver. (This is equivalent to an information-theoretic projective garbling scheme [BHR12] or decomposable randomized encoding [IKOS08] for $\widehat{\mathcal{F}}$.) (2) Use the certified OT oracle to enforce honest behavior of the sender. Below, we will refer to this certified OT functionality *sender-certified* OT (SCOT), to distinguish it from other forms of certified OT we will consider.

3. **Reducing certified OT to parallel OT.** The third step is an information-theoretic protocol for SCOT using parallel OTs. This step is implemented using a variant of the "MPC-in-the-head" approach

of [IKOS07], using a virtual MPC protocol in which each transmitted OT string is received by a different party, and an honest majority of servers is used to guarantee global consistency. The SCOT protocol is inherently susceptible to a benign form of input-dependent selective failure attacks, but these can be eliminated at a relatively low cost by using a local randomization technique [Kil88, LP07, IKO+11].

The main instance of the IKOPS protocol is based on the first variant of $\widehat{\mathcal{F}}$, which authenticates the PRG outputs. This protocol has a polylog(s) communication overhead that comes from two sources. First, the implementation of $\mathcal{F}$ given $\widehat{\mathcal{F}}$ (Step 1 above) is subject to selective failure attacks by a malicious sender. These attacks make the receiver abort if some *disjunctive* predicate of the wire values is satisfied. (The second variant of $\widehat{\mathcal{F}}$ from [IKO+11] is subject to more complex selective failure predicates, and hence is not used for the main result.) Such a disjunctive selective failure is eliminated in [IKO+11, IKL+13] by using leakage-resilient circuits, which incur a polylogarithmic overhead. We eliminate this overhead by defining an alternative NC^0 functionality $\widehat{\mathcal{F}}$ that introduces a BMR-style randomization of the wire labels (as done in [LPSY15, WRK17], but using the first variant of $\widehat{\mathcal{F}}$ from [IKO+11]). This implies that the receiver's input to $\widehat{\mathcal{F}}$ grows with additional $O(|\mathrm{C}|)$ random bits, which results in a protocol that uses $O(|\mathrm{C}|)$ OTs of $O(\kappa)$-bit strings. To reduce the number of OTs, we use a local s-wise PRG [MST03] to make the receiver's input to $\widehat{\mathcal{F}}$ small while still ensuring that the probability of the receiver detecting failure is essentially independent of its secret input. We note that the reduction in the number of OTs is essential in order to get a protocol with constant communication overhead in *the plain model* by using only a (parallel) bit-OT protocol and a PRG in a black box way.

Another source of polylogarithmic overhead in [IKO+11] comes from the SCOT construction, which relies on perfectly secure honest-majority MPC protocols. The best known protocols of this type have a polylogarithmic communication overhead [DIK10]. Our approach for reducing this overhead is to obtain an interactive variant of SCOT that can rely on any *statistically secure* honest-majority MPC protocol, and in particular on ones with constant communication overhead [DI06, CC06, IPS09]. Our new SCOT protocol extends in a natural way the recent MPC-based zero-knowledge protocol from [AHIV17]. This gives an efficient realization of the SCOT functionality, albeit one that is vulnerable to an input-dependent selective failure similar to [IKO+11].

We remark that the inputs to the original functionality can be protected against selective failures via the encoding techniques from [Kil88, LP07, IKO+11]. Concretely, the original functionality is modified so that it accepts an encoded version of the receiver's inputs, and this encoding is picked at random by the receiver. However, the BMR-style randomization step introduces additional auxiliary random inputs that are also subject to selective failure attack. And while these auxiliary BMR inputs are independent of the original inputs, they are used to mask them, and so leakage on BMR inputs reveals information about the original inputs. The latter fact was overlooked in the conference version of this paper [HIV17].

To make the leakage on BMR inputs harmless, we need to redesign the functionality $\widehat{\mathcal{F}}$ so it takes an encoded form of the BMR inputs which protects against the above leakage. In contrast to the original inputs, whose protection can be done by modifying the original functionality $\mathcal{F}$ (which is not restricted to be in NC^0), here we need the modified $\widehat{\mathcal{F}}$ to remain in NC^0, since this is crucial for achieving constant communication overhead. As a result, we cannot apply the input encoding techniques from [Kil88, LP07, IKO+11] to protect the BMR random inputs.[7]

Instead, we will encode the inputs of $\widehat{\mathcal{F}}$ (including the BMR inputs) by using a multiplicative secret

[7] For instance, the encoding used in [Kil88, LP07] requires the decoder to take the XOR of s bits for each encoded input bit. Note that here we cannot use the Free XOR optimization from [KS08], since it does not apply with constant overhead to NC^0 computations.

sharing scheme based on AG codes [CC06]. This encoding will suffice to protect the computation of $\widehat{\mathcal{F}}$ against selective failures while keeping it in NC^0 and while only increasing the circuit size of $\widehat{\mathcal{F}}$ by a constant factor. However, this modified $\widehat{\mathcal{F}}$ is susceptible to a new attack, by a sender or receiver who provide an invalid secret sharing of their inputs.[8] We will be able to protect both parties against this new attack by considering a stronger variant of the SCOT functionality. Namely, we introduce a *doubly certified oblivious-transfer* functionality (DCOT) that, in addition to certifying the sender's inputs, will provide the result to the receiver *only if* the receiver's inputs satisfy some given predicate. Finally, we show how to efficiently reduce the DCOT functionality to the SCOT functionality.

Our first variant of authenticated garbling uses the above DCOT protocol for a consistency predicate defined by Boolean circuits. As in [IKO+11], these Boolean circuits employ information-theoretic MACs based on linear-time encodable codes [Spi95]. To compute such a predicate with constant communication overhead, we rely on statistical honest-majority MPC based on algebraic geometric codes [CC06, IKOS07]. This results in poor concrete efficiency and $\Omega(s)$ computational overhead.

The second variant of our protocol eliminates the above heavy machinery and obtains good concrete efficiency by making the following two changes: (1) using the second variant of the NC^0 functionality $\widehat{\mathcal{F}}$ for Step 1 (which only authenticates color bits); (2) relaxing security to allow correlated abort, which eliminates some of the above complications and allows us to directly use SCOT bypassing the need for DCOT; (3) implementing an optimized SCOT protocol for a predicate defined by an *arithmetic* circuit over a field of size $2^{O(s)}$. Switching from boolean to arithmetic circuits allows us to use simpler constant-rate honest-majority MPC protocols for arithmetic circuits over large fields. Such protocols are simpler than their Boolean counterparts and have better concrete efficiency (see [IPS09], Appendix C, and [AHIV17]). Another (asymptotic) advantage is polylogarithmic computational overhead. The second variant of our protocol has a multiplicative communication overhead of $(1 + s/\kappa)$ over the passive Yao protocol, with a small additive overhead. The latter holds even when allowing FreeXOR optimization [KS08]. If we allow the "half-gate" optimization of [ZRE15], then the overhead becomes $(1 + 2s/\kappa)$.

1.3 Comparison with Prior Works

Comparison with Wang et al. [WRK17]. The recent results of [WRK17] are the most relevant to our work. Like the second variant of our protocol, the protocol from [WRK17] uses a combination of: (1) an "authenticated garbled circuit functionality" which is similar to the second variant from [IKO+11] that only authenticates the color bits, and (2) a BMR-style randomization to defeat selective failure attacks. (In contrast, the first variant of our protocol that we use to get our main asymptotic results relies on the first variant of the functionality from [IKO+11] that authenticates the entire PRG outputs, since in this variant the selective failure predicate is simple.) The main difference between the second variant of our protocol and the protocol from [WRK17] is in how the NC^0 functionality $\widehat{\mathcal{F}}$ is securely realized against active adversaries. While the work of [WRK17] uses a "GMW-style" interactive protocol for realizing $\widehat{\mathcal{F}}$, we rely on the non-interactive SCOT-based approach of IKOPS [IKO+11] at the cost of settling for security with correlated abort.

In slightly more detail, the protocol of [WRK17] for evaluating $\widehat{\mathcal{F}}$ first creates a large number of authenticated "AND triples" using a variant of the "TinyOT" protocol [NNOB12]. Then, using the AND triples, the parties securely compute $\widehat{\mathcal{F}}$. This protocol, which follows an optimized cut-and-choose approach, has $\Omega(s/\log|\mathrm{C}|)$ communication overhead. Alternatively, [WRK17] also proposes using a protocol from [IPS08] to make the communication overhead constant, but this only holds in the bit-OT hybrid model

[8] This is in contrast to the IKOPS transformation, where the receiver has no space for cheating.

that cannot be instantiated in our black-box model and leads to prohibitive concrete overhead. In contrast, our protocols realize $\widehat{\mathcal{F}}$ using a passively secure non-interactive protocol in the κ-bit OT-hybrid, and apply an improved implementation of SCOT to achieve security against active adversaries with constant communication overhead. The good concrete efficiency of the second variant of our protocol is inherited from a careful implementation of the passively secure protocol for $\widehat{\mathcal{F}}$ and a sublinear communication implementation of SCOT.

Comparison with dual execution protocols. The dual execution paradigm for actively secure 2PC [MF06, HKE12] has an asymptotic overhead of 2 compared to the passively secure protocol. This paradigm is based on the observation that the passively secure protocol of Yao (in the OT-hybrid model) is already actively secure against an active receiver, who evaluates the garbled circuit. In a dual execution protocol, the parties engage in two instances of the passive Yao protocol, with the roles reversed in each instance, followed by a secure equality check for the outputs. The intuition is that for at least one of the two instances the result will be correct, since in one of the two instances the corrupted party is the evaluator (and therefore cannot attack the protocol) while the other party remains honest. This protocol allows the corrupted party to obtain an arbitrary bit of leakage, regardless of who is corrupted. This is weaker than the "correlated abort" notion achieved by the second variant of our protocol, where full security is achieved against a malicious receiver. Indeed, in dual execution protocols the adversary always learns one bit of leakage from the honest party's input since the predicate that makes the honest party abort is defined by a badly formed garbled circuit via the equality check.

The cost of implementing the secure equality check in dual execution protocols may be significant, despite some recent optimizations [KMRR15, RR16]. When the statistical security parameter s is set to 40 and computation security parameter κ is set to 128, the communication overhead of our protocol (compared to passive Yao) is less than 2 for circuits with more than 1000 AND gates, regardless of the output size. In contrast, the overhead of dual execution protocols is always bigger than 2 with a significant additive term that grows with the output size. As stated above, our overhead is $1 + 2s/\kappa$ and is less than 2 for reasonable parameter regimes. In comparison with fully-secure implementations, the work of [KRRW18] provides the most communication efficient protocol and incurs an overhead of $(4B + 3) + (4B + 2)s/\kappa$ where $B = s/\log|C|$ (roughly 24 for $s = 40$, $\kappa = 128$ and circuits with more than 3100 AND gates).

Both dual execution protocols and our protocol can be benefit from preprocessing in an offline/online setting. In dual execution protocols, both parties will each have to store the entire garbled circuit, while in our protocol only one of the parties maintains an "authenticated" garbled circuit. A more significant advantage of our protocol is that its online phase can be made non-interactive while dual execution protocols require non-trivial online interaction to perform the equality test.

Finally, we can modify our protocol so that it achieves a stronger notion of security with correlated abort, namely one where the abort predicate is limited to a *disjunction* of circuit wires or their negations. This variant requires a relatively mild assumption on the underlying PRG (which holds in the random oracle model).

A summary of qualitative and quantitative improvements. We conclude with a summary of the improvements introduced by our second construction compared to prior work.

Security. Our protocol archives worse security than "fully secure" protocols. However, its security is better than dual execution security in the sense that we get full security against a malicious receiver whereas in dual execution both parties can get a bit of leakage about the other party's input.

Communication complexity. Our cost is comparable to dual execution, and significantly better than the best fully secure protocols (e.g., a factor of 13 better than [KRRW18]). For instance, the total concrete

communication cost of computing a single instance of AES circuit with $\kappa = 128$ and $s = 40$ is, 15.12 MB for [NST17], 6.29 MB for [WRK17], 4.25 MB for [KRRW18], 432 KB for the dual execution protocol (estimated) and only 425 KB for our protocol.

Interaction. Our protocol has the qualitative advantage of supporting "non-interactive" secure computation [IKO+11] (NISC), whereas other recent works on the concrete efficiency of 2PC, including those based on dual execution, inherently require 3 or more messages. A NISC protocol allows the receiver to post an "encryption" of its input and get back from the sender an "encryption" of the output. Our protocol can be implemented in this setting given pre-computed OTs, or using 2-message (parallel) OT protocols such as the PVW protocol [PVW08]. Recall that our NISC protocol employs $\kappa + 2s$ OTs, which is independent of the circuit size, therefore using the PVW OT protocol will not introduce a high overhead.

Reusability. An attractive feature of NISC is the potential to reuse the receiver's message for multiple computations, where the receiver's input is fixed and the sender's input changes. This can be an important feature in practice. In fact, if we use a PVW-style 2-message OT (as opposed to precomputed OTs), the second message can originate from multiple independent senders who can each contribute one or more inputs. Similarly to previous OT-based NISC protocols, our protocol is not fully reusable in the sense that if a malicious sender can repeatedly learn whether the receiver aborts, it can eventually learn the receiver's OT selections and violate security [IKO+11, AMPR14, CDI+18]. In practice, however, this issue is not always significant for several reasons. First, there is no problem with reusing the receiver's message as long as the receiver keeps its output to itself, or alternatively makes sure that no information about the output is leaked to the sender. Second, for such a selective failure attack to be successful, cheating should be detected by the receiver multiple times. This can come at a high price for the sender(s), either in money or in reputation. Finally, as discussed in [IKO+11], the receiver can prevent the attack by just posting a fresh encryption of its input once the number of aborting executions passes the threshold of becoming risky.

2 Preliminaries

Basic notations. We denote a security parameter by κ. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time and denote by $[n]$ the set of elements $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$.

We assume functions to be represented by a Boolean circuit C (with AND,OR,XOR gates of fan-in 2 and NOT gates), and denote the size of C by $|\mathrm{C}|$. By default we define the size to include the total number of gates, excluding NOT gates but including input gates. In the context of protocols that employ the FreeXOR garbled circuit optimization [KS08], the size does not include XOR gates.

We specify next the definitions of computationally indistinguishable distributions and statistical distance.

Definition 2.1 *Let $X = \{X(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ and $Y = \{Y(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \overset{\mathrm{c}}{\approx} Y$, if for every* $\mathsf{poly}(\kappa)$*-time non-uniform $\mathcal{D}$, every $a \in \{0,1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large κ's,*

$$\left|\Pr\left[\mathcal{D}(X(a,\kappa), 1^\kappa) = 1\right] - \Pr\left[\mathcal{D}(Y(a,\kappa), 1^\kappa) = 1\right]\right| < \frac{1}{p(\kappa)}.$$

Note that the computational indistinguishability requirement effectively requires the inputs a to be polynomially bounded in κ (as longer inputs cannot even be fully read by $\mathcal{D}$).

Definition 2.2 *Let X_κ and Y_κ be random variables accepting values taken from a finite domain $\Omega \subseteq \{0,1\}^\kappa$. The* statistical distance *between X_κ and Y_κ is*

$$SD(X_\kappa, Y_\kappa) = \frac{1}{2} \sum_{\omega \in \Omega} \left| \Pr[X_\kappa = \omega] - \Pr[Y_\kappa = \omega] \right|.$$

We say that X_κ and Y_κ are ε-close *if their statistical distance is at most $SD(X_\kappa, Y_\kappa) \leq \varepsilon(\kappa)$. We say that X_κ and Y_κ are* statistically close*, denoted $X_\kappa \approx_s Y_\kappa$, if $\varepsilon(\kappa)$ is negligible in κ.*

2.1 Length-Doubling Pseudorandom Generators

Informally speaking, a pseudorandom generator (PRG) is an efficiently computable deterministic function that looks like a truly random string to any (non-uniform) polynomial time observer. The garbling schemes of Yao [Yao86] and BMR [BMR90], which we employ, can be based on any length-doubling PRG $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$, defined below.

Definition 2.3 (Length doubling pseudorandom generator) *Let $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ be an efficient, length-doubling, deterministic function. We say that $\mathrm{G}_{\mathrm{PRG}}$ is a* pseudorandom generator *if for every polynomial-time non-uniform distinguisher $\mathcal{D}$, there exists a negligible function* negl *such that:*

$$\left| \Pr[\mathcal{D}(\mathrm{G}_{\mathrm{PRG}}(r)) = 1] - \Pr[\mathcal{D}(R) = 1] \right| \leq \mathsf{negl}(\kappa).$$

where $r \leftarrow \{0,1\}^\kappa$ and $R \leftarrow \{0,1\}^{2\kappa}$. The probability is taken over the choice of either r or R.

2.2 s-Wise Pseudorandom Generators

An s-wise pseudorandom generator (PRG) $\mathrm{G}_{\mathrm{sPRG}} : \{0,1\}^\delta \mapsto \{0,1\}^n$ satisfies the property that for a random r the bits in $\mathrm{G}_{\mathrm{sPRG}}(r)$ are s-wise independent, in the sense that their projection to any s coordinates is a uniformly random s-bit string. Standard constructions of such PRGs exist based on random $(s-1)$-degree polynomials in a finite field. In our work, we will require s-wise PRGs that additionally have the property of being computed by an NC^0 circuit, namely ones where every output bit depends on a constant number of input bits. Such "local" s-wise PRGs can be based on unique-neighbor bipartite expander graphs [MST03].

In more detail, consider a bipartite expander graph with left degree d, such that any subset of $v \leq s$ vertices on the left has at least $\frac{3}{4}vd$ neighbors on the right. Then we associate every left vertex with an output bit and and every right vertex with an input bit. An s-wise PRG can now be obtained setting an output bit as the XOR of its neighbors. If we further assume that the bipartite graph has constant-degree d for the left vertices, we obtain an s-wise PRG that can be computed by an NC^0 circuit.

Some of our results require an s-wise PRGs with polynomial stretch. Concretely, for every $0 < \epsilon < 1$ we need an explicit NC^0 construction of an s-wise PRG $\mathrm{G}_{\mathrm{sPRG}}$ from $\delta = O(n^\epsilon + s)$ to n bits. (In fact, $\delta = O(n^\epsilon) + s^{O(1)}$ would suffice for obtaining slightly weaker but qualitatively similar results.) Expander graphs with the corresponding parameters are known to exist, and in fact a random graphs has the required expansion property with high probability (cf. [IKL+13], Theorem 2). While no provable explicit constructions are known, assuming the existence of such an explicit construction (e.g., by using the binary expansion of π) can be viewed as a mild assumption compared to standard cryptographic assumptions. Some of our results rely on such an assumption, which is necessary for the existence of explicit polynomial-stretch local PRGs. See, e.g.,[IKOS08, ADI+17] for further discussion.

2.3 Message Authentication Codes

2.3.1 Simple MAC for a Single Bit

Our first construction for message space $\{0,1\}$ is a trivial MAC that picks two random strings $\{\sigma_0, \sigma_1\}$ as the key and assigns σ_b as the MAC for bit $b \in \{0,1\}$.

2.3.2 Low-Depth MAC for Strings

We consider a second MAC that will allow for a sender to authenticate a κ-bit string via a secure computation of an NC^0 function to a receiver holding the MAC key. It is easy to see that if the MAC itself is computable in NC^0 then it can only have a constant soundness error. To overcome this barrier, we follow the approach of [IKOS08, IKO+11] where the message to be authenticated is first locally encoded. Since the NC^0 computation cannot compute the encoding, we will require from the sender to provide the encoding to the NC^0 functionality along with a proof, where both the MAC computation given the encoding and the proof verification are done in NC^0. We will additionally require that the encoding procedure be efficient, since the proof verification circuit size grows with the encoding circuit size. By relying on Spielman's codes [Spi95], we obtain an asymptotically optimal code that can be encoded by linear-size circuits. More formally, such codes imply that there exist constants $\ell_{\mathsf{lin}}, \ell_{\mathsf{out}}, \ell_{\mathsf{key}}$ such that for every length κ, there exists an explicit linear-size circuit $\mathsf{Enc}_{\mathsf{lin}} : \{0,1\}^{\kappa} \to \{0,1\}^{\ell_{\mathsf{in}}\kappa}$ and an NC^0 function family $\{\mathsf{MAC}_{\mathrm{SK}} : \{0,1\}^{\ell_{\mathsf{in}}\kappa} \to \{0,1\}^{\ell_{\mathsf{out}}\kappa}\}_{\mathrm{SK}\in\{0,1\}^{\ell_{\mathsf{key}}\kappa}}$ such that $\mathsf{MAC}_{\mathrm{SK}}(\mathsf{Enc}_{\mathsf{lin}}(\sigma))$ is a $2^{-\kappa}$ information-theoretically secure MAC.

2.3.3 Special-Hiding Information-Theoretic MAC

For our concretely efficient protocol, we will employ another simple information theoretic MAC. We formalize the security requirement next and then present a construction.

Definition 2.4 (Special-hiding IT-MAC) *Let $\mathbb{F}$ be a finite field. We say that a family of functions $\mathcal{H} = \{H : \mathbb{F}^{\ell} \times \mathbb{F} \to \mathbb{F}\}$ is ϵ-secure special-hiding if the following two properties hold:*

Privacy. *For every $x, x' \in \mathbb{F}^{\ell}$ and $H \in \mathcal{H}$, the distributions $H(x;r)$ and $H(x';r)$ are identical for a random $r \in \mathbb{F}$.*

Unforgeability. *For any x, r, x', r' such that $(x,r) \neq (x',r')$, we have: $\Pr[H \leftarrow \mathcal{H} : H(x;r) = H(x',r')] \leq \epsilon$*

Proposition 2.1 *Let $\ell \in \mathbb{N}$. Define the family $\mathcal{H} = \{H_w\}_{w\in\mathcal{I}}$ where the index set $\mathcal{I}$ includes all vectors $(k_0, \ldots, k_\ell)$ such that $\sum_{i=0}^{\ell} k_i \neq 0$ and the hash function is defined as*

$$H_{(k_0,\ldots,k_\ell)}((x_1,\ldots,x_\ell), r) = \sum_{i=0}^{\ell} k_i \cdot (r + x_i)$$

where x_0 is set to 0. Then $\mathcal{H}$ is a $\frac{1}{|\mathbb{F}|}$-secure special-hiding IT-MAC.

Proof: We prove that the two properties defined in Definition 2.4 are met.

Privacy: Given x and x' in $\mathbb{F}^\ell$, we need to show that $\{r \leftarrow \mathbb{F} : H(x;r)\}$ and $\{r' \leftarrow \mathbb{F} : H(x';r')\}$ are identically distributed. This follows from the fact that given x, x' and key $w = (k_0, \ldots, k_\ell)$, for every r, we can find r' such that $H_w(x,r) = H_w(x',r')$ by setting

$$r' = r + \left(\sum_{i=0}^{\ell} k_i\right)^{-1} \left[\sum_{i=1}^{\ell} k_i(x_i - x'_i)\right]$$

where $x = (x_1, \ldots, x_\ell)$ and $x' = (x'_1, \ldots, x'_\ell)$. We remark that $(\sum_{i=0}^{\ell} k_i)$ has an inverse because $\mathcal{I}$ only includes keys such that $\sum_{i=0}^{\ell} k_i \neq 0$.

Unforgeability: Recall that robustness requires to show that for x, r, x', r' such that $(x,r) \neq (x',r')$,

$$\Pr[w \leftarrow \mathcal{I} : H_w(x;r) = H_w(x';r')] \leq \frac{1}{|\mathbb{F}|}$$

We argue this in two subcases.

- Case $r = r'$: This means $x \neq x'$ and there exists j such that $x_j \neq x'_j$ where $x = (x_1, \ldots, x_\ell)$ and $x' = (x'_1, \ldots, x'_\ell)$. Then given $w = (k_0, \ldots, k_\ell) \in \mathcal{I}$ we can rewrite the equation $H_w(x;r) = H_w(x';r)$ as:

  $$k_j = -(x_j - x'_j)^{-1} \left(\sum_{1 \leq i \leq \ell, i \neq j} k_i(x_i - x'_i)\right)$$

  This means that if the elements k_i for $0 \leq i \leq \ell$ and $i \neq j$ are fixed, then the probability that k_j satisfies the preceding equation is $\frac{1}{|\mathbb{F}|}$.

- Case $r \neq r'$: Then for any x, x' and $k_i \in \mathbb{F}$ for $1 \leq i \leq \ell$, k_0 must satisfy the following equation for the MACs to be the same.

  $$k_0 = -(r - r')^{-1} \left(\sum_{i=1}^{\ell} k_i(x_i - x'_i)\right)$$

  which happens again with probability $\frac{1}{|\mathbb{F}|}$.

■

2.4 Secret-Sharing

A secret-sharing scheme allows distribution of a secret among a group of n players, each of whom in a *sharing phase* receive a share (or piece) of the secret. In its simplest form, the goal of secret-sharing is to allow only subsets of players of size at least $t+1$ to reconstruct the secret. More formally a $t+1$-out-of-n secret sharing scheme comes with a sharing algorithm that on input a secret s outputs n shares $s_1, \ldots, s_n$ and a reconstruction algorithm that takes as input $((s_i)_{i \in S}, S)$ where $|S| > t$ and outputs either a secret s' or $\perp$. In this work, we will use the Shamir's secret sharing scheme [Sha79] with secrets in a finite field $\mathbb{F}$ such that $|\mathbb{F}| > n$ and algebraic geometric (AG) secret sharing over constant-size fields [CC06]. We present the sharing and reconstruction algorithms below:

Sharing algorithm: For any input $s \in \mathbb{F}$, pick a random polynomial $p(\cdot)$ of degree t in the polynomial-field $\mathbb{F}[x]$ with the condition that $p(0) = s$ and output $p(1), \ldots, p(n)$, where $1, \ldots, n$ denote distinct nonzero field elements.

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

Functionality $\mathcal{F}_{\text{OT}}$ communicates with sender S, receiver R and adversary $\mathcal{S}$, and is parameterized by an integer m.

1. Upon receiving from the sender the input $(sid, (v_0^1, v_1^1), \ldots, (v_0^m, v_1^m))$ from S where $v_0^i, v_1^i \in \{0,1\}^t$ for every $i \in [m]$, record $((v_0^1, v_1^1), \ldots, (v_0^m, v_1^m))$.
2. Upon receiving $(sid, u_1, \ldots, u_m)$ from R where $u_i \in \{0,1\}$, send $v_{u_i}^i$ to R for all $i \in [m]$. Otherwise, abort.

Figure 1: The parallel oblivious transfer functionality.

Reconstruction algorithm: For any input $(s'_i)_{i \in S}$ where none of the s'_i are $\perp$ and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s'_i$ for every $i \in S$. This is possible using Lagrange interpolation where g is given by

$$g(x) = \sum_{i \in S} s'_i \prod_{j \in S/\{i\}} \frac{x-j}{i-j} .$$

Finally the reconstruction algorithm outputs $g(0)$.

Packed secret-sharing. The concept of packed secret-sharing was introduced by Franklin and Yung [FY92] in order to reduce the communication complexity of secure multi-party protocols, and is an extension of standard secret-sharing. In [FY92] the authors considered Shamir's secret sharing with the difference that the number of secrets $s_1, \ldots, s_\ell$ is now ℓ instead of a single secret, evaluated by a polynomial $p(\cdot)$ on ℓ distinct points. To ensure privacy in case of t corrupted parties, the random polynomial must have degree at least $t + \ell$. We use packed secret-sharing in our underlying MPC protocol.

Packed secret-sharing from AG codes. We will use a constant-rate packed secret sharing scheme over constant-size fields that supports multiplication of d shared blocks. Such a scheme relies on a family of linear error correcting codes with the following special properties.

Fact 2.2 ([CC06, CCX12]) *For any positive integer d there exists a finite field $\mathbb{F}$ of characteristic 2 and an efficiently constructible family of linear error-correcting codes $C_K : \mathbb{F}^K \to \mathbb{F}^{N_K}$ with the following properties: (1) $N_K = O(K)$; (2) The dual distance of C_K is $\delta_K = \Omega(K)$; (3) The linear code C'_K spanned by all point-wise products of d codewords in C_K has minimal distance $\Delta_K = \Omega(K)$.*

The secret sharing scheme defined by such a code proceeds by picking a random codeword that starts with the message vector and discarding the message prefix.

2.5 Oblivious Transfer

1-out-of-2 oblivious transfer (OT) is a functionality that is engaged between a sender S and a receiver R. We consider the parallel OT functionality which captures parallel executions of several OT instance; see Figure 1 for its formal description.

2.6 The [BMR90] Garbling

The [BMR90] garbling approach demonstrates a procedure that is applicable for any number of parties and involves garbling each gate separately using PRGs, while ensuring consistency between the wires. This method was recently improved by Lindell et al. in [LPSY15] that introduced an NC^0 functionality for this task based on PRFs.[9] Their main observation was that the parties need not demonstrate the correctness of their PRF computations, as an inconsistent value would immediately cause an honest party to abort with very high probability. This is because each PRF value forms a MAC tag that is unforgeable. Moreover, this event is not correlated with that honest party's input. This is due to the fact that each wire value a transmitted via some wire w is masked with a random color bit λ_w, and is thus hidden from the parties during the evaluation. In this paper we consider a slightly modified variant where only the sender picked the PRF keys while both parties pick the masking λ's. More formally, for each gate g with input wires u, v and output wire w, the 4 entries in a two-party garbing of g are computed as follows:

$$R_g^{a,b} = F_{k_u^a}(g,a) \oplus F_{k_v^b}(g,b) \oplus (k_w^0 \oplus \Big((((\lambda_a \oplus a)(\lambda_b \oplus b)) \oplus \lambda_c)(k_w^1 \oplus k_w^0)\Big)$$

for every pair $(a,b) \in \{0,1\}^2$ where F is a PRF and the wire masks $\lambda_u, \lambda_v, \lambda_w \in \{0,1\}$ are secret-shared between the parties, while the PRF keys k_u^a, k_v^b are chosen only by the sender. That is, each wire w is associated with a pair of keys k_w^0 and k_w^1, where the output key that is encrypted under the input keys in each row is determined by the color bits $\lambda_u, \lambda_v, \lambda_w$ (where the color bits associated with the input wires point to the index of the row to be decrypted in each garbled gate).

The privacy guarantee follows similarly to the privacy guarantee of Yao's garbing. Namely, the it is possible to generate an indistinguishable fake garbling given only the function's output $\mathcal{F}(x,y)$.

2.7 Secure Multiparty Computation (MPC)

Secure Two-Party Computation. We use a standard standalone definition of secure two-party computation protocols. In this work, we only consider static corruptions, i.e. the adversary needs to decide which party it corrupts before the execution begins. Following [HL10], we use two security parameters in our definition. We denote by κ a computational security parameter and by s a statistical security parameter that captures a statistical error of up to 2^{-s}. We assume $s \leq \kappa$. We let $\mathcal{F}$ be a two-party functionality that maps a pair of inputs of equal length to a pair of outputs. Without loss of generality, our protocols only deliver output to one party (the receiver), which can be viewed as a special case in which the other party's output is fixed.

Let $\Pi = \langle P_0, P_1 \rangle$ denote a two-party protocol, where each party is given an input (x for P_0 and y for P_1) and security parameters 1^s and 1^κ. We allow honest parties to be PPT in the entire input length (this is needed to ensure correctness when no party is corrupted) but bound adversaries to time $\mathsf{poly}(\kappa)$ (this effectively means that we only require security when the input length is bounded by *some* polynomial in κ). We denote by $\mathbf{REAL}_{\Pi,\mathcal{A}(z),P_i}(x,y,\kappa,s)$ the output of the honest party P_i and the adversary $\mathcal{A}$ controlling P_{1-i} in the real execution of Π, where z is the auxiliary input, x is P_0's initial input, y is P_1's initial input, κ is the computational security parameter and s is the statistical security parameter. We denote by $\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),P_i}(x,y,\kappa,s)$ the output of the honest party P_i and the simulator $\mathcal{S}$ in the ideal model where $\mathcal{F}$ is computed by a trusted party. In some of our protocols the parties have access to ideal model

[9] We note that although it is more common to describe the [BMR90] garbling based on PRFs, it is important to measure the complexity of generating the garbling in PRG calls instead. This is because switching from PRFs notation to PRGs might incur non-constant overhead [GGM86]. In this work we rely on a single length-doubling PRG call per garbled row.

implementation of certain cryptographic primitives such as ideal oblivious-transfer ($\mathcal{F}_{\text{OT}}$) and we will denote such an execution by $\mathbf{REAL}^{\mathcal{F}_{\text{OT}}}_{\Pi,\mathcal{A}(z),P_i}(x,y,\kappa,s)$.

Definition 2.5 *A protocol $\Pi = \langle P_0, P_1\rangle$ is said to* securely compute a functionality $\mathcal{F}$ in the presence of active adversaries *if the parties always have the correct output $\mathcal{F}(x,y)$ when neither party is corrupted, and moreover the following security requirement holds. For any probabilistic $\mathsf{poly}(\kappa)$-time adversary $\mathcal{A}$ controlling P_i (for $i \in \{0,1\}$) in the real model, there exists a probabilistic $\mathsf{poly}(\kappa)$-time adversary (simulator) $\mathcal{S}$ controlling P_i in the ideal model, such that for every non-uniform $\mathsf{poly}(\kappa)$-time distinguisher $\mathcal{D}$ there exists a negligible function $\nu(\cdot)$ such that the following ensembles are distinguished by $\mathcal{D}$ with at most $\nu(\kappa) + 2^{-s}$ advantage:*

- $\{\mathbf{REAL}_{\Pi,\mathcal{A}(z),P_i}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),P_i}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y,z\in\{0,1\}^*}$

Secure circuit evaluation. The above definition considers $\mathcal{F}$ to be an infinite functionality, taking inputs of an arbitrary length. However, our protocols (similarly to other protocols from the literature) are formulated for a finite functionality $\mathcal{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ described by a Boolean circuit C. Such protocols are formally captured by a polynomial-time *protocol compiler* that, given security parameters $1^\kappa, 1^s$ and a circuit C, outputs a pair of circuits (P_0, P_1) that implement the next message function of the two parties in the protocol (possibly using oracle calls to a cryptographic primitive or an ideal functionality oracle). While the correctness requirement (when no party is corrupted) holds for any choice of κ, s, C, the security requirement only considers adversaries that run in time $\mathsf{poly}(\kappa)$. That is, we require indistinguishability (in the sense of Definition 2.5) between

- $\{\mathbf{REAL}_{\Pi,\mathcal{A}(z),P_i}(\mathrm{C},x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},\mathrm{C}\in\mathcal{C},x,y,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),P_i}(\mathrm{C},x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},\mathrm{C}\in\mathcal{C},x,y,z\in\{0,1\}^*}$

where $\mathcal{C}$ is the class of Boolean circuits that take two bit-strings as inputs and output two bit-strings, x, y are of lengths corresponding to the inputs of C, $\mathcal{F}$ is the functionality computed by C, and the next message functions of the parties P_0, P_1 is as specified by the protocol compiler on inputs $1^\kappa, 1^s, \mathrm{C}$.

MPC. Let n be the number of parties, which will be denoted by $P_1, \ldots, P_n$. All parties share the public input statement $x \in \mathcal{L}$ and a randomness share r_i. The *view* of P_i, denoted by V_i, includes x, r_i and the messages received by P_i during the execution of a protocol Π. Note that the messages sent by an uncorrupted player P_i as well as its local output can be inferred from V_i and x by invoking Π. The following definitions are taken from [IKOS09] verbatim.

Definition 2.6 (Consistent views) *We say that a pair of views V_i, V_j are consistent (with respect to the protocol Π and some public input x) if the outgoing messages implicit in V_i are identical to the incoming messages reported in V_j and vice versa.*

We consider security of protocols in both the honest-but-curious (passive) and the malicious (active) models. In the former model, one may break the security requirements into the following correctness and privacy requirements.

Definition 2.7 (Correctness) *We say that* Π *realizes a deterministic n-party functionality* $\mathcal{F}(x, r_1, \ldots, r_n)$ *with perfect (resp., statistical) correctness if for all inputs* $(x, r_1, \ldots, r_n)$*, the probability that the output of some player is different from the output of* $\mathcal{F}$ *is* 0 *(resp., negligible in* κ*), where the probability is over the independent choices of the random inputs* $r_1, \ldots, r_n$*.*

Definition 2.8 (t-Privacy) *Let* $1 \leq t < n$*. We say that* Π *realizes* $\mathcal{F}$ *with perfect* t*-privacy if there is a* PPT *simulator* $\mathcal{S}$ *such that for any inputs* $(x, r_1, \ldots, r_n)$ *and every set of corrupted players* $T \subset [n]$*, where* $|T| \leq t$*, the joint view* $\mathbf{View}_T(x, r_1, \ldots, r_n)$ *of players in* T *is distributed identically to* $\mathcal{S}(T, x, \{r_i\}_{i \in T}, \mathcal{F}_T(x, r_1, \ldots, r_n))$*. The relaxations to statistical or computational privacy are defined in the natural way. That is, in the statistical (resp., computational) case we require that for every distinguisher* $\mathcal{D}$ *(resp.,* $\mathcal{D}$ *with circuit size* $\mathsf{poly}(\kappa)$*) there is a negligible function* $\delta(\cdot)$ *such that*

$$\begin{aligned} &|\Pr[\mathcal{D}(\mathbf{View}_T(\kappa, x, r_1, \ldots, r_n)) = 1] \\ &\quad - \Pr[\mathcal{D}(\mathcal{S}(\kappa, T, x, \{r_i\}_{i \in T}, \mathcal{F}_T(x, r_1, \ldots, r_n))) = 1]| \leq \delta(\kappa). \end{aligned}$$

In the malicious model, in which corrupted players may behave arbitrarily, security cannot be generally broken into correctness and privacy as above. However, for our purposes we only need the protocols to satisfy a weaker notion of security in the malicious model that is implied by the standard general definition. Specifically, it suffices that Π be t-private as defined above, and moreover it should satisfy the following notion of correctness in the malicious model.

Definition 2.9 (t-Robustness) *We say that* Π *realizes* $\mathcal{F}$ *with perfect (resp., statistical)* t*-robustness if it is perfectly (resp., statistically) correct in the presence of a honest-but-curious adversary as in Definition 2.7, and furthermore for any computationally unbounded malicious adversary corrupting a set* T *of at most* t *players, and for any inputs* $(x, r_1, \ldots, r_n)$*, the following robustness property holds. If there is no* $(x, r_1, \ldots, r_n)$ *such that* $\mathcal{F}(x, r_1, \ldots, r_n) = 1$*, then the probability that some uncorrupted player outputs* 1 *in an execution of* Π *in which the inputs of the honest players are consistent with* $(x, r_1, \ldots, r_n)$ *is* 0 *(resp., is negligible in* κ*).*

Our main theorems are proven in the presence of a static active adversary, that corrupts one of the parties at the onset of the execution. Nevertheless, our proof utilizes an adversary that may adaptively corrupt a subset of the servers (which is needed in the MPC-in-the-head emulation of the underlying MPC protocol).

Non-interactive secure computation (NISC). We follow the notation of [IKO+11]. NISC may involve a trusted setup, an implementation of some (non-reactive) functionality $\mathcal{H}$. We therefore refer to such an NISC scheme as NISC/$\mathcal{H}$ specifying the ideal functionality $\mathcal{H}$ that is accessed by the NISC protocol. In this work we instantiate $\mathcal{H}$ with $\mathcal{F}_{\text{OT}}$.

2.7.1 Weaker Notions of Security

We consider secure two-party protocols where an abort event by the receiver is correlated with its input. This weakening is formalized by allowing the sender to submit an additional input predicate to the trusted party that computes the functionality, that is applied on the receiver's input and is leaked to the sender. We note that there is no leakage if the receiver's output is never revealed to the sender. Furthermore, to obtain the leakage from the receiver's output, a malicious sender must take a risk of being caught cheating.

Security in the presence of correlated abort. The first relaxed notion of security considers simulation based security in the presence of *correlated abort* where the malicious sender learns a predicate applied

on the receiver's input. This leakage is formalized by allowing the functionality to additionally receive a predicate $P(\cdot)$ from the adversary. The functionality then applies the predicate on the honest party's input and delivers the output to the honest party only if the predicate evaluates to 1. In this work, we consider two-party functionalities executed between a sender and a receiver where only the receiver obtains the answer. In such a scenario, an adversary is allowed to provide a predicate only if it corrupts the sender.

More formally, we consider $\mathbf{REAL}_{\Pi,\mathcal{A}(z),P_i}(x,y,\kappa,s)$ as described for Definition 2.5. The ideal world is defined differently. We denote by $\mathbf{IDEAL}^{leak}_{\mathcal{F},\mathcal{S}(z),P_i}(x,y,\kappa,s)$ the output of the honest party P_i and the simulator $\mathcal{S}$ in the ideal model where $\mathcal{S}$ provides a predicate P and $\mathcal{F}$ is computed by a trusted party that delivers the output to P_i only if P returns 0 on P_i's input, and delivers $\perp$ otherwise.

Security in the presence of IVD attacks. This relaxed notion of security considers simulation based security in the presence of *input-value disjunction (IVD) attacks*, where the malicious sender learns a predicate of the receiver's input. In Section 5 we discuss how to compile IVD security into full security. More formally, security is defined as in correlated abort with the exception that the predicate P is a disjunctive predicate on the receiver's input bits.

Security in the presence of WVD attacks. Another relaxed notion of security is *wire-value disjunction (WVD) attacks* which is specified by a disjunctive predicate $P(\cdot)$ that is applied on the wire values of the computed circuit C (rather than on the receiver's input value itself). Namely, $P(\cdot)$ includes a set of pairs W of the form (w,b) where w is a wire in C and $b \in \{0,1\}$. The functionality sends $\perp$ to the receiver in case wire w is evaluated to b. In Section 4.2.1 we present a 2PC secure protocol in the presence of WVD attacks. More formally, security is defined as in correlated abort with the exception that the predicate P is a disjunctive predicate on the wire values in the computation $C(x,y)$ where C is the Boolean circuit that describes $\mathcal{F}$.

3 Framework for Actively Secure Garbled Circuits

In this section we present a general framework for designing an actively secure two-party computation protocol for a functionality $\mathcal{F}$ given its Boolean circuit representation. It is based on (and can capture) the approach of [IKO+11], but incorporates several additional ideas. The framework consists of the following steps:

Step 1: Reduce $\mathcal{F}$ to a local $\widehat{\mathcal{F}}$. In this step, given a circuit for $\mathcal{F}$ and a (computational) security parameter κ, we obtain an NC^0 functionality $\widehat{\mathcal{F}}$ and a two-party protocol Π_1 that securely realizes $\mathcal{F}$ in the $\widehat{\mathcal{F}}$-hybrid model with active security. The protocol Π_1 will have the feature of invoking $\widehat{\mathcal{F}}$ just once and making only a black-box use of a PRG. In Section 4 we describe two implementations of this step that combine Yao-style garbling with BMR-style randomization. Our first implementation of this step is used for our main asymptotic result and the second for our concretely efficient protocol.

Step 2: Reduce $\widehat{\mathcal{F}}$ to DCOT. In this step, we obtain an actively secure protocol Π_2 for $\widehat{\mathcal{F}}$ where the parties have access to an augmented OT functionality we refer to as *doubly certified oblivious transfer* (DCOT). The DCOT functionality $\mathcal{F}_{\mathrm{DCOT}}$ in its core performs the parallel OT functionality but additionally assures the receiver that the pairs of strings transmitted satisfy a global consistency predicate as long as its input is encoded in some correct format. This step is implemented via two intermediate steps:

1. Start with a perfectly secure non-interactive protocol $\Pi_{1.5}$ for $\widehat{\mathcal{F}}$ using a standard parallel OT oracle, where security should only hold in the presence of a passive sender and an active receiver. Such protocols were referred to in [IKO+11] as NISC/OT protocols, and can be based on any decomposable

randomized encoding for $\widehat{\mathcal{F}}$ [IK02, IKOS08] (which can also be viewed as a perfectly secure projective garbling scheme [Yao86, BHR12] or a private simultaneous messages protocol [FKN94] with 1-bit inputs). We exploit the simplicity of $\widehat{\mathcal{F}}$ to get an efficient realization of this step via the standard reduction from $\binom{n}{1}$-OT to $\binom{2}{1}$-OT [BCR86].

2. Compile $\Pi_{1.5}$ into a protocol Π_2 in the $\mathcal{F}_{\text{DCOT}}$-hybrid where the sender and receiver rely on the DCOT oracle to perform the parallel OTs prescribed by $\Pi_{1.5}$ while assuring the receiver that the sender's inputs to the parallel OT oracle were constructed correctly according to $\Pi_{1.5}$ while verifying the correctness of the receiver's input. To make the DCOT predicate simpler, we allow it to be non-deterministic: the predicate depends on two additional NP witnesses provided by the sender and the receiver. Condition on the receiver's input being of the correct format, it accepts the selected strings if the witness used by an honest sender is valid, and rejects (except with negligible probability) if there is no valid witness that satisfies the global consistency predicate.

Step 3: Reduce DCOT to commit-and-prove and parallel OT. We obtain a constant-round protocol Π_3 for the DCOT functionality in a hybrid model where the parties have access to a commit-and-prove (C&P) oracle and a parallel OT oracle. Loosely speaking, the C&P functionality is a *reactive* functionality that proceeds in two phases. In the first phase, the sender commits to an input, and in the second phase it proves that this input satisfies some NP relation chosen by the receiver.

Our asymptotic result requires an extra intermediate step where the DCOT realization is reduced first to commit-and-prove and receiver certified OT (RCOT). This is required in order to enforce a global certification on the receiver's input. Using the transformation from [IKO+11], RCOT can be reduced to parallel OT. The details of this step can be found in Sections 5 and 6.

Our implementation of DCOT in this step deviates from the approach of [IKO+11] which relies on an information theoretic MPC protocol to simultaneously perform both the computations of the parallel OT and the "certification." We decouple the two by relying on the parallel OT and the C&P functionalities individually in separate steps, which leads to an efficiency improvement over the COT implementation of [IKO+11].

Step 4: Reduce commit-and-prove to parallel OT. Finally, we use a protocol Π_4 to reduce the C&P functionality to parallel OT via an MPC-in-the-head approach [IKOS07]. Prior works [IMS12, IW14] have shown how to realize C&P with sub-linear communication using PCPs and CRHFs. We provide a leaner alternative construction that realizes the C&P functionality in the parallel OT hybrid with constant communication overhead.[10] This construction is a variant of the recent sublinear zero-knowledge protocol from [AHIV17] and presented in Seciton 7.

Input-dependent failures. A (standard) issue (also present in [IKO+11]) that we have to address is that Step 2 will only achieve a slightly relaxed notion of security where an active adversary corrupting the COT sender can cause an input-dependent abort for the receiver.[11] More precisely, a corrupted sender can induce a disjunctive predicate (such as $x_3 \vee x_5 \vee \overline{x_7}$) on the receiver's input bits that if satisfied, will make the receiver abort. We refer to this as an input-value disjunction (IVD) attack and the resulting abort as IVD-abort. The IVD attack on Step 2 results in the final protocol (obtained by composing all 4 steps) realizing a relaxed functionality that allows for similar IVD attacks on $\mathcal{F}$ (and *only* such attacks). We address this issue in our asymptotic result by encoding the receiver's input to the NC_0 functionality using AG codes.

[10] We remark that our protocol can be instantiated using ideal commitments (or even one-way functions in the plain model), but we present a version based on OT as our end goal is to design an efficient secure protocol which anyway requires OT.

[11] For example, it can modify an honest sender's strategy by setting some of the OT inputs to $\perp$, which will cause the receiver to abort for those values as inputs.

4 Secure 2PC in NC^0-Hybrid

In this section, we provide our compilation from an arbitrary 2PC functionality $\mathcal{F}$ to an NC^0 functionality $\widehat{\mathcal{F}}$ and a protocol Π_1 that securely realizes $\mathcal{F}$ in the $\widehat{\mathcal{F}}$-hybrid. We provide two such compilations which will be variants of analogous constructions in [IKO+11]. Previous two-party protocols essentially have a sender who creates and delivers a garbling to a receiver. In contrast, we modify the constructions in [IKO+11] to incorporate additional randomization from the receiver inspired by the BMR approach [BMR90].

Overview. On a high-level, the BMR protocol proceeds in two phases: (1) First, in an offline phase, the parties jointly compute a garbling of the circuit they wish to evaluate on their inputs, and (2) in an online phase, the parties share the keys corresponding to their inputs and shares of the garbled circuit and output a translation table. Each party then individually reconstructs the garbled circuit, evaluates the garbled circuit using the input keys and then obtains the result of the computation.

In slight more detail and restricting our discussion to the two-party setting, there are two key differences between the BMR approach and the standard Yao approach. First, the parties provide two keys for every wire in the circuit so that the keys for the output wire are encrypted in a garbled row under (the respective) keys from both parties. Second, the association of the keys with the actual values remain hidden from both parties as both of them contribute shares (or masks) that are combined to decide the association.

One can model the offline phase in the BMR approach as an actively secure computation of a "garbling" functionality where the parties provide keys and masks for each wire. However, unless we assume some strong form of a PRG (i.e. PRGs that can be computed by a constant-depth circuits), the computation in the offline phase will not be a constant-depth circuit as it involves incorporating the circuit that computes the PRG. An important simplification considered in [LPSY15] allows the parties to locally compute the PRG values under their keys and provide the PRG values directly as inputs to the computation in the offline phase. Consequently, the computation in the offline phase becomes a constant-depth (NC^0) circuit over the inputs. However, such a functionality cannot guarantee the correctness of the PRG values provided by corrupted parties. The work of [LPSY15] demonstrates that this will not be a problem and it suffices to securely compute the modified NC^0 functionality with active security in order to achieve security against active adversaries in the overall protocol. More precisely, [LPSY15] demonstrate that bad PRG values provided by corrupted parties do not affect the security of the overall protocol. The key insight can be understood as follows: The presence of two keys (of the output wire) from both parties in every plaintext encrypted in the garbled rows forces an abort whenever a bad PRG value occurs on the evaluation path of the garbled circuit (aka active path). Since the keys to values association is randomized by both parties, the event of such an abort will be independent of the true value associated with that wire. Furthermore, absence of an abort implies that the computation has proceeded correctly.

The IKOPS protocol [IKO+11], on the other hand, is an extension of the standard Yao protocol where the garbling is computed by a sender and the keys are delivered to a receiver via an OT protocol. As previously observed [LP07, LP12], it must be ensured that an active sender does not create a bad garbled circuit. In the IKOPS protocol, the authors show how to restrict the effect of such an attack by providing some additional "authentication" information to the receiver. They propose two variants of this approach: In the first variant, the NC^0 functionality authenticates the PRG values, that are locally computed and provided as input by the sender, whereas in the second variant only the color bits (or point-and-permute bits) are authenticated. The high-level idea is that the authentication information makes the sender commit to parts of the garbling and restricts the space of attacks that can be carried out by the sender. Nevertheless, in both these variants, the sender may still cause the receiver to abort depending on its input or the actual wire values. To make the abort independent of the receiver's input, the IKOPS protocol incurs a $\mathsf{polylog}(\kappa)$ factor overhead as it

precompiles the functionality $\mathcal{F}$ to be immune to such attacks. Consequently, the resulting final protocol has a $\mathsf{polylog}(\kappa)$ communication complexity overhead over the standard passively secure Yao protocol.

Our new approach combines the benefits of the BMR protocol with the IKOPS variants to achieve a protocol that achieves communication efficiency with a constant overhead over the semi-honest Yao protocol. On a high-level, we will have the sender provide additional authentication information as in the IKOPS protocol, but will randomize the association of keys to values following the BMR approach.

4.1 Variant 1: Authenticating The PRG Values

In our first variant the functionality authenticates the PRG values submitted by the sender for creating the garbling. Following [IKO+11], the functionality will receive as input from the sender, for every garbled gate, keys and the PRG evaluations under these keys and from the receiver it receives as input a MAC key SK that is used to authenticate the PRG evaluations. The high-level idea here is to require the receiver to verify whether the PRG values obtained during the evaluation of the garbled circuit are consistent with authentication information received from the functionality and letting it abort if the authentication fails. As mentioned before, we incorporate the BMR randomization by having random bits supplied by the receiver to randomize the association of key and values for each wire.

Formally, we establish the following Lemma.

Lemma 4.1 (AuthPRG Compiler) *There exists a compiler* $\mathsf{AuthPRG}$ *that given* κ *(PRG seed length),* s *(statistical security parameter) and a two-party functionality* $\mathcal{F}(x, y)$*, expressed by a circuit* C*, outputs another two-party functionality* $\widehat{\mathcal{F}}$ *and protocol* Π_1 *that securely realizes* $\mathcal{F}$ *in the* $\widehat{\mathcal{F}}$*-hybrid with the following features:*

- $\widehat{\mathcal{F}}$ *is represented by an* NC^0 *circuit of size* $O(|\mathrm{C}|\kappa)$*. The receiver's inputs to* $\widehat{\mathcal{F}}$ *include its original input* y *to* $\mathcal{F}$ *and a string of length* $O(|\mathrm{C}| + \kappa)$ *that it will choose uniformly at random.*
- Π_1 *makes a single invocation to the* $\widehat{\mathcal{F}}$ *oracle.*
- Π_1 *makes* $O(|\mathrm{C}|)$ *black-box calls to a length-doubling PRG:* $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^{\kappa} \to \{0,1\}^{2\kappa}$.

Proof: We begin with a description of the compiled functionality $\widehat{\mathcal{F}} = \mathcal{F}_{\mathrm{AuthPRG}}$ and then continue with our protocol description. If $s > \kappa$ then the compiler sets $s = \kappa$. This is because we require our simulation error to be only bounded by $2^{-s} + \nu(\kappa)$ for some negligible function $\nu(\cdot)$.

We now describe our compiler $\mathsf{AuthPRG}$ that on input $(\kappa, s, \mathsf{desc}(\mathcal{F}))$ outputs $(\widehat{\mathcal{F}}, \Pi_1)$.

The $\mathbf{NC^0}$ Functionality $\widehat{\mathcal{F}} = \mathcal{F}_{\mathbf{AuthPRG}}$. For the functionality $\mathcal{F}$, we denote the set of wires by W and the set of gates by G. The compiled functionality $\mathcal{F}_{\mathrm{AuthPRG}}$ obtains from the sender and the receiver their respective inputs x and y to the function $\mathcal{F}$ and their making shares $\{\lambda_w^R\}_{w \in W}$ and $\{\lambda_w^S\}_{w \in W}$. Additionally, it receives keys and PRG evaluations from the sender and authentication keys from the receiver. Following $\mathcal{F}$, $\mathcal{F}_{\mathrm{AuthPRG}}$ creates the garbled circuit and authentication information which it delivers to the receiver. We provide a formal description of $\mathcal{F}_{\mathrm{AuthPRG}}$ in Figure 2.

Next, we describe our protocol.

Protocol 1 (Protocol $\widehat{\Pi}_1$) *The parties's common input is a Boolean circuit* C*, expressed by a set of wires* W *and a set of gates* G*.*

Parameters: *Let* s *be the statistical security parameter and* κ *be the computational security parameter. Let* $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^{\kappa} \to \{0,1\}^{2\kappa}$ *be a PRG and let* $(\mathsf{ECC}, \mathsf{MAC})$ *be a* $2^{-\kappa}$ *secure MAC-scheme (cf.*

Functionality $\mathcal{F}_{\text{AuthPRG}}$

Let C represent the circuit that computes the functionality $\mathcal{F}$ and comprises of a set of wires W and a set of gates G.

The sender's inputs to the functionality are:

- Input x.
- For every wire $w \in W$ (excluding output wires), keys $k_w^0, k_w^1 \leftarrow \{0,1\}^\kappa$ and mask λ_w^S.
- For every gate $g \in G$ with input wires a and b, the PRG values, $F_{a,0}^{g,0}, F_{a,0}^{g,1}, F_{a,1}^{g,0}, F_{a,1}^{g,1}, F_{b,0}^{g,0}, F_{b,0}^{g,1}, F_{b,1}^{g,0}, F_{b,1}^{g,1}$ and their encodings under $\mathsf{Enc}_{\mathsf{lin}}$, $EF_{a,0}^{g,0}, EF_{a,0}^{g,1}, EF_{a,1}^{g,0}, EF_{a,1}^{g,1}, EF_{b,0}^{g,0}, EF_{b,0}^{g,1}, EF_{b,1}^{g,0}, EF_{b,1}^{g,1}$
- For every gate g, input wire w, $b, r \in \{0,1\}$, a tag $\tau_{w,b,S}^{g,r}$ (that will be used to generate the MAC tag for the PRG value computed based on the key k_w^b).

The receiver's inputs to the functionality are:

- Input y.
- A mask for every wire $w \in W$, λ_w^R.
- A MAC key $\mathsf{SK} \in \{0,1\}^{\ell_{\mathsf{key}}\kappa}$.

The functionality performs the following computations:

1. Compute the combined masks for every wire $w \in W$ as $\lambda_w = \lambda_w^S \oplus \lambda_w^R$.
2. For every gate $g \in G$ with input wires a and b and output wire c, compute the garbled table as follows:

$$R_g^{00} = F_{a,0}^{g,0} \oplus F_{b,0}^{g,0} \oplus (k_c^0||0) \oplus \Big((\lambda_a\lambda_b \oplus \lambda_c)(k_c^1||1 \oplus k_c^0||0)\Big)$$
$$R_g^{01} = F_{a,0}^{g,1} \oplus F_{b,1}^{g,0} \oplus (k_c^0||0) \oplus \Big((\lambda_a \oplus \lambda_a\lambda_b \oplus \lambda_c)(k_c^1||1 \oplus k_c^0||0)\Big)$$
$$R_g^{10} = F_{a,1}^{g,0} \oplus F_{b,0}^{g,1} \oplus (k_c^0||0) \oplus \Big((\lambda_b \oplus \lambda_a\lambda_b \oplus \lambda_c)(k_c^1||1 \oplus k_c^0||0)\Big)$$
$$R_g^{11} = F_{a,1}^{g,1} \oplus F_{b,1}^{g,1} \oplus (k_c^0||0) \oplus \Big((1 \oplus \lambda_a \oplus \lambda_b \oplus \lambda_a\lambda_b \oplus \lambda_c)(k_c^1||1 \oplus k_c^0||0)\Big)$$

3. Send the receiver Rec the following values:
   - $\{(R_g^{00}, R_g^{01}, R_g^{10}, R_g^{11})\}_{g \in G}$.
   - $(k_w^0||0) \oplus \Big((\lambda_w \oplus x_i)(k_w^1||1 \oplus k_w^0||0)\Big)$ for every pair (w,i) where the input wire w carries the i^{th} bit of x.
   - $(k_w^0||0) \oplus \Big((\lambda_w \oplus y_i)(k_w^1||1 \oplus k_w^0||0)\Big)$ for every pair (w,i) where the input wire w carries the i^{th} bit of y.
   - The masked MAC for every PRG value, namely, $\tau_{w,b,R}^{g,r} = \tau_{w,b,S}^{g,r} \oplus \mathsf{MAC}_{\mathsf{SK}}(EF_{w,b}^{g,r})$.
   - λ_w for every output wire.

Figure 2: The offline functionality $\mathcal{F}_{\text{AuthPRG}}$.

Section 2.3.2) where $\mathsf{ECC} = \{\mathsf{Enc}_{\mathsf{lin}} : \{0,1\}^{\kappa+1} \to \{0,1\}^{\ell_{\mathsf{in}}\kappa}\}$ *and* $\mathsf{MAC} = \{\mathsf{MAC}_{\mathsf{SK}} : \{0,1\}^{\ell_{\mathsf{in}}\kappa} \to \{0,1\}^{\ell_{\mathsf{out}}\kappa}\}_{\mathsf{SK} \in \{0,1\}^{\ell_{\mathsf{key}}\kappa}}$.

Convention for expressing PRG values. *The number of random bits that we need to extract from each key*

(acting as a seed to the PRG) depends on the number of gates the wire associated with the key occurs as an input. In standard garbling, if a wire occurs as input in T gates, then each key associated with the wire will be used in $2T$ rows and in each row we will require κ (output key) $+\ell_{\mathsf{out}}$ (authentication information) $+1$ (point-and-permute bit) bits. In order to describe our protocol succinctly we will employ a PRF-type definition: $F_k(g,r)$ will represent a unique portion of $\kappa+\ell_{\mathsf{out}}+1$ bits in the output of $\mathrm{G}_{\mathrm{PRG}}(k)$ that is used for gate g in row r.

- **Input:** *The sender is given input x and the receiver is given input y. Both parties are given the security parameters $1^\kappa, 1^s$ and the description of a Boolean circuit C.*
- **The sender's input to $\mathcal{F}_{\mathrm{AuthPRG}}$:**
  - *Input x.*
  - *For every wire $w \in W$, keys k_w^0, k_w^1 sampled uniformly at random from $\{0,1\}^\kappa$ and a mask bit $\lambda_w^S \leftarrow \{0,1\}$ sampled uniformly at random.*
  - *For every gate $g \in G$, input wire $w \in W$, point-and-permute bit b and a row r, a tag $\tau_{w,b,S}^{g,r}$ (that will be used to generate the MAC tag for the PRG value computed based on the key k_w^b).*
  - *For every gate $g \in G$, with input wires a and b, the PRG values,*
    $F_{k_a^0}(g,0), F_{k_a^0}(g,1), F_{k_a^1}(g,0), F_{k_a^1}(g,1), F_{k_b^0}(g,0), F_{k_b^0}(g,1), F_{k_b^1}(g,0), F_{k_b^1}(g,1)$,
    and their encodings under $\mathsf{Enc}_{\mathsf{lin}}$,
    $EF_{a,0}^{g,0}, EF_{a,0}^{g,1}, EF_{a,1}^{g,0}, EF_{a,1}^{g,1}, EF_{b,0}^{g,0}, EF_{b,0}^{g,1}, EF_{b,1}^{g,0}, EF_{b,1}^{g,1}$
- **The receiver's input to $\mathcal{F}_{\mathrm{AuthPRG}}$:**
  - *Input y.*
  - *A mask for every wire $w \in W$, λ_w^R.*
  - *A MAC key $\mathrm{SK} \in \{0,1\}^{\gamma_2\kappa}$.*
- **The receiver's outcome from $\mathcal{F}_{\mathrm{AuthPRG}}$:**
  - *$\{(R_g^{00}, R_g^{01}, R_g^{10}, R_g^{11})\}_{g\in G}$.*
  - *$k_w||z_w$ for every input wire w.*
  - *The masked MAC for every PRG value, namely, $\tau_{w,b,R}^{g,r}$.*
  - *λ_w for every output wire.*
- *In addition, the sender encrypts the mask used to mask the MAC values and sends it to the receiver. Namely, it sends the ciphertext $c_{w,b}^{g,r} = \mathsf{Enc}_{k_w^b}(\tau_{w,b,S}^{g,r}) = F_{k_w^b}(g,(2+r)) \oplus \tau_{w,b,S}^{g,r}$.*
- **Computing output.** *Upon receiving the output from the functionality, the receiver proceeds to evaluate the garbled circuit as follows: Let the gates be arranged in some topological order. We will maintain the invariant that if the receiver has not aborted when it processes some gate g with input wires a and b, then it possess keys k_a and k_b and row indicators (color bits) Λ_a and Λ_b.*

  Base case: *For each input wire $w \in W$, the receiver obtains $k_w||z_w$ from the functionality. It sets Λ_w to z_w.*

  Induction step: *Consider an arbitrary gate $g \in G$ in the topological order with input wires a and b and output wire c. By our induction hypothesis, if the receiver has not yet aborted then it has keys k_a, k_b and colors Λ_a and Λ_b. Then the receiver first checks the correctness of the PRG values as follows:*

– *For* $\alpha \in \{0,1\}$*, compute* $\tau^{g,\alpha}_{a,\Lambda_a,S} = \mathsf{Dec}_{k_a}(c^{g,\alpha}_{a,\Lambda_a}) = c^{g,\alpha}_{a,\Lambda_a} \oplus F_{k_a}(g,(2+\alpha))$ *and check if it equals*

$$\tau^{g,\alpha}_{a,\Lambda_a,R} \oplus \mathsf{MAC}_{\mathsf{SK}}(\mathsf{Enc}_{\mathsf{lin}}(F_{k_a}(g,\alpha))).$$

If the checks fail, it aborts. Otherwise, it computes

$$k_c||\Lambda_c = R^{\Lambda_a\Lambda_b}_g \oplus F_{k_a}(g,\Lambda_a) \oplus F_{k_b}(g,\Lambda_b).$$

Finally, if the receiver has not aborted, it possesses the colors Λ_w *for every output wire* $w \in W$*. It then outputs* $\Lambda_w \oplus \lambda_w$ *as the output on wire* w *for every output wire.*

Proof of security. We prove correctness of our compilation next. Our proof follows by showing that for every adversary $\mathcal{A}$ in the $\mathcal{F}_{\mathrm{AuthPRG}}$-hybrid world there exists a simulator $\mathcal{S}$ in the ideal world with access to functionality $\mathcal{F}$. We describe our simulator for active corruption of sender and receiver and prove correctness.

Sender Sen **is corrupted.**

1. Simulator $\mathcal{S}$ internally invokes the adversary $\mathcal{A}$ on $(1^\kappa, 1^s, x)$ with uniformly sampled random tape, where x is the input given to the Sender.

2. $\mathcal{S}$ internally emulates functionality $\mathcal{F}_{\mathrm{AuthPRG}}$ as follows:

   - $\mathcal{S}$ receives from the adversary its input to $\mathcal{F}_{\mathrm{AuthPRG}}$ that is consists of an input $\widetilde{x}$, a set of keys $\{\widetilde{k}_0, \widetilde{k}_1\}_{w\in W}$ masking bits $\{\lambda^S_w\}_{w\in W}$, a set of alleged PRG values: $F^{g,0}_{a,0}, F^{g,1}_{a,0}, F^{g,0}_{a,1}, F^{g,1}_{a,1}$, $F^{g,0}_{b,0}, F^{g,1}_{b,0}, F^{g,0}_{b,1}, F^{g,1}_{b,1}$, their alleged encodings, $EF^{g,0}_{a,0}, EF^{g,1}_{a,0}, EF^{g,0}_{a,1}, EF^{g,1}_{a,1}, EF^{g,0}_{b,0}, EF^{g,1}_{b,0}$, $EF^{g,0}_{b,1}, EF^{g,1}_{b,1}$ and the tag shares $\{\tau^{g,r}_{w,b,S}\}$.
   - For every $w \in W$, $\mathcal{S}$ generates a mask λ^R_w.
   - It then computes a simulated garbled circuit as follows. $\mathcal{S}$ first chooses for each wire $w \in W$ that is not an input wire, a color Λ_w. It then defines an active path which is induced by these color bits. Namely, for each gate $g \in G$ with input a and b and output wire c, the simulator sets the row (Λ_a, Λ_b) in the garbled gate as

     $$F^{g,\Lambda_a}_{a,\Lambda_a} \oplus F^{g,\Lambda_b}_{b,\Lambda_b} \oplus (k^{\Lambda_c}_c||\Lambda_c)$$

     whereas the remaining three rows are uniformly sampled at random from $\{0,1\}^{\kappa+1}$. Denote the outcome by the set of tuples $\{(\widetilde{R}^{00}_g, \widetilde{R}^{01}_g, \widetilde{R}^{10}_g, \widetilde{R}^{11}_g)\}_{g\in G}$.
     Note that the simulator does not need to sample a MAC key and generate the masked MACs for every PRG value as it knows these PRG values and can directly check for correctness during the evaluation of the active path it chose above. Specifically, the inputs provided by the sender to $\mathcal{F}_{\mathrm{AuthPRG}}$ imply a disjunctive predicate P that can be defined as follows. For every $(g,c) \in G \times \{0,1\}$ and an input wire a such that $F^{g,c}_{a,c} \neq F_{k^c_a}(g,a)$ it includes the literal v_g if $c = 0$ and $\neg v_j$ in case $c = 1$. Applying this predicate on the active path chosen by the simulator will determine whether the receiver's aborts the computation or not.
   - Finally, the simulator evaluates the garbled circuit on $(\widetilde{x}, \widetilde{y})$, where $\widetilde{y}$ is an arbitrary input. More concretely, given these input strings, the simulator uses the input keys that correspond to these values and completes the evaluation of the garbled circuit as would the honest receiver do when given an input $\widetilde{y}$.

3. If, at some point, the receiver decrypts some garbled gate $g \in G$ such that the MAC verification fails for the active path, then the simulator aborts, sending $\perp$ to the trusted party that computes $\mathcal{F}$. Else, the simulator submits $\widetilde{x}$ to its trusted party as the adversary's input.

4. The simulator outputs whatever the adversary does.

We next prove that conditioned on not aborting both in the real and in the ideal execution, then the adversary's view in computationally indistinguishable.

Note first that in the $\mathcal{F}_{\mathrm{AuthPRG}}$-hybrid execution, the sender does not receive any messages, as functionality $\mathcal{F}_{\mathrm{AuthPRG}}$ only communicates with the receiver, and the parties do not communicate directly. It is therefore left to prove that the sender cannot cause any damage by providing inconsistent PRG values to the offline functionality. Intuitively speaking, there may be two scenarios: either the adversary will get caught by the receiver, or not. In our proof, we first claim that the adversary is caught with very high probability due to the fact that a successful attacks boils down to forging the MAC, and that the abort event does not leak any information about the honest party's input. Formally, we prove the following claims.

Claim 4.2 *For any two inputs y, y', it holds that the probabilities that the receiver outputs $\perp$ (i.e. aborts) in* $\mathbf{REAL}^{\widehat{\mathcal{F}}}_{\widehat{\Pi}_1,\mathcal{A}(z),\mathrm{Rec}}(x, y, \kappa, s)$ *and* $\mathbf{REAL}^{\widehat{\mathcal{F}}}_{\widehat{\Pi}_1,\mathcal{A},\mathrm{Rec}}(x, y', \kappa, s)$ *are equal.*

Proof: Given the sender's input to the NC^0 functionality, we will show that for every pair of inputs $y, \widetilde{y}$, there is a reversible mapping from any set of wire masks $\{\lambda^R_w\}_w$ to another set $\{\widetilde{\lambda}^R_w\}_w$ such that if the receiver aborts on input $(y, \{\lambda^R_w\}_w, \{\tau^{g,r}_{w,b,R}\}, \mathrm{SK})$, then it will abort on input $(\widetilde{y}, \{\widetilde{\lambda}^R_w\}_w, \{\tau^{g,r}_{w,b,R}\}, \mathrm{SK})$. The intuition here is that, in the evaluation, the receiver uses keys on the active path determined by the color bits Λ_w. Given y and $\{\lambda^R_w\}_w$, we will show that there is an equivalent set of masks $\{\widetilde{\lambda}^R_w\}_w$ for $\widetilde{y}$ that will result in the same sequence of color bits Λ_w and the same active path which in turn means that the same set of keys are decrypted in every garbled gate.

Let the sender's inputs be $(x, \{\lambda^S_w\}_w, \{(k^0_w, k^1_w)\}_w, \{F^{g,0}_{w,0}, F^{g,1}_{w,0}, F^{g,0}_{w,1}, F^{g,1}_{w,1}\}_{g=(a,b,c),w\in\{a,b\}}, \{\tau^{g,r}_{w,b,S}\})$. We will inductively construct the shares $\widetilde{\lambda}^R_w$ and show that the receiver's evaluation will proceed identically.

Base case: When the receiver's input are $y, \{\lambda^R_w\}_w$, the key received corresponding to the wire w carrying the i^{th} bit of sender's input $\widetilde{x}_i$ is computed as follows:

$$(k^0_w||\Lambda_0) \oplus \Big((\lambda_w \oplus \widetilde{x}_i) \wedge (k^1_w||\Lambda_1 \oplus k^0_w||\Lambda_0)\Big)$$

where recall $\lambda_w = \lambda^S_w \oplus \lambda^R_w$. We will set $\widetilde{\lambda}^S_w = \lambda^S_w$. This means that corresponding to the sender's inputs, the receiver obtains the same set of keys and masks Λ_b which would result in the same color bits Λ_w for these input wires.

For the keys corresponding to the receiver's input, the computation for the i^{th} input wire w is:

$$(k^0_w||\Lambda_0) \oplus \Big((\lambda_w \oplus y_i) \wedge (k^1_w||\Lambda_1 \oplus k^0_w||\Lambda_0)\Big).$$

We will now set $\widetilde{\lambda}^R_w = y_i \oplus \widetilde{y}_i \oplus \lambda^R_w$. This means that

$$\lambda^S_w \oplus \widetilde{\lambda}^R_w \oplus \widetilde{y}_i = \lambda^S_w \oplus \lambda^R_w \oplus y_i$$

which in turn means that the receiver will obtain the same key and result in the same computation to obtain Λ_w. Let $\rho_w, \widetilde{\rho}_w$ be the actual wire values in wire w relative to y and $\widetilde{y}$, respectively.

Induction case: When processing gate g, let the input keys k_a, k_b and color bits Λ_a, Λ_b be the same for both inputs. In the computation with input y, the receiver for gate g chooses ciphertext $R_g^{\Lambda_a \Lambda_b}$ which is computed as follows:

$$R_g^{\Lambda_a \Lambda_b} = F_{a,\Lambda_a}^{g,\Lambda_b} \oplus F_{b,\Lambda_b}^{g,\Lambda_a} \oplus (k_c^0 || \Lambda_0) \oplus \Big(((\lambda_a \oplus \Lambda_a)(\lambda_b \oplus \Lambda_b) \oplus \lambda_c) \wedge (k_c^1 || \Lambda_1 \oplus k_c^0 || \Lambda_0) \Big)$$

The only part of the computation that differs in the two computation paths is the selector bit

$$((\lambda_a \oplus \Lambda_a)(\lambda_b \oplus \Lambda_b) \oplus \lambda_c) = ((\lambda_a^R \oplus \lambda_a^S \oplus \Lambda_a)(\lambda_b^R \oplus \lambda_b^S \oplus \Lambda_b) \oplus \lambda_c^R \oplus \lambda_c^S).$$

The evaluation will proceed identically if

$$((\lambda_a^R \oplus \lambda_a^S \oplus \Lambda_a)(\lambda_b^R \oplus \lambda_b^S \oplus \Lambda_b) \oplus \lambda_c^R \oplus \lambda_c^S) \\ = ((\widetilde{\lambda}_a^R \oplus \lambda_a^S \oplus \Lambda_a)(\widetilde{\lambda}_b^R \oplus \lambda_b^S \oplus \Lambda_b) \oplus \widetilde{\lambda}_c^R \oplus \lambda_c^S).$$

We can compute $\widetilde{\lambda}_c^R$ so that this equation holds. This completes the induction step and the proof of Claim 4.2.

□

Next we prove the following claim.

Claim 4.3 *Condition on not aborting in the real execution, the receiver outputs $\mathcal{F}(\widetilde{x}, y)$ with overwhelming probability where $\widetilde{x}$ is the adversary's input extracted by $\mathcal{S}$.*

Proof: This proof follows similarly to the proof from [LPSY15]. Specifically, assuming that the sender is cheating (without getting caught) in the garbled construction by providing inconsistent PRG values to functionality $\mathcal{F}_{\text{AuthCol}}$. In this case, it must be that the active path viewed by the receiver is correctly formed. That is, for each output wire, the value $k_c || \Lambda_c$ obtained by the receiver during the evaluation implies that the receiver can verify the correctness of the PRG values that are computed using the key k_c. Denoting by Λ_c the color associated with the output wire w_c as observed by the receiver during the evaluation and by ρ_w the actual value in this wire, there may be two potential cases:

1. $\Lambda_c = \lambda_c \oplus \rho_c$.
2. $\Lambda_c = \lambda_c \oplus \bar{\rho}_c$. This event implies that the adversary successfully carried out an attack, flipping the bit to be transferred within w_c. Since that implies that the sender must correctly guess $\mathsf{MAC}_{\mathsf{SK}}(EF_{w_c,\Lambda_c}^{g,\Lambda_c}) \in \{0,1\}^s$ (where MAC is an information theoretic object), it follows that this event occurs with probability 2^{-s}.

□

Since the adversary $\mathcal{A}$ can only select inputs for a corrupted sender to be fed to $\widehat{\mathcal{F}}$ the view of $\mathcal{A}$ in $\mathbf{REAL}_{\widehat{\Pi}_1,\mathcal{A}(z),\text{Rec}}(x, y, \kappa, s)$ and $\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),\text{Rec}}(x, y, \kappa, s)$ are identically distributed. Using Claims 4.2 and 4.3, we have that the output of the receiver in the real world and in $\mathbf{REAL}_{\widehat{\Pi}_1,\mathcal{A}(z),\text{Rec}}(x, y, \kappa, s)$ and $\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),\text{Rec}}(x, y, \kappa, s)$ are statistically close. Therefore, we can conclude with the following claim regarding the correctness of our simulation for sender corruption.

Claim 4.4 *The following executions are computationally indistinguishable:*

- $\{\mathbf{REAL}^{\widehat{\mathcal{F}}}_{\widehat{\Pi}_1,\mathcal{A}(z),\mathrm{Rec}}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y\in\{0,1\}^n,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),\mathrm{Rec}}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y\in\{0,1\}^n,z\in\{0,1\}^*}$

Next, we move on to the case when an active sender corrupts the receiver.

Receiver Rec **is corrupted.**

1. Upon receiving the adversary's input $(1^\kappa, 1^s, y)$, the simulator $\mathcal{S}$ internally invokes the adversary $\mathcal{A}$ on y and uniformly generated randomness.
2. $\mathcal{S}$ internally emulates functionality $\mathcal{F}_{\mathrm{AuthPRG}}$ as follows:
   - $\mathcal{S}$ receives from the adversary its input to $\mathcal{F}_{\mathrm{AuthPRG}}$ that is consists of an input $\widetilde{y}$, a mask λ^R_w for every wire w and a random seed β to an s-wise PRG.
   - It then computes a simulated garbled circuit as follows. $\mathcal{S}$ first chooses for each wire $w \in W$ that is not an input wire, a color Λ_W. It then defines an active path which is induced by these color bits. Namely, for each gate $g \in G$ with input a and b and output wire c, the simulator computes the (Λ_a, Λ_b)th entry in the garbled gate by
     $$F^{g,\Lambda_a}_{a,\Lambda_a} \oplus F^{g,\Lambda_b}_{b,\Lambda_b} \oplus (k^{\Lambda_c}_c || \Lambda_c)$$
     whereas the remaining three rows are uniformly sampled at random from $\{0,1\}^{\kappa+1}$. Denote the outcome by the set of tuples $\{(\widetilde{R}^{00}_g, \widetilde{R}^{01}_g, \widetilde{R}^{10}_g, \widetilde{R}^{11}_g)\}_{g\in G}$.
3. Finally, the simulator submits $\widetilde{y}$ to its trusted party as the adversary's input and obtains the output z. For every output wire $w \in W$, the simulator fixes $\lambda_w = \Lambda_w \oplus z_i$ where z_i is the ith bit associated with wire w.
4. The simulator sends the adversary the values the garbled circuit, the input keys associated with the input wires, the masked MAC for every PRG value in the active path, namely, $\tau^{g,r}_{w,\Lambda_w,R}$, whereas the remaining masks are picked at random, and the set of $\{\lambda_w\}_w$ for all the output wires.

   In addition, the simulator hands the adversary the encrypted masks for the masks chosen above, and random strings for the remaining unchosen masks (namely, the simulator sends a random string instead of a ciphertext that should encrypt the mask that corresponds to a PRG value that is not associated with active path).
5. The simulator outputs whatever the adversary does.

Note that the adversary's sees a simulated garbled circuit which is created differently than the real garbling. We prove that the adversary's view is indistinguishable due to the pseudorandomness of the underlying PRG.

Claim 4.5 *The following executions are computationally indistinguishable:*

- $\{\mathbf{REAL}^{\widehat{\mathcal{F}}}_{\widehat{\Pi}_1,\mathcal{A}(z),\mathrm{Sen}}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y\in\{0,1\}^n,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),\mathrm{Sen}}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y\in\{0,1\}^n,z\in\{0,1\}^*}$

Proof: Our proof follows via a sequence of hybrids games denoted by H_τ for $0 \leq \tau \leq |G|-1$, where we sequentially replace real garbled gates with simulated ones. For that, we consider a topological ordering on the set G of gates in C. Concretely, C is viewed as a Directed Acyclic Graph (DAG), where the gates are the nodes in the graph and an output wire of gate g_1 which enters as input wire to gate g_2 indicates the edge (g_1, g_2) in the graph. We compute a topological orderings of the graph, that is, if the output wire of gate g_1 enters to gate g_2 then the index of gate g_1 in our ordering is lower than the index of gate g_2. Whenever we use the notation g_i we refer to the ith element in this topological ordering.

Hybrid H_τ. In this hybrid execution there exists a simulator $\mathcal{S}_\tau$ that follows the instructions of simulator $\mathcal{S}$ with the following changes. Specifically, $\mathcal{S}_\tau$ creates the first τ garbled gates in the topological ordering as in the original simulation, whereas the remaining $|G|-\tau$ gates are created as in the real protocol. Moreover, $\mathcal{S}_\tau$ computes the masked MAC for every PRG value and its encryption for the first τ gates as in the simulation, whereas the remaining masked MACs and their encryptions are computed as in the real execution.

Let $\mathbf{Hyb}^\tau_{\mathcal{A}}(1^\kappa, 1^s, z, R, x, y)$ denote the output distribution of $\mathcal{A}$ in hybrid H_τ. Note that the distribution in $\mathsf{H}_{|G|}$ is as in the simulation with $\mathcal{S}$, whereas H_0 is an execution for which the garbled circuit is generated honestly. Our proof follows by a reduction to the pseudorandomness of the PRG F. More formally, assume by contradiction the existence of an adversary $\mathcal{A}$ and a distinguisher $\mathcal{D}$ for which the above distributions are distinguishable by $\mathcal{D}$ with a non-negligible probability $1/q(\kappa)$ for infinitely many κ's. We construct a distinguisher $\mathcal{D}_{\mathrm{PRG}}$ that breaks the security of F with probability $1/q(\kappa) \cdot \tau$ as follows.

Fix κ and τ, then upon given the honest party's input x and κ, τ, and inputs $k_a^{\bar{\Lambda}_a}$ and $k_a^{\bar{\Lambda}_b}$ to the PRG or random strings. $\mathcal{D}_{\mathrm{PRG}}$ invokes the adversary as in the simulation with $\mathcal{S}$, internally emulating functionality $\mathcal{F}_{\mathrm{AuthPRG}}$. More concretely, $\mathcal{D}_{\mathrm{PRG}}$ is defined identically to $\mathcal{S}$ with the exception that it does not pick the three inactive rows in the garbled circuit at random for the τth gate g_τ, but rather uses the PRG/random values it obtained as an input.

More concretely, let $k_a^{\bar{\Lambda}_a}$ and $k_b^{\bar{\Lambda}_b}$ be the inactive keys associated with the input wires to gate g_τ. Then $\mathcal{D}_{\mathrm{PRG}}$ extracts from its input two strings $(\mathcal{O}^0_{k_a^{\bar{\Lambda}_a}}, \mathcal{O}^1_{k_a^{\bar{\Lambda}_a}})$ (that correspond to $F_{k_a^{\bar{\Lambda}_a}}(g_\tau, \Lambda_a)$ and $F_{k_a^{\bar{\Lambda}_a}}(g_\tau, \Lambda_a \oplus 1)$ and two strings $(\mathcal{O}^0_{k_b^{\bar{\Lambda}_b}}, \mathcal{O}^1_{k_b^{\bar{\Lambda}_b}})$ (that correspond to $F_{k_b^{\bar{\Lambda}_b}}(g_\tau, \Lambda_b)$ and $F_{k_b^{\bar{\Lambda}_b}}(g_\tau, \Lambda_b \oplus 1)$). $\mathcal{D}_{\mathrm{PRG}}$ then completes the description of the four entries in the garbling of g_τ as follows:

$$R_g^{\Lambda_a \bar{\Lambda}_b} = F_{k_a^0}(g_\tau, \Lambda_a \oplus 1) \oplus \mathcal{O}^{\bar{\Lambda}_b \oplus 1}_{k_b^{\bar{\Lambda}_b}}) \oplus (k_c^0 || 0) \oplus \Big((\lambda_{\Lambda_a} \oplus \lambda_{\Lambda_a}\lambda_{\Lambda_b} \oplus \lambda_{\Lambda_c})(k_c^1 || 1 \oplus k_c^0 || 0) \Big)$$

$$R_g^{\bar{\Lambda}_a \Lambda_b} = \mathcal{O}^{\bar{\Lambda}_a \oplus 1}_{k_a^{\bar{\Lambda}_a}} \oplus F_{k_b^{\Lambda_b}}(g_\tau, \Lambda_b \oplus 1) \oplus (k_c^0 || 0) \oplus \Big((\lambda_{\Lambda_b} \oplus \lambda_{\Lambda_a}\lambda_{\Lambda_b} \oplus \lambda_{\Lambda_c})(k_c^1 || 1 \oplus k_c^0 || 0) \Big)$$

$$R_g^{\bar{\Lambda}_a \bar{\Lambda}_b} = \mathcal{O}^{\bar{\Lambda}_a}_{k_a^{\bar{\Lambda}_a}} \oplus \mathcal{O}^{\bar{\Lambda}_b}_{k_b^{\bar{\Lambda}_b}} \oplus (k_c^0 || 0) \oplus \Big((1 \oplus \lambda_{\Lambda_a} \oplus \lambda_{\Lambda_b} \oplus \lambda_{\Lambda_a}\lambda_{\Lambda_b} \oplus \lambda_{\Lambda_c})(k_c^1 || 1 \oplus k_c^0 || 0) \Big)$$

The distinguisher further samples a random secret key SK for the MAC computations and honestly computes the masked MAC for every PRG value (for both active and inactive rows). Finally, $\mathcal{D}_{\mathrm{PRG}}$ obtains the values $R_{g,2}, R_{g,3}$ in order to encrypt the masking values $\{\tau^{g,r}_{w,b,S}\}$ for every PRG.

Note that the inactive keys associated with the input wires of g_τ are not in use in any prior gate $g_{\tau'}$ for $\tau' < \tau$ due to the topological order of the gates. Moreover, in case the distinguisher obtains values computed based on the PRG then the garbling description of g_τ is as in hybrid $\mathsf{H}_{\tau-1}$ whereas if it obtains truly random strings then this gate is garbled as in H_τ as the remaining three rows are random. This concludes the proof of Claim 4.5 and Lemma 4.1. □ ■

Next, we provide another variant of our first compiler where we further reduce the number of random bits input by the receiver to $\widehat{\mathcal{F}}$. This will be important in our compilation as the number of bits input by the

receiver to $\widehat{\mathcal{F}}$ will directly correspond to the number of calls made in the final protocol to the parallel OT functionality.

Lemma 4.6 (AuthPRG2 Compiler) *Assume the existence of explicit constant-degree unbalanced unique-neighbor expanders. There exists a compiler* AuthPRG2 *that takes as input κ (PRG seed), s (statistical parameter), ϵ (statistical PRG parameter) and a two-party deterministic functionality $\mathcal{F}(x, y)$, computable via a circuit C, between a sender with input x and a receiver with input y and outputs another two-party functionality $\widehat{\mathcal{F}}$ and protocol $\Pi_{\widehat{\mathcal{F}}}$ that securely realizes $\mathcal{F}$ in the $\widehat{\mathcal{F}}$-hybrid against static corruptions by active adversaries with the following features:*

- *$\widehat{\mathcal{F}}$ can be computed via an NC^0 circuit of size $O(|\mathrm{C}|\kappa)$. The receiver's inputs to $\widehat{\mathcal{F}}$ include its original input y to $\mathcal{F}$ and a string of length $O(|\mathrm{C}|^\epsilon + s)$ is chosen uniformly at random.*
- *$\Pi_{\widehat{\mathcal{F}}}$ makes a single invocation to the $\widehat{\mathcal{F}}$ oracle.*
- *$\Pi_{\widehat{\mathcal{F}}}$ makes $O(|\mathrm{C}|)$ black-box calls to a length-doubling PRG: $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$.*

Proof Sketch: We obtain this theorem by derandomizing the random bits provided by the receiver for $\{\lambda_w^R\}_{w \in W}$. Recall that in the AuthPRG compiler, the receiver's input length is $n + O(|\mathrm{C}| + \kappa)$ which includes n bits for y that specify the input for the original functionality $\mathcal{F}$, $O(|\mathrm{C}|)$ random bits for $\{\lambda_w^R\}_{w \in W}$ and $O(\kappa)$ random bits for a key of the information-theoretic MAC to the $\widehat{\mathcal{F}}$ functionality. Next, we show how we can reduce the random bits provided for $\{\lambda_w^R\}_{w \in W}$ by derandomizing it.

We begin by recalling that in our previous protocol, an active adversary corrupting the sender can cause the receiver to abort by giving bad PRG values. In Claim 4.2, we showed that the abort condition is independent of the receiver's input. Next, we argue that if the receiver chooses the random bits for $\{\lambda_w^R\}_{w \in W}$ as an output of an s-wise PRG, the abort condition is still independent of the receiver's input. Then, we can consider a variant of our previous construction where the receiver provides as input a seed for an s-wise PRG and then augmenting the $\widehat{\mathcal{F}}$ from our previous construction to first decode the original input and then compute $\widehat{\mathcal{F}}$. Next we recall that there exists an NC^0 implementation of s-wise PRG based on expander graphs from $O(\alpha^\epsilon + s)$ bits to α bits of size $O(\alpha)$ (c.f. Section 2.2). Instantiating this PRG with $\alpha = O(|\mathrm{C}|)$ in our construction we can conclude the proof of Lemma 4.6. It only remains to show that the abort condition is independent of the receiver's input and we prove this next.

Analysis when the $\{\lambda_w^R\}_{w \in W}$ are chosen according to a s-wise PRG. First we observe that the correctness of the protocol is unaltered by switching to an s-wise PRG as the masks λ_w^R only affect the permutation in each gate. However, the probability with which the receiver aborts will change as a result of this. To analyze this, we first compute the predicate over the λ_w^R variables that determines when the receiver aborts.

From the proceeding argument, we know that on any active path, the receiver aborts the first point the key and the PRG value are inconsistent. Such an abort event corresponds to the color Λ_w for that wire being 0 or 1, i.e. this event corresponds to a predicate that can be expressed by either Λ_w or its complement $\overline{\Lambda_w}$. As long as the PRG values are consistent on an active path, we know that the equation $\Lambda_w = \lambda_w^R \oplus \lambda_w^S \oplus \rho_w$ holds where ρ_w is the actual value on the wire. Moreover, λ_w^S and ρ_w are fixed if we fix the sender's inputs x and mask bits $\{\lambda_w^S\}_w$, and the receiver's input y. Therefore, the abort event can be rewritten as λ_w^R or $\overline{\lambda_w^R}$. As every abort event has such a form, the abort predicate is a disjunction of the literals $\{\lambda_w^R\}_w$ or their complements. We denote this predicate on the literals $\{\lambda_w^R\}_w$ by P. If we replace the literals from being chosen independent to being computed based on s-wise PRG, the following lemma follows immediately:

Lemma 4.7 *For every disjunctive predicate $P(\{\lambda_w^R\}_w)$,*

1. *If it involves fewer than s literals,*

$$\{\lambda_w^R \leftarrow \{0,1\} : P(\{\lambda_w^R\}_w)\} \equiv \{s \leftarrow \{0,1\}^\kappa; \lambda_w^R \leftarrow \mathrm{G}_{\mathrm{sPRG}}(s)_w : P(\{\lambda_w^R\}_w)\},$$

and

2. *If it involves more than s literals, then both*

$$\Pr[\lambda_w^R \leftarrow \{0,1\} : P(\{\lambda_w^R\}_w) = 1]$$

and

$$\Pr[s \leftarrow \{0,1\}^\kappa; \lambda_w^R \leftarrow \mathrm{G}_{\mathrm{sPRG}}(s)_w : P(\{\lambda_w^R\}_w)]$$

are at least $1 - 2^{-O(s)}$ which implies that the two distributions are $2^{-O(s)}$-close.

We know that when the masks $\{\lambda_w^R\}_w$ are independently chosen, then the probability of abort is independent of the receiver's input y. Now, since the abort probability is 2^{-s} close when the masks are replaced with the output of an s-wise PRG, we can conclude by a simple hybrid argument that the abort probability is statistically independent of receiver's input. ■

We can modify this variant to incorporate (by now standard) optimization of Free XOR [KS08]. Implicit in this optimization is a mechanism that restricts the space of keys sampled for the wires. On a high-level, we will redefine our NC^0 functionality that will enforce this constraint. Then, if the modification of the NC^0 functionality is compatible with the proof of Claim 4.2 which shows that the abort condition is independent of the receiver's input, the rest of the proof will essentially follow. To incorporate the Free XOR technique, we modify the NC^0 functionality where the sender only provides one key per wire and a global constant Δ that will define the other key. By construction, it follows that an active sender cannot violate the Free XOR invariant between the pair of keys for each wire. Moreover, the proof of Claim 4.2 is consistent with the Free XOR constraint and continues to hold. The rest of the proofs follow essentially as before.

4.2 Variant 2: Authenticating The Color Bits

In the second concretely efficient variant the color bits are encrypted within each garbled row in an authenticated manner using an information-theoretic MAC, where the MAC secret-key is chosen by the receiver. In contrast to the protocol described in Section 4.1, the receiver will not need to provide extra randomness and the prover will locally construct the garbled circuit. This implies that the abort predicate $P(\cdot)$ (that is implicit in the prior simulation) cannot be viewed as a disjunctive function any longer but as an arbitrary function. Furthermore, the outcome of this function will not be independent of the receiver's input as in Section 4.1, and may leak a bit of information. On the other hand, this variant enjoys the property that the corresponding COT protocol requires an NP relation that can only be based on arithmetic computations over a large field. Therefore, we do not have to rely on the protocol of [DI06] over small fields that require algebraic geometric codes. Instead, we can work over a large field and use Reed Solomon codes. Consequently, we achieve a much faster and concretely efficient protocol at the price of weakening its security. Finally, we note that by making a relatively mild assumption on the underlying PRG that holds in the random oracle model, we can simplify the abort function; see more details in Section 4.2.1. We next prove the following lemma.

Lemma 4.8 (AuthCol Compiler) *There exists a compiler* $\mathsf{AuthCol}$ *that, given κ (PRG seed length), s (statistical parameter) and a two-party deterministic functionality $\mathcal{F}(x, y)$ expressed by a circuit* C*, outputs another two-party functionality $\widehat{\mathcal{F}}$ and protocol Π_1 that securely realizes $\mathcal{F}$ in the $\widehat{\mathcal{F}}$-hybrid with correlated abort with the receiver's input and the following features:*

- *$\widehat{\mathcal{F}}$ is represented by an NC^0 circuit of size $O(|\mathrm{C}| \cdot (\kappa + s))$. The receiver's inputs to $\widehat{\mathcal{F}}$ include its original input y to $\mathcal{F}$ and a string of length $2s$ that it will chosen uniformly at random.*
- *Π_1 makes a single call to the $\widehat{\mathcal{F}}$ oracle.*
- *Π_1 makes $O(|\mathrm{C}|)$ black-box calls to a length-doubling PRG: $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$.*

The $\mathbf{NC^0}$ Functionality $\widehat{\mathcal{F}} = \boldsymbol{\mathcal{F}}_{\mathbf{AuthCol}}$. In this variant the $\mathbf{NC^0}$ functionality $\mathcal{F}_{\mathrm{AuthCol}}$ computes a BMR-style garbling for some function $\mathcal{F}$ that is expressed by a set of wires W and a set of garbled gates G, where only the sender provides the keys and PRG values to be used for this generation. The main difference over the $\mathbf{NC^0}$ functionality from Section 4.1 is that in this case the functionality authenticates the color bits instead of the PRG values submitted by the sender, where authentication is computed based on the receiver's secret-key for an information theoretic MAC (see Section 2.3). Moreover, the receiver does not provide any masking bits. More concretely, the functionality obtains the parties' inputs (x, y) to the function $\mathcal{F}$, as well as the PRG evaluations from the sender, and the authenticated information from the receiver, and creates the garbling for all gates $g \in G$. It further notifies the sender whether the receiver has aborted upon evaluating the garbled circuit; the complete details can be found in Figure 3.

Protocol 2 (Protocol Π_1) *The parties' common input is a Boolean circuit C, expressed by a set of wires W and a set of gates G. Let s be the statistical security parameter and κ be the computational security parameter. Let $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ be a PRG and let $\{\mathsf{MAC}_{\mathsf{SK}} : \{0,1\} \to \{0,1\}^s\}_{\mathsf{SK} \in \{0,1\}^{2s}}$ be an information theoretically secure MAC computable in NC^0.*

- **Input:** *The sender is given input x and the receiver is given input y. Both parties are given the security parameters $1^\kappa, 1^s$ and the description of a Boolean circuit C.*
- **The sender's input to $\mathcal{F}_{\mathrm{AuthCol}}$:**
  - *Input x.*
  - *For every wire $w \in W$, keys k_w^0, k_w^1 sampled uniformly at random from $\{0,1\}^\kappa$, and mask bit $\lambda_w^S \leftarrow \{0,1\}$ sampled uniformly at random.*
  - *For every gate $g \in G$, with input wires a and b, the following PRG values, $F_{k_a^0}(g,0), F_{k_a^0}(g,1), F_{k_a^1}(g,0), F_{k_a^1}(g,1), F_{k_b^0}(g,0), F_{k_b^0}(g,1), F_{k_b^1}(g,0), F_{k_b^1}(g,1)$.*
- **The receiver's input to $\mathcal{F}_{\mathrm{AuthCol}}$:**
  - *Input y.*
  - *for every $w \in W$, a random mask bit $\lambda_w^R \leftarrow \{0,1\}$.*
  - *Two strings $\sigma_0, \sigma_1 \leftarrow \{0,1\}^s$ chosen uniformly at random.*
- **The receiver's outcome from $\mathcal{F}_{\mathrm{AuthCol}}$:**
  - *$\{(R_g^{00}, R_g^{01}, R_g^{10}, R_g^{11})\}_{g \in G}$.*
  - *$k_w || z_w$ for every input wire w.*
  - *A mask λ_w for every output wire.*
- **Concluding the output.** *The receiver then proceeds to evaluate the garbled circuit as follows: Let the gates be arranged in some topological order. We will maintain the invariant that if the receiver has not aborted when it processes some gate g with input wires a and b, then it possess keys k_a and k_b and color bits Λ_a and Λ_b.*

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

The functionality runs with parties S, R and an adversary $\mathcal{S}$. The parties' joint input is a Boolean circuit C, expressed by a set of wires W and a set of garbled gates G.

The sender's inputs to the functionality are:

- Input x.
- For every wire $w \in W$, keys $k_w^0, k_w^1 \leftarrow \{0,1\}^\kappa$ and a mask λ_w.
- For every gate $g \in G$ with input wires a and b, the PRG values $F_{a,0}^{g,0}, F_{a,0}^{g,1}, F_{a,1}^{g,0}, F_{a,1}^{g,1}, F_{b,0}^{g,0}, F_{b,0}^{g,1}, F_{b,1}^{g,0}, F_{b,1}^{g,1}$.

The receiver's inputs to the functionality are:

- Input y.
- Two strings $\sigma_0, \sigma_1 \leftarrow \{0,1\}^s$.

The functionality performs the following computations:

1. For every gate $g \in G$, compute the garbled table as follows:

$$R_g^{00} = F_{a,0}^{g,0} \oplus F_{b,0}^{g,0} \oplus (k_c^0||\sigma_0) \oplus \Big((\lambda_a\lambda_b \oplus \lambda_c)(k_c^1||\sigma_1 \oplus k_c^0||\sigma_0)\Big)$$
$$R_g^{01} = F_{a,0}^{g,1} \oplus F_{b,1}^{g,0} \oplus (k_c^0||\sigma_0) \oplus \Big((\lambda_a \oplus \lambda_a\lambda_b \oplus \lambda_c)(k_c^1||\sigma_1 \oplus k_c^0||\sigma_0)\Big)$$
$$R_g^{10} = F_{a,1}^{g,0} \oplus F_{b,0}^{g,1} \oplus (k_c^0||\sigma_0) \oplus \Big((\lambda_b \oplus \lambda_a\lambda_b \oplus \lambda_c)(k_c^1||\sigma_1 \oplus k_c^0||\sigma_0)\Big)$$
$$R_g^{11} = F_{a,1}^{g,1} \oplus F_{b,1}^{g,1} \oplus (k_c^0||\sigma_0) \oplus \Big((1 \oplus \lambda_a \oplus \lambda_b \oplus \lambda_a\lambda_b \oplus \lambda_c)(k_c^1||\sigma_1 \oplus k_c^0||\sigma_0)\Big)$$

2. Send the receiver R the following values:
   - $\{(R_g^{00}, R_g^{01}, R_g^{10}, R_g^{11})\}_{g \in G}$.
   - $(k_w^0||\sigma_0) \oplus \Big((\lambda_w \oplus x_i) \wedge (k_w^1||\sigma_1 \oplus k_w^0||\sigma_0)\Big)$ for every pair (w,i) where input wire w carries the i^{th} bit of x.
   - $(k_w^0||\sigma_0) \oplus \Big((\lambda_w \oplus y_i) \wedge (k_w^1||\sigma_1 \oplus k_w^0||\sigma_0)\Big)$ for every pair (w,i) where input wire w carries the i^{th} bit of y.
   - λ_w for every output wire.

Figure 3: The offline functionality $\mathcal{F}_{\mathrm{AuthCol}}$.

Base case: *For each input wire $w \in W$, the receiver holds an input key k_w and a color Λ_w that is set to 0 if $z_w = \sigma_0$, and set to 1 if $z_w = \sigma_1$. In case the receiver does not have these values in the correct format, it aborts.*

Induction step: *Consider an arbitrary gate $g \in G$ in the topological sequence with input wires a and b and output wire c. By our induction hypothesis, if the receiver has not yet aborted then it has keys k_a, k_b and color bits Λ_a and Λ_b. Then the receiver computes*

$$k_c||z_c = R_g^{\Lambda_a\Lambda_b} \oplus F_{k_a}(g,\Lambda_a) \oplus F_{k_b}(g,\Lambda_b).$$

If $z_c \notin \{\sigma_0, \sigma_1\}$, the receiver aborts. Otherwise it sets the color Λ_c such that $z_c = \sigma_{\Lambda_c}$.

Finally, if the receiver has not aborted, it possesses the colors Λ_w for every output wire $w \in W$. It

then outputs $\Lambda_w \oplus \lambda_w$ as the output on wire w for every output wire and forwards the message Abort *to $\mathcal{F}_{\mathrm{AuthCol}}$. Else, in case the receiver has aborted, it forwards the message* Non − Abort.

Claim 4.9 *Let $\mathcal{F}$ a two-party functionality as above and assume that F is a PRG. Then Protocol 2 securely computes functionality $\mathcal{F}$ with correlated abort with the receiver's input in the $\mathcal{F}_{\mathrm{AuthCol}}$-hybrid.*

Proof: Our proof follows by showing that for every adversary $\mathcal{A}$ in the $\mathcal{F}_{\mathrm{AuthCol}}$-hybrid world there exists an simulator $\mathcal{S}$ in the ideal world with access to functionality $\mathcal{F}$. We begin with the description of the simulator while distinguishing two corruption cases.

The sender S is corrupted.

1. Simulator $\mathcal{S}$ internally invokes the adversary $\mathcal{A}$ on $(1^\kappa, 1^s, x)$ with uniformly sampled random tape, where x is the input given to the Sender.
2. $\mathcal{S}$ internally emulates functionality $\mathcal{F}_{\mathrm{AuthCol}}$ as follows:
   - $\mathcal{S}$ receives from the adversary its input to $\mathcal{F}_{\mathrm{AuthCol}}$ that is consists of an input $\widetilde{x}$, a set of keys $\{\widetilde{k}_0, \widetilde{k}_1\}_{w\in W}$ masking bits $\{\lambda_w^S\}_{w\in W}$ and a set of "alleged" PRG values: $F_{a,0}^{g,0}, F_{a,0}^{g,1}, F_{a,1}^{g,0}, F_{a,1}^{g,1}, F_{b,0}^{g,0}, F_{b,0}^{g,1}, F_{b,1}^{g,0}, F_{b,1}^{g,1}$.
   - $\mathcal{S}$ sends functionality $\mathcal{F}$ a leakage predicate $P : \{0,1\}^* \mapsto \{0,1\}$. More formally, P is the predicate that runs the honest receiver's algorithm using the receiver's input and keys and PRF values submitted to the functionality and outputs 1 if the receiver aborts. Recall that the trusted party implementing the functionality delivers the answer to the receiver only if P returns 0.
3. $\mathcal{S}$ outputs whatever the adversary does.

Note first that the sender's view is identical in both executions as it does not receive any messages. Moreover, the receiver aborts with the same probability in both executions since the leakage functions identically mimics the computation of $\mathcal{F}_{\mathrm{AuthCol}}$.

The receiver R is corrupted.

1. Upon receiving the adversary's input $(1^\kappa, 1^s, y)$, the simulator $\mathcal{S}$ internally invokes the adversary $\mathcal{A}$ on y and uniformly generated randomness.
2. $\mathcal{S}$ internally emulates functionality $\mathcal{F}_{\mathrm{AuthCol}}$ as follows:
   - $\mathcal{S}$ receives from the adversary its input to $\mathcal{F}_{\mathrm{AuthCol}}$ that is consists of an input $\widetilde{y}$ and two strings σ_0, σ_1.
   - It then computes a simulated garbled circuit as follows. $\mathcal{S}$ first chooses for each wire $w \in W$ that is not an input wire, a color Λ_W. It then defines an active path which is induced by these color bits. Namely, for each gate $g \in G$ with input a and b and output wire c, the simulator computes the (Λ_a, Λ_b)th entry in the garbled gate by
   $$F_{a,\Lambda_a}^{g,\Lambda_a} \oplus F_{b,\Lambda_b}^{g,\Lambda_b} \oplus (k_c^{\Lambda_c} || \sigma_{\Lambda_c})$$
   whereas the remaining three rows are uniformly sampled at random from $\{0,1\}^{\kappa+s}$. Denote the outcome by the set of tuples $\{(\widetilde{R}_g^{00}, \widetilde{R}_g^{01}, \widetilde{R}_g^{10}, \widetilde{R}_g^{11})\}_{g\in G}$.

3. Finally, the simulator submits $\widetilde{y}$ to its trusted party as the adversary's input and obtains the output z. For every output wire $w \in W$, the simulator fixes $\lambda_w = \Lambda_w \oplus z_i$ where z_i is the ith bit associated with wire w.

4. The simulator sends the adversary the values the garbled circuit, the input keys associated with the input wires and the set of $\{\lambda_w\}_w$ for all the output wires.

5. The simulator outputs whatever the adversary does.

Note that the adversary's sees a simulated garbled circuit which is created differently than the real garbling. We prove that the adversary's view is indistinguishable due to the pseudorandomness of the underlying PRG.

Claim 4.10 *The following executions are computationally indistinguishable:*

- $\{\mathbf{REAL}^{\widehat{\mathcal{F}}}_{\widehat{\Pi}_2,\mathcal{A}(z),\mathrm{Rec}}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y\in\{0,1\}^n,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),\mathrm{Rec}}(x,y,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x,y\in\{0,1\}^n,z\in\{0,1\}^*}$

We can carry out the same proof as in Claim 4.5 by providing a reduction to the pseudorandomness of the underlying PRG and omit the proof. ■

We can modify our NC_0 functionality to support the optimizations of Free XOR [KS08] (similarly to the modification made in the prior section) and half gates [ZRE15]. The latter optimization becomes possible since the receiver does not participate in the creation of the garbled circuit.

4.2.1 Security in the Presence of WVD Attacks

Recall that the key feature of functionality $\mathcal{F}_{\mathrm{AuthPRG}}$ is that whenever the receiver decrypts a row during the evaluation process, there must exist at most one possible value for the PRG that will result in the correct decryption. We will now argue that the same property holds for $\mathsf{AuthCol}$ if we assume that the PRG behaves as a random oracle. This is because that if a malicious sender needs to be able to produce two different PRG values that will result in the receiver not aborting it means that the last s bits in the output of the PRG must be the same. This is because the last s bits must decrypt to either σ_0 or σ_1 and the adversarial sender cannot guess either of these strings beyond a negligible probability. This implies that the abort predicate is a disjunction of the wire values in the computed circuit (denoted by wire-value disjunction or WVD).

More concretely, we consider a modified construction of protocol Π_1 where the PRG function is realized using the random oracle. Then, in case the sender is corrupted, the simulator defines a disjunctive predicate $P(\cdot)$ as follows. For every wire $w \in W$ that is associated with keys k^0_w, k^1_w, $\mathcal{S}$ check whether these keys are consistent with the respective PRG values provided by the sender. Namely, $\mathcal{S}$ verifies whether k^0_w (resp. k^1_w) is consistent with $F^{g,0}_{w,0}$ and $F^{g,1}_{w,0}$ (resp. $F^{g,0}_{w,1}$ and $F^{g,1}_{w,1}$) for every gate g for which w is an input wire. In case of inconsistency, $\mathcal{S}$ includes in $P(\cdot)$ the literal $(w,0)$ (resp. $(w,1)$). The rest of the simulation is as in the proof of Claim 4.9. Note that the simulation may fail only in case the adversary finds a "collision" relative to two PRG values and the last s bits in their outputs, which occurs with negligible probability in s.

We further note that the only property that we need from the PRG in the plain model is a collision-resistance property where no adversary can find two seeds for which the PRG outcomes agree on a subset of the output bits. In fact, it is sufficient to construct PRGs that are injective on the last s bits to achieve this property. This is relatively a mild assumption. We conclude with the following corollary.

Corollary 4.1 *Assuming an injective PRG on the last s bits, there exists a compiler* AuthCol *that, given κ (PRG seed length), s (statistical parameter) and a two-party deterministic functionality $\mathcal{F}(x, y)$ expressed by a circuit* C*, outputs another two-party functionality $\widehat{\mathcal{F}}$ and protocol Π_1 that securely realizes $\mathcal{F}$ in the $\widehat{\mathcal{F}}$-hybrid in the presence of WVD attacks on the receiver's input and the same features as in Lemma 4.8.*

5 Realizing NC^0 Functionalities in the $\mathcal{F}_{\mathrm{DCOT}}$-Hybrid

We will name the parties as sender Sen and receiver Rec matching their roles in the final protocol. We will design a protocol in three steps.

Step 1. Realizing NC^0 functionality $\mathcal{F}$ with NISC/$\mathcal{F}_1$ protocol for locality one functionality $\mathcal{F}_1$. We begin by considering a non-interactive protocol Π_1 for $\mathcal{F}$ in the parallel OT-hybrid that is secure in the presence of a passive sender and an active receiver. Such a protocol can be instantiated via a perfectly secure projective garbling scheme [Yao86, BHR12] or a private simultaneous messages protocol [FKN94] with 1-bit inputs. In slight more detail, the protocol from [BCR86] provides a NISC protocol in the 1-out-of-2^d OT-hybrid to compute any function with locality d. This incurs a communication cost of $2^{d+1} - 2$ bits. Using a standard transformation, this can further be reduced to d instances 1-out-of-2 OT, where one OT invocation is made for each of the d input bits from the receiver. In fact, this transformation can be generalized to any n bits output NC^0 functionality, where the protocol involves n parallel invocations of 1-out-of-2 string OTs, one corresponding to each input bit of the receiver, and achieves security against a passive sender and an active receiver.

From protocol Π_1, we construct a locality one (in the receiver's bits) NC^0 functionality $\mathcal{F}_1$ as follows. Functionality $\mathcal{F}_1$ takes the selection bits $b_1, \ldots, b_n$ from the receiver and the choice strings (s_0^i, s_1^i) for $i \in [n]$ from the sender for each of the n parallel OT invocations and delivers $s_0^i \oplus (b_i \wedge (s_0^i \oplus s_1^i))$ to the receiver. We next define a new NISC protocol Π_1' in the $\mathcal{F}_1$-hybrid that simply requires the sender and receiver to execute protocol Π_1 with the only exception that instead of feeding the inputs to the parallel OT protocol, they feed the same inputs to the $\mathcal{F}_1$ functionality.

Formally, this transformation provides the following features:

- $\mathcal{F}_1$ has locality one in the input bits of the receiver.
- $|\mathcal{F}_1| = O(|\mathcal{F}|)$ and the receiver's input size to $\mathcal{F}_1$ is equal to the receiver's input size to $\mathcal{F}$.
- Π_1' is a NISC protocol in the $\mathcal{F}_1$-hybrid that realizes $\mathcal{F}$ with security against a passive sender and an active receiver.

Step 2. Realizing NC^0 functionality $\mathcal{F}_1$ with s-wise independent random $\mathcal{F}_2$. In our next step, we wish to compile $\mathcal{F}_1$ to an NC_0 functionality $\mathcal{F}_2$ such that the receiver's bits are s-wise independent, i.e. the distribution of the receiver's input satisfies the property that for every s subset of the input bits, the receiver's inputs are distributed uniformly at random.

On a high level, $\mathcal{F}_2$ will take as input an encoding of the input bits provided by the receiver to $\mathcal{F}_1$. If we rely on packed secret sharing with AG codes over a binary field (see Section 2.4), $\mathcal{F}_2$ will be a constant times larger than $\mathcal{F}_1$ and the input bits of the receiver will be s-wise independent. Furthermore, $\mathcal{F}_2$ will be an NC_0 Boolean function over the bits shares of the receiver. Nevertheless, as the receiver needs to encode its original input for the functionality, we will only achieve security against passive receivers. In order to upgrade the receiver's security to active, we will need the receiver to demonstrate that its input shares correspond to a valid encoding. This will be enforced in the transformation presented in the next step.

More formally, in this step, we design a protocol Π_2 and functionality $\mathcal{F}_2$ such that Π_2 realizes $\mathcal{F}_1$ in the $\mathcal{F}_2$-hybrid. Namely, the receiver first follows protocol Π_1' and generates its input $b_1, \ldots, b_n$ for $\mathcal{F}_1$. Next, it encodes its input via packed secret sharing with privacy threshold s. Denote the concatenation of the receiver's shares by $\sigma_1, \ldots, \sigma_m$. If we rely on algebraic geometric codes then we can have $m = O(n + s)$. Recall that the functionality $\mathcal{F}_1$ has locality one, and the sender's inputs to $\mathcal{F}_1$ can be represented as pairs (s_0^i, s_1^i) for $i \in [n]$ where the receiver learns $s_{b_i}^i$ for every i. Next, the sender encodes its inputs by packing each of the vectors $(s_0^1, \ldots, s_0^n)$ and $(s_1^1, \ldots, s_1^n)$ using the same packing structure as the receiver, i.e., if the receiver encodes the secret b_i in block j and position k, then the sender encodes in position k the two secrets, s_0^i within the first block and s_1^i within the second block. Then functionality $\mathcal{F}_2$ takes the sender's shares and emulates $\mathcal{F}_1$ over the shares to get secret sharing of the encoded output of $\mathcal{F}_1$. (This can be viewed as emulating an MPC protocol where each party has one share corresponding to the receiver's input and the corresponding two shares from the sender.) In addition, corresponding to every input bit of the receiver, the sender chooses a pair of random strings (r_0^i, r_1^i) of length s and the functionality delivers $r_{\sigma_i}^i$ for every i. This additional randomness will be required in the next step in order to upgrade the receiver's security.

To summarize, protocol Π_2 proceeds as follows. The sender and receiver encode their inputs and send these encoding to $\mathcal{F}_2$. The receiver interprets the outputs from $\mathcal{F}_2$ as a set of shares for which it decodes and concludes the outcome by following the receiver's actions from the previous protocol interpreting the output as the output received from $\mathcal{F}_1$.

Formally, this transformation provides the following features:

- $\mathcal{F}_2$ is an NC^0 functionality.
- $|\mathcal{F}_2| = O(|\mathcal{F}_1|)$.
- Π_2 is a NISC protocol in the $\mathcal{F}_2$-hybrid that realizes $\mathcal{F}$ with security against a passive sender and a passive receiver.
- The distribution of the inputs fed by an honest receiver to Π_2 is s-wise independent.

Step 3. Realizing $\mathcal{F}_2$ in the $\mathcal{F}_{\text{DCOT}}$-IVD hybrid. We first recall that $\mathcal{F}_2$ is an NC^0 functionality. Then just as in Step 1, we design a non-interactive protocol Π_3 for $\mathcal{F}_2$ in the parallel OT-hybrid. However, in this step, we rely on oracle $\mathcal{F}_{\text{DCOT}}$ (cf. Figure 4) instead of the parallel OT oracle, in order to enforce honest behavior of both the sender and the receiver. This is achieved by instantiating the NP relations in the $\mathcal{F}_{\text{DCOT}}$ functionality with "correctness" predicates that specifies honest sender and receiver behaviors. Additionally, an adversary controlling the sender can send an input value disjunctive (IVD) predicate P to be applied on the input bits of the receiver. $\mathcal{F}_{\text{DCOT}}$-IVD delivers the output to the receiver only if the predicate is false.

This protocol achieves resilience to IVD attacks against a malicious sender because the receiver's input to $\mathcal{F}_{\text{DCOT}}$ meets s-wise independent. In more detail, if the IVD predicate P contains more than s clauses then it will result in the receiver aborting with probability at least $1 - 2^{-s}$, independent of the original inputs of the receiver. Whereas, if fewer than s clauses are present then at most s input bits of the receiver, which are distributed uniformly at random, are leaked and in particular, statistically independent of original input of the receiver. Formally, this transformation provides the following features:

- The total size of the sender's input to the $\mathcal{F}_{\text{DCOT}}$ functionality is $O(|\mathcal{F}_2|)$.
- The number of OT calls made to the $\mathcal{F}_{\text{DCOT}}$ functionality is the same as the receiver's input size to $\mathcal{F}_2$.

- Π_3 is a NISC protocol in the $\mathcal{F}_{\text{DCOT}}$-IVD hybrid and realizes $\mathcal{F}$ with security against an active sender and an active receiver, where additionally the sender can specify an IVD predicate on the receiver's input bits. Recall that $\mathcal{F}_{\text{DCOT}}$-IVD delivers the output of the computation to the receiver only if the predicate is false.

6 Realizing $\mathcal{F}_{\text{DCOT}}$-IVD

In this section, we design our protocol that securely realizes the $\mathcal{F}_{\text{DCOT}}$-IVD functionality (cf. Figure 4) with security in the presence of active adversaries up to IVD-abort. Recall that the $\mathcal{F}_{\text{DCOT}}$ has two NP-relations, one for the sender and the other for the receiver. The main contribution here is that our protocol is in the OT hybrid and employs the underlying PRG in black-box way where the total communication complexity is linear in the size of the sender's NP-relation and subquadratic in the receiver's NP-relation.

Functionality $\mathcal{F}_{\text{DCOT}}$-IVD

Functionality $\mathcal{F}_{\text{DCOT}}$-IVD communicates with sender S_{DCOT} and receiver R_{DCOT}, and adversary $\mathcal{S}$ and is parameterized by two NP relations $\mathcal{R}_S(\cdot,\cdot)$, $\mathcal{R}_R(\cdot,\cdot)$ and integers m, n, κ.

1. Upon receiving input $(P, \{(s_1^0, s_1^1)\}_{j\in[m]}, w_S)$ from S_{DCOT} where $s_j^b \in \{0,1\}^\kappa$ and $w_S \in \{0,1\}^{\mathsf{poly}(n,\kappa)}$ it checks if the predicate P is a disjunction of literals and records $(P, \bar{\mathbf{s}}, w_S)$ if it is a disjunction where $\bar{\mathbf{s}} = \{(s_1^0, s_1^1)\}_{j\in[m]}$.
2. Upon receiving $((u_1, \ldots, u_m), w_R)$ from R_{DCOT} where $u_j \in \{0,1\}$ and $w_R \in \{0,1\}^{\mathsf{poly}(n,\kappa)}$, record $(\bar{\mathbf{u}}, w_R)$ where $\bar{\mathbf{u}} = (u_1, \ldots, u_m)$). If there is no record from the sender the functionality waits until it receives a message from S_{DCOT}. If there is a record $(\bar{\mathbf{s}}, w_S)$ then it sends $(\{s_j^{u_j}\}_{j\in[m]}, \mathcal{R}_S(\bar{\mathbf{s}}, w_S))$ to R_{DCOT} only if $P(\bar{\mathbf{u}}) \neq 1$ and $\mathcal{R}_R(\bar{\mathbf{u}}, w_R) = 1$, and $\perp$ otherwise.

Figure 4: The doubly certified oblivious transfer functionality with IVD.

On a high-level, we will use the MPC-in-the-head approach of [IKOS07] to "certify" the sender inputs to the oblivious transfer executions. To certify the inputs of the receiver, we will use a one-sided variant of $\mathcal{F}_{\text{DCOT}}$, that we described below as $\mathcal{F}_{\text{RCOT}}$ which only certifies the receiver's inputs. Looking ahead, in our instantiations, the receiver certification will require an NP relation that is proportional only to the receiver's input size and sublinear in the circuit size. This will allow us to realize $\mathcal{F}_{\text{RCOT}}$ with a mildly inefficient protocol,[12] using the work of [IKO+11]. In more detail, we will describe our protocol Π_{DCOT} in the $(\mathcal{F}_{\text{RCOT}}, \mathcal{F}_{\text{CnP}})$-hybrid where $\mathcal{F}_{\text{RCOT}}$ is the parallel OT functionality which allows for receiver certification (cf. Figure 5). Whereas $\mathcal{F}_{\text{CnP}}$ is a slight variation of the standard commit-and-prove functionality that allows a sender to first commit to a witness w and then, given a function H from the receiver and an image y from the sender, delivers the output of the predicate $H(w) = y$; see Figure 6 for the formal description. We specify more details about these realizations in Section 9.1.

Beside employing functionalities $\mathcal{F}_{\text{RCOT}}$ and $\mathcal{F}_{\text{CnP}}$, our protocol uses a length doubling PRG and a special-hiding information theoretic MAC that preserves the properties of privacy and robustness in a way that enforces the sender to properly commit to its inputs; see Definition 2.4 for more details. More formally,

Protocol 3 (Protocol Π_{DCOT} for realizing functionality $\mathcal{F}_{\text{DCOT}}$-IVD)

[12] Namely, with non-constant overhead that is polylogarithmic in the functionality size.

Functionality $\mathcal{F}_{\text{RCOT}}$

Functionality $\mathcal{F}_{\text{RCOT}}$-IVD communicates with sender S_{RCOT} and receiver R_{RCOT}, and adversary $\mathcal{S}$ and is parameterized by an NP relation $\mathcal{R}_R(\cdot,\cdot)$ and integers m, n, κ.

1. Upon receiving input $(\{(s_1^0, s_1^1)\}_{j\in[m]})$ from S_{RCOT} where $s_j^b \in \{0,1\}^\kappa$ it records $(\bar{\mathbf{s}}_S)$ where $\bar{\mathbf{s}} = \{(s_1^0, s_1^1)\}_{j\in[m]}$.
2. Upon receiving $((u_1, \ldots, u_m), w_R)$ from R_{RCOT} where $u_j \in \{0,1\}$ and $w_R \in \{0,1\}^{\mathsf{poly}(n,\kappa)}$, record $(\bar{\mathbf{u}}, w_R)$ where $\bar{\mathbf{u}} = (u_1, \ldots, u_m)$). If there is no record from the sender the functionality waits until it receives a message from S_{RCOT}. If there is a record $\bar{\mathbf{s}}$ then it sends $(\{s_j^{u_j}\}_{j\in[m]})$ to R_{RCOT} only if $\mathcal{R}_R(\bar{\mathbf{u}}, w_R) = 1$, and $\perp$ otherwise.

Figure 5: The receiver certified oblivious transfer functionality.

Functionality $\mathcal{F}_{\text{CnP}}$

Functionality $\mathcal{F}_{\text{CnP}}$ communicates with sender S_{CnP} and receiver R_{CnP}, and adversary $\mathcal{S}$ and is parameterized by an NP relation $\mathcal{R}(\cdot,\cdot)$ and integers n, κ.

Commit phase. Upon receiving input (z, w) from S_{CnP} where $z = \{(s_j^b, r_j^b)\}_{j\in[m],b\in\{0,1\}}$ and $w \in \{0,1\}^{\mathsf{poly}(n,\kappa)}$, record this message.

Prove phase. Upon receiving H from R_{CnP}, forward H to S_{CnP}. Upon receiving y from S, check if there exists a record (z, w) that it received from S_{CnP}. Ignore if no such record exists. Otherwise, send 1 to R_{CnP} only if $H(z) = y$ and $\mathcal{R}(\bar{\mathbf{s}}, w) = 1$ where $\bar{\mathbf{s}} = \{(s_1^0, s_1^1)\}_{j\in[m]}$. Return 0 otherwise.

Figure 6: The commit-and-prove functionality.

- **Inputs:** *The sender S_{DCOT}'s input is $\{(s_j^0, s_j^1)\}_{j\in[m]}$ and a witness w_S with respect to some NP relation $\mathcal{R}_S$, and the receiver R_{DCOT}'s input is $u_1, \ldots, u_m$ and a witness w_R with respect to some NP relation $\mathcal{R}_R$.*
- **The protocol:**
  1. $S_{\text{DCOT}} \overset{\mathcal{F}_{\text{RCOT}}}{\longleftrightarrow} R_{\text{DCOT}}$: *The parties engage in m oblivious transfers in parallel using $\mathcal{F}_{\text{RCOT}}$ where S_{DCOT} uses $((k_j^0, r_j^0), (k_j^1, r_j^1))$ and R_{DCOT} uses u_j, as their respective inputs in the j^{th} $(j \in [m])$ oblivious transfer execution, where k_j^b is seed for the PRG and r_j^b is a sufficiently long string. Additionally R_{DCOT} provides w_R as the witness to $\mathcal{F}_{\text{RCOT}}$.*
  2. $S_{\text{DCOT}} \rightarrow R_{\text{DCOT}}$: *The sender sends $s_j^b \oplus G(k_j^b)$ where G is a PRG that expands k_j^b to the appropriate length.*
  3. $S_{\text{DCOT}} \overset{\mathcal{F}_{\text{CnP}}}{\longleftrightarrow} R_{\text{DCOT}}$: *The sender commits to the witness $(\{(s_j^b, r_j^b)\}_{j\in[m],b\in\{0,1\}}, w_S)$ by sending it to the $\mathcal{F}_{\text{CnP}}$ functionality.*
  4. $S_{\text{DCOT}} \leftarrow R_{\text{DCOT}}$*: The receiver chooses a random MAC key $H \leftarrow \mathcal{H}$ and sends it to the sender via functionality $\mathcal{F}_{\text{CnP}}$.*
  5. $S_{\text{DCOT}} \rightarrow R_{\text{DCOT}}$*: The sender sends the MAC of every string, namely $\{H(s_j^b; r_j^b)\}_{j\in[m],b\in\{0,1\}}$ to R_{DCOT}. If the MACed value transmitted for $(s_j^{u_j}, r_j^{u_j})$ does not match $H(s_j^{u_j}; r_j^{u_j})$ for some*

$j \in [m]$, *then* R_{DCOT} *rejects.*

6. $S_{\text{DCOT}} \overset{\mathcal{F}_{\text{CnP}}}{\longleftrightarrow} R_{\text{DCOT}}$: *The sender and receiver interact via the* $\mathcal{F}_{\text{CnP}}$ *functionality where* S_{DCOT} *submits* $\{H(s_j^b; r_j^b)\}_{j\in[m],b\in\{0,1\}}$ *and* R_{DCOT} *submits* H. $\mathcal{F}_{\text{CnP}}$ *checks if* H *was computed correctly on every pair* (s_j^b, r_j^b) *committed to before as part of the witness and if* $\mathcal{R}(\{(s_j^0, s_j^1)\}_{j\in[m]}, w)$. *If both these checks pass, it delivers 1 to* R_{DCOT} *and otherwise 0.*

Theorem 6.1 *Let* $\mathcal{H}$ *be a family of special-hiding MAC according to Definition 2.4 for* κ*-bit strings and* G *be a length-doubling PRG. Then protocol 3 securely computes functionality* $\mathcal{F}_{\text{DCOT}}$*-IVD in the presence of static active adversaries in the* $(\mathcal{F}_{\text{RCOT}}, \mathcal{F}_{\text{CnP}})$*-hybrid.*

Proof: Our proof follows by showing that for every adversary $\mathcal{A}$ in the $(\mathcal{F}_{\text{RCOT}}, \mathcal{F}_{\text{CnP}})$-hybrid world there exists a simulator $\mathcal{S}$ in the ideal world with access to functionality $\mathcal{F}_{\text{DCOT}}$ that generates an indistinguishable view. We begin with the description of the simulator while distinguishing two corruption cases.

The sender S_{DCOT} is corrupted. The simulator incorporates the code of the adversary $\mathcal{A}$ and emulates an execution as follows:

- First, it simulates $\mathcal{F}_{\text{RCOT}}$ and $\mathcal{F}_{\text{CnP}}$ and obtains $((k_j^0, r_j^0), (k_j^1, r_j^1))$ from the first round intercepting the inputs sent to $\mathcal{F}_{\text{RCOT}}$. Then it obtains s_j^0 and s_j^1 by decrypting the message sent by the sender in the second round and $(\{(S_j^b, R_j^b)\}_{j\in[m],b\in\{0,1\}}, w)$ from the third round intercepting the inputs sent to $\mathcal{F}_{\text{CnP}}$. Note that a corrupted sender can submit $S_j^b \neq s_j^b$ or $R_j^b \neq r_j^b$.
- Next, the simulator samples a random MAC $H \leftarrow \mathcal{H}$ and feeds it to $\mathcal{A}$ which responds with $\{h_j^b\}_{j\in[m],b\in\{0,1\}}$. If for any (j, b), $H(S_j^b, R_j^b) \neq h_j^b$, the simulator aborts.
- Then, the simulator defines a disjunctive predicate P as follows. For every (j, b) such that $H(s_j^b; r_j^b) \neq h_j^b$, it includes the literal v_j if $b = 0$ and $\neg v_j$ in case $b = 1$.

  Denote the set of literals within P by $\overline{\mathbf{v}}$.
- Finally, the simulator sends $(P, \{(S_j^0, S_j^1)\}_{j\in[m],b\in\{0,1\}}, w)$ to $\mathcal{F}_{\text{DCOT}}$.

Note first that the sender's view in the real execution is identically distributed to the simulated view because the sender only receives a MAC key H in an execution and this is identically distributed in both the real and simulated experiment. Therefore, it suffices to show that the outputs received by the honest receiver in both executions are indistinguishable. This follows from the next two claims.

Claim 6.1 *The probability that the honest receiver aborts in the real world is at least the probability with which the the honest receiver aborts in the ideal world.*

Proof: First, we observe that the honest receiver aborts in the ideal world if one of the following events occurs in the internal emulation by the simulator:

F_1 = **the MAC value does not match the committed input or the NP relation returns 0.** Specifically, the simulator aborts if $H(S_j^b, R_j^b) \neq h_j^b$ for any (j, b). Furthermore, $\mathcal{F}_{\text{DCOT}}$ delivers $\perp$ if the NP relation returns 0.

F_2 = $\mathcal{F}_{\text{DCOT}}$ **sends $\perp$ to the receiver when the predicate P is satisfied:** Specifically, this event occurs whenever $P(\overline{\mathbf{u}}) = 1$.

As mentioned at the beginning, the view of a malicious sender in the internal emulation by the simulator is identically distributed to the view of a malicious sender in the real world. This means that whenever F_1 occurs, it follows that functionality $\mathcal{F}_{\text{CnP}}$ outputs 0 as it checks both if the MACs of the committed values are correct and if the NP relation is satisfied. Next, if F_2 occurs in the internal emulation, it means that $H(s_j^{u_j}, r_j^{u_j}) \neq h_j^{u_j}$ for some j. Recall that whenever this happens in the real execution, the receiver aborts in Step 5. Therefore, we have that the probability with which the receiver aborts in the real world is at least the probability it aborts in the ideal world. □

Claim 6.2 *Conditioned on the receiver not aborting, the distribution of the output of the honest receiver in the real and simulated world is statistically close*

Proof: Consider the events F_1 and F_2 as described in the proof of Claim 6.1. We will prove that if these events do not occur then the output of the honest receiver in the real world is equal to the output received by the honest receiver in the ideal world except with negligible probability over the choice of H. Namely, condition on events F_1 and F_2 not occurring, it follows that the ideal receiver receives $S_j^{u_j}$ for every $j \in [m]$. On the other hand, the real receiver obtains $s_j^{u_j}$ for every $j \in [m]$. Next, suppose that $(s_j^{u_j}, r_j^{u_j}) \neq (S_j^{u_j}, R_j^{u_j})$ for some j and u_j. Now, since H is chosen uniformly at random after the malicious sender commits to $s_j^{u_j}$ and $S_j^{u_j}$, we conclude by the robustness of the special-hiding MAC (cf. Definition 2.4) that except with negligible probability either $H(S_j^{u_j}, R_j^{u_j}) \neq h_j^{u_j}$ or $H(s_j^{u_j}, r_j^{u_j}) \neq h_j^{u_j}$. The former event implies that F_1 occurs whereas the latter implies that F_2 occurs, and this concludes the proof of the claim. □

The receiver R_{DCOT} is corrupted. In this case, we need to simulate the messages from $\mathcal{F}_{\text{RCOT}}$ and $\mathcal{F}_{\text{CnP}}$ to the receiver. More precisely, the simulator internally incorporates $\mathcal{A}$ and then proceeds as follows:

- The receiver first provides to $\mathcal{F}_{\text{RCOT}}$ the input $\overline{\mathbf{u}} = (u_1, \ldots, u_m)$ and the witness w_R which the simulator forwards to $\mathcal{F}_{\text{DCOT}}$.
- Upon receiving from $\mathcal{F}_{\text{DCOT}}$ the values $(\{s_j^{u_j}\}_{j\in[m]}, 1)$, the simulator internally feeds $\{k_j^{u_j}, r_j^{u_j}\}_{j\in[m]}$ to $\mathcal{A}$ as the response from $\mathcal{F}_{\text{RCOT}}$, where $k_j^{u_j}$ and $r_j^{u_j}$ are sampled according to the honest strategy.
- In round two the simulator sends the ciphertexts, where corresponding to (j, u_j) it sends $s_j^{u_j} \oplus G(k_j^{u_j})$ and for the other ciphertexts it sends random strings.
- Next, upon receiving a MAC key H from the receiver, the simulator sends the adversary $\{h_j^b\}_{j\in[m],b\in\{0,1\}}$ such that $h_j^{u_j} = H(s_j^{u_j}; r_j^{u_j})$. All the other MACs are computed relative to an arbitrary set of inputs $\{s_j'^{1-u_j}\}_{j\in[m]}$.
- Finally, the simulator emulates the output of functionality $\mathcal{F}_{\text{CnP}}$ as 1.

Indistinguishability of our simulation in the $\{\mathcal{F}_{\text{RCOT}}, \mathcal{F}_{\text{CnP}}\}$-hybrid follows directly from the pseudorandomness of G and the privacy property of the special-hiding MAC (cf. Definition 2.4), which states that for every $x, x' \in \{0,1\}^\ell$ and $H \in \mathcal{H}$ the following two distributions are identical:

- $\{r \leftarrow \{0,1\}^s : H(x; r)\}$,
- $\{r' \leftarrow \{0,1\}^s : H(x'; r')\}$

This concludes our proof. ■

7 Realizing $\mathcal{F}_{\text{CnP}}$ via MPC-in-the-Head Approach

In order realize functionality $\mathcal{F}_{\text{CnP}}$, we follow the MPC-in-the-head paradigm of [IKOS07]. However, we will rely on a slightly different MPC model and make use of the watchlist mechanism in order to ensure consistency (a-la [IPS08]) to achieve better soundness and communication complexity. Recall that the $\mathcal{F}_{\text{CnP}}$ functionality defined in the previous section proceeds in two phases. In the first phase, the sender commits to an input x whereas in the second phase it receives a function H from the receiver and string y from the sender, and outputs 1 to the receiver only if x is a valid witness corresponding to some statement and an NP relation and $H(x) = y$. We simplify our functionality to simply take f and y from the receiver and sender and output 1 if $f(x) = y$. This is without loss of generality as f can incorporate the NP relation with the statement hard coded. Towards realizing the $\mathcal{F}_{\text{CnP}}$ functionality, we first introduce our MPC model that will be incorporated in our MPC-in-the-head paradigm.

MPC model. In the core of our improved analysis lies an alternative MPC model that captures a different network topology than the complete topology considered thus far. In our model, we will consider a *sender client* S, n servers $P_1, \ldots, P_n$, and a *receiver client* R. The sender has input x and the receiver has input f, whereas the servers do not provide any input. The MPC functionality is defined in Figure 7. We note that the underlying MPC only checks if $f(x) = y$ and does not realize it since it is easier and more efficient to run the verification procedure because we can flatten the verification circuit.

(Reactive) Functionality $\mathcal{F}_{\text{MPC}}$

Functionality $\mathcal{F}_{\text{MPC}}$ communicates with sender S and receiver R, and adversary $\mathcal{S}$.

Commit Phase: Upon receiving input x from S store this input.

Prove Phase: Upon receiving f from R and y from S it sends 1 to R only if $y = f(x)$.

Figure 7: The reactive MPC functionality.

More precisely, we consider the specific network where we restrict the communication to:

- Point-to-point channels between sender S and the servers and between receiver R and the servers.
- A broadcast channel between the servers. We note here that this deviates from the previous approaches [IKOS07, IKO+11] where additionally the servers had point-to-point channels between them. Excluding these channels is one of the ways we are able to improve the soundness.

For simplicity and convenience, we will allow the servers to have access to a coin-flipping oracle that can sample strings uniformly at random and broadcast to the servers. We will further assume that the underlying MPC invokes the coin-flipping oracle only once (we will be able to construct such an MPC protocol). More formally, our MPC protocol execution can be divided into the following phases:

1. **Commit Phase:** The servers receive inputs from the sender and perform local computation. The servers may additionally employ the broadcast channel.
2. **Statement Reveal Phase:** At the end of the commit phase, the receiver sends its input f to the servers.
3. **Compute Phase 1:** In this phase, the servers receive inputs from the sender and perform local computation and may use the broadcast channel.

4. **Coin-Flipping Phase:** At the end of the first compute phase, the servers obtain a public random string r of length ℓ that is sampled using a coin-flipping oracle.

5. **Compute Phase 2:** The servers perform additional local computation that may involve the broadcast channel and at the end of the phase, each server sends its outputs to the receiver R.

Definition 7.1 (Adaptive (t_p, t_r)-simulation certified oblivious-transfer) *Let Π be an MPC protocol in the framework described above between a sender client, n servers, and receiver R as described above. We say that a protocol Π realizes f with adaptive statistical (t_p, t_r)-simulation if the following properties hold:*

Completeness: *The protocol is statistically correct, namely, the receiver learns $f(x)$ except with negligible probability.*

Static t_p-privacy: *Consider a passive static adversary $\mathcal{A}$ that corrupts the receiver R and at most t_p servers (at the beginning of the execution), then the view of $\mathcal{A}$ can be simulated only using the outputs received by the receivers.*

Adaptive t_r-simulation: *Consider an active, adaptive adversary $\mathcal{A}$ that corrupts the sender and a total of at most t_r servers throughout the protocol. We will require standard UC simulation of the protocol Π against an adaptive adversary $\mathcal{A}$ for the MPC functionality f.*

We note that t_r is a constant fraction of n and t_p can be set to s where s is the statistical security-parameter.

Protocol description. Let Π be an MPC protocol satisfying Definition 7.1 and realizes $\mathcal{F}_{\text{MPC}}$ (we discuss implementations of Π in the next section). We will now design a malicious secure protocol Π_{CnP} that realizes the $\mathcal{F}_{\text{CnP}}$ functionality.

Protocol 4 (Protocol Π_{CnP} for realizing functionality $\mathcal{F}_{\text{CnP}}$ in the $\mathcal{F}_{\text{OT}}$-hybrid with active security)

- **Inputs.** *The sender's input is x and the receiver's input is f. Let $\Pi_{\text{ENC}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA (multi-message) symmetric-key encryption scheme that is constructed from PRG G.*

- **The protocol.**

  1. *The sender S_{CnP} runs the MPC protocol Π "in the head" as follows. It first picks a key k_i for each server $i \in [n]$, these keys will be used to establish the watchlists.*
  2. *Let u be an integer such that $1/u \in [t/2n, t/n]$. The sender S_{CnP} and receiver R_{CnP} apply one parallel call to $\mathcal{F}_{\text{OT}}$ in which the receiver obtains k_i to monitor the view of server P_i with probability $1/u$ for each $i \in [n]$. Let W be the set of servers for which the receiver obtains the key.*
  3. *S_{CnP} picks a random input r_S and invokes S on x and a random input r_i for every server P_i. It computes the views of the servers up to the end of commit phase, denoted by $(V_1^{\text{COM}}, \ldots, V_n^{\text{COM}})$. It then sends the view of the i^{th} server over the i^{th} watchlist, namely, it sends $c_i^{\text{COM}} = \mathsf{Enc}_{k_i}(V_i^{\text{COM}})$ to R_{CnP}. In addition, S_{CnP} sends the messages that the servers broadcast in the clear. We remark here that the broadcast message is also included as part of the view of each server. The receiver records the broadcast messages and decrypts and obtains V_i^{COM} for every $i \in W$.*
  4. *R_{CnP} sends its input f to S_{CnP}. Recall that in our MPC model, which S_{CnP} emulates in its head, the receiver R sends its input f to each servers. Using the message f received from R_{CnP}, S_{CnP} computes $y = f(x)$ and sends y to R_{CnP}.*

5. *Next, S_{CnP} computes the views of the servers up to the end of Compute Phase 1, denoted by $(V_1^1, \ldots, V_n^1)$. It then sends them encrypted as $c_i^1 = \mathsf{Enc}_{k_i}(V_i^1)$ to R_{CnP}. Additionally, it sends the broadcast messages exchanged in this phase. Once again, the receiver decrypts and obtains V_i^1 for $i \in W$.*
6. *R_{CnP} picks a random challenge r of length ℓ and sends it to the sender. S_{CnP} will feed r to the servers as the message received from the coin-flipping oracle in the Challenge Phase.*
7. *S_{CnP} next computes the views of the servers in Compute Phase 2, denoted by $(V_1^2, \ldots, V_n^2)$, sends them to R_{CnP} over the watchlists as $c_i^2 = \mathsf{Enc}_{k_i}(V_i^2)$ and includes the broadcast messages. In addition, it also sends the view of R in the internal emulation. The receiver obtains V_i^2 for $i \in W$.*
8. *The receiver next checks if there are any inconsistencies in the views obtained over the watchlist. More precisely, for every server $i \in W$, the receiver (re-)performs the computation and checks if all messages sent by the server were correct. Messages include the broadcast message between the servers and the message relayed to the receiver R. The broadcast message were sent in the clear and can be checked directly. The final message sent by the server to R can be checked as it is included in the view of R. It aborts if any of the checks fail.*
9. *Finally, R_{CnP} outputs what R outputs in its view.*

We proceed with the proof to the following theorem.

Theorem 7.2 *Let G be a length-doubling PRG. Then, protocol 4 securely computes $\mathcal{F}_{\text{CnP}}$ in the $\mathcal{F}_{\text{OT}}$-hybrid.*

Proof: Our proof follows by showing that for every adversary $\mathcal{A}$ in the $\mathcal{F}_{\text{OT}}$-hybrid execution there exists an simulator $\mathcal{S}$ in the ideal execution with access to functionality $\mathcal{F}_{\text{CnP}}$. We begin with the description of the simulator while distinguishing two corruption cases.

The sender S_{CnP} is corrupted. It is easy to see from the protocol that the only message that the sender receives from the receiver are the function is Step 4 and the random challenge r in Step 6. Hence generating a view of an adversary $\mathcal{A}$ controlling the sender only requires generating r uniformly at random within the simulation. The more challenging part in the simulation is extracting an input on behalf of the sender for the $\mathcal{F}_{\text{CnP}}$ functionality and deciding when to deliver the output to the receiver. Towards this $\mathcal{S}$ defines an adversary $\mathcal{B}$ for the underlying MPC protocol Π. By the guarantee of the adaptive t_r-(UC) simulation property of Π there exists a corresponding simulator $\mathcal{S}_{\mathcal{B}}$ in the $\mathcal{F}_{\text{MPC}}$-hybrid. Our simulation follows by simultaneously executing $\mathcal{B}$ and $\mathcal{S}_{\mathcal{B}}$. In more detail, $\mathcal{S}$ constructs an adversary $\mathcal{B}$ as follows:

- Upon receiving the input $(1^\kappa, x)$, it internally incorporates the code of $\mathcal{A}$ and begins emulating an execution by running the adversary $\mathcal{A}$ on this input and uniformly generated randomness.
- $\mathcal{B}$ internally emulates functionality $\mathcal{F}_{\text{OT}}$ where in Step 2 it receives from $\mathcal{A}$ keys k_i for every server P_i within $\mathcal{A}$'s message to the $\mathcal{F}_{\text{OT}}$ functionality. $\mathcal{B}$ records these keys.
- Next, $\mathcal{B}$ receives encryptions of views of the servers $c_1, \ldots, c_n$ in the commit phase from which it extracts the view V_i^{COM} for each server i by decrypting the ciphertexts as $V_i^{\text{COM}} = \mathsf{Dec}_{k_i}(c_i)$ where k_i are the keys recorded in the previous step.
- $\mathcal{B}$ corrupts the sender and then sends as input to each server P_i according to what is specified in V_i^{COM}.

- Recall that every message that is broadcast by the servers in any phase is provided in the clear by $\mathcal{A}$ and is included in the view of each server $(V_1^{\mathrm{COM}}, \ldots, V_n^{\mathrm{COM}})$. Let T_{COM} be the subset of servers for which the broadcast messages incorporated in their views differ from the broadcast messages relayed by $\mathcal{A}$. $\mathcal{B}$ adaptively corrupts the servers in T_{COM} and use the (incorrect) broadcast message included in the view for its computation.

- Next, $\mathcal{B}$ receives f from the receiver R (emulated by $\mathcal{S}$) and forwards it internally to $\mathcal{A}$ as received from $\mathcal{R}_{\mathrm{CnP}}$. In response, $\mathcal{A}$ provides encrypted views of the servers in the Compute Phase from which $\mathcal{B}$ extracts $V_1^1, \ldots, V_n^1$. On behalf of the corrupted sender, $\mathcal{B}$ sends to each server the corresponding message recorded in the server's view V_i^1. Let T^1 be the servers for which the broadcast message is inconsistent with what is sent in the clear by $\mathcal{A}$. If $|T^1 \cup T^{\mathrm{COM}}| > t_r$, then $\mathcal{B}$ aborts. Otherwise it corrupts the servers in $T^1 - T^{\mathrm{COM}}$ and uses the incorrect broadcast message incorporated in the view.

- The servers receive a random string r from the coin-flipping oracle next in the MPC execution. $\mathcal{B}$ feeds r internally to $\mathcal{A}$ as received from $\mathcal{R}_{\mathrm{CnP}}$. In response, $\mathcal{A}$ gives encrypted views of the servers in Compute Phase 2 from which $\mathcal{B}$ extracts $V_1^2, \ldots, V_n^2$ and proceeds exactly as in Compute Phase 1 by computing the set of servers T^2 with inconsistent views. Again, $\mathcal{B}$ aborts if $|T^2 \cup T^1 \cup T^{\mathrm{COM}}| > t_r$ and otherwise it corrupts the servers in $T^2 - T^1 - T^{\mathrm{COM}}$. Finally, on behalf of the corrupted servers $T^{\mathrm{COM}} \cup T^1 \cup T^2$, $\mathcal{B}$ sends a message to R as recorded in the corresponding view V_i^2.

- Finally, $\mathcal{B}$ outputs the view of $\mathcal{A}$ in the internal emulation.

Our simulator $\mathcal{S}$ for protocol Π_{CnP} will internally run $\mathcal{S}_{\mathcal{B}}$ with $\mathcal{B}$ (where the input for $\mathcal{S}_{\mathcal{B}}$ is the input $\mathcal{S}$ obtains for $\mathcal{A}$. Upon extracting $\mathcal{B}$'s input x, $\mathcal{S}_{\mathcal{B}}$ sends it on behalf of the corrupted sender to $\mathcal{F}_{\mathrm{MPC}}$ emulated by $\mathcal{S}$, which intercepts x and sends it to $\mathcal{F}_{\mathrm{CnP}}$. Next, $\mathcal{S}_{\mathcal{B}}$ expects to receive f from the receiver. Specifically, $\mathcal{S}$ receives f from $\mathcal{F}_{\mathrm{CnP}}$ which it internally emulates for $\mathcal{S}_{\mathcal{B}}$ as received from R. In response, $\mathcal{S}_{\mathcal{B}}$ provides y which $\mathcal{S}$ forwards to $\mathcal{F}_{\mathrm{CnP}}$. Finally, $\mathcal{S}$ outputs whatever $\mathcal{S}_{\mathcal{B}}$ outputs.

We first claim that indistinguishability of the simulation and the correctness of the receiver's output follow directly from the adaptive t_r-simulation of $\mathcal{B}$. This is because as long as fewer than t_r servers are corrupted overall then we have that the MPC execution with adversary $\mathcal{B}$ proceeds exactly as specified in the views submitted by S_{CnP}.

Next, we observe that if more than t_r servers are corrupted $\mathcal{S}$ aborts. We prove here that in this case R_{CnP} aborts except with negligible probability. Specifically, since R_{CnP} uses the watchlist mechanism to observe each server with probability $1/u$ where $u \in [t_p/2n, t_p/n]$, it holds that the expected number of servers watched by the receiver is t_p. We will show that with probability at most $(1 - \frac{t_r}{n})^{t_p}$ the receiver in protocol Π_{CnP} misses all the servers for which the view is inconsistent with the watchlist views. This is negligible if $\frac{t_r}{n}$ is $O(1)$.

We calculate this probability by computing the probability that the watchlist includes a server carrying an incorrect broadcast message or carrying an incorrect final message to the receiver R. Towards this, we can define a graph G with nodes for each server and two special nodes B and R where B corresponds to all broadcast messages and R denotes the receiver. Then an edge is included in G between a server and B if some broadcast message included in the view of this server is different from the message relayed by $\mathcal{A}$. Moreover, an edge between R and a server is included in G if the message sent by the server included in the view of the server is different from the message received by R included in the view of R. No we observe that if any server that has any edge incident on it in included in the watchlist W, then the adversary is caught. Since by our assumption there are more than t_r edges, it follows that the probability that the adversary is not caught is at most $(1 - \frac{t_r}{n})^t$.

The receiver R_{CnP} is corrupted.

- Upon receiving the receiver's input $(1^\kappa, f)$, the simulator $\mathcal{S}$ internally invokes the adversary $\mathcal{A}$ on this input and uniformly generated randomness.
- $\mathcal{S}$ internally emulates functionality $\mathcal{F}_{\text{OT}}$ as follows:
  1. $\mathcal{S}$ generates a key k_j for each server P_j using Gen.
  2. $\mathcal{S}$ receives from the adversary its input to $\mathcal{F}_{\text{OT}}$ which consists of $\widetilde{u}_1, \ldots, \widetilde{u}_m$. Recall that the receiver also obtains keys for each server with probability $1/u$ where $u \in [t/2n, t/n]$, $\mathcal{S}$ sends k_i to the receiver with probability $1/u$. Let T be the subset of $[n]$ containing the indexes of the servers for which the adversary learns the key. The adversary, acting as the receiver, expects to receive in the commit phase commitments to the servers views. To emulate these, $\mathcal{S}$ relies on the t_p-privacy of the underlying MPC protocol Π to generate the views V_i^{Com} for the servers in $\{P_i\}_{i \in T}$. More specifically, $\mathcal{S}$ defines a passive adversary $\mathcal{B}$ for Π that corrupts all the servers in T, and runs the simulation for the MPC protocol with this adversary $\mathcal{B}$. Specifically, let $\mathcal{S}_{\mathcal{B}}$ denote the corresponding simulator for $\mathcal{B}$ guaranteed by the t_p-privacy property, where $\mathcal{S}_{\mathcal{B}}$ provides the views for the servers in T. $\mathcal{S}$ defines the views for $i \in T$ according to what it received from $\mathcal{S}_{\mathcal{B}}$ and sets the rest of the views to the all 0's string. In addition it extracts the broadcast messages from the views and feeds it in the clear.
  3. $\mathcal{S}$ obtains the function f from the adversary which it forwards the $\mathcal{F}_{\text{CnP}}$ functionality, receiving the value 1. Following this, $\mathcal{S}_{\mathcal{B}}$ provides views of the servers in T in Compute Phase 1 which it encrypts and feeds to $\mathcal{A}$ just as in the Commit Phase.
  4. Next, it receives from the adversary a challenge r and instructs $\mathcal{S}_{\mathcal{B}}$ to send this challenge to $\mathcal{B}$. Following this, $\mathcal{S}_{\mathcal{B}}$ provides the message sent by the servers in T and the view of R in the internal emulation. $\mathcal{S}$ simply forwards this view to the receiver R to $\mathcal{A}$.

Note that the differences between the two executions are with the way the V_i's views are generated (for the corrupted servers these are generated from the corrupted receivers outputs whereas the rest of the views are set to zero). Moreover, the broadcast messages are computed by the simulator $\mathcal{S}_{\mathcal{B}}$. Then indistinguishability of simulation follows directly from the t_p-privacy of the underlying MPC protocol and the privacy of the symmetric key encryption. ■

To instantiate our MPC protocol we will rely on a variant of the MPC protocol of [IPS09] found in Appendix C. We specify its details in the next section.

8 Information-Theoretic Protocol for Realizing $\mathcal{F}_{\text{MPC}}$

In this section, we present our instantiations for an MPC protocol to realize $\mathcal{F}_{\text{MPC}}$. Recall that in our MPC model, we consider a *sender client* S, n servers $P_1, \ldots, P_n$ and a *receiver client* R. The sender has input x and the receiver has input f, whereas the servers have no input. The security guarantee requires the MPC protocol to be t_p-private and statistical t_r-robust. We will instantiate the MPC protocols for our two main results as follows:

Communication efficient variant. For our communication efficient protocol, that achieves constant overhead, we will instantiate the underlying MPC protocol with a variant of the protocol from [DI06] further used in [IKOS07]. We point out that our MPC functionality is different from the functionality used in

[IKOS07] as we introduce an initial phase where the sender *stores* the input within the servers and another phase where the receiver provides the function f. Nevertheless, we can rely on essentially the same MPC protocol since [DI06], as well as other prior protocols such as [BGW88, CCD88] and [DI05] already have an input sharing phase where all parties share their inputs. Therefore, incorporating an input sharing phase as such will not require any modification of the MPC protocol.

We recall next that the MPC protocol of [DI06] incurs a communication complexity of $O(\mathsf{poly}(\kappa) \cdot |\mathrm{C}|)$ where C is the circuit representing the function computed by the servers and κ is a computational parameter. In order to avoid this blowup, [IKOS07] squash the circuit C that represents a polynomial-sized verification circuit for an NP-relation, by considering an extended witness of size $|\mathrm{C}|$ that includes the intermediate values in the computation of $\mathrm{C}(w)$ in addition to w itself. In order to verify the computation of $\mathrm{C}(w)$ it suffices to verify the computation of each gate in the computation of $\mathrm{C}(w)$ from the extended witness. The squashed circuit takes as input the extended witness and has output of size $|\mathrm{C}|$ which includes for every gate the result of checking the output value w.r.t to its two input values. Now, an output of $1^{|\mathrm{C}|}$ of the squashed circuit is interpreted as an output 1 for the original computation and any other output is interpreted as 0. In our MPC protocol, the circuit C is not determined before the receiver reveals function f. Hence, we need to modify the MPC protocol to first share the original input and then only after f is revealed, the sender additionally shares the intermediate values of the computation under f with the servers.

The resulting MPC protocol will have the same asymptotic communication complexity as the variant used in [IKOS07] and therefore we obtain the following lemma (which is a reformulation of Lemma 4.8 [IKOS07]) for our MPC protocol).

Lemma 8.1 (Reformulation of Lemma 4.8 [IKOS07]) *Let f be a function which can be computed by a circuit C. Then, f can be realized by a perfectly t-private and statistically t-robust MPC protocol Π_f with $t = O(n)$ and the following efficiency features:*

- *Commit Phase and Compute Phase 1 involves a total of $O(|\mathrm{C}|) + \mathsf{poly}(t, n, \log(|\mathrm{C}|))$ sent from the sender to the n servers.*
- *The random challenge r between Compute Phase 1 and Compute Phase 2 is of length $\mathsf{poly}(t, n, \log(|\mathrm{C}|))$.*
- *Phase 2 involves broadcast messages whose total length is $O(|\mathrm{C}|) + \mathsf{poly}(t, n, \log(|\mathrm{C}|))$.*

Concretely efficient variant. In our concretely efficient variant, we will not use the MPC protocol from [DI06, IKOS07] that applies to *boolean* circuits and has a high concrete communication and computation overhead. In particular, this protocol employs secret sharing via AG codes [CC06]. Instead we will rely on the variant of the MPC protocol from [DI06] described in [IPS09], which applies to arithmetic circuits over large fields and has a small concrete overhead. Our concretely efficient variant will employ a function f that can be cast as an arithmetic circuit. In a companion work [AHIV17], we employ a similar MPC protocol to obtain concretely efficient sublinear zero-knowledge arguments. As mentioned above, we will have to consider an analogous variant to incorporate the additional phases in our MPC model.

Below, we summarize the result in the following lemma.

Lemma 8.2 (Reformulation of Lemma 4 [AHIV17]) *Let $s \in \mathbb{N}$ be the statistical parameter and f be a function which can be computed by an arithmetic circuit C over a finite field $|\mathbb{F}| > 2^s$. Let $\alpha = \sqrt{\frac{t_p|\mathrm{C}|}{2}}$ and $\beta = \sqrt{\frac{18|\mathrm{C}|}{t_p}}$. Then, f can be realized by a perfectly t_p-private and statistically t_r-robust MPC protocol Π_f with 2^{-s} error where $t_p = s(-\log(4/5))^{-1}$, $t_r = \lfloor \frac{n}{5} \rfloor - 1$ and $n = 5(\alpha + t_p) + 1$ and the following efficiency features:*

- *The view size of each server is $s(\beta+1)$ bits.*
- *The random challenge r between the Compute Phase 1 and Compute Phase 2 is of length s.*
- *Phase 2 involves broadcast messages whose total length is $3s(\alpha+t)$.*

9 Putting it Together

In this section we instantiate our framework for two-party computation by instantiating the computation of our two NC^0 functionalities and the information-theoretic MPC protocols and obtain different efficiency guarantees, both in the asymptotic and concrete regimes. We use the following convention:

- We use κ and s for the computational and statistical security parameter respectively.
- We use n to denote the input lengths of the parties and m to denote the output length of the function $\mathcal{F}$ that the parties want to securely compute.

Both of our variants will have constant overhead communication complexity over the passively secure Yao protocol. The second uses a large number of OTs but has better concrete efficiency.

9.1 Variant 1: Asymptotically Optimal Construction

The first variant incurs communication complexity of $O(|\mathrm{C}|\kappa)$ bits in the κ-bit string OT hybrid. We first provide a basic result for this variant that will employ $O(|\mathrm{C}|)$ calls to κ-bit string OT functionality. Next, by relying on an information-theoretic PRG, we will be able to reduce the number of calls to $n + O(s \cdot |\mathrm{C}|^{\epsilon})$ for an arbitrary constant $\epsilon > 0$. Such information-theoretic PRGs exist assuming explicit constant-degree unbalanced unique-neighbor expanders.

The basic result we obtain in this variant is the following theorem.

Theorem 9.1 *There exists a protocol compiler that given κ (PRG seed length), s (statistical security parameter), and a two-party deterministic functionality $\mathcal{F}$ expressed as a Boolean circuit $\mathrm{C} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, outputs a protocol Π_{C} that securely realizes $\mathcal{F}$ in the κ-bit string OT hybrid, namely using ideal calls to κ-bit string OT. The protocol Π_{C} has the following efficiency features:*

- *It makes $O(|\mathrm{C}|) + \mathsf{poly}(\log(|\mathrm{C}|), s)$ black-box calls to a PRG $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^{\kappa} \to \{0,1\}^{2\kappa}$.*
- *It makes $n + O(s \cdot |\mathrm{C}|^{\epsilon})$ calls to a κ-bit string OT oracle.*
- *It communicates $O(\kappa \cdot |\mathrm{C}|) + \mathsf{poly}(\log(|\mathrm{C}|), \log \kappa, s)$ bits.*

Remark 9.1 *Recall that we require the distinguishing advantage to be bounded by $2^{-s} + \nu(\kappa)$ for some negligible function $\nu(\cdot)$. We state our asymptotic result with s as a parameter as we would like to make the distinction between protocols that achieve 2^{-s} error versus negligible in s error. Furthermore, it allows us to compare our protocols with prior works that achieve the same simulation error. We remark that we can assume $s < \kappa$ without loss of generality as we require the distinguishing error to be bounded by a negligible function in κ and if s is bigger than κ, we can let $s = \kappa$.*

Proof of Theorem 9.1. We follow the framework described in Section 3.

1. Following an approach based on [LP07, SS13], we first transform the original functionality $\mathcal{F}$ into a new functionality $\mathcal{F}_{\text{IVD}}$ that will resist input-dependent attack.

   - The circuit size of $\mathcal{F}_{\text{IVD}}$ is $O(\kappa \cdot |\text{C}| + \kappa \cdot s)$ for any circuit C that computes the original functionality $\mathcal{F}$.
   - The receiver's input length in $\mathcal{F}_{\text{IVD}}$ is $O(|\text{C}|) + O(\max(n, s)) = O(|\text{C}| + s)$.

2. We next compile $\mathcal{F}_{\text{IVD}}$ using the AuthPRG2 compiler to obtain a protocol in the $\mathcal{F}_{\text{DCOT}}$-hybrid following the transformations presented in Sections 5.

   This protocol $\mathcal{F}_2$ has the following features:

   - The receiver's input length (which corresponds to the number of parallel OT invocations) is $n + O(s \cdot |\text{C}|^\epsilon)$.
   - The sender's algorithm makes $O(|\text{C}| + s \cdot |\text{C}|^\epsilon)$ black-box calls to a length-doubling PRG $\text{G}_{\text{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$.
   - The total length of the sender's inputs to the functionality is $O(\kappa \cdot |\text{C}| + \kappa \cdot s \cdot |\text{C}|^\epsilon)$.
   - The NP relation associated with the sender in the $\mathcal{F}_{\text{DCOT}}$-IVD can be computed via a boolean circuit of size $O(\kappa \cdot |\text{C}| + \kappa \cdot s)$ and the NP relation associated with the receiver can be computed via a circuit of size $O(n + s \cdot |\text{C}|^\epsilon)$.

   We will track the number of OTs and the sum total of the lengths of the sender OT inputs. However, at the end, we will rely on a standard transformation that takes n_{OT} parallel OTs where the sum of OT input lengths is ℓ_{OT} and compile it to n_{OT} parallel OTs with κ-bit inputs that will require the sender to make $\lceil \frac{\ell_{\text{OT}}}{\kappa} \rceil$ calls to the underlying length doubling PRG G_{PRG} and send one additional message to the receiver of length ℓ_{OT}. This transformation simply requires the sender to use κ-bit keys sampled independent from a semantically-secure encryption scheme as the OT sender inputs and send encryptions of the corresponding inputs with that key.

3. We first replace the oracle call to $\mathcal{F}_{\text{DCOT}}$-IVD within $\mathcal{F}_2$ with protocol Π_{DCOT} from Section 6 that is realized in the $(\mathcal{F}_{\text{RCOT}}, \mathcal{F}_{\text{CnP}})$-hybrid. Then we replace the oracle call to $\mathcal{F}_{\text{CnP}}$ with our protocol Π_{CnP} where we instantiate our MPC protocol using a variant of the protocol from [DI06] further used in [IKOS07]. The resulting protocol is in the $\mathcal{F}_{\text{RCOT}}$-hybrid and realizes $\mathcal{F}_{\text{DCOT}}$-IVD against active adversaries. The communication complexity of the protocol can be computed as follows:

   (a) The sender and receiver first invoke the $\mathcal{F}_{\text{RCOT}}$-functionality where the number of OT calls is $n + O(s \cdot |\text{C}|^\epsilon)$ and the sender's input length for each OT choice in each invocation is κ. The NP relation associated with the receiver can be computed via a circuit of size $O(n + s \cdot |\text{C}|^\epsilon)$.

   (b) The protocol makes $O(|\text{C}| + s \cdot |\text{C}|^\epsilon)$ black-box calls to a length-doubling PRG $\text{G}_{\text{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$.

   (c) The total bits communicated in the protocol can be computed from the following:

      - The sender transmits "encrypted" values whose sum total is $O(\kappa \cdot |\text{C}| + \kappa \cdot s \cdot |\text{C}|^\epsilon)$ bits.
      - The sender transmits a MAC of length s corresponding to each OT input. There are totally $O(|\text{C}| + s)$ strings transmitted via 1-out-of-2 OTs. Therefore, sending the MACs will require the sender to transmit $2 \cdot s \cdot O(n + s \cdot |\text{C}|^\epsilon) = O(s \cdot n + s^2 \cdot |\text{C}|^\epsilon)$ bits.

   Therefore, the protocol involves $O(\kappa \cdot |\text{C}|) + \kappa \cdot s \cdot |\text{C}|^\epsilon$ bits of communication.

(d) The NP-relation associated with $\mathcal{F}_{\text{CnP}}$ is of size $O(\kappa \cdot \mathrm{C} + \kappa \cdot s \cdot |\mathrm{C}|^{\epsilon})$. We can conclude the communication complexity of the protocol realizing $\mathcal{F}_{\text{CnP}}$ to be $O(\kappa \cdot \mathrm{C} + \kappa \cdot s \cdot |\mathrm{C}|^{\epsilon})$ and involves $O(|\mathrm{C}|) + s \cdot |\mathrm{C}|^{\epsilon}$ calls to the PRG.

4. In our final step of the transformation we replace the oracle call to $\mathcal{F}_{\text{RCOT}}$ with the protocol from [IKO+11] to achieve full security. Since the overhead induced by [IKO+11] is polylogarithmic in the circuit size that computes this functionality, we get a protocol with following features.

   - The protocol makes $O(|\mathrm{C}| + s \cdot |\mathrm{C}|^{\epsilon})$ black-box calls to a length-doubling PRG $\mathrm{G}_{\text{PRG}} : \{0,1\}^{\kappa} \to \{0,1\}^{2\kappa}$.
   - The protocol involves $O(\kappa \cdot |\mathrm{C}| + s \cdot \kappa \cdot |\mathrm{C}|^{\epsilon}) + \kappa \cdot n \cdot \mathsf{poly}(\log s, \log |\mathrm{C}|)$ bits of communication.
   - The protocol incurs $n + O(s \cdot |\mathrm{C}|^{\epsilon})$ calls to $O(\kappa)$-bit string OTs.

This concludes the proof of Theorem 9.1.

Remark 9.2 *As discussed in Footnote 6 and Section 2.2, the combinatorial assumption about explicit s-wise PRGs is a seemingly mild assumption that was already used in other contexts.*

This theorem provides the first black-box protocol that simultaneously achieves asymptotically constant overhead communication complexity over Yao's passively secure protocol and requires sublinear (in circuit size) number of calls to a OT protocol. In contrast, prior works have either obtained constant overhead (eg, [WRK17], albeit in the bit-OT hybrid model) or a small number of calls to the OT oracle (eg, protocols based on cut-and-choose).

9.2 Variant 2: Concretely Efficient Variant

Our second variant will also achieve a communication complexity of $O(\kappa \cdot |\mathrm{C}|)$ bits and employ $O(n + s)$ calls to a κ-bit string OT oracle. We will identity the precise constant in the overhead. In this variant we will be able to incorporate the Free XOR optimization [KS08] and half-gates optimizations [ZRE15]. However, we will settle for security with correlated abort.

Theorem 9.2 *There exists a protocol compiler that, given κ (PRG seed length), s (statistical security parameter) and a functionality $\mathcal{F}(x, y)$ expressed as a circuit $\mathrm{C} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, outputs a protocol Π_{C} which securely realizes $\mathcal{F}$ in the κ-bit string OT-hybrid with correlated abort with the receiver's input and the following features:*

- *The protocol makes $|\mathrm{C}| + 2 \cdot s + \max(4 \cdot n, 8 \cdot s)$ calls to κ-bit string OT.*
- *The protocol communicates (in bits)*

$$4 \cdot (\kappa + s) \cdot |\mathrm{C}| + 2 \cdot s \cdot (n + 2 \cdot s + \max(4 \cdot n, 8 \cdot s)) + 18 \cdot s^{1.5} \cdot \sqrt{|\mathrm{C}|}.$$

Proof of Theorem 9.2. The compilation takes as input a circuit C and security parameter κ and proceeds by following the same approach as in our first variant with the exception that we use our transformation in the $\mathcal{F}_{\text{AuthCol}}$-hybrid as described in Section 4.2 and the MPC protocol instantiated above. More precisely,

1. We transform the original functionality $\mathcal{F}$ into $\mathcal{F}_{\text{IVD}}$ that is resistant to IVD attacks just as in the previous compilation. At the end of the section, we provide bounds for the additive terms. The circuit size of $\mathcal{F}_{\text{IVD}}$ will therefore be $|\text{C}|$ and the reciever input size will be $n + \max(4 \cdot n, 8 \cdot s)$ where $\max(4 \cdot n, 8 \cdot s)$ is the length of the encoding of the receiver's input following [LP07].

2. Next, we compile $\mathcal{F}_{\text{IVD}}$ to $\widehat{\mathcal{F}}$ using $\mathcal{F}_{\text{AuthCol}}$-hybrid as described in Section 4.2 which correlates the abort event on the receiver's side with its input. The NC^0 functionality $\widehat{\mathcal{F}}$ has the following features:

   - The receiver's input size is $n + 2 \cdot s + \max(4 \cdot n, 8 \cdot s)$.
   - The output length of the NC^0 functionality that depends on the receiver's input bits is $4 \cdot |\text{C}| \cdot s$. Note that $\widehat{\mathcal{F}}$ includes an additional n^2 XOR gates compared to $\mathcal{F}_{\text{IVD}}$. These are required to decode the receiver's input before the computation begins. We will not include these gates in our circuit size as we can rely on the Free XOR optimization.

3. We next consider an information-theoretic protocol Π that realizes $\widehat{\mathcal{F}}$-IVD in the $\mathcal{F}_{\text{SCOT}}$-IVD-hybrid. This proceeds in two steps: (1) Take a non-interactive protocol for $\widehat{\mathcal{F}}$ using a parallel OT oracle, where the protocol only needs to be secure in the presence of a passive sender and an active receiver. (2) Use the $\mathcal{F}_{\text{SCOT}}$-IVD oracle to enforce honest behavior of the sender up to IVD attacks. We describe the steps in more detail to understand the concrete overhead of the steps.

   First, we compute the communication complexity of the passive protocol that realizes the NC^0 functionality in (1). Note that the computation of $\widehat{\mathcal{F}}$ involves a computation with locality one. This means for every input bit of the receiver we need to execute one 1-out-of-2 string OT. The sum total of the OT sender input lengths will be $4 \cdot |\text{C}| \cdot s$.

   This protocol incurs the following costs:

   - The receiver's input size is $n + 2 \cdot s + \max(4 \cdot n, 8 \cdot s)$.
   - The sum total of the sender OT inputs is $4 \cdot s \cdot |\text{C}|$ bits.
   - The length of the sender's message is $4 \cdot \kappa \cdot |\text{C}|$.
   - The sender predicate that will be the NP relation used in the $\mathcal{F}_{\text{SCOT}}$-IVD oracle can be expressed as an arithmetic circuit over the $\mathbb{GF}(2^s)$ field. We will only count the number of multiplication gates, as addition will be free. Recall that the global predicate is required to enforce honest behavior of the sender in Π. Given the sender inputs to the parallel OT, we compute the size of the global predicate as follows:

     Witness size of the NP relation. The witness to the NP statement includes (1) The PRF values which totals to $4 \cdot |\text{C}|$ field elements, and (2) for each wire w, λ_w.
     The total witness size (i.e. input size of the NP-relation) is $4 \cdot |\text{C}| + |\text{C}|$ field elements.

     Number of multiplications: We compute the number of multiplications in $\mathbb{GF}(2^s)$ in the NP-relation.
     Consider one of the garbled rows for a gate g with input wires a, b and output wire c. The MAC part of the row can be expressed as:

$$F^a_{\mathsf{prg}} + F^b_{\mathsf{prg}} + \sigma_0 + f_{g,row}(\lambda_a, \lambda_b, \lambda_c) \cdot (\sigma_1 - \sigma_0).$$

     where $F^a_{\mathsf{prg}}, F^b_{\mathsf{prg}}, \lambda_a, \lambda_b, \lambda_c, k^0_c, k^1_c$ will be included in the witness for the predicate. The function $f_{g,row}$ can further be expressed as

$$c_1 \cdot \lambda_a \cdot \lambda_b + c_2 \cdot \lambda_a + c_3 \cdot \lambda_b + c_4 \cdot \lambda_c + c_5 \tag{1}$$

for some coefficient c_1 through c_5.
First, we observe that σ_0 and σ_1 are provided by the receiver. We consider the computation of each bit of this string. For every position i, if the i^{th} bit of σ_0 and $\sigma_1 - \sigma_0$ are b_1 and b_2 respectively, the result will be $v_{g,row}^{b_1 b_2}$, where

$$v_{g,row}^{b_1 b_2} = F_{\mathsf{prg}}^{a} + F_{\mathsf{prg}}^{b} + b_1 + f_{g,row}(\lambda_a, \lambda_b, \lambda_c) \cdot b_2.$$

We note here that $v_{g,row}^{b_1 b_2}$ is independent of the position in the MAC part. This means we need to compute $f_{g,row}$ only once per row. This involves only one multiplication $\lambda_a \cdot \lambda_b$. Recall that the masked MAC values are transmitted via the OT. $v_{g,row}^{b_1 b_2}$ can be computed in a different way from the sender's OT inputs. In fact, this involves simply addition operations. In the NP-relation we check using addition gates if the $v_{g,row}^{b_1 b_2}$ computed two different ways are equal.

Binary constraints. The λ_w values need to be sampled from $\{0, 1\}$ and since we are operating in $\mathbb{GF}(2^s)$ we need enforce this constraint. We express this as

$$\lambda_w^2 - \lambda_w = 0$$

This will require a single multiplication per wire for a total of $|\mathrm{C}|$ multiplications.

Combining the above, we have a total of $2 \cdot |\mathrm{C}|$ multiplications.

4. We replace the oracle call to $\mathcal{F}_{\mathrm{SCOT}}$-IVD in Π with a slight variant of the protocol Π_{DCOT} from Section 6. Recall that Π_{DCOT} is the protocol that realizes $\mathcal{F}_{\mathrm{DCOT}}$-IVD in the $(\mathcal{F}_{\mathrm{RCOT}}, \mathcal{F}_{\mathrm{CnP}})$-hybrid. To realize $\mathcal{F}_{\mathrm{SCOT}}$ we can use the same protocol with the exception that the parties will use $\mathcal{F}_{\mathrm{OT}}$ instead of the $\mathcal{F}_{\mathrm{RCOT}}$ functionality. This is sufficient as $\mathcal{F}_{\mathrm{SCOT}}$ only requires to certify the sender's inputs. We then replace the oracle call to $\mathcal{F}_{\mathrm{CnP}}$ with our protocol Π_{CnP} in the $\mathcal{F}_{\mathrm{OT}}$-hybrid where we instantiate our MPC protocol using [AHIV17]. The resulting protocol realizes $\mathcal{F}_{\mathrm{COT}}$-IVD against static corruptions by active adversaries. This communication complexity of the protocol can be computed as follows:

   (a) The sender communicates $4 \cdot s \cdot |\mathrm{C}|$ bits to the OT functionality and $4 \cdot \kappa \cdot |\mathrm{C}|$ bits in a direct message to the receiver in the first step of the protocol as part of the passively secure protocol for realizing the $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ functionality.

   (b) Transmitting a MAC for each OT input. We transmit a MAC value of length s for each OT string independent of its length. There are $2 \cdot (n + 2 \cdot s + \max(4 \cdot n, 8 \cdot s))$ strings transmitted via OTs. Therefore, sending the MACs will require the sender to transmit $2 \cdot s \cdot (n + 2 \cdot s + \max(4 \cdot n, 8 \cdot s))$ bits.

   (c) The commit-and-prove protocol. The communication complexity of this protocol can be bounded by $8 \cdot s^{1.5} \cdot \sqrt{I + 3 \cdot M}$ bits, where M represents the number of field multiplications over $\mathbb{GF}(2^s)$ involved in the computation of the NP-relation and I denotes the any additional witness bits (involved only in additions). Our NP-relation can be expressed as an arithmetic circuit over $\mathbb{GF}(2^s)$, including the global predicate from the previous step and an additional check to ensure that the MACs are correct. From the previous step we know that the first part requires $2 \cdot |\mathrm{C}|$ multiplications for verifying the OT inputs whereas the second part, verifying the MACs does not require any multiplication. The input size of the NP relation is $5 \cdot |\mathrm{C}|$. For an arithmetic circuit C we denote by $|\mathrm{C}|$ the number of multiplication gates. Our proof length is given by

$$5.5 \cdot s^{1.5} \cdot \sqrt{5 \cdot |\mathrm{C}| + 3 \cdot 2 \cdot |\mathrm{C}|)}$$

Circuit Size	Yao (passive)	Our Comm	Overhead
6800	0.21	0.42	1.96
1024	0.03	0.09	2.89
2048	0.06	0.15	2.36
4096	0.13	0.26	2.07
8192	0.25	0.48	1.9
16384	0.5	0.91	1.81
32768	1.01	1.78	1.76
65536	2.02	3.49	1.73
131072	4.03	6.89	1.71
262144	8.06	13.66	1.69
524288	16.13	27.17	1.69
1048576	32.25	54.13	1.68
2097152	64.5	107.97	1.67
4194304	129	215.52	1.67

Table 1: We give our total estimated communication cost in MB of our concretely efficient variant where $\kappa = 128$ and $s = 40$.

Finally, the overall communication complexity of the protocol in the κ-bit string OT-hybrid for circuits with more than 30000 AND gates is

$$\underbrace{4 \cdot (\kappa + s) \cdot |\mathrm{C}|}_{\text{Garbled Circuit}} + \underbrace{2 \cdot s \cdot (n + 2 \cdot s + \max(4 \cdot n, 8 \cdot s))}_{\text{MAC for every OT input}} + \underbrace{18 \cdot s^{1.5} \cdot \sqrt{|\mathrm{C}|}}_{\text{CnP protocol}}.$$

As stated in Section 4.2.1, we conclude with a corollary that simplifies the leakage function into WVD-style leakage and further holds in the random oracle model.

Corollary 9.3 *Assuming an injective PRG on the last s bits, there exists a protocol compiler that, given κ (PRG seed length), s (statistical security parameter) and a functionality $\mathcal{F}(x, y)$ expressed as a circuit $\mathrm{C} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, outputs a protocol Π_{C} which securely realizes $\mathcal{F}$ in the κ-bit string OT-hybrid in the presence of WVD attacks on the receiver's input and the same features as in Theorem 9.2.*

Computational efficiency. In contrast to the concrete communication complexity, which is implementation independent, the concrete computation cost is sensitive to many implementation details. Although we have not implemented our protocol, we believe that it can be reasonably fast. The parties engage in $O(|\mathrm{C}|)$ instances of parallel OT execution which can be implemented efficiently via OT-extensions [IKNP03, KOS16]. Reconstructing the garbled circuit from the output of the parallel OTs relies only on simple bitwise XOR operation on bit strings. The computationally intensive part of the COT protocol is sharing several blocks of secrets via packed secret-sharing and evaluating polynomials by both the sender and the receiver. Since we instantiate our packed-secret sharing scheme over a large finite field, this can be done efficiently via Fast Fourier Transforms. An implementation of a similar FFT-based protocol is provided in [AHIV17]. Furthermore, the communication complexity of the COT protocol is significantly smaller than the overall

[12] 6800 is the size of the AES circuit excluding XOR gates.

communication (as can be seen in the calculations above). This allows trading a slight increase in the communication cost for more significant improvements in computational cost by using a larger number of FFTs on shorter blocks.

Incorporating Free XOR [KS08] and half gates [ZRE15] optimizations. The concretely efficient variant described is compatible with the standard garbled circuit optimizations.

In the Free XOR optimization, key-pairs for every wire are chosen so that they have a fixed difference. This allows to discard the XOR gates while garbling, where we can incorporate the same optimization for the garbled circuit in our case as well. Moreover, we recall that our concretely efficient variant implies that an encrypted MAC is transmitted for each garbled row along with the garbled circuit. Then we point out that the MACs need not be transmitted for XOR gates. This is because the MAC authenticates the color bits for the evaluator, where the color bit of the output wire of an XOR gate is a deterministic computation over the color bits of its input gates, namely, the XOR of the color bits. In other words, no additional authentication is required for the color bit of the output of an XOR gate.

In the half gate optimization, an AND gate is decomposed into two special types of gates which are further optimized via row-reduction. As before, this optimization can be incorporated for the garbled circuit in our concretely efficient variant. However, row reduction does not apply to the MACs. In other words, a MAC needs to be transmitted for rows that are "reduced". This means for every AND gate, 2 ciphertexts but 4 MACs are transmitted and the overhead of our communication becomes $(1+2s/\kappa)$. This outperforms dual-execution for various parameter regimes, eg, $(s = 40, \kappa = 128)$, $(s = 60, \kappa = 128)$, and $(s = 80, \kappa = 256)$.

Non-interactive variant. With function independent preprocessing for generating random OTs between the sender and the receiver, we can make our protocol non-interactive in the sense of [IKO+11], namely implement the protocol with one message from the receiver to the sender, followed by one message from the sender to the receiver. At a high-level, this is done by executing the passively secure information theoretic "Yao-style" protocol, followed by our commit and prove protocol. By OT preprocessing we can make the passively secure NISC/OT protocol a two-message protocol in the online phase. Our commit and prove protocol is public coin and can be made non-interactive via a standard Fiat-Shamir transform [FS87]. However, in the non-interactive case, the statistical security parameter s becomes a computational parameter, since a malicious sender can just try sampling 2^s instances of its message until finding one that would lead the receiver to accept a badly formed transcript. It is therefore needed in this case to use a larger value of s, say $s = 80$. A useful feature of non-interactive protocols is that the sender can use the same encrypted receiver input for multiple evaluations.[13]

Offline-online variant. Our protocol is particularly attractive in the offline-online setting. In an offline preprocessing phase, before the inputs are known, the sender and the receiver can run the entire protocol except for the oblivious transfers that depend on the receiver's input. Following this offline interaction, the receiver verifies that the information obtained from the sender is consistent, and can then "compress" this information into a single authenticated garbled circuit whose size is comparable to a standard garbled circuit. In an online phase, once the inputs are known, the receiver uses a small number of OTs to obtain the input keys, and performs garbled circuit evaluation and verification whose total cost is comparable to a single garbled circuit evaluation.

[13] As discussed in [IKO+11], the multiple evaluation setting is subject to selective failure attacks when the sender can learn the receiver's output in each evaluation.

10 Acknowledgments

We thank Peter Rindal, Mike Rosulek and Xiao Wang for helpful discussions and the anonymous TCC reviewers for their helpful comments.

The first author was supported by the European Research Council under the ERC consolidators grant agreement n. 615172 (HIPS), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. The second author was supported by a DARPA/ARL SAFEWARE award, DARPA Brandeis program under Contract N66001-15-C-4065, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, ERC grant 742754, NSF-BSF grant 2015782, ISF grant 1709/14, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of Google, the Department of Defense, the National Science Foundation, or the U.S. Government. The third author was supported by Google Faculty Research Grant and NSF Awards CNS-1526377 and CNS-1618884.

References

[ADI+17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO*, pages 223–254, 2017.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proc. ACM CCS 2017, to appear*, 2017.

[AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In *EUROCRYPT*, pages 387–404, 2014.

[BCR86] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *CRYPTO*, pages 234–238, 1986.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT*, pages 163–193, 2017.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS*, pages 784–796, 2012.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.

[cal] Calctopia. https://www.calctopia.com/.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536, 2006.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[CCX12] Ignacio Cascudo, Ronald Cramer, and Chaoping Xing. The arithmetic codex. In *IEEE Information Theory Workshop*, pages 75–79, 2012.

[CDI+18] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. *IACR Cryptology ePrint Archive*, 2018:940, 2018.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, pages 445–465, 2010.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, pages 554–563, 1994.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1987.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710, 1992.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[HIV17] Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Actively secure garbled circuits with constant communication overhead in the plain model. In *TCC*, pages 3–39, 2017.

[HKE12] Yan Huang, Jonathan Katz, and David Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE Symposium on Security and Privacy*, pages 272–284, 2012.

[HKE13] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *CRYPTO*, pages 18–35, 2013.

[HKK+14] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J. Malozemoff. Amortizing garbled circuits. In *CRYPTO*, pages 458–475, 2014.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

[IKL+13] Yuval Ishai, Eyal Kushilevitz, Xin Li, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and David Zuckerman. Robust pseudorandom generators. In *ICALP*, pages 576–588, 2013.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *EUROCRYPT*, pages 406–425, 2011.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30, 2007.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, pages 433–442, 2008.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[IMS12] Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge pcps. In *TCC*, pages 151–168, 2012.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *TCC*, pages 121–145, 2014.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. *IACR Cryptology ePrint Archive*, 2013:491, 2013.

[KMRR15] Vladimir Kolesnikov, Payman Mohassel, Ben Riva, and Mike Rosulek. Richer efficiency/security tradeoffs in 2pc. In *TCC*, pages 229–259, 2015.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *CRYPTO*, pages 724–741, 2015.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842, 2016.

[KRRW18] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In *CRYPTO*, pages 365–391, 2018.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, pages 486–498, 2008.

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

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.

[LP12] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, 2012.

[LPSY15] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In *CRYPTO*, pages 319–338, 2015.

[LR14] Yehuda Lindell and Ben Riva. Cut-and-choose Yao-based secure computation in the online/offline and batch settings. In *CRYPTO*, pages 476–494, 2014.

[MF06] Payman Mohassel and Matthew K. Franklin. Efficiency tradeoffs for malicious two-party computation. In *PKC*, pages 458–473, 2006.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *FOCS*, pages 136–145, 2003.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, pages 681–700, 2012.

[NO16] Jesper Buus Nielsen and Claudio Orlandi. Cross and clean: Amortized garbled circuits with constant overhead. In *TCC*, pages 582–603, 2016.

[NST16] Jesper Buus Nielsen, Thomas Schneider, and Roberto Trifiletti. Constant round maliciously secure 2pc with function-independent preprocessing using LEGO. *IACR Cryptology ePrint Archive*, 2016:1069, 2016.

[NST17] Jesper Buus Nielsen, Thomas Schneider, and Roberto Trifiletti. Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In *NDSS*, 2017.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[RR16] Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 297–314, 2016.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Spi95] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. In *STOC*, pages 388–397, 1995.

[SS13] Abhi Shelat and Chih-Hao Shen. Fast two-party secure computation with minimal assumptions. In *CCS*, pages 523–534, 2013.

[WMK17] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. Faster secure two-party computation in the single-execution setting. In *EUROCRYPT*, pages 399–424, 2017.

[WRK17] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure multi-party computation. In *Proc. ACM CCS, to appear*, 2017. Full version: Cryptology ePrint Archive, Report 2017/030.

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

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT*, pages 220–250, 2015.

A Comparison with the Conference Version of This Paper

The main difference between this full version and the earlier conference version [HIV17] is a correction to the protocol proving the main asymptotic result (Informal Theorem 1.1, Theorem 9.1). While the correction fixes the proof of the main theorem, this comes at the cost of poor concrete efficiency and higher asymptotic computational complexity.

To recover a concretely efficiency variant of the protocol, we need to relax the standard notion of security and settle for the weaker notion of "security with correlated abort." The current protocol can be seen as a leaner and more efficient variant of the concretely efficient protocol presented in [HIV17], which was incorrectly claimed to be fully secure. See Section 1.1 for a more detailed comparison between the current concretely efficient protocol and alternative protocols from the literature, including protocols based on the dual execution paradigm.

In what follows, we explain the problem and our fix. (See Section 1.2 for a more detailed overview.) The problem arises in Step 2 of the framework from Section 3 where we reduce the security to the certified OT (COT) functionality. As specified in the text, this reduction holds only in the presence of IVD attacks, where a malicious sender can cause a selective failure attack on the receiver's input. We recall that the inputs to the

original functionality are randomized using [Kil88, LP07, IKO+11] and are immune to these attacks, but the BMR randomization introduced in our functionality introduces additional input bits from the receiver and an input-dependent selective failure can result from these bits. In the conference version [HIV17] it is argued that an abort resulting from the IVD attack on these bits would be independent of the original inputs. However, since the receiver obtains circuit wire values that are masked with these random bits, leakage on the random bits leads to indirect leakage on the input.

In order to explain our fix, we recall the two instantiations for our NC^0 functionality. The first instantiation from Section 4.1 is used to obtain the asymptotic result, whereas the instantiation presented in Section 4.2 implies a concretely efficient protocol. We modify the former construction so that it achieves full security in the presence of malicious attacks, whereas our concrete result only satisfies the relaxed notion of security with correlated abort discussed above.

Specifically, we design a new COT protocol for NC^0 predicates where the receiver's input is encoded via an s-wise independent encoding to protect the receiver against against selective failure attacks. To make the communication overhead constant, this encoding is realized using algebraic geometric (AG) codes. This further requires that the receiver proves that its encoded input is well formed. To this end we define a new "doubly certified OT" functionality DCOT (see Section 6) to support the proof on the receiver's side, where a technique from [IKO+11] is used to directly ensure that the sender's input is well formed. We explain these steps in details in Section 5.

Due to the use of AG codes, the above approach is not concretely efficient. In the concretely efficient variant of our protocol we provide a leaner version of the protocol, which avoids the use of AG codes and only satisfies security with correlated abort. This protocol is leaner than the concretely efficient protocol from [HIV17] in that it avoids the use of the BMR randomization technique. We can in fact slightly strengthen the notion of security achieved by this protocol, showing that in the random oracle model the predicate that makes the receiver abort is of the wire-value disjunction (WVD) type, namely it can be expressed as a disjunction of circuit wires or their negations. See Section 4.2.1 for more details.