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

Carmit Hazay[1], Yuval Ishai[2], and Muthuramakrishnan Venkitasubramaniam[3]

[1] Bar-Ilan University
carmit.hazay@biu.ac.il,
[2] Technion and UCLA
yuvali@cs.technion.ac.il,
[3] University of Rochester
muthuv@cs.rochester.edu

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. The communication complexity of the second variant of our protocol can beat the best previous protocols 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.

1 Introduction

Secure two-party computation allows two parties to perform a distributed computation while protecting to the extent possible the secrecy of the inputs and the correctness of the outputs. The most practical approach to *constant-round* secure two-party computation is Yao's garbling paradigm [50]. It is convenient to describe Yao's protocol for the case of computing deterministic two-party functionalities, described 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[4] (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 [13] or alternatively require public-key operations for every gate in the circuit [28, 12], 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.[5] In the following, we will refer to a protocol that only makes a black-box use of a PRG (or a stronger "symmetric" primitive)[6] 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 [35] (following [39]) 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., [19, 43, 42, 48, 49] 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 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 [15, 34]. However, the multiplicative

[4] 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.

[5] 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 [17, 29].

[6] It is sometimes helpful to replace the PRG by a stronger symmetric primitive, such as symmetric encryption, a correlation-robust hash function [17], 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.

communication overhead[7] 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 $\mathsf{polylog}(s)$ [20, 19], 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 security in the presence of active adversaries, another line of works analyzed the *amortized* setting, where multiple evaluations of the same circuit are conducted by the two parties [38, 16, 45, 42]. In this setting, a recent work of Nielsen and Orlandi [42] 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 [49] 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.[8] 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 [49] inherits from [25, 7] 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-

[7] 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 $c(s) \cdot O(\kappa|\mathrm{C}|) + |\mathrm{C}|^{\epsilon} \cdot \mathsf{poly}(n, s, \kappa)$ for every constant $\epsilon > 0$.

[8] 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 [23, 5] or polynomial-stretch local PRGs [22]. 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 [17, 32].

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.

As a secondary goal, we will also be interested in minimizing the *computational overhead*. The computational cost of Yao's protocol is dominated by a constant number of PRG calls per gate.

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.

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.*[9]

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. In this variant we do not attempt to minimize the number of string OTs, but only attempt to minimize the complexity in the κ-bit string-OT hybrid. Efficient OT extension techniques [17, 29] can be used to get a similar complexity in the plain model. An additional advantage of the second variant is that it avoids the heavy machinery of linear-time encodable error-correcting codes and AG codes that are used in the first variant. Eliminating the use of AG codes makes the protocol realizable with polylogarithmic *computational* overhead compared to Yao's protocol (as opposed to $\Omega(s)$ computational overhead in the first variant, which is incurred by applying the encoding of an AG code on a message of length $\Omega(s)$).

Optimizing our second variant, we get better concrete communication complexity than the best previous protocols. For instance, for computational security $\kappa = 128$ and statistical security $s = 40$ and sufficiently large circuits, our multiplicative communication overhead is roughly 7 (compared to 40 in optimized cut-and-choose and roughly

[9] This assumption is needed for the existence of polynomial-stretch local s-wise PRGs. It is a mild assumption (arguably more so than standard cryptographic assumptions) that can be instantiated heuristically (see, e.g., [40, 22, 2]). One can dispense with this assumption by allowing $O(|\mathrm{C}|)$ OTs of κ-bit strings, or using a stronger symmetric primitive such as a correlation-robust hash function.

10 in [49]). Similarly to [49], our technique is compatible with the standard Free XOR garbled circuit optimization [33], but not with the "half-gate" optimization from [51]. Thus, for a fair comparison with the best passively secure cut-and-choose based protocols in the random oracle model, our overhead should be multiplied by 2. We leave the question of combining our technique with half-gate optimizations for future work.

For the case of evaluating a single AES circuit, the communication complexity of our optimized protocol is 3.39MB, roughly half of that of [49]. Our concrete efficiency advantages are even bigger when choosing bigger values of s. This is needed for the non-interactive setting discussed below. See Section 6.2 (and also Section 1.3) for a detailed concrete analysis of this variant of our construction and comparison with the concrete efficiency of other recent protocols.

We note that, similarly to the protocol from [49], the second variant of our protocol is particularly attractive in the offline-online setting. In this setting, the overhead of handling active adversaries is mainly restricted to an input-independent offline phase, where the concrete cost of the online phase is comparable to the passively secure variant. Moreover, the amount of data the receiver needs to store following the offline phase is comparable to a single garbled circuit. This should be contrasted with (single-instance) cut-and-choose based protocols, where only roughly half of the factor-s multiplicative communication overhead can be moved to an offline phase [38].

Another useful feature of our protocol is that, following a function-independent preprocessing, it can be made *non-interactive* in the sense of [20] 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. The protocol from [49] cannot be made non-interactive in this sense.

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 [20, 49]. To eliminate selective failure attacks by a malicious sender, we apply the multiparty circuit garbling technique of Beaver, Micali, and Rogaway (BMR) [3], which was used for a similar purpose in the two-party protocols of [37, 49]. 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 [20].

We now give a more detailed account of our techniques. Our starting point is the work of Ishai, Kushilevitz, Ostrovsky, Prabhakaran, and Sahai [20] (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 a functionality circuit $\mathcal{F}$ can be broken into the following three non-interactive reductions.

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 [20]: 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* (COT) oracle, namely a parallel OT oracle in which the receiver is assured that the pairs of transmitted strings (which also include strings it does not receive) satisfy some global 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 oracle, 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 [4] or decomposable randomized encoding [22] for $\widehat{\mathcal{F}}$.) (2) Use the COT oracle to enforce honest behavior of the sender.
3. **Reducing certified OT to parallel OT.** The third step is an information-theoretic protocol for COT using parallel OTs. This step is implemented using a variant of the "MPC-in-the-head" approach of [21], 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 COT 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 [31, 35, 20].

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 $\mathsf{polylog}(s)$ 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 [20] 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 [20, 19] 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 [37, 49], but using the first variant of $\widehat{\mathcal{F}}$ from [20]). This requires $\widehat{\mathcal{F}}$ to take $O(|\mathrm{C}|)$ random bits from the receiver, which results in the protocol using $O(|\mathrm{C}|)$ OTs of $O(\kappa)$-bit strings. To reduce the number of OTs, we use a local s-wise PRG [40] 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 [20] comes from the COT construction, which relies on perfectly secure honest-majority MPC protocols. The best known protocols of this type have a polylogarithmic communication overhead [9]. Our approach for reducing this overhead is to obtain an interactive variant of COT that can rely on any *statistically secure* honest-majority MPC protocol, and in particular on ones with constant communication overhead [8, 7, 26]. Our new COT protocol extends in a natural way the recent MPC-based zero-knowledge protocol from [1].

The first variant of our protocol uses the above COT protocol for a consistency predicate defined by Boolean circuits. As in [20], these boolean circuits employ information-theoretic MACs based on linear-time encodable codes [47]. To compute such a predicate with constant communication overhead, we rely on statistical honest-majority MPC based on algebraic geometric codes [7, 21]. 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; (2) applying COT for a predicate defined by an *arithmetic* circuit over a field of size $2^{O(s)}$. The latter change allows us to use simpler honest-majority MPC protocols for *arithmetic* circuits over large fields. Such protocols are simpler than their Boolean analogues and have better concrete efficiency (see [26], Appendix C, and [1]). Another advantage is polylogarithmic computational overhead. A disadvantage of this approach is that the corresponding functionality $\widehat{\mathcal{F}}$ does not allow us to use an s-wise PRG for reducing the number of OTs. As a result, the protocol requires $O(|\mathrm{C}|)$ OTs of $O(\kappa)$-bit strings.

1.3 Comparison with Wang et al. [49]

The recent results of [49] are the most relevant to our work. Like the second variant of our protocol, the protocol from [49] uses a combination of: (1) an "authenticated garbled circuit functionality" which is similar to the second variant from [20] 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 [20] 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 [49] is in how the NC^0 functionality $\widehat{\mathcal{F}}$ is securely realized against active adversaries. While the work of [49] uses a "GMW-style" interactive protocol for realizing $\widehat{\mathcal{F}}$, we rely on the non-interactive COT-based approach of IKOPS [20].

In slightly more detail, the protocol of [49] for evaluating $\widehat{\mathcal{F}}$ first creates a large number of authenticated "AND triples" using a variant of the "TinyOT" protocol [41]. 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, [49] also propose using a protocol from [25] to make the communication overhead constant, but this only holds in the bit-OT hybrid model that cannot be instantiated in the plain model 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 COT 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 COT.

Protocol	Total Comm.
[44]	15.12 MB
[49]	6.29 MB
This work	3.39 MB

Table 1. Total concrete communication cost of computing a single instance of AES circuit with $\kappa = 128$ and $s = 40$. The data about [49] and [44] was obtained from [49].

2 Preliminaries

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 [33], the size does not include XOR gates.

We use a standard notion of secure two-party computation in the standalone model, in the presence of static, adaptive corruptions. See Appendix A for details.

2.1 Local s-Wise PRGs

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 [40].

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. [19], 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 such an assumption, which is necessary for the existence of explicit polynomial-stretch local PRGs. See, e.g.,[22, 2] for further discussion.

2.2 Message Authentication Codes

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 $b \in \{0,1\}$.

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 [22, 20] 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 [47], 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.

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 2.1 (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.

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 [20], 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. In Section 4 we describe two implementations of this step that combine Yao-style garbling with BMR-style randomization. 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. 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 COT. 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 *certified oblivious transfer* (COT). The COT functionality $\mathcal{F}_{\mathrm{COT}}$ in its core performs the parallel OT functionality but additionally assures the receiver that the pairs of strings transmitted satisfy a global consistency predicate. 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 [20] as NISC/OT protocols, and can be based on any decomposable randomized encoding for $\widehat{\mathcal{F}}$ [18, 22] (which can also be viewed as a perfectly secure projective garbling scheme [50, 4] or a private simultaneous messages protocol [10] 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 [6].
2. Compile $\Pi_{1.5}$ into a protocol Π_2 in the $\mathcal{F}_{\mathrm{COT}}$-hybrid where the sender and receiver rely on the COT 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 $\Pi_{1.5}$. To make the COT predicate simpler, we allow it to be non-deterministic: the predicate depends on an additional NP-witness provided by the sender. The receiver 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 COT to commit-and-prove and parallel-OT. We obtain a constant-round protocol Π_3 for the COT 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 implementation of COT in this step deviates from the approach of [20] 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 [20].

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 [21]. Prior works [24, 27] 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 [1].

Input-dependent failures. A (standard) issue (also present in [20]) 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 as in [20] by precompiling $\mathcal{F}$ into another functionality $\mathcal{F}_{\text{IVD}}$ such that securely realizing $\mathcal{F}$ reduces to securely realizing $\mathcal{F}_{\text{IVD}}$ up to IVD attacks. Finally, for our concretely efficient protocol we will rely on efficient variants of this reduction from [35, 46] that increase the length of the receiver's input by a constant (≤ 4) factor and adds only XOR gates in $\mathcal{F}_{\text{IVD}}$ which can be handled efficiently via the Free XOR optimization [33]. An alternative implementation of $\mathcal{F}_{\text{IVD}}$ that increases the input length by only a sublinear amount is given in [20].

4 Secure 2PC in NC^0-Hybrid

In this section, we provide our compilation from an arbitrary 2PC functionality $\mathcal{F}$ to a 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 [20]. Previous two-party protocols essentially have a sender who creates and delivers a

[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.

garbling to a receiver. In contrast, we modify the constructions in [20] to incorporate additional randomization from the receiver inspired by the BMR approach [3].

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 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. When instantiated in the two-party setting, the BMR protocol differs from the standard Yao's protocol [50] in that the garbling is constructed jointly by both parties as opposed to just by one party in [50].

In slight more detail and restricting our discussion to the two-party setting, both parties provide two keys for every wire in the circuit so that the keys for the output wires of each gate are encrypted in each garbled row under both keys associated with the input wires of this gate. A key difference between the BMR and the Yao protocol is that 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. This allows both parties to evaluate the garbled circuit while maintaining privacy. One can model the offline phase as a actively secure computation of a "garbling" functionality where 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.

An important optimization considered in [37], observes that if the parties provide both the keys and PRG values under these keys as inputs in the offline phase, then the garbling can be computed via an NC^0 circuit over the inputs. However, such a functionality cannot guarantee the correctness of the PRG values provided by corrupted parties. Nevertheless, [37] show that to achieve security against active adversaries in the overall protocol, it suffices to securely compute the NC^0 functionality with active security. In other words, [37] demonstrate that bad PRG values provided by corrupted parties do not affect the correctness or privacy in the overall protocol. The key idea here is that a bad PRG value can at most trigger an abort when the particular wire associated with the PRG value assumes some particular value in the computation. However, since the associations of keys to values are randomized by both parties, the event of such an abort will be independent of the true value associated with that wire. Using an induction argument, it is possible to establish an invariant that if the evaluation does not abort due to a bad PRG value then the computation will be correct because the protocol guarantees correctness of the NC^0 computation. Combining the invariant with the key idea implies that the event of an abort is independent of of the actual honest parties' inputs (as opposed to just the particular intermediate wire value) thereby guaranteeing privacy. Finally, [37] demonstrate that if the evaluation path in the garbled circuit never hits a bad PRG value, then the computation will be correct.

The IKOPS protocol [20], 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 [35, 36], 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 effects of such an attack by introducing two variants of

functionalities. In the first variant, the NC^0 functionality authenticates the PRG values that are 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 continue to have a sender that provides the keys and PRG values to the garbling functionality as in the IKOPS protocol, but will randomize the association of keys to values following the BMR approach.

4.1 Variant 1: Authenticating PRG Values

In our first variant the functionality authenticates the PRG values submitted by the sender for creating the garbling. Following [20], 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. We deviate from [20] by including additional randomization in the form of 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 $\nu(\cdot)$.

The $\mathbf{NC^0}$ Functionality $\widehat{\mathcal{F}} = \mathcal{F}_{\mathbf{AuthPRG}}$. In more details, in this variant the $\mathbf{NC^0}$ functionality $\mathcal{F}_{\mathrm{AuthPRG}}$ computes a BMR-style garbling for the functionality $\mathcal{F}$ that is expressed by a set of wires W and a set of gates G, where only the sender provides the keys and PRG values to be used for this generation. Namely, the functionality obtains the parties' respective inputs (x, y) to the function $\mathcal{F}$ and shares for masking $\{\lambda_w\}_{w \in W}$, as well as the PRG evaluations from the sender and the authenticated information from the receiver, and creates the garbling for all gates $g \in G$; the complete details can be found in Figure 1.

Protocol 1 (Protocol Π_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. Section 2.2) where* $\mathsf{ECC} = \{\mathsf{Enc}_{\mathsf{lin}} : \{0,1\}^{\kappa} \to \{0,1\}^{\ell_{\mathsf{in}}\kappa}\}$ *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,* $F_{k_a^0}(g,0), F_{k_a^0}(g,1), F_{k_a^1}(g,0), F_{k_a^1}(g,1), F_{k_b^0}(g,0), F_{k_b^0}(g,1), F_{k_b^1}(g,0), F_{k_b^1}(g,1)$*.*
- **The receiver's input to $\mathcal{F}_{\mathrm{AuthPRG}}$:**
  - *Input* y*.*
  - *A random seed* β *to an* s*-wise PRG* $\mathrm{G}_{\mathrm{sPRG}} : \{0,1\}^{\kappa} \mapsto \{0,1\}^{t}$*.*
  - *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.*

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

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 random seed β to an s-wise PRG $\mathrm{G}_{\mathrm{sPRG}} : \{0,1\}^\kappa \mapsto \{0,1\}^t$.
- 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 $\lambda_w = \lambda_w^S \oplus \lambda_w^R$, where λ_w^R is computed by choosing the w^{th} bit in $\mathrm{G}_{\mathrm{sPRG}}(\beta)$.
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:
   - $\{(R_g^{00}, R_g^{01}, R_g^{10}, R_g^{11})\}_{g \in G}$.
   - $(k_w^0||0) \oplus \Big((\lambda_w \oplus x_i) \wedge (k_w^1||1 \oplus k_w^0||0)\Big)$ for every pair (w,i) where the input wire w carries the i^{th} bit of x.
   - $(k_w^0||0) \oplus \Big((\lambda_w \oplus y_i) \wedge (k_w^1||1 \oplus k_w^0||0)\Big)$ 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.

Fig. 1. The offline functionality $\mathcal{F}_{\mathrm{AuthPRG}}$.

– *In addition, the sender encrypts the mask used to mask the MAC values and sends it to the receiver. Namely, it sends the ciphertext* $c^{g,r}_{w,b} = \mathsf{Enc}_{k^b_w}(\tau^{g,r}_{w,b,S}) = F_{k^b_w}(g||(2+r)) \oplus \tau^{g,r}_{w,b,S}$.
– **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 colors* Λ_a *and* Λ_b*.*

Base case: *For each input wire* $w \in W$*, the receiver obtains* $k_w||z_w$*, where the key is* k_w *and the color* Λ_w *is set to* z_w*.*

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 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*

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

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.*

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 4.2 (AuthPRG2 Compiler) *Suppose there exist explicit* s*-wise PRGs in* NC^0 *with* $O(s)$ *seed size and an arbitrary polynomial stretch. Then there exists a compiler* AuthPRG2 *that, given* κ *(PRG seed),* s *(statistical parameter),* ϵ *(statistical PRG 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* $O(|\mathrm{C}|^\epsilon + s)$ *that it chooses 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:* $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$.

Moreover, any active corruption of the sender induces a disjunctive predicate P *on the random bits* r *input by the receiver to* $\widehat{\mathcal{F}}$ *such that the receiver aborts whenever* P *on the receiver's input is satisfied.*

4.2 Variant 2: Authenticating Color Bits

In the second variant, the parties submit their inputs to an NC^0 functionality that computes the garbled circuit. In this 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 number of OTs in this protocol will be proportional to the circuit's size since the abort predicate cannot be viewed as a disjunctive function any longer. On the other hand, the main advantage of this variant will be that the NP relation between the OT inputs of the sender and the sender's inputs and randomness can be expressed by a constant-degree arithmetic circuit over a large field. As we rely on an MPC protocol to boost the security of the passive protocol to the active case by certifying that the OT inputs of the sender satisfy the NP relation, we can rely on efficient MPC protocols for arithmetic circuits over large fields. In the full version we prove the following Lemma.

Lemma 4.3 (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 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* $W + 2s$ *that it will chosen uniformly at random where* $W = |\mathrm{C}|$ *is the number of distinct*[12] *wires in the circuit.*
- 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 2.2). More concretely, the functionality obtains the parties' inputs (x, y) to the function $\mathcal{F}$ and masking $\{\lambda_w\}_{w \in W}$, as well as the PRG evaluations from the sender, and the authenticated information from the receiver, and creates the garbling for all gates $g \in G$; the complete details can be found in Figure 2.

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* $\{\mathsf{MAC}_{\mathrm{SK}} : \{0,1\} \to \{0,1\}^{s}\}_{\mathrm{SK} \in \{0,1\}^{2s}}$ *be an information theoretically secure MAC computable in* NC^0.

[12] Wires as part of a fan out from a gate are considered the same wire.

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^S.
- For every gate $g \in G$ with input wires a and b, the PRG values $F_{a,0}^{g,0}, F_{a,0}^{g,1}, F_{a,1}^{g,0}, F_{a,1}^{g,1}, F_{b,0}^{g,0}, F_{b,0}^{g,1}, F_{b,1}^{g,0}, F_{b,1}^{g,1}$.

The receiver's inputs to the functionality are:

- Input y.
- For every wire $w \in W$, a mask λ_w^R.
- Two strings $\sigma_0, \sigma_1 \leftarrow \{0,1\}^s$.

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 $g \in G$, compute the garbled table as follows:

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

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

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

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

3. 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.

Fig. 2. The offline functionality $\mathcal{F}_{\mathrm{AuthCol}}$.

- **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, 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 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.*

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.

Claim 4.4 *Let $\mathcal{F}$ a two-party functionality as above and assume that F is a PRG. Then Protocol 2 securely computes $\mathcal{F}$ in the $\mathcal{F}_{\mathrm{AuthCol}}$-hybrid.*

We can modify all our variants to incorporate (by now standard) optimization of Free XOR [33]. Implicit in this optimization is a mechanism that restricts the space of keys sampled for the wires.

5 Realizing $\mathcal{F}_{\mathrm{COT}}$ in the Presence of IVD Attacks

In this section, we design our protocol that securely realizes the COT functionality (cf. Figure 3) with security in the presence of active adversaries up to IVD-abort. On a high-level, we combine the MPC-in-the-head approach of [21] to "certify" the inputs to the OT executions. A similar approach was taken in the work of [20]. However, our approach significantly deviates from the previous approaches in the following way:

– In the [20] approach, the receiver obtains the output of the individual OTs by obtaining the view of the corresponding receivers in the MPC network. In our approach, the sender and receiver first engage in the OT protocol as in a normal OT execution

and later a "zero-knowledge" proof for the correctness of the values, that are transferred via the OT protocol, is provided. The main savings of our approach is in the communication complexity. In the [20] approach, the view of the receivers contain redundant information from each of the servers and we avoid this redundancy.

Functionality $\mathcal{F}_{\mathrm{COT-IVD}}$

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

1. Upon receiving input $(P, \{(s_1^0, s_1^1)\}_{j\in[m]}, w)$ from S where $s_j^b \in \{0,1\}^\kappa$ and $w \in \{0,1\}^{\mathsf{poly}(n,\kappa)}$ it checks if the predicate P is a disjunction of literals and records $(P, \bar{\mathbf{s}}, w)$ if it is a disjunction where $\bar{\mathbf{s}} = \{(s_1^0, s_1^1)\}_{j\in[m]}$.
2. Upon receiving $(u_1, \ldots, u_m)$ from R where $u_j \in \{0,1\}$, record $(\bar{\mathbf{u}})$ 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. If there is a record $(\bar{\mathbf{s}}, w)$ then it sends $(\{s_j^{u_j}\}_{j\in[m]}, \mathcal{R}(\bar{\mathbf{s}}, w))$ to R only if $P(\bar{\mathbf{u}}) \neq 1$ and $\perp$ if $P(\bar{\mathbf{u}}) = 1$.

Fig. 3. The certified oblivious transfer functionality with IVD.

We will describe our protocol Π_3 in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{CnP}})$-hybrid where $\mathcal{F}_{\mathrm{OT}}$ is the parallel OT functionality and $\mathcal{F}_{\mathrm{CnP}}$ is a slight variant 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 4 for the formal description.

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

Functionality $\mathcal{F}_{\mathrm{CnP}}$ communicates with sender S and receiver R, 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 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, forward H to S. Upon receiving y from S, check if there exists a record (z, w) that it received from S. Ignore if no such record exists. Otherwise, send 1 to R 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.

Fig. 4. The commit-and-prove functionality.

Beside employing functionalities $\mathcal{F}_{\mathrm{OT}}$ and $\mathcal{F}_{\mathrm{CnP}}$, our protocol uses a special-hiding information theoretic MAC that preserves the properties of privacy and unforgeability

in a way that enforces the sender to properly commit to its inputs; see Definition 2.1 for more details. More formally,

Protocol 3 (Protocol Π_3 for realizing functionality $\mathcal{F}_{\mathrm{COT-IVD}}$)

- **Inputs:** *The sender S_{COT}'s input is $\{(s_j^0, s_j^1)\}_{j\in[m]}$ and a witness w with respect to some NP relation $\mathcal{R}$, and the receiver R_{COT}'s input is $b_1, \ldots, b_m$.*
- **The protocol:**
  1. $S_{\mathrm{COT}} \overset{\mathcal{F}_{\mathrm{OT}}}{\longleftrightarrow} R_{\mathrm{COT}}$: *The parties engage in m oblivious transfers in parallel using $\mathcal{F}_{\mathrm{OT}}$ where S_{COT} uses $((s_j^0, r_j^0), (s_j^1, r_j^1))$ and R_{COT} uses b_j, as their respective inputs in the j^{th} $(j \in [m])$ oblivious transfer execution, where r_j^b is a sufficiently long string. (Looking ahead, this string will serve as the randomness for some MAC function H.)*
  2. $S_{\mathrm{COT}} \overset{\mathcal{F}_{\mathrm{CnP}}}{\longleftrightarrow} R_{\mathrm{COT}}$: *The sender commits to the witness $(\{(s_j^b, r_j^b)\}_{j\in[m],b\in\{0,1\}}, w)$ by sending it to the $\mathcal{F}_{\mathrm{CnP}}$ functionality.*
  3. $S_{\mathrm{COT}} \leftarrow R_{\mathrm{COT}}$*: The receiver chooses a random MAC key $H \leftarrow \mathcal{H}$ and sends it to the sender via functionality $\mathcal{F}_{\mathrm{CnP}}$.*
  4. $S_{\mathrm{COT}} \rightarrow R_{\mathrm{COT}}$*: The sender sends the MAC of every string, namely it sends $\{H(s_j^b; r_j^b)\}_{j\in[m],b\in\{0,1\}}$ to R_{COT}. 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_{COT} rejects.*
  5. $S_{\mathrm{COT}} \overset{\mathcal{F}_{\mathrm{CnP}}}{\longleftrightarrow} R_{\mathrm{COT}}$: *The sender and receiver interact via the $\mathcal{F}_{\mathrm{CnP}}$ functionality where S_{COT} submits $\{H(s_j^b; r_j^b)\}_{j\in[m],b\in\{0,1\}}$ and R_{COT} submits H. $\mathcal{F}_{\mathrm{CnP}}$ 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_{COT} and otherwise 0.*

Since we can only realize a relaxed functionality $\mathcal{F}_{\mathrm{COT-IVD}}$ that allows IVD attacks, we need to understand how the attack propagates into the protocol for $\mathcal{F}$ (the original functionality that the parties want to compute) in the $\mathcal{F}_{\mathrm{COT}}$-hybrid. The key point is that the receiver's inputs to $\mathcal{F}_{\mathrm{COT}}$ in the latter protocol consist of either actual inputs y for $\mathcal{F}$ or independently random bits (for the BMR masking and MAC keys). Thus, any disjunctive predicate on the receiver's inputs to $\mathcal{F}_{\mathrm{COT}}$ can be emulated by a (randomized) disjunctive predicate on the receiver's inputs y to $\mathcal{F}$.

Theorem 5.1. *Let $\mathcal{H}$ be a family of special-hiding MAC according to Definition 2.1 for κ-bit strings. Then Protocol 3 securely computes $\mathcal{F}_{\mathrm{COT-IVD}}$ in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{CnP}})$-hybrid.*

6 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.

6.1 Variant 1: Asymptotically Optimal Construction

The first variant incurs communication complexity of $O(|\mathrm{C}|\kappa)$ bits in the κ-bit string OT oracle. We first provide a basic result for this variant that will employ $O(|\mathrm{C}|)$ calls to κ-bit string OT oracle. 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 6.1. *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 $O(|\mathrm{C}| + s)$ calls to κ-bit string OT oracle.*
- *It communicates $O(\kappa \cdot |\mathrm{C}|) + \mathsf{poly}(\log(|\mathrm{C}|), \log \kappa, s)$ bits.*

Remark 6.1. 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 6.1. We follow the framework described in Section 3.

1. Following an approach based on [35], we first transform the original functionality $\mathcal{F}$ into a new functionality $\mathcal{F}_{\mathrm{IVD}}$ that will resist input-dependent attack.
   - The circuit size of $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ is $O(\kappa \cdot |\mathrm{C}| + \kappa \cdot s)$ for any circuit C that computes the original functionality $\mathcal{F}$.
   - The receiver's input length in $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ is $O(|\mathrm{C}|) + O(\max(n, s)) = O(|\mathrm{C}| + s)$.
2. We next consider an information-theoretic protocol Π_2 that realizes $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ in the $\mathcal{F}_{\mathrm{COT-IVD}}$-hybrid (where functionality $\mathcal{F}_{\mathrm{COT-IVD}}$ is defined in Section 5). Such a protocol is obtained in two steps: (1) First, we take a non-interactive protocol $\Pi_{1.5}$ for $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ using a standard parallel OT oracle, where this protocol only needs to

be secure in the presence of a passive sender and an active receiver. (which can also be viewed as a perfectly secure projective garbling scheme [50, 4] or a private simultaneous messages protocol [10] with 1-bit inputs. See next variant for more details.) (2) We then use the $\mathcal{F}_{\mathrm{COT-IVD}}$ oracle to enforce honest behavior of the sender up to IVD attacks.

This protocol Π_2 has the following features:

- The receiver's input size is $O(|\mathrm{C}| + s)$.
- The sender's algorithm makes $O(|\mathrm{C}| + s)$ black-box calls to a length-doubling PRG $\mathrm{G}_{\mathrm{PRG}} : \{0,1\}^{\kappa} \to \{0,1\}^{2\kappa}$.
- The total length of the sender's OT inputs across all OTs is $O(\kappa \cdot |\mathrm{C}| + \kappa \cdot s)$.

We remark that we only track the number of OTs and the sum total of the lengths of the sender OT inputs as we can 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 $\lceil \frac{\ell_{\mathrm{OT}}}{\kappa} \rceil$ calls to the underlying length doubling PRG $\mathrm{G}_{\mathrm{PRG}}$ 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 replace the oracle call to the $\mathcal{F}_{\mathrm{COT-IVD}}$ in Π_2 by replacing it with the protocol Π_3 from Section 5 in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{CnP}})$-hybrid. Then we replace the oracle call to $\mathcal{F}_{\mathrm{CnP}}$ with our protocol Π_4 where we instantiate our MPC protocol using a variant of the protocol from [8] further used in [21]. The resulting protocol is in the $\mathcal{F}_{\mathrm{OT}}$-hybrid and realizes $\mathcal{F}_{\mathrm{COT-IVD}}$ against active adversaries. The communication complexity of the protocol can be computed as follows:
   (a) The sender and receiver first engage in parallel OTs where they execute only the oblivious-transfer part of the COT protocol. This involves $O(|\mathrm{C}| + s)$ inputs from the receiver and the sum total of the lengths of the sender's OT inputs across all OTs is $O(\kappa \cdot |\mathrm{C}| + \kappa \cdot s)$.
   (b) The sender transmits a MAC of length s corresponding to each OT input. There are totally $O(|\mathrm{C}| + s)$ strings transmitted via 1-out-of-2 OTs. Therefore, sending the MACs will require the sender to transmit $2 \cdot s \cdot O(|\mathrm{C}| + s) = O(s \cdot |\mathrm{C}| + s^2)$ bits.
   (c) The NP-relation associated with $\mathcal{F}_{\mathrm{CnP}}$ is of size $O(\kappa \cdot \mathrm{C} + \kappa \cdot s) + O(s \cdot |\mathrm{C}| + s^2) = O(\kappa \cdot \mathrm{C}) + \kappa \cdot \mathsf{poly}(s)$. We can conclude the communication complexity of the protocol realizing $\mathcal{F}_{\mathrm{CnP}}$ to be $O(\kappa \cdot \mathrm{C}) + \mathsf{poly}(\log |\mathrm{C}|, \log \kappa, s)$ and involves $O(|\mathrm{C}|) + \mathsf{poly}(\log |\mathrm{C}|, s)$ calls to the PRG.

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

This concludes the proof of Theorem 6.1.

Based on Theorem 6.1, we obtain the first construction of actively secure 2PC protocol that achieves constant overhead communication complexity over Yao's passively

secure protocol in a model where all parties have black-box access to any protocol realizing the OT oracle. In contrast, prior works based on the cut-and-choose paradigm induce a multiplicative overhead of $\Omega(s)$.

Next, we improve our construction from Theorem 6.1 to one that requires fewer calls to the OT oracle to something that is sublinear in the circuit size. This is obtained by replacing the compilation from $\mathcal{F}_{\text{IVD}}$ to $\widehat{\mathcal{F}}_{\text{IVD}}$ in Step 2 using Lemma 4.2. This compilation results in $\widehat{\mathcal{F}}_{\text{IVD}}$ where the receiver's input length is $n + O(s \cdot |\text{C}|^{\epsilon})$ assuming s-wise PRGs. Then we observe that the number of calls made to the OT in our final protocol is equal to the receiver's input length to $\widehat{\mathcal{F}}_{\text{IVD}}$. We thus get the following corollary.

Corollary 6.2 *Suppose there exist explicit s-wise PRGs in NC^0 with $O(s)$ seed size and an arbitrary polynomial stretch. Then, for every $\epsilon > 0$, there exists a protocol compiler that, given (κ, s) and a functionality $\mathcal{F}$ expressed as a Boolean circuit $\text{C} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, outputs a protocol Π_{C} that securely realizes $\mathcal{F}$ in in the κ-bit string OT hybrid with the following efficiency features:*

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

Remark 6.2. As discussed in Footnote 9 and Section 2.1, the combinatorial assumption about explicit s-wise PRGs is a seemingly mild assumption that was already used in other contexts.

This corollary 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, [49], albeit in the bit-OT hybrid model) or a small number of calls to the OT oracle (eg, protocols based on cut-and-choose).

6.2 Variant 2: Concretely Efficient Variant

Our second variant will also achieve a communication complexity of $O(\kappa \cdot |\text{C}|)$ bits and employ $O(|\text{C}|)$ 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 FreeXOR optimization.

More precisely, we have the following theorem:

Theorem 6.3. *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 $\text{C} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, outputs a protocol Π_{C} which securely realizes $\mathcal{F}$ in the κ-bit string OT-hybrid with the following features:*

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

$$(16 \cdot \kappa + 26 \cdot s) \cdot |\text{C}| + 2 \cdot s \cdot (|\text{C}| + 2 \cdot s + \max(4 \cdot n, 8 \cdot s)) \\ + 8 \cdot s^{1.5} \cdot \sqrt{|\text{C}| \cdot (55 \cdot \lceil \kappa/s \rceil + 6 \cdot \kappa + 73)}.$$

Proof of Theorem 6.3. 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 $\mathcal{F}_{\mathrm{AuthCol}}$-hybrid as described in Section 4.2 and the MPC protocol instantiated above. More precisely,

1. We transform the original functionality $\mathcal{F}$ into $\mathcal{F}_{\mathrm{IVD}}$ that is resistant to IVD attacks just as in the previous compilation. In our initial computation in this section, we ignore the additive overhead that is incurred as a result of this transformation. At the end of the section, we provide bounds for the additive terms. The circuit size of $\mathcal{F}_{\mathrm{IVD}}$ will therefore be $|\mathrm{C}|$ and the recipe input $|\mathrm{C}| + n + 2s$.
2. Next, we compile $\mathcal{F}_{\mathrm{IVD}}$ to $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ using $\mathcal{F}_{\mathrm{AuthCol}}$-hybrid as described in Section 4.2. The NC^0 functionality $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ has the following features:
   - The receiver's input size is $|\mathrm{C}| + 2 \cdot s + \max(4 \cdot n, 8 \cdot s)$ where $\max(4 \cdot n, 8 \cdot s)$ is the length of the encoding of the receiver's input following [35].
   - The output length of the NC^0 functionality is $4 \cdot |\mathrm{C}| \cdot (\kappa + s)$. Note that $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ includes an additional n^2 XOR gates compared to $\mathcal{F}_{\mathrm{IVD}}$. 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 FreeXOR optimization.

   We will compute the precise size in the next step.
3. We next consider an information-theoretic protocol Π that realizes $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ in the $\mathcal{F}_{\mathrm{COT-IVD}}$-hybrid. As before, 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 $\mathcal{F}_{\mathrm{COT-IVD}}$ oracle to enforce honest behavior of the sender up to IVD attacks.

   First, we compute the communication complexity of the passive protocol that realizes the NC^0 functionality in (1). Note that the computation of $\widehat{\mathcal{F}}_{\mathrm{IVD}}$ involves a computation with constant locality, in fact, at most 4 locality on the receiver's inputs. We recall from [6] that there is 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. Following this construction naively results in a total communication complexity of $2^{4+1} - 2 = 30$ per output bit for total of $30 \cdot 4 \cdot (\kappa + s) \cdot |\mathrm{C}|$ bits for computing $\widehat{\mathcal{F}}_{\mathrm{IVD}}$.

   We tighten the analysis in two ways:
   - First, we observe that each bit in the output of the NC^0 functionality we are computing can be expressed as a sum of monomials. This means we can break the monomials into different sections where the locality of each section is small. Then, we compute each section using the standard approach prescribed above. Additionaly, to ensure privacy we will have to mask the outputs each section with shares of 0.
   - Certain monomials (or sum of monomials) appear in multiple expression (for eg, the four garbled rows share monomials as we describe below) and we can compute the shared monomials only once.

   The general formula for computing the garbled row (r_1, r_2) in gate g with input wires a, b and output wire c is given by

$$F_{k_a^{r_1}}(g, r_1, r_1) \oplus F_{k_b^{r_2}}(g, r_1, r_2) \oplus [(\lambda_a^R \oplus \lambda_a^S \oplus r_1) \wedge (\lambda_b^R \oplus \lambda_b^S \oplus r_2) \oplus \lambda_c^R \oplus \lambda_c^A] \wedge (k_c^0 \oplus k_c^1)$$

Next, we consider the following monomials and explain how the first κ bits of the four rows of a garbled gate can be computed from them.

$$M_1 = [(\lambda_a^R \wedge \lambda_b^R) \oplus (\lambda_a^R \wedge \lambda_b^S) \oplus (\lambda_a^S \wedge \lambda_b^R) \oplus \lambda_c^A] \wedge (k_c^0 \oplus k_c^1) \oplus R_1$$
$$M_2 = [\lambda_a^R \wedge (k_c^0 \oplus k_c^1)] \oplus R_2$$
$$M_3 = [\lambda_b^R \wedge (k_c^0 \oplus k_c^1)] \oplus R_3$$
$$M_4 = [\lambda_c^R \wedge (k_c^0 \oplus k_c^1)] \oplus R_3$$

We will also have the sender send the receiver the following four strings (ciphertexts): For $r_1, r_2 \in \{0,1\}$

$$c_{g,r_1,r_2} = F_{k_a^{r_1}}(g, r_1, r_1, 0) \oplus F_{k_b^{r_2}}(g, r_1, r_2, 0) \oplus R_1 \oplus (r_1 \wedge R_2) \oplus (r_2 \wedge R_3) \oplus R_4$$

In the evaluation, if the receiver obtains $k_a^{r_1}$ and $k_b^{r_2}$, then it can obtain key for the c wire by computing

$$c_{g,r_1,r_2} \oplus M_1 \oplus (r_1 \wedge M_2) \oplus (r_2 \wedge M_3) \oplus M_4$$

By our preceding calculations, M_1 is a monomial over two variable λ_a^R, λ_b^R from the receiver and can be computed with overhead 6 per bit of the key. The other three monomials involve only one variable from the receiver and can be computed with overhead 2. Overall the communication of transmitting this will be $6 \cdot \kappa + (2+2+2) \cdot \kappa$ as part of the oblivious-transfer and $4 \cdot |\mathrm{C}| \cdot \kappa$ bits in the clear.
The next s bits which will encrypt the color bits can be computed analogously, where each of the terms above will additionally involve a multiplicand $\sigma_0 \oplus \sigma_1$ from the receiver. Following a similar analysis the number of bits transmitted will be $14 \cdot s + (6+6+6) \cdot s$ bits as part of the oblivious transfer and $4 \cdot |\mathrm{C}| \cdot s$ bits in the clear. We can improve this further because we can compress the sender's input to the OT when it is communicating strings that are long and chosen uniformly random. For example, in the OTs involving each bit of $\sigma_0 \oplus \sigma_1$ as receiver's input, the sender's input length is $O(|C|)$. This can be reduced to sending a PRG seed of length κ and the receiver expanding it to $O(|C|)$ bits. This reduces the cost to

$$2 \cdot \kappa \cdot s + 10 \cdot s + (4+4+4) \cdot s = 2 \cdot \kappa \cdot s + 22 \cdot s$$

Looking ahead, in our final protocol we will employ protocol Π directly as a sub-protocol. We will need two measures of complexity from this protocol. First, we need the communication complexity which we compute by calculating the receiver's input size (which translates to number of parallel OT invocations) and the sums of the lengths of the sender's inputs in all the parallel OT. Second, we estimate the size of the global predicate defined by the NISC/OT protocol which will dictate the complexity of our commit-and-prove protocol in the next step.
Following the calculations described above, we can conclude that this protocol incurs the following costs:

- The receiver's input size is $|\mathrm{C}| + 2s + \max(4 \cdot n, 8 \cdot s)$.

- The sum total of the sender OT inputs is $(12 \cdot \kappa + 22 \cdot s) \cdot |\mathrm{C}| + 2 \cdot \kappa \cdot s$ bits.
- The length of the sender's message is $4 \cdot (\kappa + s) \cdot |\mathrm{C}|$.
- The global predicate that will be the NP relation used in the $\mathcal{F}_{\mathrm{COT-IVD}}$ 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:

  Input size to NP relation. The witness to the NP statement includes (1) the strings for the OT, the sum of the lengths of inputs of which are $(12 \cdot \kappa + 22 \cdot s) \cdot |\mathrm{C}| + 2 \cdot \kappa \cdot s$, (2) The PRF values which totals to $4 \cdot |\mathrm{C}| \cdot (\kappa + s)$, and (3) for each wire w, $\lambda_w^R, k_w^0, k_w^1$ that sums up $|\mathrm{C}| + 2|\mathrm{C}| \cdot \kappa$.

  Key part of the output. Consider one of the garbled rows for a gate g with input wires a, b and output wire c. For very possible assignment (a_1, a_2, a_3) of $\lambda_a^R, \lambda_b^R, \lambda_c^R$ the first κ bits of a garbled row can be expressed as

  $$F_{\mathsf{prg}}^a + F_{\mathsf{prg}}^b + k_c^0 + f_{g,row}^{a_1,a_2,a_3}(\lambda_a^S, \lambda_b^S, \lambda_c^S) \cdot (k_c^1 - k_c^0)$$

  where $F_{\mathsf{prg}}^a, F_{\mathsf{prg}}^b, \lambda_a^S, \lambda_b^S, \lambda_c^S, k_c^0, k_c^1$ will be include in the witness for the predicate. The function $f_{g,row}^{a_1,a_2,a_3}$ can further be expressed as

  $$c_1 \cdot \lambda_a^S \cdot \lambda_b^S + c_2 \cdot \lambda_a^S + c_3 \cdot \lambda_b^S + c_4 \cdot \lambda_c^S + c_5$$

  for some coefficient c_1 through c_5 that will depend on the particular assignment for $\lambda_a^R, \lambda_b^R, \lambda_c^R$.

  Each garbled row can also be computed from the sender's OT inputs (again included in the witness) using only addition operations (this is exactly the computation of the receiver once it receivers the OT outputs). The predicate will check that the garbled row computed the two ways are equal.

  We only include the number of multiplication operations in our circuit size. It suffices to compute the product of each of $\lambda_a^S \cdot \lambda_b^S, \lambda_a^S, \lambda_b^S, \lambda_c^S$ with $(k_c^1 - k_c^0)$ to compute all of $f_{g,row}^{a_1 a_2 a_3}$ (where $a_1, a_2, a_3 \in \{0, 1\}$ and $row \in \{1, 2, 3, 4\}$). Since we first split k_c^0, k_c^1 into chunks of s bits, there will be $\lceil \frac{\kappa}{s} \rceil$ chunks and for each garbled row, the predicate will include $5 \cdot \lceil \frac{\kappa}{s} \rceil$ multiplications per gate.

  MAC part of the output. Again, for every combination of λ_w^R values, we compute the MAC part in two ways and check if they are equal. However, we will not do this check for every garbled row, we will do this for every column of the matrix where the MAC part of the garbled rows across all gates are stacked up. Furthermore, as we describe below, it will incur no additional multiplication gates.

  As before, for very possible assignment (a_1, a_2, a_3) of $\lambda_a^R, \lambda_b^R, \lambda_c^R$ corresponding to a gate, the last s bits (i.e. the MAC part) of a garbled row can be expressed as

  $$F_{\mathsf{prg}}^a + F_{\mathsf{prg}}^b + \sigma_0 + f_{g,row}^{a_1,a_2,a_3}(\lambda_a^S, \lambda_b^S, \lambda_c^S) \cdot (\sigma_1 - \sigma_0).$$

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^t bit of σ_0 and $\sigma_1 - \sigma_0$ are b_1 and b_2 respectively, the result will be $v^{b_1 b_2}_{g,row}$, where

$$v^{b_1 b_2}_{g,row} = F^a_{\mathsf{prg}} + F^b_{\mathsf{prg}} + b_1 + f^{a_1,a_2,a_3}_{g,row}(\lambda^S_a, \lambda^S_b, \lambda^S_c) \cdot b_2.$$

Since $f^{a_1,a_2,a_3}_{g,row}$ was already computed in the previous part, the values $v^{b_1 b_2}_{g,row}$ can be achieved with no additional multiplication gates. We further note that $v^{b_1 b_2}_{g,row}$ is independent of the position in the MAC part.

Again, we can compute $v^{b_1 b_2}_{g,row}$ using the sender OT input strings by using only an addition operation for each position in the MAC part. We can check for every position if this value matches the computation above from the witness. There is no additional cost for this part.

Binary constraints. The λ^S_w values need to be sampled from $\{0, 1\}$ and since we are operating in $\mathbb{GF}(2^s)$ we need enforce this constraint. This will require a single multiplication[13] per wire for a total of $|\mathrm{C}|$ multiplications. However, it will not affect the communication length and only the computations that need to performed.

Combining the above, we have a total of $|\mathrm{C}| \cdot 5 \cdot \lceil \frac{\kappa}{s} \rceil$ multiplications.

4. As in our previous compilation, we replace the oracle call to $\mathcal{F}_{\mathrm{COT-IVD}}$ in Π with the protocol Π_3 from Section 5 in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{CnP}})$-hybrid. We then replace the oracle call to $\mathcal{F}_{\mathrm{CnP}}$ with our protocol Π_4 in the $\mathcal{F}_{\mathrm{OT}}$-hybrid where we instantiate our MPC protocol using [1]. 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 $(12 \cdot \kappa + 22 \cdot s) \cdot |\mathrm{C}| + 2 \cdot \kappa \cdot s$ bits to the OT functionality and $4 \cdot (\kappa + s) \cdot |\mathrm{C}|$ 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 $2 \cdot (|\mathrm{C}| + 2 \cdot s + \max(4 \cdot n, 8 \cdot s))$ strings transmitted via OTs. Therefore, sending the MACs will require the sender to transmit $2 \cdot s \cdot (|\mathrm{C}| + 2 \cdot s + \max(4 \cdot n, 8 \cdot s))$ 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 $5 \cdot |\mathrm{C}| \cdot \lceil \frac{\kappa}{s} \rceil$ multiplications for verifying the OT inputs whereas the second part, verifying the MACs requires one multiplication per s bits of the OT sender inputs. This results in $\lceil \Gamma / s \rceil$ multiplications where

[13] We express this as $x^2 - x = 0$.

Γ is the total length of OT sender inputs. From the previous step, we know $\Gamma = (12 \cdot \kappa + 22 \cdot s) \cdot |\mathrm{C}| + 2 \cdot \kappa \cdot s$. In addition, as part of the witness, the PRF values that are used only for additions need to be included this sums up to $4 \cdot |\mathrm{C}| \cdot (\kappa + s)$. For an arithmetic circuit C we denote by $|\mathrm{C}|$ the number of multiplication gates. Our proof length is given by

$$\begin{aligned} &8 \cdot s^{1.5} \cdot \sqrt{4 \cdot |\mathrm{C}| \cdot (\kappa + s) + 3 \cdot |\mathrm{C}| \cdot (5 \cdot \lceil \kappa/s \rceil + 12\lceil \kappa/s \rceil + 22 + 2 \cdot \kappa \cdot s)} \\ &= 8 \cdot s^{1.5} \cdot \sqrt{|\mathrm{C}| \cdot (55 \cdot \lceil \kappa/s \rceil + 6 \cdot \kappa + 73)} \end{aligned}$$

Finally, the overall communication complexity of the protocol in the κ-bit string OT-hybrid for circuits with more than 5000 AND gates is

$$\underbrace{(12 \cdot \kappa + 22 \cdot s) \cdot |\mathrm{C}| + 4 \cdot (\kappa + s) \cdot |\mathrm{C}|}_{\text{passive NISC/ OT communication}} + \underbrace{2 \cdot s \cdot (|\mathrm{C}| + 2 \cdot s + \max(4 \cdot n, 8 \cdot s))}_{\text{MAC for every OT input}}$$
$$+ \underbrace{8 \cdot s^{1.5} \cdot \sqrt{|\mathrm{C}| \cdot (55 \cdot \lceil \kappa/s \rceil + 6 \cdot \kappa + 73)}}_{\text{CnP protocol}}.$$

In Table 2, we provide estimate communication cost incurred by our protocol. We set $\kappa = 128$ and $s = 40$ and 80. The communication cost for the OT invocations were computed assuming an implementation based on the actively secure OT extension protocol of [29] and can be bounded by $3 \cdot (\#OT) \cdot \kappa$. Furthermore, to accommodate arbitrary length strings for the sender's inputs the communication cost of OT is computed on κ'-bit strings from the sender, where $\kappa' = 128$ in practice, and longer strings are handled by transferring random keys and encrypting the bigger strings via the keys.

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 [17, 30]. 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 [1]. Furthermore, the communication complexity of the COT protocol is significantly smaller than the overall 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.

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 [20], namely implement the protocol with one message from the receiver

[13] 6800 is the size of the AES circuit excluding XOR gates.

Circuit Size	Comm.(MB) ($s = 40$)	Overhead ($s = 40$)	Comm.(MB) ($s = 80$)	Overhead ($s = 80$)
1024	0.71	11.33	1.07	17.16
2048	1.19	9.49	1.89	15.09
4096	2.12	8.46	3.16	12.65
6800 (AES)	3.39	8.16	4.80	11.56
8192	3.94	7.88	5.63	11.27
16384	7.53	7.53	10.46	10.46
32768	14.64	7.32	19.96	9.98
65536	28.76	7.19	38.74	9.69
131072	56.86	7.11	75.99	9.50
262144	112.85	7.05	150.05	9.38
524288	224.54	7.02	297.55	9.30
1048576	447.52	6.99	591.66	9.24
2097152	892.90	6.98	1178.65	9.21
4194304	1782.85	6.96	2350.87	9.18

Table 2. We give our total estimated communication cost of our concretely efficient variant where $\kappa = 128$ and $s = 40$ and 80. We also provide our overhead over the passively secure Yao protocol (with FreeXOR but no half-gate optimization).

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 [11]. 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.[14]

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.

[14] As discussed in [20], the multiple evaluation setting is subject to selective failure attacks when the sender can learn the receiver's output in each evaluation.

7 Acknowledgments

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

1. 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.
2. Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO*, pages 223–254, 2017.
3. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.
4. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS*, pages 784–796, 2012.
5. Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT*, pages 163–193, 2017.
6. Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *CRYPTO*, pages 234–238, 1986.
7. Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multiparty computations over small fields. In *CRYPTO*, pages 521–536, 2006.
8. Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.
9. Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, pages 445–465, 2010.
10. Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, pages 554–563, 1994.
11. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1987.
12. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

13. 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.
14. Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.
15. Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *CRYPTO*, pages 18–35, 2013.
16. Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J. Malozemoff. Amortizing garbled circuits. In *CRYPTO*, pages 458–475, 2014.
17. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.
18. Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.
19. Yuval Ishai, Eyal Kushilevitz, Xin Li, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and David Zuckerman. Robust pseudorandom generators. In *ICALP*, pages 576–588, 2013.
20. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *EUROCRYPT*, pages 406–425, 2011.
21. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30, 2007.
22. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, pages 433–442, 2008.
23. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
24. Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge pcps. In *TCC*, pages 151–168, 2012.
25. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.
26. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.
27. Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *TCC*, pages 121–145, 2014.
28. Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.
29. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *CRYPTO*, pages 724–741, 2015.
30. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. *IACR Cryptology ePrint Archive*, 2016:505, 2016.
31. Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
32. Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. *IACR Cryptology ePrint Archive*, 2013:491, 2013.
33. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, pages 486–498, 2008.
34. Yehuda Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. *J. Cryptology*, 29(2):456–490, 2016.
35. 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.
36. Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, 2012.
37. 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.

38. 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.
39. Payman Mohassel and Matthew K. Franklin. Efficiency tradeoffs for malicious two-party computation. In *PKC*, pages 458–473, 2006.
40. Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *FOCS*, pages 136–145, 2003.
41. 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.
42. Jesper Buus Nielsen and Claudio Orlandi. Cross and clean: Amortized garbled circuits with constant overhead. In *TCC*, pages 582–603, 2016.
43. 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.
44. Jesper Buus Nielsen, Thomas Schneider, and Roberto Trifiletti. Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In *24. Annual Network and Distributed System Security Symposium (NDSS'17)*. The Internet Society, February 26-March 1 2017.
45. 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.
46. Abhi Shelat and Chih-Hao Shen. Fast two-party secure computation with minimal assumptions. In *CCS*, pages 523–534, 2013.
47. Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. In *STOC*, pages 388–397, 1995.
48. Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. Faster secure two-party computation in the single-execution setting. In *EUROCRYPT*, pages 399–424, 2017.
49. 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.
50. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.
51. 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 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 [14], 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 implementation of certain cryptographic primitives such as ideal oblivious-transfer ($\mathcal{F}_{\mathrm{OT}}$) and we will denote such an execution by $\mathbf{REAL}^{\mathcal{F}_{\mathrm{OT}}}_{\Pi,\mathcal{A}(z),P_i}(x,y,\kappa,s)$.

Definition A.1. *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 A.1) 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 $1^\kappa, 1^s, \mathrm{C}$.