Privacy-Free Garbled Circuits for Formulas: Size Zero and Information-Theoretic

Yashvanth Kondi[1] and Arpita Patra[2]

[1] International Institute of Information Technology - Bangalore, India.
yashvanth.kondi@iiitb.org
[2] Indian Institute of Science, India. arpita@csa.iisc.ernet.in

Abstract. Garbled circuits are of central importance in cryptography, finding widespread application in secure computation, zero-knowledge (ZK) protocols, and verifiable outsourcing of computation to name a few. We are interested in a particular kind of garbling scheme, termed *privacy-free* in the literature. We show that Boolean formulas can be garbled *information-theoretically* in the privacy-free setting, producing *no* ciphertexts at all. Existing garbling schemes either rely on cryptographic assumptions (and thus require cryptographic operations to construct and evaluate garbled circuits), produce garbled circuits of non-zero size, or are restricted to low depth formulaic circuits. Our result has both theoretical and practical implications for garbled circuits as a primitive. On the theory front, our result breaks the known theoretical lower bound of one ciphertext for garbling an AND gate in this setting. As an interesting implication of producing size zero garbled circuits, our scheme scores adaptive security for free. On the practical side, our garbling scheme involves only cheap XOR operations and produces size zero garbled circuits. As a side result, we propose several interesting extensions of our scheme. Namely, we show how to garble threshold and high fan-in gates.
An aspect of our garbling scheme that we believe is of theoretical interest is that it does *not* maintain the invariant that the garbled circuit evaluator must not at any point be in possession of both keys of any wire in the garbled circuit.
Our scheme directly finds application in ZK protocols where the verification function of the language is representable by a formulaic circuit. Such examples include Boolean formula satisfiability. The ZK protocols obtained by plugging in our scheme in the known paradigm of building ZK protocols from garbled circuits offer better proof size, while relying on standard assumptions. Furthermore, the adaptivity of our garbling scheme allows us to cast our ZK protocols in the offline-online setting and offload circuit dependent communication and computation to the offline phase. As a result, the online phase enjoys communication and computation (in terms of number of symmetric key operations) complexity that are linearly proportional to the witness size alone.

Keywords: Garbled Circuits, Privacy-free, Formula, Information-theoretic, Authenticity

1 Introduction

Garbled circuits (GC) are of paramount importance in cryptographic protocol theory, lending their power in building vital cryptographic primitives such as secure computation in two party [Yao86, LP07, LP11, LR15] and multiparty [BELO16, CKMZ14, MRZ15] settings, zero-knowledge protocols [JKO13, FNO15, ZRE15], verifiable outsourcing of computation [GGP10], and functional encryption [SS10] to name a few. Roughly speaking, a GC allows evaluation of a circuit in its encoded form on an encoded input, and produces an encoded output. Based on the application that a GC serves, the information required to decode the output may be provided to the evaluator, or retained by the GC constructor if she wishes to keep the function output private. GCs first made their appearance in Yao's secure two-party computation protocol [Yao86]. Following multiple optimizations [BMR90, MNPS04, NPS99, PSSW09, BHKR13, GLNP15, ZRE15, KMR14, KS08], GCs today are an indispensable primitive used in various secure protocols. Their theoretical importance and potential to serve as a cryptographic primitive has been recognized by Bellare *et al.* [BHR12b], who elevate GCs from a technique to be used in other protocols, to a cryptographic primitive. To facilitate abstraction as a primitive, the fundamental work of Bellare *et al.* [BHR12b] formalizes three notions of security that a garbling scheme may achieve; namely privacy, obliviousness, and authenticity, and shows separation between them. Informally, privacy aims to protect the privacy of encrypted inputs, while obliviousness hides both the input and the output when the output decoding information is withheld. However once the output decoding information is revealed, obliviousness does not necessarily imply privacy of inputs. Lastly, authenticity captures the unforgeability of the output of a garbled circuit evaluation. Different applications of GC often use different combinations of the above properties of garbling schemes. Majority of the schemes in the literature, including the classical scheme of Yao [Yao86], satisfy all the three aforementioned properties.

In the original scheme of Yao [Yao86], each wire in the GC was assigned two strings called "keys", each corresponding to bit values zero and one on that wire. A garbled gate in the circuit was represented by ciphertexts encrypting its output wire keys using the corresponding input wire keys as per the gate's truth table. A garbled gate for a gate with fan-in two is thus constituted of four ciphertexts. An evaluator who knows one key for each input wire can only open one of the ciphertexts and therefore obtain only one key for the gate output wire, corresponding to the bit output of the gate. The final garbled circuit was a composition of the garbled gates, and its size was defined as the number of bits of ciphertext needed overall. The encoded input consisted of the keys on the input wires corresponding to the input bits. On receiving an encoded input, an evaluator evaluates the gates topologically, finding the output key for every gate, and stops when the keys for the output gates are obtained. The efficiency of a GC is determined by the computation cost (for the constructor and the evaluator) and its size. The latter directly impacts the communication complexity of protocols that employ the GC. Towards making secure computation practically efficient, tremendous efforts have been made in boosting the performance and efficiency of GCs. Some of the outstanding lines of work are highlighted below.

The work of the evaluator is significantly cut down via a technique called point-and-permute [BMR90, MNPS04]. Specifically, it cuts down the computation cost of an evaluator to one quarter, by introducing a pointing mechanism for every gate that imparts to the evaluator the knowledge of the particular ciphertext that she needs to decrypt in order to evaluate a garbled gate. Put differently, an evaluator simply decrypts the relevant ciphertext, skipping the remaining three for a two-input gate. Next, the celebrated Free-XOR technique [KS08] shows a simple yet brilliant way of garbling and evaluating XOR gates with zero ciphertexts and no cryptographic operations. Garbled Row Reduction (GRR) techniques [NPS99, PSSW09, GLNP15, ZRE15] are devoted towards making concise garbled gates by fixing some of the ciphertexts to constant values (therefore removing the need to transmit them). Both free-XOR and GRR techniques are instrumental in reducing the size of GCs. To date, the best known garbling scheme can garble an XOR gate with zero ciphertexts, and an AND gate with just two ciphertexts using free-XOR and clever GRR techniques [ZRE15]. Precluding further improvement in this domain, the work of [ZRE15] shows optimality of two ciphertexts (or 2μ bits; μ is the computational security parameter) for garbling an AND gate. Specifically, the lower bound holds true for any scheme that is captured by their characterization of linear garbling techniques. Informally, a garbling scheme qualifies to be linear when the circuit constructor and evaluator need to perform linear operations apart from making random oracle calls. Several other techniques for improving the computation cost of the constructor and evaluator are reported in [BHR12b, BHKR13]. The efficiency study of GCs are further enriched and extended by considering a number of interesting relaxations that lead to further optimizations. In one, some of the security properties of GCs are compromised. In the others, specific classes of circuits are used for garbling. As we discuss below, these relaxed notions of GCs are not only interesting from an application perspective, but they also show significant savings in terms of both size and computation cost. Since our work makes further inroad in the study of the GCs exploiting some relaxations, we take a detailed look at the relevant literature in order to set the stage for our contribution.

Privacy-free Garbling. In a breakthrough result, Jawurek *et al.* [JKO13] show that efficient zero knowledge (ZK) protocols for non-algebraic languages can be constructed based on garbling schemes achieving only authenticity. Frederiksen *et al.* [FNO15] termed this class of garbling schemes as *privacy-free*. There has since been significant interest in garbled circuit based ZK protocols [CGM16, HMR15]. A privacy-free garbling scheme does not need to satisfy privacy nor obliviousness, instead it only requires authenticity and a notion of *verifiability*. Informally, verifiability ensures that even a malicious constructor cannot create a garbled circuit that can be evaluated to different garbled output values, for inputs which when applied to the circuit in clear give the same output. This property is needed to mitigate selective failure attacks that a malicious verifier could possibly mount in a ZK protocol. Though as of writing this paper, the primary motivation of work in privacy-free garbling is to plug into the GC based ZK protocols which can prove that a 'prover' knows x such that $f(x) = 1$ in zero knowledge efficiently for non-algebraic f, verifiable outsourcing of computation provides another potential application for privacy-free GCs [BHR12b, BHR12a]. Motivated by the important use-cases of privacy-free garbling schemes, [FNO15] and [ZRE15] study the efficiency of privacy-free garbling. Both works show that more efficient GCs than the most optimized Yao's GC can be constructed by leveraging privacy-freeness. In terms of individual gate garbling computation and communication cost in the privacy-free setting, the Half Gates approach [ZRE15] which is currently the most efficient, requires one ciphertext per garbled AND gate, and no ciphertexts to garble XOR gates (with two calls to a hash function H per AND gate). Zahur *et al.* [ZRE15] also argue a lower bound of one ciphertext (or μ bits; μ is the computational security parameter) required to garble an AND gate for any linear scheme, and conclude optimality of their privacy-free construction.

Garbling for Formulaic Circuits. Formulaic circuits or formulas, informally, are circuits comprised solely of gates with fan-out of one. Formulaic circuits have several use-cases, such as Boolean formula satisfiability and membership in a language to name a few. By Cook's theorem, there exists a Boolean formula of size polynomial in $|x|$ that can verify an NP-witness of membership of x in language L. Formula satisfiability and language membership are well studied languages in the study of ZK protocols [CD97, KR06, Gro10, Lip12, GGPR13]. There are examples abound showing that treating Boolean formulas as a separate case from a general circuit may be apt [Kol05,KR06,KKKS15]. In the context of garbling, Kempka *et al.* [KKKS15] show how to garble a formulaic circuit with just four bits to represent each garbled gate. In contrast, even the best known garbling scheme for general circuits [ZRE15] needs $\mathcal{O}(\mu)$ bits where μ denotes computational security parameter. However, the garbling scheme of Kempka *et al.* [KKKS15] requires expensive public-key operations (which also disqualifies it from being a linear scheme). In yet another attempt, Kolesnikov [Kol05] shows how to garble a formula information-theoretically under the umbrella of "Gate Evaluation Secret Sharing", or *GESS*. The underlying garbling scheme achieves privacy (and though not explicitly proven or defined, authenticity) using only information-theoretic operations and produces a GC of size *zero*. On the down side, the keys associated with the wires have their length dependent on the depth of the circuit. Specifically, for a circuit of depth d and a statistical security parameter κ, a key on an input wire can be of size $\mathcal{O}(d(\kappa+d))$. Thus the input circuit needs to be of low depth, apart from being formulaic. The blow-up in key size also means that it does not meet the requirement of linearity as per [ZRE15]. Information-theoretic schemes are attractive in practice due to their highly efficient computation cost.

The schemes reported in [FNO15,ZRE15,KKKS15] are neither information-theoretic nor do they produce size-zero GCs. On the other hand, while the scheme of [Kol05] produces size-zero GCs, it is restricted to low-depth formulaic circuits. This leaves open the question of achieving best of the both worlds and sets the stage for our contribution.

1.1 Our Contribution

In this work, we explore privacy-free garbling for formulas (of arbitrary depth). Our findings are presented below.

Privacy-free garbling for formulas with size-zero GCs and information-theoretic security. The main contribution in this paper is to present a privacy-free garbling scheme for formulas of arbitrary depth that achieves information-theoretic security, size-zero GCs, and circuit-depth independence for the keys. Unlike in the information-theoretic scheme of [Kol05], the key length for the wires in our scheme is independent of the circuit depth. Unlike the schemes of [FNO15, ZRE15, KKKS15], ours is information-theoretic and is extremely fast due to the usage of cheap XOR operations. A couple of interesting theoretical implications of our result are given below.

– ***Breaking the lower bound of [ZRE15].*** The proven lower bound on the number of ciphertexts (bits) for garbling an AND gate is one (μ bits; μ is a security parameter) as per any linear garbling scheme. We show that our scheme is linear and yet requires *no* ciphertext at all to garble any gate. This breaks the lower bound shown in [ZRE15] for linear garbling schemes in the privacy-free setting.

– ***Achieving Adaptive Security for Free.*** A garbling scheme is said to achieve static security if its security properties are guaranteed as long as the choice of input to the circuit is not allowed to depend on the garbled circuit itself. A scheme is adaptively secure when there is no such restriction. Several applications, notably one-time programs [GKR08], secure outsourcing [GGP10], and ZK protocols cast in offline-online setting [KR06] need adaptive security, where the input may depend on the garbled circuit. An interesting implication of size-zero GC is that, in the terminology of Bellare *et al.* [BHR12b, BHR12a] achieving static security for our construction is equivalent to achieving adaptive security[3].

Several works confirm that privacy-freeness brings along efficiency both in terms of size and computation complexity of garbled circuits. We reaffirm this belief. Specifically, garbling an XOR gate requires three XOR operations (which can be improved in the multi-fan-in setting), while garbling an AND gate requires only one XOR operation. Evaluating any gate requires at most one XOR operation. Interestingly, contrary to the norm in secure multiparty computation, AND gates are handled (garbled, evaluated, and verified) more efficiently than XOR gates in our construction. Furthermore, our scheme requires only one κ-bit random string to generate all keys for both incoming wires to an AND or XOR gate. In Table 1, we compare our work against other schemes operating in the privacy-free setting to garble formulaic circuits, namely with [FNO15, ZRE15, Kol05]. The performance is measured with respect to a statistical security parameter κ for the information-theoretic constructions and with respect to a computational security parameter μ for cryptographic constructions, for a formulaic circuit of depth d. For most of the practical purposes, the value of μ can be taken as 128, while the value of κ can be taken as 40. Apart from usual measures (size and computation cost) of GCs, we also take into account the input key length of the GCs resulted from various schemes for comparison. The input key length impacts the communication required in the input encoding phase, which is frequently done by expensive Oblivious Transfer (OT) [EGL85] instances.

Technically, our scheme is very simple. We garble "upwards" from the output wire similar to the garbling schemes of [Kol05, KKKS15]. As with many secure computation protocols, at the heart of our scheme is our method for handling AND gates. Here, we provide a preview of how our scheme garbles an AND gate g. Denote the keys corresponding to bit b on the left and right incoming wires, and the gate output wire, as L^b, R^b, and K^b respectively. Our garbling scheme proceeds as follows. L^1 and R^1 are defined as additive shares of K^1 so that $L^1 \oplus R^1 = K^1$. Therefore, an evaluator can derive $K^1 = L^1 \oplus R^1$ only if she has both L^1 and R^1. We then copy the value of K^0 to the zero keys of both incoming wires; $L^0 = R^0 = K^0$. An evaluator hence has the output key corresponding to bit value zero if she has a zero key on either incoming

[3] Specifically, our scheme achieves `aut1` security in the terminology of [BHR12a]

Table 1: Performance and Security comparison of various Privacy-free garbling schemes for formulaic circuits. Calls to H refers to the number of hash function invocations. μ and κ refer to the computational and the statistical parameter respectively. d is circuit depth.

Garbling Scheme	**Cost per gate**						**Input key size**	**Security**
	Size (in bits)		Computation (Calls to H)					
			Constructor		Evaluator			
	XOR	AND	XOR	AND	XOR	AND		
Row reduction (GRR1) [FNO15]	μ	μ	0	3	0	1	μ	Static Computational
freeXOR+GRR2 [FNO15]	0	2μ	0	3	0	1	μ	Static Computational
Half Gates [ZRE15]	0	μ	0	2	0	1	μ	Static Computational
GESS [Kol05]	0	0	0	0	0	0	$\mathcal{O}\left(d(\kappa + d)\right)$	Adaptive Unconditional
This work	**0**	**0**	**0**	**0**	**0**	**0**	$\boldsymbol{\kappa}$	**Adaptive Unconditional**

wire. Note that in the case that the left incoming wire has value 0 flowing on it, and the right incoming wire 1, an evaluator will effectively possess both keys R^0 and R^1 on the right incoming wire; R^1 obtained legitimately, and R^0 as it is equal to L^0. We show that our scheme tolerates the leakage of certain keys within the garbled circuit (both directly and indirectly due to the observation above), at no cost of security.

The above aspect of our scheme is of theoretical interest as we do not maintain the invariant that an evaluator is allowed to know only one key on each wire. Our scheme achieves authenticity despite conceding both keys to an evaluator on certain wires. In fact, this property is taken advantage of in order to gain much in terms of efficiency. To the best of our knowledge, ours is the first garbling scheme where this invariant is not maintained. A direct implication of violating this invariant is that the standard proof paradigms for garbled circuits (which assume the invariant to hold) are not applicable here. We exploit the fact that the only gate that is necessarily "uncompromised" is the circuit output gate, and reduce (with no security loss) the authenticity of the circuit output gate in the context of an arbitrarily large circuit, to the authenticity of a single-gate circuit.

Extensions for high fan-in gates and general circuits. To optimize our garbling scheme, we propose efficient garbling of ℓ-fan-in gates. Apart from handling ℓ-fan-in XOR and AND gates, we consider *threshold* gates and provide a new garbling scheme for them. A threshold gate with fan-in ℓ and threshold t with $\ell > t$ outputs 1 when at least $t+1$ inputs carry the bit 1, and zero otherwise. The threshold range $1 < t < \ell - 1$ is of interest to us, as the gate otherwise degenerates into an ℓ-fan-in AND or NAND gate, which can be handled more efficiently by our scheme. Boolean threshold gates are considered and motivated by Ball *et al.* [BMR16], who construct a scheme to garble them natively (generating $\mathcal{O}\left(\log^3 \ell / \log\log \ell\right)$ ciphertexts) as opposed to garbling a composition of AND, XOR and NOT gates (yielding $\mathcal{O}\left(\ell \log \ell\right)$ ciphertexts using the best known garbling scheme of [ZRE15]). Here, we present a method of garbling Boolean threshold gates (embedded in formulaic circuits) directly in privacy-free setting, producing no ciphertext, and using only information-theoretic operations; specifically two independent instances of Shamir secret sharing [Sha79] per threshold gate.

The power of threshold gates is brought out in the fact that $\mathsf{NC}^0 \subsetneq \mathsf{AC}^0 \subsetneq \mathsf{TC}^0$, where circuits deciding languages in TC^0 contain majority gates in addition to AND, OR and NOT. More practically, threshold gates implement natural expressions in the settings of zero-knowledge [JKO13] and attribute-based credentials [KKL$^+$16]. In the former case, threshold gates can implement statements of the form, "I have witnesses for at least t out of these ℓ statements", without revealing for which statements the prover has witnesses. In the case of attribute-based credentials, one can prove that her attributes satisfy at least t criteria out of ℓ in a policy, without revealing which ones, or how many exactly.

We show how to garble and evaluate ℓ fan-in XOR and AND gates with fewer XOR operations than are needed when we express such gates in terms of two fan-in XOR and AND gates respectively. Specifically, garbling an ℓ-fan-in XOR gate directly takes 2ℓ XOR operations, as opposed to $3(\ell - 1)$ XOR operations to garble $\ell - 1$ XOR gates individually. Evaluating an ℓ-fan-in AND gate, in $2^\ell - 1$ cases out of 2^ℓ, will take zero XOR operations. In the final case, the evaluation is done at the same cost as evaluating $\ell - 1$ individual AND gates.

For completeness, we describe how to adapt our scheme to garble generic circuits in the privacy-free setting in the full version of the paper. While the adaptation itself is not generally efficient for circuits that are not largely formulaic, it establishes the feasibility of violating the single-key invariant when garbling any generic circuit, at least in the privacy-free setting. Our approach relies on cryptographic assumptions. For generic circuits that are not largely formulaic in nature, the construction of [FNO15, ZRE15] can be used. However, both the constructions rely on non-standard assumptions. In [FNO15], it is a customized notion of key derivation function (KDF) where random oracle can be shown to be a secure KDF. In [ZRE15], the construction needs a circular correlation robust hash function. We take a look at the scheme of [GLNP15] which works under standard pseudo-random function (PRF) assumption and propose several optimizations in privacy-free setting in the full version of the paper.

Application to ZK Protocols. Lately, ZK protocols from garbled circuits has gained a lot of momentum [JKO13, FNO15, CGM16], with applications such as attribute-based key exchange built on top of them [KKL$^+$16]. We apply our garbling scheme to the domain of ZK protocols where the verification function of the language is representable by an almost formulaic circuit such as Boolean formula satisfiability. When we plug in our scheme in the paradigm of [JKO13] (with a slight tweak), we get ZK protocols that rely on standard assumption (PRG) in the OT-hybrid model and results in a better proof size for right choice of the security parameters than the known instantiations in the same paradigm. The best known GC-based ZK instantiation that results from the composition of the privacy-free construction of [ZRE15] and the ZK protocols of [JKO13] needs to rely on KDF and circular correlation-robust hash function.

Leveraging the adaptivity of our garbling scheme, we cast our ZK protocols in the offline-online paradigm and offload circuit dependent expensive communication and computation to the offline phase. As a result, the witness size alone linearly impacts the communication and computation (in terms of number of symmetric key operations) complexities of the online phase. The existing ZK protocols relying on statically secure garbling schemes cannot match the online complexities of our protocol as the garbled circuit needs to be sent in the online phase. In contrast to the garbled circuit based ZK protocols (including ours) where public key operations are proportional to the witness size, the theoretically interesting ZK proofs/arguments [CD97, KR06, Gro10, Lip12, GGPR13] for satisfiability employ public key operations proportional to the circuit size. We focus on the protocols that are practically relevant. A practical non-interactive alternative can be found in ZKBoo [GOM16], however at the cost of a large proof size; the proof is linear in the size of the statement, and computed (and communicated) only after the witness is available. A comparison of our ZK protocol for Boolean formula satisfiability with [JKO13] instantiated in the offline-online paradigm with the state of the art privacy-free garbling scheme [ZRE15] is provided in Table 2.

Table 2: Complexities of GC based ZK for Boolean formula satisfiability. The last two rows correspond to the protocols in offline-online setting. The computational and statistical security parameters are μ and κ respectively, and the size of the statement is m, while the size of the witness is n.

Protocol	**Communication**		**Computation (Input encoding and GC evaluation)**	
	Offline	Online	Offline	Online
[JKO13] + [ZRE15]	0	$\mathcal{O}(\mu m + \kappa n)$	0	$\mathcal{O}(n)$ PKE + $\mathcal{O}(m)$ Hash invocations
Our protocol	0	$\boldsymbol{\mathcal{O}(\kappa m + \kappa n)}$	0	$\boldsymbol{\mathcal{O}(n)}$ **PKE +** $\boldsymbol{\mathcal{O}(m)}$ **XORs**
[JKO13] + [ZRE15] (Offline-online)	$\mathcal{O}(\mu n)$	$\mathcal{O}(\mu m)$	$\mathcal{O}(n)$ PKE	$\mathcal{O}(m)$ Hash invocations
Our protocol (Offline-online)	$\mathcal{O}(\kappa m)$	$\boldsymbol{\mathcal{O}(\mu n)}$	$\mathcal{O}(n)$ PKE	$\boldsymbol{\mathcal{O}(n)}$ **PRG invocations +** $\boldsymbol{\mathcal{O}(m)}$ **XORs**

1.2 Organization

In Section 2, we recall the necessary definitions. In Section 3, we present our privacy-free information-theoretic garbling scheme for formulas. The full proof of security appears in Section 4. The definition of a privacy-free linear garbling scheme and the proof that our scheme qualifies to be a linear scheme is presented in Section 5. We present the optimizations for ℓ fan-in gates in Section 6. Our ZK protocol appears in Section 7 and the required functionalities are recalled in Appendix A.

2 Preliminaries

We use $a \leftarrow \{0,1\}^n$ to denote that a is assigned a uniformly random n-bit string, and $a \leftarrow \mathsf{alg}(x)$ to denote that a is assigned the value output by randomized algorithm alg when supplied the input x. We use $b := a$ to denote that b is deterministically assigned the value a. The operator $a||b$ denotes the concatenation of a and b. PPT denotes probabilistic polynomial time. The value κ is used throughout this paper to denote the statistical security parameter, which is reflected in the key length of the instance of the garbling scheme. For all practical purposes, the value of κ can be taken as 40. We also use the terms "zero key" and "key corresponding to bit value zero" interchangeably. In what follows, we present the required definitions. We denote by $[x]$, the set of elements $\{1, \ldots, x\}$.

2.1 Formulaic Circuits

Informally, a formula is a circuit which has a fan-out of one for every gate. The implication of this is that a gate's output wire can either be a circuit output wire, or an input wire for only one other gate. Formally, we use a modified version of the syntax for circuits in [BHR12b]. In GC based ZK protocols [JKO13], the verification circuit that needs to be garbled has one bit output. The output zero indicates that the proof is rejected, whereas the output one indicates that the proof is accepted.

Definition 1. *A formulaic circuit is characterized by a tuple $f = (n, q, A, B, G)$. The parameters n, q define the number of input, and non-input wires respectively. Wires are indexed from 1 to $n + q$, with 1 to n being input wires, and $n + q$ being the output wire. A gate is identified by its outgoing wire index. For a gate $g \in [n + 1, n + q]$, $A(g)$ and $B(g)$ are* injective *functions that map to left and right incoming wire indices respectively*[4]*. We have $B(g) \in [1, n+q-1]$, and $A(g) \in [0, n+q-1]$; $A(g) = 0$ if g has fan-in of 1. We also require that $A(g) < B(g) < g$. Additionally, we require that for every gate g, if $\exists g', A(g') = g$, then $\nexists g'', B(g'') = g$, and vice versa. This is to ensure that a gate can be an incoming wire to at most one other gate. The gate functionality $G(g)$ is a map $G(g) : \{0,1\}^2 \mapsto \{0,1\}$.*

The terms "wire" and "gate" are used interchangeably throughout the paper, as a gate is identified by the index of its outgoing wire.

2.2 Privacy-Free Garbling Scheme

A garbling scheme, as defined in [BHR12b], is defined by a tuple $(\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev})$. Their arguments and outputs are as follows:

- Gb: $(f, 1^\kappa) \mapsto (F, e, d)$. Given the function f to garble, the PPT algorithm Gb outputs the garbled circuit F, encoding information e, and decoding information d.
- En: $(x, e) \mapsto X$. Given clear function input x and valid encoding information e, the deterministic algorithm En outputs garbled input X.
- Ev: $(F, X) \mapsto Y$. Given a garbled circuit F and garbled input X for that circuit, Ev deterministically outputs garbled output Y.
- De: $(Y, d) \mapsto y$. Given garbled output Y, and valid decoding information d, De deterministically outputs the clear function output y. If Y is not consistent with d, then De outputs $\perp$.

Definition 2 (Correctness). *A garbling scheme satisfies* correctness *if for every valid circuit f and its input x, we have*

$$\forall (F, e, d) \leftarrow \mathsf{Gb}\left(f, 1^\kappa\right), \ \mathsf{De}\left(\mathsf{Ev}\left(F, \mathsf{En}(x, e)\right), d\right) = f(x)$$

[4] This is a departure from the [BHR12b] definition for conventional circuits. The injection property required here ensures that every gate in the circuit has a fan-out of one.

We consider only *projective* garbling schemes, where the encoding information e is of the form $\left(\left(k_i^0, k_i^1\right)_{i\in[n]}\right)$. We refer the reader to [BHR12b] for a formal treatment and discussion.

In [BHR12b], definitions for the security notions of privacy, obliviousness, and authenticity are provided. However, as we are not interested in achieving privacy or obliviousness, we will only consider authenticity, and the notion of verifiability defined in [JKO13].

Definition 3 (Authenticity). *A garbling scheme satisfies* unconditional authenticity *if for every computationally unbounded $\mathcal{A}$, and for every $f : \{0,1\}^n \mapsto \{0,1\}$ and $x \leftarrow \{0,1\}^n$, we have*

$$\Pr\left[(Y \neq \mathsf{Ev}(F,X)) \ \wedge \ (\mathsf{De}(Y,d) \neq \bot) \ : \ Y \leftarrow \mathcal{A}\left(F,X\right)\right] \leq 2^{-\kappa}$$

where $(F,e,d) \leftarrow \mathsf{Gb}\left(f, 1^\kappa\right)$, and $X := \mathsf{En}(e,x)$.

The definition for unconditional authenticity in Definition 3 is stronger than that of Bellare *et al.* [BHR12b], as it places no bound on the computational power of the adversary, and specifies that no such adversary should be able to perform better than randomly guessing a garbled output. Intuitively, schemes delivering such guarantees should rely only on information theoretic operations.

Finally, we also consider the property of *verifiability* introduced in [JKO13]. A privacy-free garbling scheme that can be plugged into their ZK protocol must have an additional 'verification function' $\mathsf{Ve} : (F, f, e) \mapsto b$. The purpose of this function is to enable the Prover (who evaluates the garbled circuit) to verify that the garbled circuit that she was given was legitimately constructed, which is important in ensuring that the garbled output obtained upon evaluation doesn't reveal any input bits, ie. the Prover's witness. This function outputs a single bit b, given a garbled circuit F, the underlying clear function f, and encoding information e. Informally, when Ve outputs 1 for a certain F, f, e, then evaluating F on garbled input X corresponding to x such that $f(x) = 1$ will produce garbled output that matches the expected garbled output that can be extracted given F, e.

Definition 4 (Verifiability). *A verifiable garbling scheme contains a poly-time computable function* Ve *such that there exists a poly-time algorithm* Ext*, which for every computationally unbounded adversary $\mathcal{A}$, function f within the domain of* Gb*, input x where $f(x) = 1$, ensures the following,*

$$\Pr\left[\mathsf{Ext}(F,e) = \mathsf{Ev}\left(F, \mathsf{En}(e,x)\right)\right] = 1, \textit{ when } \mathsf{Ve}\left(F,f,e\right) = 1; \ (F,e) \leftarrow \mathcal{A}\left(1^\kappa, f\right)$$

For completeness, we require: $\forall (F,e,d) \leftarrow \mathsf{Gb}\left(f, 1^\kappa\right), \ \mathsf{Ve}\left(F,f,e\right) = 1$.

Note that like Definition 3 the above definition for verifiability is stronger than the original one in [JKO13], owing to the fact that our Definition 4 does not place a bound on the running time of the adversary, and does not permit even a negligible error for the Ext algorithm.

An unconditionally secure privacy-free garbling scheme is defined by a tuple $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, \mathsf{Ve})$, and satisfies the correctness, authenticity, and verifiability properties detailed in Definitions 2, 3, and 4.

3 Privacy-Free Garbling for Formulas

In this section, we define our construction for an unconditionally secure, verifiable privacy-free garbling scheme whose domain of circuits that can be garbled are formulaic. As per previous paradigms of garbling formulaic circuits in [Kol05, KKKS15], our garbling scheme proceeds upwards from the output wire.

3.1 Garbling Individual Gates

As per Yao's paradigm of garbling circuits [Yao86], every wire in the circuit is assigned two κ-bit string tokens, called "keys"; one each for bit values zero and one on that wire. For a gate g, let the output wire keys corresponding to zero and one be K^0 and K^1 respectively. The zero and one keys of the left incoming wire are L^0, L^1 respectively, and those of the right incoming wire are R^0, R^1 respectively. The bit value flowing on wire w is b_w. A gate garbling routine is a randomized algorithm that accepts the gate keys K^0, K^1 as arguments, and returns constructed keys L^0, L^1, R^0, R^1 for the gate's input wires. A gate evaluation routine deterministically returns a key $K^{G_g(b_L, b_R)}$ where G_g is the gate functionality, upon being supplied with input wire keys L^{b_L}, R^{b_R} (and possibly input bits b_L, b_R). In this section, we define gate garbling and evaluation routines for XOR, AND, and NOT gates.

Garbling XOR Gates. Garbling and evaluation of XOR gates is relatively simple. Our garbling scheme for XOR gates is similar to that of Kolesnikov's [Kol05]. The wire keys produced by our garbling scheme maintain the same relation, namely $L^{b_L} \oplus R^{b_R} = K^{b_L \oplus b_R}$. However, while the construction of [Kol05] requires four XOR operations to garble an XOR gate, our construction requires only three (tending to two in the l-fan-in setting), hence saving on computation cost.

First, K^0 is split into two additive shares, assigned to L^1 and R^1 respectively. Therefore, $L^1 \oplus R^1 = K^0$. Next, K^1 is masked with R^1 and assigned to L^0, and independently masked with L^1 and assigned to R^0. ie. $L^0 := K^1 \oplus R^1$ and $R^0 := K^1 \oplus L^1$. This ensures that $L^0 \oplus R^1 = R^0 \oplus L^1 = K^1$. Conveniently, $L^0 \oplus R^0 = L^1 \oplus R^1 = K^0$.

Evaluation can hence be defined as follows: if the evaluator has keys L^{b_L} and R^{b_R}, corresponding to bits b_L and b_R on the left and right wires respectively, she can obtain the output key as $K^{b_L \oplus b_R} = L^{b_L} \oplus R^{b_R}$. Correctness of evaluating an XOR gate as per this scheme is implicit.

The VeXOR routine defined in Fig. 1 ensures that any combination of L^{b_L}, R^{b_R} taken from L^0, L^1, R^0, R^1 consistently evaluates to a $K^{b_L \oplus b_R}$. This can be considered a "consistency check", that a given tuple of keys (L^0, L^1, R^0, R^1) maintain correctness of a garbled XOR gate.

Garbling AND Gates. Our construction for garbling AND gates is as simple as the one defined for XOR gates, however the proof of authenticity is not as straightforward. Interestingly, our scheme requires only one XOR operation to garble an AND gate, and at most one XOR operation to evaluate a garbled AND gate (in three out of four cases, evaluation is completely free). This makes garbling, evaluation, and verification

```
GbXOR $\left(K^0, K^1, 1^\kappa\right)$

The zero and one keys of the left and right incoming wires will be $L^0$, $R^0$ and $L^1$, $R^1$ respectively

1. Split $K^0$ into additive secret shares, $L^1 \leftarrow \{0,1\}^\kappa$;  $R^1 := K^0 \oplus L^1$
2. Mask $K^1$ for the incoming zero keys, $L^0 := K^1 \oplus R^1$; $R^0 := K^1 \oplus L^1$
3. return $L^0, L^1, R^0, R^1$

EvXOR $\left(L^{b_L}, R^{b_R}\right)$

1. return $L^{b_L} \oplus R^{b_R}$

VeXOR $\left(L^0, L^1, R^0, R^1\right)$

1. Generate both output keys in all combinations
    i. $K^{00} := L^0 \oplus R^0$; $K^{01} := L^0 \oplus R^1$
    ii. $K^{11} := L^1 \oplus R^1$; $K^{10} := L^1 \oplus R^0$
2. if $K^{00} \neq K^{11}$ or $K^{10} \neq K^{01}$ then the keys are inconsistent, return 0, $\perp$, $\perp$. else
   return 1, $K^{00}, K^{01}$
```

Fig. 1: Garbling, evaluation and verification of an XOR gate

of AND gates cheaper than that of XOR gates. Figure 2 formalizes the construction discussed in the Introduction.

Correctness of evaluating an AND gate as per this scheme is hence implicit. Note that if an evaluator has key L^0, she will be missing L^1, therefore making whatever key she has on the right incoming wire irrelevant; K^1 remains completely hidden unless both L^1 and R^1 are available. A similar argument applies in case she has R^0. Additionally, if she is able to derive K^1 during evaluation, it implies that she started with L^1 and R^1, keeping K^0 inaccessible for the lack of L^0 and R^0. Therefore, during an evaluation of the gate for the first time (when no gate $g' > g$ has been evaluated yet), the evaluator will be unable to forge the output key that she is missing.

It can be observed that knowledge of L^0 implies knowledge of R^0. Due to the earlier argument regarding K^1 being perfectly hidden unless both L^1 and R^1 are known, this does not pose a problem. Intuitively, the worst that an adversary could do with this knowledge (eg. given L^0 and R^1) is obtain both keys on the right incoming wire, but the damage is “contained”; wires occurring after this gate are not affected. Examining what an adversarial evaluator is capable of doing with this information (beyond just one ‘pass’ of evaluation) requires a more comprehensive analysis, which we defer to Section 4. We show that despite the information leaked by the key structure of the AND gates, our scheme achieves unconditional authenticity.

The routine VeAND defined in Fig. 2 verifies that both incoming wires of a gate g have the same zero key, which will also be the zero key for g. The key corresponding to bit value one for wire g is defined such that it requires no consistency checking with

```
GbAND $(K^0, K^1, 1^\kappa)$

The zero and one keys of the left and right incoming wires are $L^0$, $R^0$ and $L^1$, $R^1$ respectively

1. Set both zero keys, $L^0 := K^0$; $R^0 := K^0$
2. Split $K^1$ into additive secret shares, $L^1 \leftarrow \{0,1\}^\kappa$; $R^1 := K^1 \oplus L^1$
3. **return** $L^0, L^1, R^0, R^1$
```

```
EvAND $(L^{b_L}, R^{b_R}, b_L, b_R)$

Note that we require the bit values on the incoming wires to evaluate AND gates

1. **if** $b_L = 0$ **then return** $L^{b_L}$
2. **else if** $b_R = 0$ **then return** $R^{b_R}$
3. **else return** $L^{b_L} \oplus R^{b_R}$
```

```
VeAND $(L^0, L^1, R^0, R^1)$

1. **if** $L^0 \neq R^0$ **then** zero keys are inconsistent, **return** $0, \perp, \perp$. **else return** $1, L^0, L^1 \oplus R^1$
```

Fig. 2: Garbling, evaluation and verification of an AND gate

respect to its incoming wires' keys. This routine can hence be considered a "consistency check" that a given tuple of keys (L^0, L^1, R^0, R^1) maintain correctness of a garbled AND gate.

Garbling NOT Gates. NOT gates can be garbled for free, like in [Kol05], by switching the association of the zero and one keys. If wire w has keys K_w^0, K_w^1 corresponding to bit values zero and one respectively, and is input to a NOT gate g, the outgoing wire of g will have keys $K_g^0 = K_w^1$, $K_g^1 = K_w^0$ corresponding to values 0 and 1 respectively.

Note that none of the above schemes require ciphertexts to be published. Given that XOR, NOT, and AND gates can be garbled without ciphertexts, we therefore have a scheme to garble any formula without ciphertexts in the information-theoretic, privacy-free setting. Note that unlike the GESS construction of [Kol05], in our scheme the key size on every wire is the same (κ bits), hence allowing the online communication complexity of encoding the input x to be dependent only on the size of the input x, and not circuit depth of f.

3.2 Garbling an Entire Circuit

We can combine the routines defined in Fig. 1 and Fig. 2 in order to construct a garbling scheme for an entire formulaic circuit. Our garbling scheme $\mathcal{G}$ is defined by the tuple $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, \mathsf{Ve})$, as detailed in Fig. 3, Fig. 4, Fig. 5, Fig. 4, and Fig. 6 respectively.

```
Gb(f, 1^κ)

– Parse $n, q$ from $f$
– Denote the keys on wire $w$ as $K_w^0, K_w^1$ corresponding to bit values 0 and 1 respectively

  1. Start with the the circuit output gate, $g = n + q$
  2. Set circuit output gate keys, $K_g^0 \leftarrow \{0,1\}^\kappa;\ K_g^1 \leftarrow \{0,1\}^\kappa$
  3. while $g > n$ do
       i. $\alpha := A(g);\ \beta := B(g)$
       ii. if $g$ is an XOR Gate then $K_\alpha^0, K_\alpha^1, K_\beta^0, K_\beta^1 \leftarrow \mathsf{GbXOR}\left(K_g^0, K_g^1, 1^\kappa\right)$
       iii. else if $g$ is an AND Gate then $K_\alpha^0, K_\alpha^1, K_\beta^0, K_\beta^1 \leftarrow \mathsf{GbAND}\left(K_g^0, K_g^1, 1^\kappa\right)$
       iv. else $g$ is a NOT gate, $K_\beta^0 := K_g^1;\ K_\beta^1 := K_g^0$
       v. Proceed to the previous gate, $g := g - 1$
  4. Prepare encoding information, $e := \left(\left(K_i^0, K_i^1\right)_{i\in[n]}\right)$
  5. Prepare decoding information, $d := \left(K_{n+q}^0, K_{n+q}^1\right)$
  6. return $\varnothing$, $e$, $d$
```

Fig. 3: Garbling an entire circuit

```
En(x, e)

Let $x_i$ denote the $i^{th}$ bit of $x$

  1. Parse $e$ into keys, $\left(\left(K_i^0, K_i^1\right)_{i\in[n]}\right) := e$
  2. return $(K_i^{x_i} || x_i)_{i\in[n]}$

De(Y, d)

  1. if $Y = d[0]$ then return 0
  2. else if $Y = d[1]$ then return 1
  3. else return $\perp$
```

Fig. 4: Encoding a clear function input and Decoding a garbled output

We can further optimize our scheme to handle ℓ-fan-in gates with better concrete efficiency. A detailed discussion is deferred to Section 6. The full proof of security appears in the next section.

4 Full Proof of Security

Theorem 1. *The garbling scheme $\mathcal{G}$ is an unconditionally secure privacy-free garbling scheme.*

```
$\mathsf{Ev}(F, X)$

– The clear circuit $f$ is assumed to be known
– Let $K_w, b_w$ denote the key obtained on wire $w$, and the bit on that wire respectively

 1. Parse $(K_w||b_w)_{i\in[n]} := X$
 2. Start with the first input gate $g := n + 1$
 3. **while** $g \leq n + q$ **do**
      i. $\alpha := A(g);\ \beta := B(g)$
     ii. **if** $g$ is an XOR Gate **then** compute $b_g := b_\alpha \oplus b_\beta$ and $K_g \leftarrow \mathsf{EvXOR}(K_\alpha, K_\beta)$
    iii. **else if** $g$ is an AND Gate **then** compute $b_g := b_\alpha \wedge b_\beta$ and $K_g \leftarrow$
         $\mathsf{EvAND}(K_\alpha, K_\beta, b_\alpha, b_\beta)$
     iv. **else** $g$ is a NOT gate, $b_g := \neg b_\beta$
      v. Proceed to the next gate, $g := g + 1$
 4. The key on the last wire is the garbled output, **return** $K_{n+q}$
```

Fig. 5: Evaluating a Garbled Circuit on Garbled Input

```
$\mathsf{Ve}(F, f, e)$

– The consistency of each gate is verified, and if found to be consistent, the corresponding
  keys are derived
– Let $K_w^0, K_w^1$ denote the keys corresponding to values 0 and 1 respectively on wire $w$
– Parse $n, q$ from $f$

 1. Parse $e$ into keys $\left(\left(K_i^0, K_i^1\right)_{i\in[n]}\right) := e$
 2. Start with the first gate, $g := n + 1$
 3. **while** $g \leq n + q$ **do**
      i. $\alpha := A(g);\ \beta := B(g)$
     ii. **if** $g$ is an XOR gate **then**
         – $b, K_g^0, K_g^1 := \mathsf{VeXOR}\left(K_\alpha^0, K_\alpha^1, K_\beta^0, K_\beta^1\right)$
         – **if** $b = 0$ **then return** 0
    iii. **else if** $g$ is an AND gate **then**
         – $b, K_g^0, K_g^1 := \mathsf{VeAND}\left(K_\alpha^0, K_\alpha^1, K_\beta^0, K_\beta^1\right)$
         – **if** $b = 0$ **then return** 0
     iv. **else** $g$ is a NOT gate, $K_g^0 := K_\beta^1;\ K_g^1 := K_\beta^0$
      v. Proceed to the next gate, $g := g + 1$
 4. All keys are consistent, **return** 1
```

Fig. 6: Verifying a Garbled Circuit

Correctness follows from the correctness of the garbling schemes for individual gates, discussed in Section 3.1. Verifiability follows from the consistency-checks of individual gates conducted in the Ve algorithm, discussed in Section 3.1.

We now construct a proof of authenticity by reducing the authenticity of our scheme for a generic formulaic circuit to the authenticity of a single garbled gate. We start by showing that a garbling of a circuit consisting of one gate is authentic. We then show that forging an output for an n-input garbled formulaic circuit is exactly as hard as forging an output for the same circuit with one of its gates deleted, when garbled with the same randomness[5]. The "hidden core" of our argument is that any compromise in the keys of a gate *allowed* by our scheme will not concede the gate's child's keys; the damage will only spread 'upward' to its incoming wires. We denote an adversary wishing to compromise the authenticity of a circuit with n inputs as $\mathcal{A}_n$.

4.1 Single Gate Case

Lemma 1. *The garbling scheme $\mathcal{G}$ achieves unconditional authenticity as per Definition 3 when the domain is restricted to circuits f with input size $n = 2$.*

Proving that an adversarial evaluator will be unable to forge an output key, given her requested input keys for any single gate will prove Lemma 1. This can be done by considering the garbling of AND and XOR gates, as per Fig. 2 and Fig. 1 respectively.

Let the keys on the left input wire be L^0, L^1, right input wire be R^0, R^1, and output wire be K^0, K^1. The evaluator has input bits b_L and b_R on the left and right input wires respectively. Consequently, she is given the keys L^{b_L} and R^{b_R}. We denote the adversarial evaluator as $\mathcal{A}_2$, and show that she can not forge the key $K^{\neg b_K}$, where b_K is the output bit (either $b_L \wedge b_R$ or $b_L \oplus b_R$ as per the case).

XOR Gate. The authenticity of XOR gate garbling is relatively straightforward. As per the output of the GbXOR routine, we have,

$$L^0 \oplus R^0 = L^1 \oplus R^1 = K^0, \text{ and } L^1 \oplus R^0 = L^0 \oplus R^1 = K^1$$

Let $b_K = b_L \oplus b_R$. The evaluator computes $K^{b_K} = L^{b_L} \oplus R^{b_R}$. The adversarial evaluator $\mathcal{A}_2$ wishing to forge $K^{\neg b_K}$ will notice that the only relations connecting her input keys to $K^{\neg b_K}$ are as follows: $K^{\neg b_K} = L^{\neg b_L} \oplus R^{b_R} = L^{b_L} \oplus R^{\neg b_R}$. Clearly, she will be unable to forge $K^{\neg b_K}$ without guessing either $L^{\neg b_L}$ or $R^{\neg b_R}$.

AND Gate. To show authenticity of a garbled AND gate, we have to take into account that one of the input wires may compromise both keys. We analyze all four cases, based on the input bits. Keep in mind that $L^0 = R^0 = K^0$, and $L^1 \oplus R^1 = K^1$.

1. $b_L = b_R = 0$: In this case, $\mathcal{A}_2$ has absolutely no information about K^1, and can do no better than directly guessing it.
2. $b_L = b_R = 1$: In this case, $\mathcal{A}_2$ has absolutely no information about K^0, and can do no better than directly guessing it.
3. $b_L = 1, b_R = 0, b_K = b_L \wedge b_R = 0$: $\mathcal{A}_2$ has $K^0 = R^0$, as well as L^1. Due to the key structure, she also obtains $L^0 = R^0$. However, this information is useless, as the missing output key $K^1 = L^1 \oplus R^1$ requires knowledge of R^1, which $\mathcal{A}_2$ does not have.

[5] ie. the random tapes used in the garbling of f and f' are identical.

4. $b_L = 0$, $b_R = 1$, $b_K = b_L \wedge b_R = 0$: This case is identical to Case 3, as the left and right input wires are treated symmetrically.

NOT Gate. A NOT gate may be added on or removed from any wire at will, with no implications for authenticity, as the distributions of input and output keys for the individual gates remain unchanged.

Hence, we have shown on a case-by-case basis that there exists no gate or input combination in which an adversary $\mathcal{A}_2$ can do better than guessing the output key $K^{\neg b_K}$ that she is missing. Therefore, even a computationally unbounded adversary will be successful in forging a gate output with probability no greater than $2^{-\kappa}$, which proves Lemma 1.

4.2 Reduction Step

In this section, we perfectly reduce the authenticity of the garbling of an n-input formulaic circuit to that of an $(n-1)$-input one. We denote the garbling (ie. collection of keys on each wire, generated within Gb) of a function f as $\mathcal{K} = \left(K_i^0, K_i^1\right)_{i \in [1, n+q]}$.

Simply put, given that garbling an n-input formulaic circuit f produces $\mathcal{K}$, an adversary loses no advantage by deleting an input gate g (gate fed only by circuit input wires), as Lemma 1 demonstrates that the keys on input wires $A(g)$ and $B(g)$ are completely useless in forging an unknown key for g. Hence, an adversary $\mathcal{A}_n$ wishing to forge an output key as per $\mathcal{K}$ will be as successful in forging an output key as per $\mathcal{K}'$, a garbling of f with any input gate g deleted. An adversary for the latter procedure is denoted by $\mathcal{A}_{n-1}$. As there is no security loss in the reduction from $\mathcal{A}_n$ to $\mathcal{A}_{n-1}$, we finally conclude that $\mathcal{A}_n$ is as successful in forging an output as per $\mathcal{K}$ as $\mathcal{A}_2$ is in forging an output for a single-gate circuit. We know from Lemma 1 that no such computationally unbounded $\mathcal{A}_2$ succeeds with probability greater than $2^{-\kappa}$.

Given an adversary $\mathcal{A}_n$ that can forge an output for an n-input formulaic circuit f, we construct adversary $\mathcal{A}_{n-1}$ (in Fig. 7), that can forge an output for an $(n-1)$-input formulaic circuit f' with the same probability of success. For readability, for a scheme $\mathcal{G}$, denote the event that a computationally unbounded adversary $\mathcal{A}$ succeeds in forging a garbled output Y given F, X for some f, x (where $(F, e, d) \leftarrow \mathsf{Gb}(f, 1^\kappa); X \leftarrow \mathsf{En}(e, x)$), by the outcome of $\mathsf{Aut}_{\mathcal{G}}\left(\mathcal{A}, 1^\kappa\right)$. Specifically,

$$\mathsf{Aut}_{\mathcal{G}}\left(\mathcal{A}, 1^\kappa\right) = \begin{cases} 1 & \text{if } \mathcal{A}(F, X) = Y; Y \neq \mathsf{Ev}(F, X), \mathsf{De}(Y, d) \neq \perp \\ 0 & \text{otherwise} \end{cases}$$

It is clear to see that a garbling scheme $\mathcal{G}$ is authentic if, and only if, $\Pr\left[\mathsf{Aut}_{\mathcal{G}}\left(\mathcal{A}, 1^\kappa\right) = 1\right] \leq 2^{-\kappa}$, $\forall \mathcal{A}$. Therefore, as there is no security loss in our reduction from $\mathcal{A}_n$ to $\mathcal{A}_{n-1}$, we have:

$$\Pr\left[\mathsf{Aut}_{\mathcal{G}}\left(\mathcal{A}_n, 1^\kappa\right) = 1\right] = \Pr\left[\mathsf{Aut}_{\mathcal{G}}\left(\mathcal{A}_{n-1}, 1^\kappa\right) = 1\right] =$$
$$\cdots = \Pr\left[\mathsf{Aut}_{\mathcal{G}}\left(\mathcal{A}_2, 1^\kappa\right) = 1\right] \leq 2^{-\kappa}$$

Note that the reduction from $\mathcal{A}_n$ to $\mathcal{A}_{n-1}$ detailed in Fig. 7 only works for formulaic circuits; deleting a gate with fan-out of l will produce l different input wires, each with

$\mathcal{A}_{n-1}$

- $\mathcal{A}_{n-1}$ has black-box access to $\mathcal{A}_n$, which is capable of forging Y for the garbled circuit F and corresponding encoded input X for a certain formulaic circuit f and corresponding n-bit input x.
- Using $\mathcal{A}_n$, $\mathcal{A}_{n-1}$ forges garbled output Y' for a (F', X'), for some formulaic circuit f' and corresponding $(n-1)$-bit input x'.
- f', x' are derived from f, x as follows:
  - i. Choose some gate g from f such that both parents $A(g)$ and $B(g)$ are input wires[a].
  - ii. Construct f' identical to f, with the exception that $g, A(g), B(g)$ are replaced with a single input wire numbered g'.
  - iii. Parse x into bits $x_1 x_2 \cdots x_i \cdots x_n$, copy them to create x', with the exception of $x_{A(g)}$ and $x_{B(g)}$, which are replaced with $x'_{g'} = G_g\left(x_{A(g)}, x_{B(g)}\right)$[b]
  - iv. f', x' are now an $(n-1)$-input function and its corresponding input such that $f(x) = f'(x')$

```
1. Parse $X'$ into keys $X'_1 X'_2 \cdots X'_{n-1}$, and copy them into $X$ at the appropriate locations.
2. $X$ will be missing keys at locations $A(g)$ and $B(g)$. They can be generated as follows[c]:
   - if $g$ was an XOR gate then $X_{A(g)} \leftarrow \{0,1\}^\kappa$; $X_{B(g)} := X_{A(g)} \oplus X'_{g'}$
   - else if $g$ was an AND gate then
       i. if $x_{A(g)} = x_{B(g)} = 0$ then $X_{A(g)} := X_{B(g)} := X'_{g'}$
       ii. else if $x_{A(g)} = x_{B(g)} = 1$ then $X_{A(g)} \leftarrow \{0,1\}^\kappa$, $X_{B(g)} := X_{A(g)} \oplus X'_{g'}$
       iii. else if $x_{A(g)} = 0$ then $X_{A(g)} := X'_{g'}$, $X_{B(g)} \leftarrow \{0,1\}^\kappa$
       iv. else $X_{B(g)} := X'_{g'}$, $X_{A(g)} \leftarrow \{0,1\}^\kappa$
3. Send $X$ to $\mathcal{A}_n$ and output the response, output $\mathcal{A}_n(\varnothing, X, 1^\kappa)$
```

[a] Even g such that its parent is a NOT gate $A(g)$ with its parent as an input wire $w = B(A(g)) < n$ will work. In this case, consider $\neg x_w$ in place of x_w wherever relevant in this algorithm.

[b] G_g is the gate functionality of gate g, ie. XOR or AND

[c] This subroutine effectively garbles the missing gate g such that the keys for parents $A(g), B(g)$ consistently evaluate to the keys on wire g'. Note that this leaves all the original keys generated when garbling f' undisturbed, hence implying that a forged key returned by $\mathcal{A}_n$ for its garbling of f can directly be output as a forged key for the garbling of f' given to $\mathcal{A}_{n-1}$. Also note a minor technical detail, that we ignore that X_i is actually $K_i || b_i$ on an input wire, for readability.

Fig. 7: Constructing Adversary $\mathcal{A}_{n-1}$ given $\mathcal{A}_n$

its own independent pair of keys. For $\mathcal{A}_{n-1}$ to ensure that the deleted gate's keys are consistent with l different outgoing wires is undefined as per our garbling scheme.

Hence, there exists no computationally unbounded adversary that succeeds in forging an output for a formulaic circuit of any size when garbled by $\mathcal{G}$, with probability greater than $2^{-\kappa}$. This proves Theorem 1.

4.3 Adaptive Security

We had mentioned in an earlier section that our scheme achieves adaptive security, or $\mathsf{aut1}$ in the terminology of [BHR12a], as opposed to Definition 3 which they term static security, or aut.

We show this by illustrating that an adversary in the $\mathsf{Aut1}_{\mathcal{G}}$ game (which forms the basis for the definition of adaptive security) is at no advantage in forging a garbled output, as compared to an adversary wishing to break the 'static' authenticity of our scheme as per Definition 3.

In the $\mathsf{Aut1}_{\mathcal{G}}$ game, the adversary is allowed to request from the game the garbled circuit F for her function f *before* she chooses x for which she receives encoded input $X = \mathsf{En}(e, x)$. The $\mathsf{Aut1}_{\mathcal{G}}$ game consists of three stages:

1. The GARBLE stage accepts from $\mathcal{A}$ a circuit f, computes $(F, e, d) \leftarrow \mathsf{Gb}(1^\kappa, f)$, and returns F to $\mathcal{A}$.
2. The INPUT stage accepts from $\mathcal{A}$ an input x, outputs $\perp$ if it is not in the domain of f, otherwise returns $X = \mathsf{En}(e, x)$ to $\mathcal{A}$.
3. The FINALIZE stage accepts from $\mathcal{A}$ a garbled output Y, and outputs 1 if $Y \neq \mathsf{Ev}(F, X)$ while still being a valid garbled output (ie. $\mathsf{De}(Y, d) \neq \perp$), and 0 otherwise.

The output of the experiment $\mathsf{Aut1}_{\mathcal{G}}(\mathcal{A}, 1^\kappa)$ is the value output by the FINALIZE stage. An unconditionally adaptively authentic scheme will ensure that $\Pr[\mathsf{Aut1}_{\mathcal{G}}(\mathcal{A}, 1^\kappa) = 1] \leq 2^{-\kappa}$ for all computationally unbounded $\mathcal{A}$.

It is immediately evident that this extra concession granted to the adversary is useless in our setting, as our scheme does not produce any ciphertexts to represent a garbled circuit. An adversary $\mathcal{A}'$ for the $\mathsf{Aut1}_{\mathcal{G}}$ game can be given a null string to serve as the garbled circuit F of any function f that it may submit to the GARBLE stage. Therefore, $\mathcal{A}'$ is forced to choose x completely independently of the garbling of f, effectively having to commit to f, x simultaneously. Hence, the task of $\mathcal{A}'$ is equivalent to that of a static adversary $\mathcal{A}(F, X)$ attempting to forge a garbled output as per Definition 3, which is proven not to succeed with probability better than $2^{-\kappa}$ by Theorem 1.

5 Breaking the Lower Bound of [ZRE15]

Zahur *et al.* [ZRE15] observe that most known garbling schemes fit into their characterization of *linear* garbling techniques. Informally, a linear garbling scheme proceeds gate by gate, at each gate generating a vector $\mathbf{S} = (R_1, \cdots, R_r, Q_1, \cdots, Q_q)$, where R_is are fresh random values, and Q_is are obtained by independent calls to a random oracle (queries may depend on R_i values). The gate ciphertexts as well as the keys on each wire touching the gate are derived by linearly combining the values in $\mathbf{S}$. The only non-linearity allowed in their model is through the random oracle invocations, and permutation bits. All elements are μ bits long, where μ is the security parameter. They prove that an ideally secure garbling scheme that is linear as per their characterization must adhere to certain lower bounds in terms of bits of ciphertext produced when garbling a *single atomic* AND gate. An ideally secure garbling scheme ensures that no computationally unbounded adversary (with bounded calls to the random oracle) will have advantage better than $\mathsf{poly}(\mu)/2^{\mu}$ in the security games of Bellare *et al.* [BHR12b]. The following are the bounds in the private and privacy-free settings respectively, as argued by Zahur *et al.* [ZRE15].

Lower bound for garbling schemes achieving privacy. Linear garbling schemes are shown to require at least 2μ bits of ciphertext to garble an AND gate privately. This bound was circumvented (but not contradicted) in the works of Ball *et al.* [BMR16] and Kempka *et al.* [KKS16] by a different treatment of permutation bits. Both schemes garble a single AND gate privately but non-composably with just one ciphertext.

Lower bound for privacy-free garbling schemes. Linear garbling schemes achieving authenticity are argued to require at least μ bits of ciphertext to garble an AND gate. To the best of our knowledge, this bound is currently unchallenged. Our scheme is clearly linear (with no requirement of a random oracle) and yet garbles AND gates with *no ciphertexts* for any μ. Moreover, our scheme composes to garble a non-trivial class of circuits (ie. formulas) with no ciphertexts.

5.1 Linear Garbling

We recall the formal definition of linear garbling [ZRE15], but simplified for the privacy-free setting. Specifically, we enforce that the permutation bit always be 0, as there is no reason for the semantic value of a wire key to be hidden from an evaluator in this setting. Indeed, both previous privacy-free schemes [ZRE15, FNO15] rely on an evaluator knowing the semantic value of the key she has. A garbling scheme $\mathcal{G}$ is linear if its routines are of the form described in Fig. 8.

Claim ([ZRE15]). Every linear ideally secure privacy-free garbling scheme for AND gates must have $p \geq 1$. The garbled gate consists of at least μ bits.

Our privacy-free garbling scheme is a linear garbling scheme with the following parameters for an AND gate and with $\mu = \kappa$:

- Number of ciphertexts $p = 0$, random values $r = 3$ and random oracle queries $q = 0$.
- The same vector to obtain all zero keys, $\mathbf{L}_0 = \mathbf{R}_0 = \mathbf{K}_0 = [1\ 0\ 0]$
- Vectors to select independent input 1-keys, $\mathbf{L}_1 = [0\ 1\ 0]$, $\mathbf{R}_1 = [0\ 0\ 1]$
- Output 1-key vector as the sum of both input 1-keys, $\mathbf{K}_1 = \mathbf{L}_1 + \mathbf{R}_1 = [0\ 1\ 1]$
- $\left(C^i\right)_{i\in[p]}$ is an empty set as there are no ciphertexts required.
- Evaluation vectors $\left(\mathbf{V}_{\alpha,\beta}\right)_{\alpha,\beta\in\{0,1\}}$ as follows:
  - When the evaluator has a zero key, output the zero key. So, $V_{0,0} = V_{0,1} = [1\ 0]$, $V_{1,0} = [0\ 1]$.
  - When both keys correspond to 1, output their sum. So $V_{1,1} = [1\ 1]$.

Succinctness of our garbling scheme. As Zahur *et al.* [ZRE15] note, almost all practical techniques so far for garbling Boolean circuits qualify as linear as per their characterization. If we use their parameters to define $s = p + r + q$ as a measure of 'program succinctness' of a linear garbling scheme, then we observe that our garbling scheme has the most succinct program ($s = 3$) of all garbling schemes in the literature.

- We describe here a simplified characterization of linear garbling [ZRE15] for the privacy-free setting. Note that garbling by default is for a single gate.
- The integers p, q, r, and vectors $\mathbf{L}_0$, $\mathbf{L}_1$, $\mathbf{R}_0$, $\mathbf{R}_1$, $\mathbf{K}_0$, $\mathbf{K}_1$, $\left(\mathbf{C}^i\right)_{i\in[p]}$, $\left(\mathbf{V}_{\alpha,\beta}\right)_{\alpha,\beta\in\{0,1\}}$ parameterize garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$. p denotes the number of ciphertexts. r and q denote the number of uniformly random elements and the number of random oracle calls needed. Each of the above vectors is of size $r + q$ (except $\mathbf{V}_{\alpha,\beta}$ which is of size $p + q + 2$) with entries in $GF\left(2^\mu\right)$.

$\mathsf{Gb}\left(\cdot, 1^\mu\right)$

```
1. for $i \in [r]$ do Choose $R_i \leftarrow GF\left(2^\mu\right)$
2. for $i \in [q]$ do Make a query to the random oracle, store the response in $Q_i$
3. Construct $\mathbf{S} = \left(R_1, \cdots, R_r, Q_1, \cdots, Q_q\right)$
4. for $i \in \{0,1\}$ do Corresponding to semantic value $i$, compute keys on the two input
   wires as $L_i := \langle \mathbf{L}_i, \mathbf{S}\rangle$ and $R_i := \langle \mathbf{R}_i, \mathbf{S}\rangle$, and the output wire as $K_i := \langle \mathbf{K}_i, \mathbf{S}\rangle$
5. for $i \in [p]$ do Compute the $i^{\text{th}}$ gate ciphertext $C_i := \langle \mathbf{C}^i, \mathbf{S}\rangle$
6. Construct and output encoding information $e := \left(\left(L_0, L_1\right), \left(R_0, R_1\right)\right)$, and gate cipher-
   texts $F = \left(C_i\right)_{i\in[p]}$
```

$\mathsf{En}\left(x, e\right)$

```
1. Parse $\left(x_0, x_1\right) := x$, and $\left(\left(L_0, L_1\right), \left(R_0, R_1\right)\right) := e$
2. Output $X = \left(L_{x_0} || x_0, R_{x_1} || x_1\right)$
```

$\mathsf{Ev}\left(F, X\right)$

```
1. Parse input labels $\left(L_\alpha || \alpha, R_\beta || \beta\right) := X$, and ciphertexts $\left(C_i\right)_{i\in[p]} := F$
2. for $i \in [q]$ do Make a query to the random oracle, store the response in $Q'_i$
3. Construct $\mathbf{T} = \left(L_\alpha, R_\beta, Q'_1, \cdots, Q'_q, C_1, \cdots, C_p\right)$
4. Output $\langle \mathbf{V}_{\alpha,\beta}, \mathbf{T}\rangle$
```

Fig. 8: Form of linear garbling schemes

5.2 Where the [ZRE15] Technique for Bounding Privacy-Free Garbling Fails

As illustrated above, our garbling scheme is clearly linear and achieves ideal security, but can still garble an AND gate in the privacy-free setting with *no* ciphertext. Our scheme is therefore a simple and direct counterexample to the argument of Zahur *et al.* [ZRE15] that a linear garbling scheme achieving ideal authenticity must produce at least μ bits of ciphertext when garbling and AND gate.

In more detail, the ciphertext generating $\mathbb{G}_{a,b}$ becomes a dimension 0 matrix. At the core of the linear garbling model is that the evaluator's behaviour must depend only on the public α, β 'signal' bits, a property which is adhered to by our privacy-free scheme. In our setting, the signal bits convey the actual semantic values with which the keys are associated. However, the lower bound proof in [ZRE15] relies on the property that changing a 'permute' bit a/b which is defined when garbling, must also change the corresponding signal bit on which the evaluator acts. In our setting it is immediate that this assumption does not need to hold (as α, β are not tied to a, b), and our scheme takes advantage of this to break the claimed lower bound.

6 ℓ-fan-in Gates

In this section, we describe how to handle ℓ-fan-in gates efficiently. We first provide a new garbling scheme for *threshold* gates in Section 6.1, then describe how to save computation in garbling and evaluating ℓ-fan-in XOR and AND gates respectively in Sections 6.2 and 6.3.

6.1 Threshold Gates

An ℓ-input threshold gate, parameterized by a threshold t, realizes the following function:

$$f_t(x_1, \cdots, x_i, \cdots, x_\ell) = \begin{cases} 1, & \text{if} \sum_{i=1}^{\ell} x_i > t \\ 0, & \text{otherwise} \end{cases}$$

The threshold range $1 < t < \ell - 1$ is of interest to us, as the gate otherwise degenerates into an ℓ-fan-in AND or NAND gate, which can be handled more efficiently by our scheme. Boolean threshold gates are considered and motivated by Ball *et al.* [BMR16], who construct a scheme to garble them natively (generating $\mathcal{O}\left(\log^3 \ell / \log\log \ell\right)$ ciphertexts) as opposed to garbling a composition of AND, XOR and NOT gates (yielding $\mathcal{O}\left(\ell \log \ell\right)$ ciphertexts using the best known garbling scheme of [ZRE15]). Here, we present a method of garbling Boolean threshold gates (embedded in formulaic circuits) directly, producing no ciphertext, and using only information-theoretic operations; specifically two independent instances of Shamir secret sharing [Sha79] per threshold gate assuming the underlying field to be $GF(2^\kappa)$.

The idea is as follows; an evaluator having inputs $x_1 \cdots x_\ell$ to the threshold gate computing f_t, such that $\sum_{i=1}^{\ell} x_\ell = m$, will possess m input 1-keys, and $\ell - m$ input 0-keys. Let the gate output keys be denoted as K^0 and K^1, and denote the keys on the i^{th} input wire as K_i^0, K_i^1. As the requirement of the threshold gate is that more than t of the evaluator's inputs must be 1 in order to output 1, we need to devise a garbled evaluation scheme which allows the evaluator to obtain K^1 when she has more than t K_i^1s. A natural candidate for this construction is a threshold secret sharing scheme, where the K_i^1s form a t-out-of-l sharing of K^1; ie. any $t+1$ of the K_i^1s are sufficient to reconstruct K^1, while having t or fewer K_i^1s renders K^1 unconditionally hidden except with a probability of $2^{-\kappa}$.

Note that in order to correctly realise f_t, our garbled gate evaluation scheme also needs to ensure that if (and only if) the evaluator has fewer than $(t+1)$ input values equal to 1, she should obtain K^0. In this case, her $\ell - m$ zero keys K_i^0 should be sufficient to reconstruct K^0. Therefore, we define the K_i^0s to form an $(\ell - (t+1))$-out-of-l sharing of K^0, ie. any $(\ell - t)$ of the K_i^0s are sufficient to reconstruct K^0. This also

$\mathsf{GbTHR}\left(\ell, t, K^0, K^1, 1^\kappa\right)$

The zero and one keys of the i^{th} incoming wire will be K_i^0, K_i^1. We denote the set of all t-degree polynomials with constant s as $\mathcal{P}^{s,t}$.

```
1. Choose a uniformly random $t$-degree polynomial with $K^1$ as its constant, $h_{K^1} \leftarrow \mathcal{P}^{K^1,t}$
2. Generate the input 1-keys to be Shamir shares of $K^1$, **for all** $i \in [\ell]$ **do** $K_i^1 := h_{K^1}(i)$
3. Choose a uniformly random $(\ell - (t+1))$-degree polynomial with $K^0$ as its constant,
   $h_{K^0} \leftarrow \mathcal{P}^{K^0,(\ell-(t+1))}$
4. Generate the input 0-keys to be Shamir shares of $K^0$, **for all** $i \in [\ell]$ **do** $K_i^0 := h_{K^0}(i)$.
5. **return** $\left(K_i^0, K_i^1\right)_{i\in[1,\ell]}$
```

$\mathsf{EvTHR}\left(t', (j_i, K_i)_{i\in[t+1]}\right)$

```
1. The input to this routine is assumed to be a set of $t'+1$ unique (index, key) pairs, where
   each key corresponds to the same value. Note that $t'$ may be $t$ or $\ell-(t+1)$ depending
   on the gate output.
2. Using Lagrange interpolation, we obtain the unique $t$-degree polynomial $h$, such that
   $h(j_i) = K_i, \forall i \in [t+1]$.
3. Compute the output key by retrieving the constant of $h$; $K := h(0)$.
4. **return** $K$
```

$\mathsf{VeTHR}\left(t, \left(K_i^0, K_i^1\right)_{i\in[\ell]}\right)$

```
1. Using Lagrange interpolation, we obtain the unique $t$-degree polynomial $h_{K^1}$, such that
   $h_{K^1}(i) = K_i^1, \forall i \in [t+1]$.
2. **if** $\exists j \in [t+2, \ell]$ such that $h_{K^1}(j) \neq K_j^1$ **then return** $0, \perp, \perp$
3. Using Lagrange interpolation, we obtain the unique $(\ell-(t+1))$-degree polynomial
   $h_{K^0}$, such that $h_{K^0}(i) = K_i^0, \forall i \in [\ell-t]$.
4. **if** $\exists j \in [\ell-t+1, \ell]$ such that $h_{K^0}(j) \neq K_j^0$ **then return** $0, \perp, \perp$
5. The input 0-keys and 1-keys each define unique polynomials of degrees $\ell-(t+1)$ and $t$
   respectively. Compute the output keys to be the constants of the curves, $K^0 := h_{K^0}(0)$
   and $K^1 := h_{K^1}(0)$
6. **return** $1, K^0, K^1$
```

Fig. 9: Garbling, evaluation and verification of a threshold gate

ensures that when $m > t$ (ie. $f_t(x_1 \cdots x_\ell) = 1$), she will be unable to reconstruct K^0, as $(\ell - m) < (\ell - t)$, and she only has $(\ell - m)$ K_i^0s.

We formalize the described scheme in Fig. 9. It is evident how to invoke the GbTHR, EvTHR, and VeTHR routines within the Gb, Ev, and Ve algorithms respectively. To formally prove the authenticity of our threshold gate garbling routine, we describe how the adversary $\mathcal{A}_{n-\ell+1}$, given black-box access to $\mathcal{A}_n$, can forge an output for an $n-\ell+1$ input formula obtained by deleting an ℓ-fan-in input threshold gate from an n-input formula used by $\mathcal{A}_n$, in Fig. 10.

```
$\mathcal{A}_{n-\ell+1}(f', x', F', X')$

– This procedure is a modification of the adversary from Fig. 7, to accommodate threshold
    gates.
– Without loss of generality, $f'$ was generated by deleting an input threshold gate $g$ from $f$,
    which was fed by input bits $x_1 \cdots x_\ell$.
– This routine adds a clause to Step 2 of the original $\mathcal{A}_{n-1}$ to detail how to generate
    $X_1 \cdots X_\ell$ for $\mathcal{A}_n$, given the input key $X'_g$ on wire $g'$.

  2.    - else if $g$ was a threshold gate $f_t$ then
          i. $b := f_t(x_1 \cdots x_\ell)$
         ii. if $b = 1$ then Choose a uniformly random $t$-degree polynomial with $X'_g$ as its
             constant, $h \leftarrow \mathcal{P}^{X'_g, t}$
        iii. else Choose a uniformly random $(\ell - (t+1))$-degree polynomial with $X'_g$ as
             its constant, $h \leftarrow \mathcal{P}^{X'_g, (\ell-(t+1))}$
         iv. for $i \in [1, \ell]$ do
               • if $x_i = b$ then $X_i := h(i)$
               • else $X_i \leftarrow \{0,1\}^\kappa$
```

Fig. 10: Deleting a threshold gate to reduce $\mathcal{A}_n$ to $\mathcal{A}_{n-\ell+1}$ as per the gate deletion proof strategy

Security. As discussed earlier, the unconditional authenticity of our threshold gate garbling in the single gate case is implied by the unconditional security of Shamir's secret sharing [Sha79]. Observe that our threshold gate garbling scheme is also made possible by the violation of Yao's invariant; the nature of threshold secret sharing is such that once the curve is reconstructed, the missing shares can be computed as well. Specifically, possessing the 1-key on $t+1$ input wires to an ℓ-fan-in threshold gate computing f_t, allows the reconstruction of the 1-keys on the remaining $\ell - (t+1)$ input wires in addition to the gate output 1-key. However, this information is useless in reconstructing the 0-key of the gate, and hence has no impact on authenticity.

Extension to Circuits. It is easy to see that our threshold gate garbling gadget can be used to augment any privacy-free circuit garbling scheme $\mathcal{G}_c$, at the cost of cryptographic assumptions no stronger than required by $\mathcal{G}_c$. Every input key k_i^b to the threshold gate g can be mapped to a corresponding K_i^b output by GbTHR. During evaluation, K_i^b is made accessible given k_i^b by means of a ciphertext $T[g]_{i,b} = H(g, i, k_i^b, b) \oplus K_i^b$, where H is the cryptographic primitive used to implement encryption in $\mathcal{G}_c$, eg. PRF [GLNP15], KDF [FNO15], circular correlation robust hash [ZRE15].

While this gadget costs only 2ℓ ciphertexts to implement, we can additionally optimize this construction to cut down the ciphertexts by half. Intuitively, we can set the curves h_{K^1} and h_{K^0} pseudorandomly rather than uniformly at random. Specifically, the polynomial h_{K^1} in GbTHR (Fig. 9) can be set by fixing $t-1$ points as $h_{K^1}(i) = H(g, i, 1, k_i^1), \forall i \in [t-1]$, so that cipherexts are needed to convey only the remaining $\ell - t + 1$ points. The same optimization applied to h_{K^0} yields that the total number of ciphertexts that need to be communicated for this gadget is now $\ell + 2$.

Performance. Our base construction for formulas is significantly more efficient than a naive approach, as representing threshold gates in a formula is highly non-trivial, with upper bounds of $\mathcal{O}\left(\ell^{3.04}\right)$ [Ser14]. As for general circuits, the construction of Ball *et al.* [BMR16] will cost $\mathcal{O}\left(\log^3 \ell / \log\log \ell\right)$ more ciphertexts than our construction when embedded directly in a Boolean circuit (accounting for ℓ 'projection' gates) despite relying on a circular correlation robust hash function.

6.2 Improved ℓ-fan-in XOR

The routine to garble an individual XOR gate described in Fig. 1 performs 3 XOR operations in order to derive the incoming wire keys corresponding to a given pair of gate keys. Hence, in order to garble ℓ XOR gates, repeating this routine $\ell - 1$ times will cost $3(\ell - 1)$ XOR operations.

Consider a subtree (with ℓ leaves) consisting only of XOR gates, contained within the tree representation of a formulaic circuit. Note that there are $\ell - 1$ gates in this subtree. Without loss of generality, let the subtree be collapsed into a single gate accepting ℓ incoming wires. For convenience, the incoming wires (leaves of the subtree) are assumed to be numbered consecutively from w to $w + \ell - 1$, with the final XOR gate itself (root of the subtree) being numbered g such that the internal nodes of the subtree are numbered consecutively from $w + \ell$ to $g - 1$. As usual, the keys on wire i are denoted K_i^0, K_i^1, corresponding to bit values 0 and 1 respectively.

Consider the keys $\left(K_i^0, K_i^1\right)_{i \in [w,g]}$ to be produced by $\ell - 1$ instances of the GbXOR routine from Fig. 1; starting from the root K_g^0, K_g^1 and ending at the leaves to produce $\left(K_i^0, K_i^1\right)_{i \in [w, w+\ell-1]}$. Observe that the zero and one keys on each wire differ by the same offset; ie. $\forall i \in [w, g]$:

$$K_w^0 \oplus K_w^1 = \cdots = K_i^0 \oplus K_i^1 = \cdots = K_g^0 \oplus K_g^1 \qquad (1)$$

We make use of the property observed in Equation (1) in order to garble such an ℓ-fan-in XOR gate more efficiently. Essentially, the 0-keys of the incoming wires are chosen so as to form an additive secret sharing of the gate's 0-key. The 1-keys are then generated by offsetting the 0-keys by the same offset as the gate key pair (ie. $K_g^0 \oplus K_g^1$). The formal description is given in Fig. 11.

The routine detailed in Fig. 11 produces keys that adhere to the exact same distribution as the result of invoking the original GbXOR routine $\ell - 1$ times in an appropriate sequence. The evaluation and verification algorithms for garbled XOR gates (Fig. 1) are directly compatible. A separate proof of authenticity is therefore not required.

As for the computation cost, the new GbXOR routine of Fig. 11 requires one XOR operation to find the gate offset, $\ell - 1$ XOR operations to additively secret share one of the gate keys, and ℓ XOR operations to offset each of the 1-keys on the incoming wires, bringing the total to 2ℓ. This beats the $3(\ell - 1)$ cost of using multiple instances of the original routine when $\ell > 3$.

```
GbXOR $(\ell, K^0, K^1, 1^\kappa)$

– We have to generate $\ell$ key pairs, which will produce either $K^0$ or $K^1$ appropriately upon
    being combined by XORing
– The resultant keys are locally indexed here as $K_i^0, K_i^1, i \in [1, \ell]$

  1. Calculate the offset, $\Delta := K^0 \oplus K^1$
  2. Choose the 0-keys on all but one wire randomly, **for all** $i \in [1, \ell-1]$ **do** $K_i^0 \leftarrow \{0,1\}^\kappa$
  3. Set the final 0-key so that all the incoming wires' 0-keys form an additive secret sharing
     of $K^0$, $K_\ell^0 := \left(\bigoplus_{i=1}^{\ell-1} K_i^0\right) \oplus K^0$
  4. Offset the 0-keys to generate the 1-keys on the incoming wires. **for all** $i \in [1, \ell]$ **do**
     $K_i^1 = K_i^0 \oplus \Delta$
  5. **return** $(K_i^0, K_i^1)_{i \in [1, \ell]}$
```

Fig. 11: Garbling an ℓ-fan-in XOR gate

6.3 Improved ℓ-fan-in AND

The cost of garbling an AND gate is already minimal, at a single XOR operation per gate. Instead, we focus on optimizing the evaluation of AND gates.

Similar to the ℓ-fan-in case of XOR gates, consider a subtree consisting solely of AND gates, contained in a formulaic circuit. The gates in the subtree are numbered as described in the ℓ-fan-in XOR section; w to $w + \ell - 1$ for the inputs, $w + \ell$ to g for the intermediate gates, and g for the root of the subtree. The subtree is collapsed into a single ℓ-fan-in AND gate. We follow the standard naming convention for wire keys and bit values.

Observe that if any of the bit values on wires w to $w + \ell - 1$ are 0, then the entire subtree (the ℓ-fan-in AND gate) will evaluate to 0, as $b_g = b_w \wedge \cdots \wedge b_{w+\ell-1}$. Also observe that as per the GbAND routine defined in Fig. 2, the following relation holds:

$$K_w^0 = \cdots = K_i^0 = \cdots = K_g^0, \ \ \forall i \in [w, g] \tag{2}$$

We exploit the above relation in order to save time during evaluation; if a wire $j \in [w, w+\ell-1]$ is found to be carrying a bit value of 0, then the ℓ-fan-in AND gate output is set to 0, with the key K_j being assigned to the gate output key K_g. The routine is formally detailed in Fig. 12.

The only case where XOR operations are performed in the EvAND routine in Fig. 12 is when all input bit values are 1; ie. $b_i = 1, \forall i \in [w, w+\ell-1]$. Even so, only $\ell-1$ XOR operations are performed, which is the same as when $\ell-1$ instances of the original EvAND routine from Fig. 2 are executed. However, if there exists at least one incoming wire carrying bit value 0, ie. $\exists j \in [w, w+\ell-1], b_j = 0$, no XOR operations are performed to evaluate the entire ℓ-fan-in AND gate. This occurs for $2^\ell - 1$ out of the 2^ℓ input cases. The number of XOR operations saved will be equal to the number of gates in the (now collapsed) subtree that evaluate to bit value 1. As there is no modification to the garbling routine, there is no additional proof of authenticity required here.

$$\mathsf{EvAND}\left((K_i, b_i)_{i \in [1,\ell]}, 1^\kappa\right)$$

– We have to process ℓ (key, bit value) pairs that effectively correspond to an ℓ-fan-in AND gate.
– The incoming wire keys and bit values are locally indexed as K_i, b_i, where $i \in [1, \ell]$, and the resultant key and bit value are locally indexed as K_g, b_g.

```
1. if $\exists j \in [1, \ell]$ such that $b_j = 0$ then Gate output is zero, $K_j$ is also the output key. $K_g := K_j$
2. else Gate output is 1, XOR all input keys. $K_g := \bigoplus_{i=1}^{\ell} K_i$
3. return $K_g, b_g$
```

Fig. 12: Evaluating an l-fan-in AND gate

7 Online-Efficient Zero-Knowledge

Privacy-free GCs are motivated by applications to ZK protocols. Specifically, when plugged into the ZK protocol of [JKO13], a privacy-free GC yields an efficient method to prove non-algebraic statements. In this section, we show that when instantiated with our scheme, we obtain a ZK protocol for Boolean formula satisfiability (SAT) statements in the online-offline paradigm, where the communication in the online phase is linearly proportional only to the size of the witness.

A SAT verification function can be realised by a formula. A witness bit may occur a number of times in the formula. While realizing the formula as a formulaic circuit, each occurence of a witness bit in the formula is treated as a separate input wire. Denoting the i^{th} witness bit of the formula to be represented by input wires $\mathbf{I}_i = \{i_0, i_1, \cdots i_l\}$ in the formulaic circuit, in order for the formula to correctly check a witness $w = (w_i)_{i \in [n]}$ we must ensure that $\forall j \in \mathbf{I}_i$, $x_j = w_i$. We stress that the cumulative size of the $\mathbf{I}_i$s may be much bigger than the witness length. We denote the size of $\cup_i^n \mathbf{I}_i$ as n'. We denote the size of the (formulaic) circuit by m and the size of the witness w as n. So we have $n \le n' \le m$.

Our ZK protocol $\pi = (\pi_{\mathsf{off}}, \pi_{\mathsf{on}})$ is described in Fig. 13. Informally, protocol π is a direct adaptation of the ZK protocol in [JKO13] to the online-offline setting, by shifting the public-key operations (OTs) to the offline phase. However, we observe that the same witness bit is used to select multiple GC keys, and accordingly use a domain extension technique for the OTs in order to encode n' garbled inputs with just n OT instances. This is the core of why the communication required in π_{on} is only proportional to the witness size, and not the size of the statement. The proof of security appears in the full version of the paper and the formal definitions of the necessary ideal functionalities $\mathcal{F}_{\mathsf{COM}}, \mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{ZK}}^{\mathsf{R}}$ are postponed to Appendix A.

Computation cost. The offline phase will require $\mathcal{O}(n)$ PRG invocations by $\mathcal{V}$, and $\mathcal{O}(n)$ public key operations (OTs) by both $\mathcal{P}$ and $\mathcal{V}$. The online phase will require $\mathcal{O}(n)$ PRG invocations by $\mathcal{P}$ to unmask the input keys, $\mathcal{O}(m)$ XORs to evaluate the

- The witness checking formulaic circuit f is the only input available during π_{off}. During π_{on}, the prover $\mathcal{P}$ additionally has her witness $w = w_1 w_2 \cdots w_n$ as input.
- $G(\cdot)$ is a length-expanding PRG.
- The protocol is in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{COM}})$-hybrid model (formal definitions of these functionalities appear in Appendix A).

π_{off}

1. $\mathcal{V}$ garbles the circuit, $\left(\emptyset, \left(K_i^0, K_i^1\right)_{i\in[n']}, Z\right) \leftarrow \mathsf{Gb}\left(1^\kappa, f\right)^a$ and groups together input keys for the same witness bit as $\boldsymbol{K}_i^b = \left(K_j^b\right)_{j\in\mathbf{I}_i}$ for all $i \in [n]$ and $b \in \{0,1\}$.
2. For all $i \in [n]$ and $b \in \{0,1\}$, $\mathcal{V}$ samples seeds $S_i^b \leftarrow \{0,1\}^\mu$, computes $T_i^b := \boldsymbol{K}_i^b \oplus G(S_i^b)$ and sends T_i^b to $\mathcal{P}$.
3. For all $i \in [n]$, $\mathcal{P}$ samples $r_i \leftarrow \{0,1\}$ and sends $(\texttt{choose}, id, r_i)$ to $\mathcal{F}_{\mathsf{COT}}$.
4. On receiving messages $(\texttt{chosen}, id)$ for $i \in [n]$ from $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{V}$ samples $R_i^0, R_i^1 \leftarrow \{0,1\}^\mu$ and sends $\left(\texttt{transfer}, id, R_i^0, R_i^1\right)$ as input to $\mathcal{F}_{\mathsf{COT}}$ for all $i \in [n]$.
5. $\mathcal{P}$ receives $(\texttt{transferred}, id, R_i^{r_i})$, for $i \in [n]$ from $\mathcal{F}_{\mathsf{COT}}$.

π_{on}

6. For all $i \in [n]$, $\mathcal{P}$ computes $d_i := r_i \oplus w_i$ and sends d_i to $\mathcal{V}$.
7. For all $i \in [n]$, $\mathcal{V}$ computes $M_i^0 := S_i^0 \oplus R_i^{d_i}$, $M_i^1 := S_i^1 \oplus R_i^{\neg d_i}$, and sends $\left(M_i^0, M_i^1\right)$ to $\mathcal{P}$.
8. $\mathcal{P}$ does the following:
   i. computes $S_i^{w_i} := M_i^{w_i} \oplus R_i^{r_i}$ for all $i \in [n]$
   ii. computes $\boldsymbol{K}_i^{w_i} := T_i^{w_i} \oplus G\left(S_i^{w_i}\right)$ for all $i \in [n]$
   iii. parses $\{K_i^{w_i}\}_{i\in[n']}$ from $\{\boldsymbol{K}_i^{w_i}\}_{i\in[n]}$
   iv. evaluates the garbled circuit to obtain $Z' := \mathsf{Ev}\left(\emptyset, \{K_i^{w_i}\}_{i\in[n']}\right)$
   v. sends $(\texttt{commit}, id, Z')$ to $\mathcal{F}_{\mathsf{COM}}$.
9. On receiving $(\texttt{committed}, id, |Z'|)$ from $\mathcal{F}_{\mathsf{COM}}$, $\mathcal{V}$ sends the message $(\texttt{open-all}, id)$ to $\mathcal{F}_{\mathsf{COT}}$.
10. On receiving $\left(\texttt{transfer}, id, R_i^0, R_i^1\right)$ for all $i \in [n]$ from $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{P}$ does the following
   i. computes S_i^0, S_i^1 for all $i \in [n]$ using M_i^0, M_i^1 and R_i^0, R_i^1
   ii. computes $\boldsymbol{K}_i^b := T_i^b \oplus G(S_i^b)$ for all $i \in [n]$ and $b \in \{0,1\}$
   iii. and parses $\left\{K_i^b\right\}_{i\in[n']}$ from $\{\boldsymbol{K}_i^b\}_{i\in[n]}$ for $b \in \{0,1\}$
   iv. aborts if $\mathsf{Ve}\left(\emptyset, f, \left\{K_i^0, K_i^1\right\}_{i\in[n']}\right) \neq 1$, sends $(\texttt{reveal}, id)$ to $\mathcal{F}_{\mathsf{COM}}$ otherwise.
11. On receiving the message $(\texttt{reveal}, id, Z')$ from $\mathcal{F}_{\mathsf{COM}}$, $\mathcal{V}$ outputs $\texttt{accept}$ if $Z = Z'$.

[a] Instead of returning d, Gb is tweaked to return the 1-key on the output wire.

Fig. 13: Online-efficient ZK from our Privacy-free Garbling Scheme

GC, and another $\mathcal{O}(n)$ public key operations to verify that the GC is valid. $\mathcal{V}$ need only perform $\mathcal{O}(n)$ XOR operations in the online phase, and open one commitment.

Communication cost. The preprocessing phase will require $\mathcal{O}(m)$, and the online phase will require $\mathcal{O}(n)$ bits of communication. The ZK protocol when instantiated with a statically secure garbling scheme can not possibly yield an online phase which is independent of the size of the statement. This is because the garbled circuit will necessarily have to be sent after the evaluator commits to her input.

Our ZK Protocol without Offline Phase. Our protocol in Fig. 13 in its monolithic form can be obtained by running the OTs in an online fashion where the inputs of $\mathcal{V}$ are the seeds of the PRG and the inputs of $\mathcal{P}$ are the witness bits directly. We compare this protocol with that of [JKO13] composed with [ZRE15]. While our protocol offers the qualitative advantage of relying on weaker primitives (PRGs), we also note that since our garbling scheme is instantiated with a statistical security parameter, it can offer a better proof size.

References

BELO16. Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 578–590, New York, NY, USA, 2016. ACM.

BHKR13. Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *Security and Privacy (SP) '13, IEEE Symposium on*, pages 478–492. IEEE, 2013.

BHR12a. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *ASIACRYPT 2012*, pages 134–153. Springer (LNCS 7658), 2012.

BHR12b. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM CCS 2012*, pages 784–796. ACM, 2012.

BMR90. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *Proceedings of the twenty-second annual ACM Symposium on Theory of Computing (STOC 1990)*, pages 503–513. ACM, 1990.

BMR16. Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *ACM CCS 2016*, pages 565–577. ACM, 2016.

CD97. Ronald Cramer and Ivan Damgård. Linear zero-knowledge - A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 436–445, 1997.

CGM16. Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *CRYPTO 2016*. Springer (LNCS 9816), 2016.

CKMZ14. Seung Geol Choi, Jonathan Katz, Alex J. Malozemoff, and Vassilis Zikas. Efficient three-party computation from cut-and-choose. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 513–530, 2014.

EGL85. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

FNO15. Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *EUROCRYPT 2015*, pages 191–219. Springer (LNCS 9057), 2015.

GGP10. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO 2010*, pages 465–482. Springer (LNCS 6223), 2010.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT 2013*, pages 626–645. Springer (LNCS 7881), 2013.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO 2008*, pages 39–56. Springer (LNCS 5157), 2008.

GLNP15. Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In *ACM CCS '15*, pages 567–578. ACM, 2015.

GOM16. Irene Giacomelli, Claudio Orlandi, and Jesper Madsen. Zkboo: Faster zero-knowledge for boolean circuits. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*, pages 321–340. Springer (LNCS 6477), 2010.

HMR15. Zhangxiang Hu, Payman Mohassel, and Mike Rosulek. Efficient zero-knowledge proofs of non-algebraic statements with sublinear amortized cost. In *CRYPTO 2015*, pages 150–169. Springer (LNCS 9216), 2015.

JKO13. Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS 2013*, pages 955–966. ACM, 2013.

KKKS15. Carmen Kempka, Ryo Kikuchi, Susumu Kiyoshima, and Koutarou Suzuki. Garbling scheme for formulas with constant size of garbled gates. In *ASIACRYPT 2015*, pages 758–782. Springer (LNCS 9452), 2015.

KKL$^+$16. Vladimir Kolesnikov, Hugo Krawczyk, Yehuda Lindell, Alex J. Malozemoff, and Tal Rabin. Attribute-based key exchange with general policies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1451–1463, 2016.

KKS16. Carmen Kempka, Ryo Kikuchi, and Koutarou Suzuki. How to circumvent the two-ciphertext lower bound for linear garbling schemes. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 967–997, 2016.

KMR14. Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. Flexor: Flexible garbling for xor gates that beats free-xor. In *CRYPTO 2014*, pages 440–457. Springer (LNCS 8617), 2014.

Kol05. Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In *ASIACRYPT 2005*, pages 136–155. Springer (LNCS 3788), 2005.

KR06. Yael Tauman Kalai and Ran Raz. Succinct non-interactive zero-knowledge proofs with preprocessing for LOGSNP. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 355–366, 2006.

KS08. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *International Colloquium on Automata, Languages, and Programming (ICALP 2008)*, pages 486–498. Springer, 2008.

Lin11. Yehuda Lindell. Highly-efficient universally-composable commitments based on the ddh assumption. In *EUROCRYPT 2011*, pages 446–466. Springer, 2011.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC 2012*, pages 169–189. Springer, 2012.

LP07. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT 2007*, pages 52–78. Springer (LNCS 4515), 2007.

LP11. Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *TCC 2011*, pages 329–346, 2011.

LR15. Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *ACM CCS 2015*, pages 579–590. ACM, 2015.

MNPS04. Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay-secure two-party computation system. In *USENIX Security Symposium 2004*, volume 4, 2004.

MRZ15. Payman Mohassel, Mike Rosulek, and Ye Zhang. Fast and secure three-party computation: The garbled circuit approach. In *ACM CCS 2015*, pages 591–602. ACM, 2015.

NPS99. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce 1999*, pages 129–139. ACM, 1999.

PSSW09. Benny Pinkas, Thomas Schneider, Nigel P Smart, and Stephen C Williams. Secure two-party computation is practical. In *ASIACRYPT 2009*, pages 250–267. Springer (LNCS 5912), 2009.

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

Ser14. I. S. Sergeev. Upper bounds for the formula size of symmetric boolean functions. *Russian Mathematics*, 58(5):30–42, 2014.

Sha79. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

SS10. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS 2010*, pages 463–472. ACM, 2010.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *EUROCRYPT 2015*, pages 220–250. Springer (LNCS 9057), 2015.

A Zero-Knowledge from Garbled Circuits: Required Functionalities

Here we describe the required ideal functionalities. The Zero-knowledge functionality is detailed in Fig. 14. The $\mathcal{F}_{\mathsf{COT}}$ and $\mathcal{F}_{\mathsf{COM}}$ functionalities are provided in Fig. 15 and Fig. 16 respectively. The $\mathcal{F}_{\mathsf{COT}}$ functionality can be securely realised in the framework of [PVW08] with an augmentation for the **Open-all** property, as discussed in [JKO13]. The $\mathcal{F}_{\mathsf{COM}}$ functionality can be securely and efficiently realised as well [Lin11].

- This is the ideal functionality for proving in zero-knowledge to a Verifier $\mathcal{V}$ that a Prover $\mathcal{P}$ possesses a witness w for instance x as per relation R. As in [JKO13] we use an ideal functionality to succinctly capture our requirements of a zero-knowledge protocol.

$$\mathcal{F}_{\mathsf{ZK}}^{\mathsf{R}}$$

1. Receive $(\texttt{prove}, sid, x, w)$ from P and $(\texttt{verify}, sid, x')$ from V
2. **if** $x = x'$ and $R(x, w) = 1$ **then** output $(\texttt{verified}, x)$ to V

Fig. 14: The Zero-knowledge functionality

- This is the ideal functionality for Committing Oblivious Transfer, borrowed from [JKO13]. A Sender S provides two messages, of which a Receiver R chooses to receive one. S doesn't know which message R chose, and R has no information about the message it didn't choose. Upon receiving a signal from S, the functionality reveals both messages to R.

$$\mathcal{F}_{\mathsf{COT}}$$

1. **Choose:** Receive $(\texttt{choose}, id, b)$ from R, where $b \in \{0, 1\}$. If no message of the form $(\texttt{choose}, id, \cdot)$ exists in memory, store $(\texttt{choose}, id, b)$ and send $(\texttt{chosen}, id)$ to S.
2. **Transfer:** Receive $(\texttt{transfer}, id, tid, m_0, m_1)$ from S, where $m_0, m_1 \in \{0, 1\}^\kappa$. If no message of the form $(\texttt{transfer}, id, tid, \cdot, \cdot)$ exists in memory, and a message of the form $(\texttt{choose}, id, b)$ exists in memory, then send $(\texttt{transferred}, id, tid, m_b)$ to R.
3. **Open-all:** Receive $(\texttt{open-all})$ from the S. Send all messages of the form $(\texttt{transfer}, id, tid, m_0, m_1)$ to R.

Fig. 15: The Ideal Committing OT functionality

- The ideal commitment functionality, borrowed from [JKO13].
- A Sender S commits to a message m, which she later reveals to the receiver R. S is 'bound' to only the message that she committed, while the message is hidden from R until S opens her commitment.

$$\mathcal{F}_{\mathsf{COM}}$$

1. **Commit:** Receive $(\texttt{commit}, id, m)$ from the sender, where $m \in \{0, 1\}^*$. If no such message already exists in memory, then store $(\texttt{commit}, id, m)$ and send $(\texttt{committed}, id, |m|)$ to R.
2. **Reveal:** Receive $(\texttt{reveal}, id)$ from S, send $(\texttt{reveal}, id, m)$ to R if corresponding $(\texttt{commit}, id, m)$ exists in memory.

Fig. 16: The Ideal Commitment Functionality