A New Framework for Garbled Circuits

Tomer Ashur[1,2], Efrat Cohen[3], Carmit Hazay[4], and Avishay Yanai[5]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
`[firstname].[lastname]@esat.kuleuven.be`
[2] TU Eindhoven, Eindhoven, the Netherlands
`t.[lastname]@tue.nl`
[3] Bar-Ilan University, Ramat Gan, Israel
`[firstname].Choen@biu.ac.il`
[4] Bar-Ilan University, Ramat Gan, Israel
`[firstname].[lastname]@biu.ac.il`
[5] VMware Research, Herzliya, Israel
`yanaia@vmware.com`

Abstract. Garbled circuits are a fundamental cryptographic building block to encode Boolean circuits as a sequence of encrypted values. This sequence allows two parties to securely evaluate the circuit, *e.g.*, without revealing their respective inputs. At the heart of any garbling scheme lies a randomized algorithm projecting the plain values into a larger domain. Emerging from a large body of work, the common paradigm meets two implicit properties: the circuit is garbled progressively gate-wise; and all underlying algorithms are linear. In this setting, the communication complexity is measured in the number of sent ciphertexts and shown to be optimal with a scheme sending two ciphertexts per AND gate and no ciphertexts per XOR gate (Zahur, Rosulek and Evans, Eurocrypt'15).
We revisit the common paradigm and extend the seminal work of Bellare, Hoang, and Rogaway from CCS 2012 to present for the first time an abstraction of the garbling algorithm itself. This abstraction highlights how Yao's work (Yao, FOCS'86) and all its optimizations focused on improving just one aspect of the garbling. We then discuss how improving the other aspects could provide new ways to overcome the limitations of existing schemes. As a proof of concept we present a non-bijective scheme avoiding Zahur *et al.*'s bound, achieving a communication complexity of a single data item which is not a ciphertext.

Keywords: Garbling Schemes· Abstraction· Lower Bound

1 Introduction

The theory and practice of garbled circuits has been the focus of a long line of research, starting from the seminal work of [Yao86]. The main application of garbled circuits is constant round secure two-party computation protocol (2PC) [LP09], where two parties perform a distributed computation, protecting to the extent possible the secrecy of their inputs and the correctness of the outputs. This protocol proceeds by having the sender randomly generate an encoded version of the circuit, referred to as a *garbled circuit*, together with *input labels*.[6] It sends the garbled circuit to the receiver along with the input labels corresponding to the sender's input values, and allows the receiver to obtain the labels corresponding to their own input values using oblivious transfer (OT). From the garbled circuit and the input labels, the receiver can compute the output.

The common paradigm in garbled circuit operates at the gate level where the truth table of each gate is encoded separately (this is also known as the 'gate-by-gate paradigm'). The garbling procedure follows by choosing a pair of labels for each wire and encrypting each gate using an encryption scheme that takes a pair of input labels as a secret key, and an output label as plaintext. The underlying cryptographic primitive is symmetric (*e.g.*, a pseudorandom function (PRF) or a correlation robust hash function) which yields extremely fast algorithms. This paradigm led to a long sequence of successful optimizations that established garbled circuits as a practical tool for achieving 2PC, *e.g.*, [BMR90,NPS99,KS08,PSSW09,KMR14,ZRE15].

The complexity of garbling schemes. Most of the effort in pushing the practical limitations of garbling schemes has been devoted to reducing the communication complexity of the garbled circuit size, which is measured by the number of ciphertexts sent per gate. By now, the communication complexity of garbled circuits (in the random oracle model) requires two ciphertexts for each AND gate and no ciphertexts for XOR gates [ZRE15], which has also been shown to be optimal. On the other hand, all garbling algorithms require at least four calls to the cryptographic object (*i.e.*, a function or an oracle) per AND gate. We are not aware of any previous works trying to reduce the computation complexity.

A property met by all garbling schemes is projectivity [BHR12]; in a projective scheme, each circuit wire is encrypted by one of two possible labels, where one label corresponds to 0 and the other to 1. (this property is also known as decomposability, see [App17]). Projective garbling schemes imply input encoding size that grows with $|x| \cdot \kappa$ where κ is the security parameter.

The [ZRE15] lower bound. Towards explaining our new abstraction, it is instructive to examine the limitations imposed by the lower bound on the number of ciphertexts per gate. In this paper, Zahur *et al.* considered a model capturing all exiting techniques for practical garbling (including followup schemes), proving that it is not possible to garble a single AND gate with a single ciphertext. This proof only applies for linear garbling schemes, namely, when both the

[6] The existing literature interchangeably uses the terms *keys* and *labels* (and sometimes also *tokens* which we do not use). In this paper we make a distinction between keys and labels. We use the term *keys* when we want to stress that we speak of a random input to a cryptographic primitive (*e.g.*, the secret-key of a block cipher); the term *label* is used when we want to stress that we speak of an encoded wire value in an analogue way to a plaintext or a ciphertext. In previous works, these notions converge since the wire labels are used as keys to *e.g.*, a PRF. However, as this is not the case in our scheme, making the distinction necessary.

garbling and the evaluation algorithms only use linear operations outside the cryptographic primitive, following the gate-by-gate paradigm.

This lower bound implies that any improvement over current schemes will require a radically different approach to garbled circuits.

Prior related work. With the aim of circumventing [ZRE15]'s lower bound, Kempka *et al.* [KKS16] introduced a new garbling scheme deviating from the linear garbling model. Their scheme allows to encode an AND gate with fewer than two κ-bit ciphertexts in cases where at least one of the gate's input wires is a circuit input wire. Similarly to ours, their proof also holds in the random oracle model. Nevertheless, our contributions are fundamentally different. Whilst [KKS16] presented a new scheme, we provide a new abstraction which allows to deviate from the standard garbling approach, offering a new scheme merely as a proof of concept.

1.1 Our Contribution

Our starting point is the observation that all previous works focused on the input domain: a gate is associated with *four* labels, each label encoding a single value of one input wire. Each pair of labels (four in total) is used as a *secret key* to an encryption scheme, hiding the output labels.[7] Recalling the distinction between keys and labels from Footnote 1, we suggest that since the goal is to hide the output labels, of which there are only two, only *two* keys (and therefore **only two primitive calls**) are required.

To provide a theoretical framework for this observation, we abstract the garbling procedure and conclude that it is fully parameterized by three functions (f_0, f_1, f_2) and an ordering on the gates.

To show that this abstraction is meaningful, we first show that it captures all previous work. In particular, we show that Yao's approach garbled circuits is but one instance of this abstraction influencing all subsequent optimizations. Furthermore, the new abstraction facilitates a nuanced discussion of properties associated with garbling schemes. For example, projectivity is a property of f_0 while linearity is a property of f_1.

Focosuing on f_1, we generalize linearity and identify a new property: injectivity. A garbling scheme is said to be *injective* if it maps the *four input labels* into *four distinct keys*. Yao's garbling, its extensions, and in particular, linear schemes, are all injective garbling schemes.

Our observation that an output wire is associated with only two labels, and therefore, requires a shorter source of randomness is addressed by the notion of non-injective schemes. Non-injective schemes map the four initial labels into fewer than four encryption keys. Non-injective schemes are by definition non-linear, allowing them in principle to circumvent [ZRE15]'s lower bound. To show that this is indeed the case, we instantiate such a scheme via what we call

[7] In [BHR12], this encryption scheme was modeled as a *dual-key encryption scheme*.

a compress-then-collapse garbling approach. We prove that this construction satisfies the privacy and authenticity security definitions due to [BHR12]. The scheme achieves a communication complexity of a *single data item* for both AND and XOR gates. This data item does not encode a value, hence we do not consider it to be a ciphertext (or a label).

Proving the security of this scheme requires new tools and proof techniques; all of independent interest. As a by-product of our novel proof technique, we achieve a new notion of authenticity, namely *strong authenticity*, which prevents the evaluator from learning the inactive labels of any wire, not just circuit output ones. Strong authenticity, which is of independent interest, is then used in our privacy proof.

We stress that this new garbling scheme is not meant as a drop-in replacement for existing garbling schemes, and in many ways it is inferior to the state of the art. Instead, it serves to show the power of our new paradigm, hoping that future work would be able to utilize the abstraction better, eventually pushing the envelope with respect to linear schemes.

2 Our New Abstraction

In this section we present our new abstraction. After recalling some preliminaries in Subsections 2.1, we present and explain our new abstraction in 2.2. Subsection 2.3 gives an overview of possible research directions arising naturally from this abstraction.

2.1 Preliminaries

We provide te definitions and notations we use throughout the paper, and recall the work of Bellare *et al.*

Basic notations. We denote the computational and statistical security parameters by κ and σ, respectively, ℓ is the labels' length. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for any positive polynomial $p(\cdot)$ and any sufficiently large κ it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote Probabilistic Polynomial-Time and denote by $X \overset{c}{\approx} Y$ computational indistinguishability between the two ensembles X and Y. We denote by $[n]$ the set of elements $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$. We use $\wedge$ for the AND operation on two bits and & for bitwise AND on strings. For a bit string X we denote by $X[i]$ the i-th bit of X.

Circuit syntax. A Boolean circuit $\mathbf{C} : \{0,1\}^n \to \{0,1\}^m$ has n input wires enumerated by the indices $1, \ldots, n$, and m output wires enumerated by $n + q - m + 1, \ldots, n + q$, where $q = |\mathbf{C}|$ is the number Boolean gates.[8] The output wire of gate j (also denoted by g^j) is $n + j$, which also implies that the gates satisfy

[8] The convention is to comprise the circuit out of XOR and AND gates with fan-in 2.

a topological order, allowing to speak of $g^i > g^j$ when $i > j$. On occasion, we abuse notation and use g as a synonym for the binary function described by this gate. Namely, $g^j(a, b)$ is the result of the binary function of gate g^j on the binary inputs a and b. For example, if g^j is an XOR gate then $g^j(a, b) = a \oplus b$. The interpretation would always be clear from the context.

The Abstraction due to Bellare *et al.* Our starting point is the abstraction due to Bellare *et al.* from [BHR12] that formalized garbling schemes using four algorithms $(\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$.[9] At the heart of every garbling scheme lies the randomized garbling algorithm Gb. As stated above, the most common type of a garbling algorithm Gb encodes the circuit gate-by-gate, returning a set of garbled gates; each such gate is described by a set of ciphertexts, obtained by encrypting the output labels using the input labels as an encryption key. [BHR12] refers to this kind of encryption as a *dual-key cipher*).

The corresponding encoding algorithm En that is associated with the above garbling algorithm, outputs a label $K^j_{x_j} \in \{K^j_0, K^j_1\}$ for every circuit input wire $j \in [n]$, where x_j is the input value on wire j.

The evaluation algorithm Ev that is carried out by the evaluator (or the receiver in the 2PC application) is defined as follows. For each gate g in some topological order: let K^A, K^B be the labels associated with the input wires of g as obtained by the evaluator. Then, if g is an XOR gate, compute $K^c = K^A \oplus K^B$ (as stated by the free-XOR optimization [KS08]). Otherwise, if g is an AND gate decrypt one of the ciphertexts within the garbled table of g. Namely, set $K^C = \mathsf{Dec}(K^A, K^B; c_i)$ for $0 \leq i \leq 3$, where c_i is chosen as stated by the point-and-permute optimization [BMR90].

The decoding algorithm De outputs the plain value $y = \mathbf{C}(x)$. This mapping translates the output label K^j, associated with an output wire j, to the bit b that K^j encodes.
The algorithms above are captured formally by Definition 1.

Definition 1 (Garbling scheme; [BHR12]). *A garbling scheme* $(\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev})$ *consists of four polynomial-time algorithms defined as follows:*

- $(F, e, d) \leftarrow \mathsf{Gb}(1^\kappa, \mathbf{C})$*: is a probabilistic algorithm that takes as input a circuit* $\mathbf{C}$ *with* n *input wires and* m *output wires; and returns a garbled circuit* F*, a set of encoding labels* e*, and a set of decoding labels* d*.*
- $X := \mathsf{En}(e, x)$ *is a deterministic algorithm that takes as input an encoding information* e *and a plain value* x*; and returns the value's encoding* X*.*
- $\{\perp, y\} := \mathsf{De}(Y, d)$*: is a deterministic algorithm that takes as input a decoding information* d *and an output label* Y*; and returns either the failure symbol* $\perp$ *or a plain value* y*.*
- $Y := \mathsf{Ev}(F, X)$*: is a deterministic algorithm that takes as input a garbled circuit* F *and a set of input labels* X*; and returns an output label* Y*.*

[9] For simplicity we omit the fifth algorithm ev and implicitly assume that the parties agree to a certain Boolean circuit as the function's description.

Properties of Garbled Circuits

Correctness. We say that a garbling scheme is correct if for any $\kappa \in \mathbb{N}$, for any polynomial-size circuit $\mathbf{C}$, all inputs x in the domain of $\mathbf{C}$, and for all (F, e, d) outputs of $\mathsf{Gb}(1^\kappa, \mathbf{C})$ such that $X := \mathsf{En}(e, x)$ and $Y := \mathsf{Ev}(F, X)$, it holds that $y = \mathbf{C}(x)$ where $y := \mathsf{De}(Y, d)$.

Privacy. We say that a garbling scheme is secure if there exists a PPT algorithm $\mathcal{S}$ such that for any family of polynomial-size circuits $\mathbf{C}_\kappa$ and sequence of polynomial-size inputs $\{x_\kappa\}_\kappa$ (polynomial in κ),

$$\{(F, d, e) \leftarrow \mathsf{Gb}(1^\kappa, \mathbf{C}_\kappa); X := \mathsf{En}(e, x_\kappa) : (F, X, d)\}_\kappa \overset{\text{c}}{\approx} \{y = \mathbf{C}(x_\kappa) : \mathcal{S}\left(1^\kappa, \mathbf{C}_\kappa, y\right)\}_\kappa.$$

Authenticity. In the authenticity game the adversary receives a set of garbled circuits, input labels, and their corresponding output labels; and outputs a new output label. Authenticity is achieved when the adversary's label corresponds to a valid output only with negligible probability. Namely, the probability that there exists a PPT adversary $\mathcal{A}$ such that the following event occurs with non-negligible probability, is negligible.

$$(F, d, e) \leftarrow \mathsf{Gb}(1^\kappa, \mathbf{C}_\kappa); X := \mathsf{En}(e, x_\kappa); y := \mathsf{De}(Y, d); y \neq \mathbf{C}_\kappa(x_\kappa) : Y \leftarrow \mathcal{A}(F, X);$$

That is, the adversary cannot output an incorrect, yet valid, decoding information.

Strong authenticity. The definition above for authenticity is generic to any (projective) garbling scheme. In our proof we consider a stronger definition. Each wire is associated with two labels and in an honest evaluation of the garbled circuit the evaluator obtains one label per wire. The labels known to the evaluator are said to be *active* labels, whereas the rest are said to be *inactive*. Strong authenticity is achieved when the adversary can obtain an inactive label of *any* wire (not necessarily an output wire) only with negligible probability.

2.2 Abstracting the Garbling Procedure Anew

Having established the garbling notation, we are now ready to discuss our new abstraction. In this work we focus our attention on the garbling algorithm Gb. As stated above, our first observation is that [BHR12]'s garbling algorithm can be further abstracted using three functions (f_0, f_1, f_2), for *label sampling*, *key extraction* and *output hiding*, respectively (see Algorithm 1).

The Label Sampling Function $\boldsymbol{f_0}$ This function assigns an ℓ-bit label K_j to each possible value that wire j can take. Collectively, the set of labels associated with the wire is denoted by $\{K_j\}$. In particular, Yao's scheme and all subsequent optimizations decompose the circuit's input into bits and each bit is assigned a label (See also [App17]).

Algorithm 1 Abstraction of the Garbling Procedure

```
1: for each wire $j$ do                                   ▷ Label sampling (Section 2.2)
2:     $\{K_j\} \leftarrow f_0()$
3: end for
4: for every gate $g^i$ do
5:     The set of input labels is $\{K_{g^i}\}$
6:     $\{S_{g^i}\} \leftarrow f_1(\{K_{g^i}\})$          ▷ Key Extraction (Section 2.2)
7:     $G^i \leftarrow f_2(\{S_{g^i}\}, \{K^C\})$        ▷ Output hiding (Section 2.2)
8: end for
9: Return $(F, e, d)$ where $F = \{G^i\}_{i \in [n+1, n+q]}$, $e = \{\{K_i\}\}_{i \in [n]}$ and $d = \{\{K_i\}\}_{i \in [n+q-m+q, n+q]}$
```

The Key Extraction Function f_1 Under the gate-by-gate paradigm, this function is applied to each gate independently. When applied to g^i it receives as input the set of labels associated with g^i's input wires, which we denote hereafter by $\{K_{g^i}\}$; and outputs a set of keys, which we denote hereafter by $\{S_{g^i}\}$. These keys are later used by the cryptographic primitive in f_2 to hide the output labels contained in the set $\{K^C\}$.

We say that a scheme is injective if it maps the input set $\{K_{g^i}\}$ to a key set $\{S_{g^i}\}$ of the same size. More specifically, following Yao's approach, all previous works suggested injective schemes with $|\{K_{g^i}\}| = |\{S_{g^i}\}| = 4$, forcing four calls to the cryptographic primitive in f_2 (*e.g.*, by means of a PRF; or more generally, by means of a dual-key cipher). Non-injective schemes deviate from this formulation by reducing the size of the output set, *i.e.*, $|\{K_{g^i}\}| > |\{X_{g^i}\}|$, thus also reducing the number of primitive calls in f_2 and motivating our distinction between labels (*i.e.*, the inputs to this function) and keys (*i.e.*, its outputs).

The Output Hiding Function f_2 This function uses the outputs of f_1, *i.e.*, the set $\{S_{g^i}\}$, as keys to a cryptographic primitive hiding the set of output labels $\{K^C\}$. Optimizing this part was the focus of almost all prior work, including finding more efficient primitives [BHKR13], fixing one of the ciphertexts to a known value [NPS99], or treating linear and non-linear gates differently [KS08]; however, they all assumed that there are four different keys and therefore, four calls to the cryptographic primitive are necessary.

2.3 A Promising Vision

In fact, all previous works have made three implicit assumptions: (i) that f_0 maps $\mathbb{F}_2$ to $\mathbb{F}_{2^\ell}$ (see [App17]); (ii) that f_0 operates in a gate-by-gate manner; and (iii) that f_1 is injective. Combining the three assumptions means that f_1 outputs four different encryption keys. These encryption keys are then used in

f_2, forcing four calls to the cryptographic primitive, and resulting in a garbled table with four rows; *i.e.*, the size of the garbled gate is 4ℓ (cf. [LP09]).[10]

The new abstraction offers several opportunities. A research direction which we do not explore at all is trying to link the individual functions (f_0, f_1, f_2) with security properties *e.g.*, authenticity, privacy, and obliviousness. Achieving this would allow generic constructions, possibly reducing the work effort in coming up with new optimizations.

Considering the individual functions (f_0, f_1, f_2), we note that they have a cascading effect: the output of f_i is the input to f_{i+1} and therefore the output size of the former drives the complexity of the latter. This motivates an approach seeking to apply optimizations as early as possible.

While this paper focuses on f_1, we cannot see a reason why such optimizations would not apply for f_0. For example, we can envision a garbling scheme which does not necessarily decompose the input into bits, but into larger chunks, thus resulting in a smaller encoding of the input space. A second idea to explore is to use labels represented by sets. That is, instead of using bitstrings of length ℓ, the label now consists of a set of bitstrings, all interpreted as encoding the same value. Allowing a plurality of encodings leads in turn to shifting away from the gate-by-gate paradigm by offering garbling schemes operating on larger gadgets.

Another approach for garbling larger gadgets can be obtained from extending our compress-then-collapse approach. Observing that the number of input labels grows with the size of the gadget, but the number of output labels remains the same, a non-injective f_1 can be used to collapse a larger number of intermediate values instead of the four we consider. Furthermore, the compress-then-collapse approach we present in this paper suffers from a few limitations making it sub-optimal for practical use. We do not claim it to be ideal, and in fact, we believe that there are more natural ways to instantiate f_1. Moreover, this work does not try at all to investigate the required security properties of the potential cryptographic primitives it may use, and just conveniently use random oracles to draw attention to the bigger picture. However, it is not clear that such heavy machinery is needed in f_2, and we conjecture that a much weaker (and thus more efficient) primitive would suffice and can be proven secure.

For f_2 we observe that reducing the output size of f_1 can be used to reduce the number of primitive calls. Although intuitively we believe that a random oracle is unavoidable here, future research should look into this in a similar manner to f_1. Finally, as we show in the sequel, the two encryption keys coming out from f_1 can be used directly as a random seed to derive the output label, thus allowing, at least in principle, to reduce the communication complexity.

A Concrete Scheme Concretely, in this paper we use a non-injective function for f_1, resulting in a smaller input to f_2. More specifically, f_1 takes four input labels $K_0^A, K_1^A, K_0^B, K_1^B$; and returns two keys S_0 and S_1 to be used in f_2.

[10] We make a distinction between the actual garbled table size and the number of table entries (ciphertexts) that are communicated. Therefore, optimizations like garbled row reduction reduce the communication but not the table size.

We achieve this by employing two sub-functions in f_1. The first of which, which we model as a random oracle, compresses a 2ℓ-bit pair of labels into an ℓ-bit intermediate value X. The second function then collapses the four ℓ-bit intermediate values into two ℓ-bit keys which are given, together with additional information describing the collapsing function, to f_2.

A second optimization we propose is to use the two ℓ-bit keys as seeds for sampling the output labels rather than as random material to an encryption scheme. This second optimization reduces the communication complexity, allowing to send only the information for collapsing the values, as everything else can be computed directly by the evaluator from known values.

3 A New Approach: Compress-Then-Collapse

Motivated by the new abstraction, we offer a scheme that makes use of non-linear operations. Our novelty is in the garbling and evaluation algorithms which can be found in Algorithms 2–3, respectively. The encoding and decoding algorithms are quite standard and are provided in Algorithms 4–5 for completeness.

Below, we provide an overview of the new scheme and an intuitive explanation, deferring a formal proof to Sections 4–5.

3.1 Garbling

Key Extraction Similarly to the classical Yao's garbled circuit, f_1 first splits the four inputs, namely $K_0^A, K_1^A, K_0^B, K_1^B$ coming out from f_0, into the pairs:

$$(K_0^A, K_0^B), (K_0^A, K_1^B), (K_1^A, K_0^B), (K_1^A, K_1^B).$$

Compress. The function $f_{1,0}$, which we model as a random oracle, is used to compress each pair into a random string of length ℓ, *i.e.*,

$$\begin{aligned} X_{00} &= f_{1,0}(K_0^A, K_0^B) = RO_0(K_0^A, K_0^B); \\ X_{01} &= f_{1,0}(K_0^A, K_1^B) = RO_0(K_0^A, K_1^B); \\ X_{10} &= f_{1,0}(K_1^A, K_0^B) = RO_0(K_1^A, K_0^B); \\ X_{11} &= f_{1,0}(K_1^A, K_1^B) = RO_0(K_1^A, K_1^B). \end{aligned}$$

Collapse. These four outputs of the random oracle are given to $f_{1,1}$ to produce ∇ (this is either $\nabla_\oplus$ or $\nabla_\wedge$, depending on the gate type):

$$\nabla \leftarrow f_{1,1}(X_{00}, X_{01}, X_{10}, X_{11})$$

which satisfies the following relation: let $(x_0, y_0) \in \{0,1\}^2$ and $(x_1, y_1) \in \{0,1\}^2$ be two possible inputs to the (ungrabled) gate g^i, where (K_{x_0}, K_{y_0}) and (K_{x_1}, K_{y_1}) denote their respective labels; then

$$\begin{aligned} &g^i(x_0, y_0) = g^i(x_1, y_1) \iff \\ &\qquad f_{1,0}(K_{x_0}, K_{y_0}) \& \nabla = f_{1,0}(K_{x_1}, K_{y_1}) \& \nabla. \end{aligned}$$

Algorithm 2 Algorithm $\mathsf{Gb}(1^\kappa, \mathbf{C})$

```
1:                                                          ▷ Label sampling
2: for every circuit input wire $j \in [n]$ do
3:     Sample $K_0^j, K_1^j \in_R \{0,1\}^\ell$ uniformly at random
4:     Set $e_j = (K_0^j, K_1^j)$                          ▷ Encoding information
5:     Denote by $e_j[x_j]$ the value $K_{x_j}^j$
6: end for
7:                                              ▷ Key Extraction (Section 3.1)
8: for each gate $g$ of $\mathbf{C}$ in topological order do
9:     Compress $(f_{1,0})$:                     ▷ See Section 3.1 (Compress)
10:    $X_{00} = RO_0(K_0^A, K_0^B)$
11:    $X_{01} = RO_0(K_0^A, K_1^B)$
12:    $X_{10} = RO_0(K_1^A, K_0^B)$
13:    $X_{11} = RO_0(K_1^A, K_1^B)$
14:
15:    Collapse $(f_{1,1})$:                      ▷ See Section 3.1 (Collapse)
16:    if $g = \wedge$ then
17:        $\nabla = \nabla_\wedge$                              ▷ See Table 1
18:    else
19:        $\nabla = \nabla_\oplus$                              ▷ See Table 2
20:    end if
21:    $S_0 = X_{00} \& \nabla$
22:    if $g = \oplus$ then
23:        $S_1 = X_{01} \& \nabla$
24:    else  $(g = \wedge)$
25:        $S_1 = X_{11} \& \nabla$
26:    end if
27:                                             ▷ Output hiding (Section 3.1)
28:    $K_0^C = RO_1(S_0)$
29:    $K_1^C = RO_1(S_1)$
30:    $F[g] \leftarrow \nabla$
31: end for
32:                                                   ▷ Decoding information
33: for For every output wire $j \in [n+q-m+1, n+q]$ do
34:     Sample $d_j \in_R \{0,1\}^\ell$ s.t. $\mathsf{lsb}(RO_2(K_0^j, d_j)) = 0$ and $\mathsf{lsb}(RO_2(K_1^j, d_j)) = 1$
35: end for
36: Return $(F, e, d)$ where $e = \{e_j\}$ and $d = \{d_j\}$.
```

To generate ∇, we scan the bits of the four intermediate values $\{X_{00}, X_{01}, X_{10}, X_{11}\}$ to determine which bits preserve the ungarbled gate's logic, and deactivate the rest.

- **AND gate.** Set $\nabla[j] = 1$ if and only if $X_{00}[j] = X_{01}[j] = X_{10}[j]$. In Table 1 this corresponds to Rows 0–1 and Rows 14–15.
- **XOR gate.** Set $\nabla[j] = 1$ if and only if $(X_{00}[j] = X_{11}[j]$ and $X_{01}[j] = X_{10}[j])$. It Table 2 this is captured by Rows 0, 6, 9, and 15.

Other Boolean gates can similarly be obtained, see Appendix B for the details.

The output of $f_{1,1}$ is ∇, which is a public value, accessible to other parts of the garbling algorithm.

Output. Formally, the output of f_1 are the two keys S_0 and S_1, which are obtained by applying $X\&\nabla$ on each of the four outputs of $f_{1,0}$.

Output Hiding At this point, a naive extension of existing approaches would use the keys S_0 and S_1 to encrypt the gate's output labels using a single-key encryption scheme, *i.e.*,

$$C_0 = E_{S_0}(K_0^C);$$
$$C_1 = E_{S_1}(K_1^C),$$

and send (C_0, C_1, ∇) to the evaluator.

However, our second optimization shows that that this is unnecessary. Instead of sampling new labels K_0^C and K_1^C, we can derive them directly from the values S_0 and S_1, even if the later have fewer than ℓ bits of entropy (as long as they have κ bits of entropy). We set

$$K_0^C = RO_1(S_0)$$
$$K_1^C = RO_1(S_1),$$

and send only ∇ to the evaluator.

3.2 Evaluation

The evaluation is presented in Algorithm 3 below. To intuitively understand how

Algorithm 3 Algorithm $\mathsf{Ev}(F, X)$

```
1: for each gate $g^i$ in a topological order do
2:     let $K^A, K^B$ be the active labels associated with the input wires of $g^i$ and let
       $\nabla_{g^i} = F[g]$ (with $\nabla_{g^i} = \nabla_\oplus$ if $g^i$ is a XOR gate and $\nabla_{g^i} = \nabla_\wedge$ if $g^i$ is an AND
       gate)
3:     Compute $K^C = RO_1(RO_0(K^A, K^B)\&\nabla_{g^i})$
4: end for
5: for Each output wire $j$ do
6:     output $Y^j = K^j$
7: end for
```

evaluation works, we focus on a single gate. The evaluator has received exactly one value of the pair $\{K_0^A, K_1^A\}$, exactly one value of the pair $\{K_0^B, K_1^A\}$, and ∇.

Table 1. This table defines $\nabla_\wedge[j]$ as a function in $X_{00}[j], X_{01}[j], X_{10}[j], X_{11}[j]$. In addition, the right side demonstrates how combining $X_{ij}[j]\&\nabla$ collapses into only two distinct values $S_0 = S_{00} = S_{01} = S_{10}$ and $S_1 = S_{11}$. Each row in the table corresponds to one bit-slice of the values $X_{ij}[j]$ for $i, j \in \{0, 1\}$.

	X_{00}	X_{01}	X_{10}	X_{11}	$\nabla_\wedge$	S_{00}	S_{01}	S_{10}	S_{11}
0	0	0	0	0	1	0	0	0	0
1	0	0	0	1	1	0	0	0	1
2	0	0	1	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0
6	0	1	1	0	0	0	0	0	0
7	0	1	1	1	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0
10	1	0	1	0	0	0	0	0	0
11	1	0	1	1	0	0	0	0	0
12	1	1	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0
14	1	1	1	0	1	1	1	1	0
15	1	1	1	1	1	1	1	1	1

Without loss of generality, suppose the received labels are K_0^A and K_0^B. The evaluator calls RO_0 to obtain $X_{00} = RO_0(K_0^A, K_0^B)$. Then, knowing ∇ they compute $S_0 = X_{00}\&\nabla$. Finally, they calls the second random oracle to obtain $K_0^C = RO_1(S_0)$ which they use as the output label.

3.3 Encoding and Decoding

The encoding and decoding algorithms are presented in Algorithms 4-5 below.

Algorithm 4 Algorithm $\mathsf{En}(e, x)$

```
1: for every $j \in [n]$ do
2:     output $K^j_{x_j} = e_j[x_j]$
3: end for
```

3.4 Properties

In this subsection we provide an intuitive discussion about the properties of the new scheme. More formal treatment can be found in Sections 4–5.

Table 2. This table defines $\nabla_{\oplus}[j]$ as a function in $X_{00}[j], X_{01}[j], X_{10}[j], X_{11}[j]$. In addition, the right side demonstrates how combining $X_{ij}[j] \& \nabla$ collapses into only two distinct values $S_0 = S_{00} = S_{11}$ and $S_1 = S_{01} = S_{10}$. Each row in the table corresponds to one bit-slice of the values $X_{ij}[j]$ for $i, j \in \{0, 1\}$.

	X_{00}	X_{01}	X_{10}	X_{11}	$\nabla_{\oplus}$	S_{00}	S_{01}	S_{10}	S_{11}
0	0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0
6	0	1	1	0	1	0	1	1	0
7	0	1	1	1	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0
9	1	0	0	1	1	1	0	0	1
10	1	0	1	0	0	0	0	0	0
11	1	0	1	1	0	0	0	0	0
12	1	1	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0
15	1	1	1	1	1	1	1	1	1

Algorithm 5 Algorithm $\mathsf{De}(Y, d)$

```
1: for $j \in [n + q - m + 1, n + q]$ do
2:     output $\mathsf{lsb}(RO_2(Y^j, d_j))$
3: end for
```

Correctness The correctness of the scheme follows from the definition of $\nabla_{\wedge}$ and $\nabla_{\oplus}$ in Tables 1–2. The ∇ value for each gate is obtained by mapping the ungarbled gate's truth table to the encoded values $\{X_{00}, X_{01}, X_{10}, X_{11}\}$ (for the exact procedure, see Section 3.1). Focusing on bit position j, given a gate and two labels K_a^A, K_b^B, first computing $X_{ab} = RO_0(K_a^A, K_b^B)$, then $S_{ab} = X_{ab} \& \nabla_{\oplus}$, collapses the set $|\{X_{00}, X_{01}, X_{10}, X_{11}\}| = 4$ into the set $|\{S_{00}, S_{01}, S_{10}, S_{11}\}| = 2$ in a way that preserves the ungarbled gate's output logic. Correctness can be verified in a bit-wise manner.

Complexity Compared to a classical Yao circuit, which implies four calls to the cryptographic primitive, our scheme requires six primitive calls: four calls in f_1 and two more calls in f_2. Another caveat is that due to the entropy loss of & (which we analyze separately), the length of labels must be longer than in other schemes, such that they retain κ bits of entropy even after the entropy loss. For example, with $\kappa = 128; \sigma = 40$ where κ, σ are the computational and statistical security parameteres, the scheme requires $\ell = 842$ bits. Other parameter combinations can be found in Section 4.2.

On the other hand, our scheme only requires sending one data item, namely ∇, breaking the bound introduced by [ZRE15] and demonstrating the potential of the new approach. Importantly, while recent optimizations for garbled circuits can achieve similar or even better communication complexity, we stress that these are the result of over 30 years of optimization, while ours is the first proof-of-concept of the new approach.

Security The security of our garbling intuitively follows via four claims. First, modeling $f_{1,0}$ as a random oracle and maintaining an invariant that the evaluator sees only one label per wire. Second, due to correlation intractability of the random oracle, the evaluator can obtain no more than one of $\{X_{00}, X_{01}, X_{10}, X_{11}\}$. Third, choosing a sufficiently large ℓ can be used to compensate for the entropy loss due to the bitwise AND. Finally, relying again on the unpredictability and correlation intractability of the random oracle, this time in f_2, full entropy is restored for each of $\{K_0^C, K_1^C\}$; supporting an inductive argument gate-by-gate. A formal analysis is provided in Sections 4-5.

Adversarial model. Our proof is shown in the presence of probabilistic polynomial-time adversaries but can be extended against super-polynomial adversaries that are restricted by the number of queries they make to the oracles. Namely, this restriction limits the number of queries to a polynomial in the security parameter.

Random orcale. Our proof is applied in the random oracle model. Specifically, the properties we use from the random oracle are that it is hard to invert, and its outputs are independent and have full entropy, regardless of the input distribution. In particular, for a given input and output of the random oracle, x and y, respectively, if the input has κ bits of entropy then with high probability, the entropy in y is full (*i.e.,* the oracle's output length).

Strong authenticity. As a by-product of our new proof technique, we establish a stronger notion of authenticity than the one suggested in [BHR12]. Whereas they defined authenticity in terms of what the adversary (does not) learn about inactive labels of *output wires*, our proof extends this to *all* wires, both internal and output ones. We refer to this new notion as *strong authenticity.*

4 Strong Authenticity

Authenticity of a garbling scheme states that an adversary cannot obtain the inactive labels of circuit *output* wires. We argue a stronger claim where the adversary cannot obtain an inactive label of *any* (both internal and output) wire. The formal definitions for authenticity and strong authenticity appear in Section 2.1. We prove the following

Theorem 1. *Our garbling scheme (Algorithms 2-5) satisfies the definition of strong authenticity (from Section 2.1).*

4.1 Proof

We denote by H and h the random oracle objects RO_0 and RO_1 used in our garbling scheme (see Algorithm 2). For each wire j we denote the active and inactive labels by K^j and $\bar{K}^j$.

Let K be the set of inactive labels and Q be the set of queries made to the random oracle by the adversary. We prove that the probability that $K \cap \hat{Q} \neq \emptyset$ is negligible, where $\hat{Q} = \{H(q) : q \in Q\}$.

Our proof follows an inductive argument by measuring the entropy of inactive labels, showing that given (F, X, d), the entropy is never smaller than κ and thus, inactive labels cannot be (computationally) guessed.

We consider the *apriori* and *posteriori* entropy in $\bar{K}^j$, where the *apriori* entropy of $\bar{K}^j$ is calculated based on a modified garbled circuit, which is a trimmed version of the garbled circuit corresponding to $\mathbf{C}$, such that all gates that depend on wire j are removed (and wire j becomes an output wire). The *posteriori* entropy is calculated based on the unmodified garbled circuit. We use the apriori entropy as a stepping stone to analyse the posteriori entropy. Specifically, suppose that the apriori entropy of $\bar{K}^j$ is r bits, we denote its posteriori entropy by $E(r)$ and prove that when $r = \ell$, for ℓ the length of the labels; then $E(r) \geq \kappa$.

Proof outline. The proof is done by induction. The induction's base is that circuit input wires have $\ell \geq \kappa$ bits of entropy by definition. The induction hypothesis is that if all inactive input labels to a gate g^j have at least κ bits of entropy, its output label has ℓ bits of entropy except with negligible probability.

The inductive argument. Let Q_j be the event that the adversary queries the random oracle on a preimage of the inactive label $\bar{K}^j$. In addition, let $\bar{Q}_j$ be its complement (*i.e.*, the adversary does not query the oracle on a preimage of $\bar{K}^j$). We are interested in the probability of the union $\bigcup_{j \in [n+q]} Q_j$. By the union bound we have

$$\Pr[\bigcup_{j=1}^{n+q} Q_j] \leq \sum_{j=1}^{n+q} \Pr[Q_j], \tag{1}$$

and we show that this probability is negligible.

Induction basis. The basis of the induction consists of all circuit input wires $j \in [n]$. Recall that the apriori entropy of wire j relates to a modified version of the circuit in which j is an output wire and is not an input to any other gate. It is obvious that the inactive labels of these wires have an apriori entropy of ℓ bits since they are chosen uniformly at random from $\{0,1\}^\ell$ by the garbling algorithm. Thus, we conclude that every circuit input wire j has a posteriori entropy of $E(\ell)$ bits.

hypothesis: For every wire $j \in [n+1, n+q]$ it holds that $\bar{K}^j$ has a posteriori entropy of $E(\ell)$ bits.

Inductive step. We assume that the hypothesis holds for every wire $j' < j$ and show that it holds for wire j.

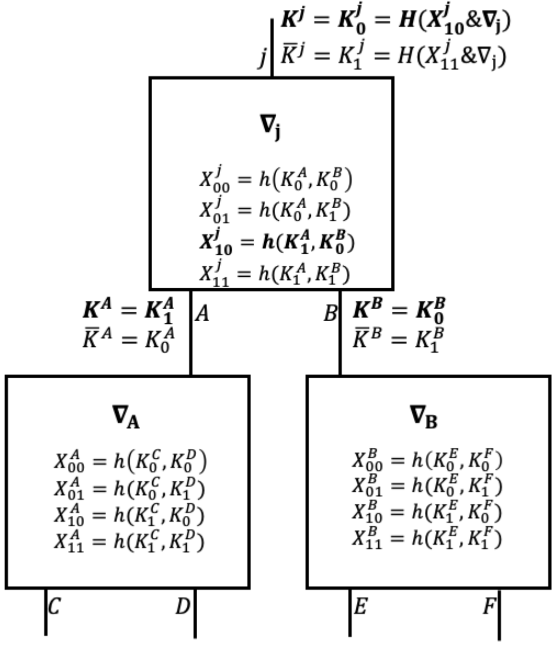
$K^j = K_0^j = H(X_{10}^j \& \nabla_j)$
$\bar{K}^j = K_1^j = H(X_{11}^j \& \nabla_j)$
j
∇_j
$X_{00}^j = h(K_0^A, K_0^B)$
$X_{01}^j = h(K_0^A, K_1^B)$
$X_{10}^j = h(K_1^A, K_0^B)$
$X_{11}^j = h(K_1^A, K_1^B)$
$K^A = K_1^A$
$\bar{K}^A = K_0^A$
A
B
$K^B = K_0^B$
$\bar{K}^B = K_1^B$
∇_A
$X_{00}^A = h(K_0^C, K_0^D)$
$X_{01}^A = h(K_0^C, K_1^D)$
$X_{10}^A = h(K_1^C, K_0^D)$
$X_{11}^A = h(K_1^C, K_1^D)$
∇_B
$X_{00}^B = h(K_0^E, K_0^F)$
$X_{01}^B = h(K_0^E, K_1^F)$
$X_{10}^B = h(K_1^E, K_0^F)$
$X_{11}^B = h(K_1^E, K_1^F)$
C
D
E
F

Fig. 1. Our notation

For a gate with input wires A, B and an output wire j (see Figure 1), by the rule of total probability, the event Q_j is conditioned on the two complementing events: $Q_A \vee Q_B$ and $\overline{Q_A \vee Q_B}$. The former means that the adversary queried the oracle on a preimage leading to any of the inactive labels $\bar{K}^A$ or $\bar{K}^B$; the latter means that it did not. Formally, set $\overline{Q_A \vee Q_B} = \bar{Q}_A \wedge \bar{Q}_B$, then,

$$\begin{aligned}\Pr[Q_j] = \Pr[Q_j \mid Q_A \vee Q_B] \cdot \Pr[Q_A \vee Q_B] \\ + \Pr[Q_j \mid \bar{Q}_A \wedge \bar{Q}_B] \cdot \Pr[\bar{Q}_A \wedge \bar{Q}_B].\end{aligned} \tag{2}$$

The probability $\Pr[Q_j \mid Q_A \vee Q_B] \cdot \Pr[Q_A \vee Q_B]$ is upper bounded by $\Pr[Q_A] + \Pr[Q_B]$ since $\Pr[Q_j \mid Q_A \vee Q_B] \leq 1$ and $\Pr[Q_A \vee Q_B] \leq \Pr[Q_A] + \Pr[Q_B]$ by the union bound. Thus, we can rewrite (2) as

$$\Pr[Q_j] \leq \Pr[Q_A] + \Pr[Q_B] + \Pr[Q_j \mid \bar{Q}_A \wedge \bar{Q}_B] \cdot \Pr[\bar{Q}_A \wedge \bar{Q}_B]$$

We now show that $\Pr[Q_j \mid \bar{Q}_A \wedge \bar{Q}_B]$ is negligible. To see this, first observe that the only data that may leak information about $\bar{K}^j$ is ∇ and the output of the random oracle, when queried on the active labels (*i.e.*, $X = h(K^A, K^B)$). We will now show that given this data the probability of Q_j is negligible.

Without loss of generality let the active labels be K_a^A and K_b^B. The probability that the adversary obtains any of the inactive intermediate values

$$\begin{aligned} X_{a(1-b)} &= h(K_a^A, K_{1-b}^B); \\ X_{(1-a)b} &= h(K_{1-a}^A, K_b^B); \\ X_{(1-a)(1-b)} &= h(K_{1-a}^A, K_{1-b}^B), \end{aligned}$$

is negligible since the posteriori entropy of the inactive labels K_{1-a}^A and K_{1-b}^B is $E(\ell)$ as their apriori entropy was ℓ. We will show in Section 4.2 that $E(\ell) \geq \kappa$. Thus, with high probability $X_{a(1-b)}, X_{(1-a)b}$ and $X_{(1-a)(1-b)}$ have ℓ bits of entropy.

Now, to have queried a preimage of $\bar{K}^j$ the adversary had to query

$$\begin{aligned} q_1 &= X_{a(1-b)} \& \nabla; \\ q_2 &= X_{(1-a)b} \& \nabla; \text{or} \\ q_3 &= X_{(1-a)(1-b)} \& \nabla. \end{aligned}$$

On the other hand, since each of $X_{a(1-b)}, X_{(1-a)b}$, and $X_{(1-a)(1-b)}$ has ℓ bits of entropy, the only source of leakage about q_1, q_2, q_3 is ∇. In particular, for a position $i \in [\ell]$, having $\nabla[i] = 0$ implies that $q_1[i] = q_2[i] = q_3[i] = 0$; on the other hand, $\nabla[i] = 1$ masks the values $q_1[i], q_2[i]$ and $q_3[i]$.

We note that the larger the number of 1's in ∇ the harder it is for the adversary to guess the preimages q_1, q_2, q_3 of $\bar{K}^j$, as each position i with $\nabla[i] = 1$ adds a single bit of entropy to q_1, q_2, q_3. In particular, if the number of 1's is κ, then the adversary guesses q_1, q_2, q_3 with only a negligible probability. In Section 4.2 we calculate the sufficient length ℓ for which the above condition holds. That is, the minimal ℓ for which, with high probability, the number of 1's in ∇ is greater than κ.

We obtain that $\Pr[Q_j \mid \bar{Q}_A \wedge \bar{Q}_B] = \mathsf{neg}(\kappa)$, and therefore we can rewrite (2) as

$$\Pr[Q_j] \leq \Pr[Q_A] + \Pr[Q_B] + \mathsf{neg}(\kappa).$$

The probability of the adversary querying the preimage of the inactive label associated with output wire j (*i.e.*, $\Pr[Q_j]$) is the sum of the probabilities that it queried the preimage of the inactive labels associated with predecessor wires A and B (*i.e.*, $\Pr[Q_A]$ and $\Pr[Q_B]$), and a negligible probability. Following from the induction's assumption, both $\Pr[Q_A]$ and $\Pr[Q_B]$ are negligible, and therefore so is $\Pr[Q_j]$. We conclude that the probability that the adversary queried the preimage of $\bar{K}^j$ is negligible and therefore its posteriori entropy is $E(\ell)$. This completes the induction.

We see that the union (1) is the sum of negligible events, and is therefore neghligible. $\square$

4.2 Setting Labels' Length

The authenticity of our construction relies on having at least κ positions i in ∇ such that $\nabla[i] = 1$. This is required since these are the positions that mask

$q_1[i], q_2[i]$, and $q_3[i]$. In this section we determine an appropriate length for ℓ. Concretely, we are looking for the minimal ℓ such that the number of positions in which ∇ equals 1 is larger than κ in the worst case. We do so by inspecting $\nabla_\wedge$ and $\nabla_\oplus$ as functions of $X_{00}, X_{01}, X_{10}, X_{11}$ (see Tables 1-2).

Note that the intermediate values $X_{00}, X_{01}, X_{10}, X_{11}$ in our garbling scheme are the output of a random oracle call; therefore, they are uniformly distributed and independent. Ergo, each entry in Tables 1-2 is accessed with probability 1/16.

We start by considering an AND gate. Given $X_{ab} = h(K_a^A, K_b^B)$ and $\nabla_\wedge$ where K_a^A, K_b^B are the active labels associated with the input wires of that gate, there are exactly four rows (in Table 1) that contribute to the entropy of the inactive key $S_{(1-c)} = (X_{(1-c)(1-c)} \& \nabla_\wedge)$ where $c = a \wedge b$.[11] Similarly, for an XOR gate, given $X_{ab} = h(K_a^A, K_b^B)$ and $\nabla_\oplus$ there are exactly four rows (in Table 2) that contribute to the entropy of the inactive key $S_{1-c} = X_{a(1-b)} \& \nabla_\oplus$ where $c = a \oplus b$. The above holds for every $a, b \in \{0, 1\}$.

Limiting our scope to those positions i for which $\nabla_\wedge[i] = 1$ (resp., $\nabla_\oplus[i] = 1$), the probability that $S_{(1-c)}[i]$ is equal 0 is 1/2. That is, such positions contribute a single bit of entropy. Since X_{00}, X_{01}, X_{10},and X_{11} are uniformly random values we have that $\Pr[\nabla_\wedge[i] = 1] = \Pr[\nabla_\oplus[i] = 1] = 1/4$ per the rows in Tables 1–2.

It remains to find what is the minimal ℓ for which, with high probability (over a statistical security parameter σ), the number of positions i satisfying $\nabla_\wedge[i] = 1$ (similarly for $\nabla_\oplus[i] = 1$) is greater than κ. Specifically, let N_1 be a random variable that counts the number of 1's in ∇ (either $\nabla_\wedge$ or $\nabla_\oplus$), we are looking for

$$\ell = \min_{\ell'} : \Pr[N_1 < \kappa] = \sum_{i=0}^{\kappa-1} \binom{\ell'}{i} 0.25^i \cdot 0.75^{\ell' - i} \leq 2^{-\sigma}$$

That is, we consider it as a Bernoulli experiment repeated ℓ times with $p = 0.25$.

By means of the script provided in Appendix A, we obtain the values above, where the minimal length ℓ is the intersection between the statistical and computational security parameters, σ and κ, respectively.

κ/σ	40	60	80
128	842	944	1035
160	1003	1112	1209
256	1468	1597	1711

[11] The intermediate value $X_{(1-c)(1-c)}$ is chosen arbitrarily among the inactive intermediate values.

5 Privacy

We follow the privacy definition of [BHR12] given in section 2.1. We first describe the simulation and then show indistinguishability via a hybrid argument.

Theorem 2. *Our garbling scheme (Algorithms 2-5) satisfies the definition of privacy (from Section 2.1).*

5.1 The simulation

As before, we use the term *active* labels to refer to the labels that are used by the honest evaluator to evaluate the garbled circuit whereas *inactive* labels refer to the rest. Note that given an input $x \in \{0,1\}^n$ to the circuit, each wire j is associated with one active and one inactive label, K^j and $\bar{K}^j$, respectively. We denote by H, h and $\tilde{H}$ the random oracle objects RO_0, RO_1 and RO_2, respectively, used in our garbling scheme (see Algorithm 2).

The simulator $\mathcal{S}$ outputs the label K_0^j as the garbled input for every circuit input wire j (as it does not know the actual input to that wire). In addition, for each gate, rather than using the actual inactive labels, the simulator uses a uniformly random labels. Formally, given $y = \mathbf{C}(x)$ (where $\mathbf{C}$ and x picked by the adversary) the simulator proceeds as follows:

Algorithm $\mathcal{S}(1^\kappa, \mathbf{C}, y)$

1. For every circuit input wire $j \in [n]$ sample independent labels $K_0^j, K_1^j \in \{0,1\}^\ell$ at uniform.
2. For each gate g of $\mathbf{C}$ in a topological order:
   (a) Let A, B and C be g's input wires and output wire.
      i. $X_{00} = h(K_0^A, K_0^B)$
      ii. $X_{01} = h(K_0^A, K_1^B)$
      iii. $X_{10} = h(K_1^A, K_0^B)$
      iv. $X_{11} = h(K_1^A, K_1^B)$
      v. If $g = \wedge$ set $\nabla = \nabla_\wedge$ and if $g = \oplus$ set $\nabla = \nabla_\oplus$, where $\nabla_\wedge$ and $\nabla_\oplus$ are defined in Tables 1-2, as a function of $X_{00}, X_{01}, X_{10}, X_{11}$.
      vi. Let c be the bit carried by wire C when evaluating $\mathbf{C}(0^n)$. Compute K_c^C as in $\mathsf{Gb}(1^\kappa, \mathbf{C})$ (see Algorithm 2) and pick $K_{(1-c)}^C$ uniformly at random.
   (b) Set $F[g] \leftarrow \nabla$
3. For every output wire $j \in [n+q-m+1, n+q]$
   (a) Let b be the bit carried by wire j when evaluating $\mathbf{C}(0^n)$ and let y_j be the correct output bit on wire j.
   (b) Sample $d_j \in_R \{0,1\}^\ell$ s.t. $\mathsf{lsb}(\tilde{H}(K_b^j, d_j)) = y_j$ and $\mathsf{lsb}(\tilde{H}(K_{1-b}^j, d_j)) = 1 - y_j$.
4. Output F, X, d, where $X = \{K_0^j\}_j$ for every circuit input wire j and $d = \{d_j\}_j$ for every circuit output wire j.

5.2 Indistinguishability Under Correlated Randomness

We reduce the privacy of our construction to the problem of indistinguishability under ∇-correlation, as defined below.

Experiment $\mathsf{Exp}^{\mathcal{A}}_{\nabla}(1^{\kappa}, b, \mathtt{type})$*.* The experiment is parameterized with a bit b and a $\mathtt{type} \in \{\oplus, \wedge\}$.

1. The adversary $\mathcal{A}$ gives the labels K^A_0, K^A_1, K^B_0, K^B_1, and the bits a, b to the experiment, where a, b indicate the active labels K^A_a and K^B_b, respectively.
2. The experiment computes $X_{\alpha\beta} = h(K^A_\alpha, K^B_\beta)$ for every $\alpha, \beta \in \{0,1\}$ and $\nabla = f_\nabla(X_{00}, X_{01}, X_{10}, X_{11})$ where f_∇ represent the function $\nabla_\oplus$ if $\mathtt{type} = \oplus$ and $\nabla_\wedge$ if $\mathtt{type} = \wedge$, as defined in Tables 1-2.
3. The experiment computes an inactive output label $\bar{K}^C$
   - If $b = 0$, pick $\bar{K}^C \in \{0,1\}^\kappa$ uniformly at random.
   - If $b = 1$, if $\mathtt{type} = \oplus$ compute $\bar{K}^C = H(\nabla \& X_{a(1-b)})$; otherwise ($\mathtt{type} = \wedge$) compute $c = 1 - (a \wedge b)$ and $\bar{K}^C = H(\nabla \& X_{cc})$.
4. The experiment returns ∇ and $\bar{K}^C$ to the adversary $\mathcal{A}$.
5. $\mathcal{A}$ outputs the bit b', which is also the output of the experiment.

The ∇-Correlation Assumption. For every PPT adversary $\mathcal{A}$ with an oracle access to h, H and $\tilde{H}$, if the adversary does not query $h(K^A_a, K^B_{1-b}), h(K^A_{1-a}, K^B_b)$ or $h(K^A_{1-a}, K^B_{1-b})$ it holds that for any $\mathtt{type} \in \{\oplus, \wedge\}$

$$|\Pr[\mathsf{Exp}^{\mathcal{A}}_{\nabla}(1^\kappa, 0, \mathtt{type}) = 1] - \Pr[\mathsf{Exp}^{\mathcal{A}}_{\nabla}(1^\kappa, 1, \mathtt{type}) = 1] \leq \mathsf{neg}(\kappa).$$

The intuition behind the assumption is that it models an adversary who does not query the random oracle on three combinations of labels, where each combination contains at least one inactive label. Therefore, the adversary does not learn the oracle result on these queries, and thus, the oracle's output is indistinguishable from a uniformly random string.

The premise of this assumption is in our proof for strong authenticity (Section 4.1) where it is guaranteed that the probability that the adversary queried a preimage of an inactive label is negligible in κ. The three combinations mentioned in the assumption, $(K^A_a, K^B_{1-b}), (K^A_{1-a}, K^B_b)$ or (K^A_{1-a}, K^B_{1-b}) take the role of that preimage.

5.3 Proof of Privacy

We define hybrid $\mathbf{Hyb}_i$, for $i \in [q]$, as the result of running the simulated garbling up until the i-th gate (including), and then running the real garbling on the rest. Formally, given a circuit $\mathbf{C}$, and input x and an output y,[12] $\mathbf{Hyb}_i$ is defined as follows:

[12] The output argument is redundant since it is possible to be computed by $\mathbf{C}(x)$, however, we include it in order to match the description of the hybrid to that of the simulation.

Algorithm $\mathbf{Hyb}_i(1^\kappa, \mathbf{C}, x, y)$

```
1. For every circuit input wire $j \in [n]$ sample independent labels $K_0^j, K_1^j \in \{0,1\}^\ell$ at uniform.
2. For each gate $g$ in $\mathbf{C}$ in a topological order
   (a) Let $A, B$ and $C$ be $g$'s input wires and output wire.
       i. $X_{00} = h(K_0^A, K_0^B)$
       ii. $X_{01} = h(K_0^A, K_1^B)$
       iii. $X_{10} = h(K_1^A, K_0^B)$
       iv. $X_{11} = h(K_1^A, K_1^B)$
       v. If $g = \wedge$ set $\nabla = \nabla_\wedge$ and if $g = \oplus$ set $\nabla = \nabla_\oplus$, where $\nabla_\wedge$ and $\nabla_\oplus$
          are defined in Tables 1-2, as a function of $X_{00}, X_{01}, X_{10}, X_{11}$.
       vi. *Simulation: If $g \in \{1, \dots, i\}$*. Let $c$ be the bit carried by wire $C$ when
           evaluating $\mathbf{C}(x)$. Compute $K_c^C$ as in $\mathsf{Gb}(1^\kappa, \mathbf{C})$ (Algorithm 2) and
           pick $K_{1-c}^C$ uniformly at random.
       vii. *Real: If $g \in \{i+1, \dots, q\}$*. Compute $K_0^C, K_1^C$ as in $\mathsf{Gb}(1^\kappa, \mathbf{C})$ (Algorithm 2), namely,
            A. Set $S_0 = X_{00} \& \nabla$. If $g = \oplus$ then set $S_1 = X_{01} \& \nabla$ and if $g = \wedge$
               then set $S_1 = X_{11} \& \nabla$.
            B. Set $K_0^C = H(S_0)$ and $K_1^C = H(S_1)$.
   (b) Set $F[g] \leftarrow \nabla$
3. For every output wire $j \in [n+q-m+1, n+q]$
   (a) Let $b_j$ be the bit carried by wire $j$ when evaluating $\mathbf{C}(x)$ and let $y_j$ be
       the $j$-th output bit.
   (b) Sample $d_j \in_R \{0,1\}^\ell$ s.t. $\mathsf{lsb}(H(K_{b_j}^j, d_j)) = y_j$ and $\mathsf{lsb}(H(K_{1-b_j}^j, d_j)) = 1 - y_j$.
4. Output $F, X, d$, where $X = \{K_{x_j}^j\}_j$ for every circuit input wire $j$ (where $x_j$
   is the $j$-th bit in $x$) and $d = \{d_j\}_j$ for every circuit output wire $j$.
```

Note that $\mathbf{Hyb}_0(1^\kappa, \mathbf{C}, x, y)$ is distributed identically to (F, X, d) as output from the real garbling $(F, e, d) \leftarrow \mathsf{Gb}(1^\kappa, \mathbf{C})$ and $X \leftarrow \mathsf{En}(e, x)$ whereas $\mathbf{Hyb}_q(1^\kappa, \mathbf{C}, 0^n, y)$ is distributed identically to (F, X, d) as output from the simulation $\mathcal{S}(1^\kappa, \mathbf{C}, y)$.

In the rest of this section we first show that for every x and y, $\mathbf{Hyb}_0(1^\kappa, \mathbf{C}, x, y)$ is indistinguishable from $\mathbf{Hyb}_q(1^\kappa, \mathbf{C}, x, y)$ and then, we show that $\mathbf{Hyb}_q(1^\kappa, \mathbf{C}, x, y)$ is indistinguishable from a $\mathbf{Hyb}_q(1^\kappa, \mathbf{C}, 0^n, y)$. That is:

$$\mathbf{Hyb}_0(1^\kappa, \mathbf{C}, x, y) \overset{c}{\approx} \mathbf{Hyb}_q(1^\kappa, \mathbf{C}, x, y) \overset{c}{\approx} \mathbf{Hyb}_q(1^\kappa, \mathbf{C}, 0^n, y)$$

Suppose there exits a distinguisher $\mathcal{D}$ such that for some circuit $\mathbf{C}$, input $x \in \{0,1\}^n$ and output $y \in \{0,1\}^m$, distinguishes between $\mathbf{Hyb}_0(1^\kappa, \mathbf{C}, x, y)$ and $\mathbf{Hyb}_q(1^\kappa, \mathbf{C}, x, y)$ with non-negligible probability $p(\kappa)$, then, there must be some $i \in [q]$ such that $\mathcal{D}$ distinguishes between $\mathbf{Hyb}_i(1^\kappa, \mathbf{C}, x, y)$ and $\mathbf{Hyb}_{i+1}(1^\kappa, \mathbf{C}, x, y)$ with non-negligible probability of at least $p(\kappa)/q$. As we show immediately, this contradicts our assumption (in Section 5.2), as we can use $\mathcal{D}$ to construct an adversary $\mathcal{A}$ with a non-negligible advantage in experiment $\mathsf{Exp}_\nabla^\mathcal{A}(1^\kappa, b, \mathtt{type})$, as follows:

Our adversary $\mathcal{A}$ first generates $\mathbf{Hyb}_i(1^\kappa, \mathbf{C}, x, y)$. This means that the inactive labels of output wires of gates $1, \ldots, i$ are chosen randomly. Then, $\mathcal{A}$ changes the $i+1$-th garbled gate according to its interaction with experiment $\mathsf{Exp}_{\nabla}^{\mathcal{A}}(1^\kappa, b, \mathtt{type})$, as follows: let the input wires and output wire of gate $i+1$ be A, B and C, respectively, and let $K_0^A, K_1^A, K_0^B, K_1^B$ be the associated labels of A, B, as produced by $\mathbf{Hyb}_i(1^\kappa, \mathbf{C}, x, y)$, with K_a^A, K_b^B being the active labels. $\mathcal{A}$ sends $K_0^A, K_1^A, K_0^B, K_1^B$ and a, b to the experiment and receives back $\nabla, \bar{K}^C$. That $\bar{K}^C$ accounts to the inactive label of the output wire C. $\mathcal{A}$ can compute K^C (the active key of wire C) on its own by $X_{ab} \leftarrow h(K_a^A, K_b^B)$ and then $K^C \leftarrow H(\nabla \& X_{ab})$. At this point, $\mathcal{A}$ sets $F[i+1] = \nabla$ and it has everything needed to continue the garbling of gates $i+2, \ldots, q$ according to the *real* garbling scheme. In particular, it uses keys K^C and $\bar{K}^C$ as the active and inactive labels for wire C. Now, $\mathcal{A}$ hands the resulting garbled circuit, garbled input and decoding information (F, X, d) to $\mathcal{D}$ and outputs whatever $\mathcal{D}$ outputs.

Observe that if the experiment is parameterized with the bit $b = 0$ then $\bar{K}^C$ (the inactive key of wire C) is uniformly random and garbled gate $i+1$ is computed exactly as in $\mathbf{Hyb}_{i+1}(1^\kappa, \mathbf{C}, x, y)$ whereas in case $b = 1$ the keys $\bar{K}^C$ is computed exactly as in $\mathbf{Hyb}_i(1^\kappa, \mathbf{C}, x, y)$. Therefore, the advantage of $\mathcal{A}$ is greater than $p(\kappa)/q$, which implies that $p(\kappa)$ is negligible.

$\mathbf{Hyb}_q(1^\kappa, \mathbf{C}, x, y) \overset{c}{\approx} \mathbf{Hyb}_q(1^\kappa, \mathbf{C}, 0^n, y)$ We now show that for every $x \in \{0,1\}^n$ it holds that the simulator's output when the garbled input is an encoding of x is indistinguishable from its output when the garbled input is an encoding of 0^n.

We rely on the strong authenticity property of the construction (see Section 4), from which it is guaranteed that, with overwhelming probability, the adversary does not obtain any inactive label (not even those of internal wires).

As before, note that labels $K^j, \bar{K}^j$, for every wire j, are uniformly random and independent, as they are a direct output of the random oracle. Furthermore, the intermediate values $X_{00}, X_{01}, X_{10}, X_{11}$ obtained in the garbling process are uniformly random and independent as well. For each gate g with input wires A, B and output wire C, the only value that depends on $X_{00}, X_{01}, X_{10}, X_{11}$ (where $X_{\alpha,\beta} = h(K_\alpha^A, K_\beta^B)$) is the garbled gate ∇_g. We have to show that for each gate g, given the active labels K_a^A, K_b^B (where a, b are the active label indicators) and ∇_g, the adversary could not tell what are the values a, b (unless it has an auxiliary information about them).

To this end, we show that the pairs $(X_{\alpha,\beta}, \nabla_g)$ are distributed identically for all $\alpha, \beta \in \{0,1\}$. Table 1 shows that for each $\alpha, \beta \in \{0,1\}$ we have

$$\Pr[(X_{\alpha,\beta}, \nabla_\wedge) = (0,0)] = \Pr[(X_{\alpha,\beta}, \nabla_\wedge) = (1,0)] = 6/16 \quad \text{and}$$
$$\Pr[(X_{\alpha,\beta}, \nabla_\wedge) = (0,1)] = \Pr[(X_{\alpha,\beta}, \nabla_\wedge) = (1,1)] = 2/16$$

Concluding that the pairs $(X_{\alpha,\beta}, \nabla_\wedge)$ are identically distributed for all $\alpha, \beta \in \{0,1\}$. This means that the adversary can always expect (regardless the actual bits a, b being carried on the wires) a pair of strings of length ℓ, such that it is expected that in 6/16 of the positions the bits are $(0, 0)$, in 6/16 of the positions the bits are $(1, 0)$, in 2/16 of the positions the bits are $(0, 1)$ and in 2/16 of the positions the bits are $(1, 1)$.

Exactly the same argument and the same distributions hold for the pairs $(X_{\alpha,\beta}, \nabla_{\oplus})$ for XOR gates, which concludes the proof.

Acknowledgments Tomer Ashur is an FWO post-doctoral fellow under Grant Number 12ZH420N. Efrat Cohen and Carmit Hazay were supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by ISF grant No. 1316/18.

A Key Length Search

We present here the code for our exhaustive search that finds the minimal ℓ for which the number of 1's in ∇ is greater than κ with high probability (i.e. except with a negligible probability in σ).

```python
import math
from mpmath import *
mp.dps=1000 # set precision


SIGMA = 80.0    # statistical sec param
KAPPA = 256.0   # computational sec param

SEARCH_FROM = 1700  # minimal ell to begin search with
SEARCH_TO = 1800  # stop search when reaching ell

mpfsigma = mpf(SIGMA)
mpfkappa = mpf(KAPPA)
mpf1 = mpf(1)
mpf3 = mpf(3)
mpf4 = mpf(4)
mpf025 = mpf1/mpf4
mpf075 = mpf3/mpf4

mpfnegl = mpf(1)/mpf(int(math.pow(2,SIGMA)))

for ell in range(SEARCH_FROM,SEARCH_TO):
  mpfl = mpf(ell)
  mpfbadprob = mpf(0)
  for i in range(int(KAPPA)-1):
    mpfi = mpf(i)
    mpfbadprob = mpfbadprob + binomial(mpfl, mpfi) * math.pow
    (0.25,mpfi) * math.pow(0.75, mpfl-mpfi)
  print("ell= ", ell, ", " , "mpfbadprob=2^{%.3f}"%log(
    mpfbadprob,2))
```

```
if mpfbadprob <= mpfnegl:
  print("found ell = ", ell, " !")
  break
```

B Garbling Other Gates

Other Boolean gates can similarly be obtained by applying the following procedure: (i) Draw the truth table corresponding to the plain gate and its complement (see *e.g.*, Table 3 for the case of an OR gate); (ii) Transpose the output columns such that they become row vectors (e.g., from Table 3, $\vee = [0, 1, 1, 1]; \overline{\vee} = [1, 0, 0, 0]$); (iii) With Table 1 as a template, initialize a new 16-row table T, whose index is the vector $[X_{00}, X_{01}, X_{10}, X_{11}]$ and its value is ∇. Initialize all ∇ values to 0 (*i.e.*, $T[X_{00}, X_{01}, X_{10}, X_{11}] = 0$ for all X_{00}, X_{01}, X_{10}, and X_{11}); (iv) Set $\nabla = 1$ in the rows indexed by the vectors from Step (ii), as well as the first and last rows. Completelting the example, in the case of an OR gate, set

$$\begin{aligned} T[0,0,0,0] &= 1; \\ T[1,1,1,1] &= 1; \\ T[\vee] &= 1; \\ T[\overline{\vee}] &= 1. \end{aligned}$$

Table 3. Truth table for an OR gate

x	y	$\vee$	$\overline{\vee}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

References

App17. Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017.

BHKR13. Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 478–492. IEEE Computer Society, 2013.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012.

BMR90. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990.

KKS16. Carmen Kempka, Ryo Kikuchi, and Koutarou Suzuki. How to circumvent the two-ciphertext lower bound for linear garbling schemes. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *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*, volume 10032 of *Lecture Notes in Computer Science*, pages 967–997, 2016.

KMR14. Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. Flexor: Flexible garbling for XOR gates that beats free-xor. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 440–457. Springer, 2014.

KS08. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.

LP09. Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptol.*, 22(2):161–188, 2009.

NPS99. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In Stuart I. Feldman and Michael P. Wellman, editors, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 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 Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2009.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 220–250. Springer, 2015.