ON THE AUTHENTICITY OF GARBLING SCHEMES

Yashvanth Mohan Kondi

Master of Technology Thesis

June 2017

International Institute of Information Technology, Bangalore

ON THE AUTHENTICITY OF GARBLING SCHEMES

Submitted to International Institute of Information Technology, Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Technology

by

Yashvanth Mohan Kondi
IMT2012053

International Institute of Information Technology, Bangalore
June 2017

Dedicated to

My family

Thesis Certificate

This is to certify that the thesis titled **On the Authenticity of Garbling Schemes** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Yashvanth Mohan Kondi**, **IMT2012053**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Ashish Choudhury

Bengaluru,
The 6th of June, 2017.

ON THE AUTHENTICITY OF GARBLING SCHEMES

Abstract

In this thesis we study one of the fundamental security notions achievable by a powerful cryptographic primitive; namely the *authenticity* of *garbling schemes*. While almost all garbling schemes in the literature (including Yao's) achieve both privacy and authenticity, Frederiksen et al. (Eurocrypt '15) were the first to show a natural separation between these two notions of security. 'Privacy-free' garbling schemes achieve authenticity with increased efficiency at the cost of privacy. Zahur et al. (Eurocrypt '15) bound the performance of private and privacy-free schemes separately within their model of linear garbling. However in this work, we show constructively that the lower bound of Zahur et al. for privacy-free linear garbling schemes does not hold. Further, our construction requires no cryptographic assumptions, and can be used to garble formulas with an asymptotic improvement over its counterpart which achieves privacy as well. An interesting aspect of our construction is that it allows an evaluator to be in possession of both keys on some non-output wires, with no impact on authenticity. Our techniques further conceptually separate private from privacy-free garbling, while also finding direct application in zero-knowledge proofs for SAT. We also introduce unconditional privacy-free garbling for threshold/majority gates, which at the cost of standard cryptographic assumptions can be embedded in Boolean *circuits* with high concrete efficiency.

We motivate and initiate the study of the orthogonal variant of 'authenticity-free' garbling schemes which achieve privacy alone, and show that known garbling techniques can not be used to compromise authenticity without affecting privacy.

Acknowledgements

I would like to thank my advisor Prof. Ashish Choudhury, for teaching the courses that captured my attention and drew me to cryptography, and for the many opportunities made available to me, in addition to his invaluable guidance.

I am grateful to have been supervised by Prof. Arpita Patra, who along with Prof. Ashish Choudhury mentored me as I made my first steps into research in secure computation. Much of my work was done at the Cryptography and Information Security Lab at the Indian Institute of Science, and I greatly appreciate the discussions I had there, and the time that I got to spend with all of its members.

I would also like to thank Prof. Madhav Rao, for providing me with a number of opportunities to gain exposure to research in Computer Science.

Finally, I am indebted to my family and friends (one very special friend in particular) for their unfailing support through the years.

List of Publications

[1] Yashvanth Kondi and Arpita Patra. “Privacy-Free Garbled Circuits for Formulas: Size Zero and Information-Theoretic”. To appear in CRYPTO 2017.

[2] Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar. “Round-efficient Zero Knowledge Arguments from Garbled Circuits”. In submission.

[3] Ashish Choudhury, Yashvanth Kondi, Arpita Patra, and Divya Ravi. “A Survey of Garbling Schemes and Their Theoretical Aspects”. In progress.

Contents

List of Figures

List of Abbreviations

aut . . . . . Authenticity

GC . . . . . Garbled Circuit

GS . . . . . Garbling Scheme

IIITB . . . . . International Institute of Information Technology Bangalore

OT . . . . . Oblivious Transfer

PRF . . . . . Pseudorandom Function

PRG . . . . . Pseudorandom Generator

RO . . . . . Random Oracle

Sim . . . . . Simulator

ZK . . . . . Zero-knowledge

CHAPTER 1

INTRODUCTION

Secure Multiparty Computation (MPC) generalizes the task of computing a function on the joint private inputs of participants in a protocol. Two-party Computation (2PC) is a special case of MPC where two mutually distrusting parties wish to compute a function on their joint inputs. Relaxations of 2PC represent natural applications such as zero-knowledge protocols [2]. A breakthrough in the design of 2PC protocols came in the form of Yao's garbled circuit technique [3]. A physical analogue of this technique as described by Lindell and Pinkas [4] is that of two sets of two identical 'input' keys, and four identical lock-boxes such that each box can be opened only by a unique combination of one key from each set. If each of the input keys are assigned semantic Boolean values, arranging for each lock-box to contain one of two 'output' keys makes the process of unlocking a box (given two input keys) implement the evaluation of a Boolean gate, as per the semantics of the keys used and obtained. When the output keys can serve as input keys to the next set of lock-boxes, we have a technique to evaluate a whole circuit.

1.1 A Brief Primer on Yao's Protocol

Informally the concrete instantiation of this technique assigns random Boolean strings as the input keys, which are used to construct four 'double-encryption' ciphertexts, each

of which function as a lock-box containing an output key.

A little more formally, consider an AND gate g to have left and right incoming wires i and j respectively. Garbling g would produce input keys (k_i^0, k_i^1) corresponding to semantic zero and one values respectively on wire i, $\left(k_j^0, k_j^1\right)$ corresponding to semantic zero and one values on wire j, and $\left(k_g^0, k_g^1\right)$ as the output keys corresponding to zero and one respectively. Each key is a random κ-bit string, where κ is the security parameter. Finally, garbling g also produces a randomly ordered set of four ciphertexts $T = \left(\mathsf{Enc}_{k_i^a}\left(\mathsf{Enc}_{k_j^b}\left(k_g^{a\wedge b}\right)\right)\right)_{a,b\in\{0,1\}}$. An evaluator who is in possession of k_i^a, k_j^b and T, should be able to obtain $k_g^{a\wedge b}$ and nothing else. Given this construction, one of the two parties in Yao's secure function evaluation protocol creates such ciphertexts and keys, and provides the required keys for evaluation to the other party, who in turn performs the evaluation by 'unlocking' one ciphertext for every gate, and sends back the output. Lindell and Pinkas [4] provide a full proof of security for Yao's protocol when Enc is the encryption algorithm of a CPA-secure encryption scheme satisfying a special correctness property.

Yao's technique was named 'garbled circuits' by Beaver et al. [5], and has been central to many theoretical and practical protocols that followed, including constant-round MPC [5, 6], 2PC [7, 8], Verifiable Computation [9], Non-interactive Secure Computation [10, 11], One-time programs [12], and Zero-knowledge [1, 13] to name a few.

1.2 Garbled Circuits as a Primitive

There have been multiple optimizations to Yao's garbled circuits [5, 14–16], as well as diverse use-cases as listed above. In recognition of a need for a formal abstraction for this powerful cryptographic technique, Bellare et al. [17] introduced the notion of a *garbling scheme* as a cryptographic primitive. A garbling scheme can now be viewed as an end in itself, with well-defined security properties enabling modular use in secure

protocols. Bellare et al. [17] define *privacy* and *obliviousness* as security notions which capture the privacy of inputs and outputs during a garbled evaluation, and *authenticity* as the unforgeability of a garbled output. These security notions are shown to be separate by the existence of garbling schemes that achieve each definition but not the others.

This language of garbling schemes has largely been adopted in the literature, with recent work on efficient garbled circuit constructions [18–20] being cast in the framework of Bellare et al. .

Besides providing a modular approach with which to use garbling schemes as a primitive, the work of Bellare et al. also points out that garbled circuits are complex objects which (until then) achieved different, possibly independent security goals simultaneously. Certain garbled circuit based protocols may not rely on all of the security properties that a garbling scheme can achieve, and a natural question that follows is whether it is possible to gain efficiency in garbling by relaxing the security goals as required. Frederiksen et al. [21] ask exactly this question when tailoring a garbling scheme to the garbled circuit based Zero-knowledge protocol of Jawurek et al. [1].

1.3 Our Contributions

In this thesis, we expand on the understanding of the separations between privacy and authenticity in particular, showing that they are conceptually different security goals instantiable by significantly different techniques. More specifically, we show that the invariant of an evaluator obtaining only one key for each non-output wire of a garbled circuit (henceforth the 'single-key invariant') does not need to hold in the privacy-free setting, whereas it is arguably the conceptual core of *every known garbling technique* [5, 14, 15, 18–22] until now. We also initiate the study of garbling without authenticity but privacy alone, and show that this relaxation does not yield any benefit using current garbling techniques. Specifically, if a garbling scheme satisfies a natural notion

of *composability*, achieving privacy inherently requires authenticity. The implication is that with current garbling techniques, authenticity can not be sacrificed meaningfully while still retaining privacy.

CHAPTER 2

PRELIMINARIES

The primitives and terminology used in this thesis are defined below.

We use the language of Bellare et al. [17] for circuits as well as garbled circuits. As most of our discussions are centred on garbling formulas, we formally describe formulaic circuits, in addition to formally specifying general circuits.

2.1 Fundamental Concepts

Definition 2.1.1 *(Negligible Function [23]) A function η from the natural numbers to the non-negative real numbers is negligible in the security parameter κ if for every positive polynomial $\mathcal{P}$ there is an N, such that for all integers $\kappa > N$ it holds that $\eta(\kappa) < \frac{1}{\mathcal{P}(\kappa)}$.*

Definition 2.1.2 *(Pseudorandom Function (PRF) [23]) Let $F : \{0,1\}^{\ell_{key}} \times \{0,1\}^{\ell_{in}} \rightarrow \{0,1\}^{\ell_{out}}$ be an efficient, keyed function[1], where ℓ_{key}, ℓ_{in} and ℓ_{out} are functions of the security parameter κ. Then F is a pseudorandom function if for all probabilistic*

[1] A keyed function is a two-input function, where the first input is called the *key* and denoted by k. We say F is efficient if there is a polynomial-time algorithm that computes $F(k, x)$ given k and x. While using a PRF, typically a key k is chosen randomly and fixed, and we are then interested in the single-input function $F_k : \{0,1\}^{\ell_{in}} \rightarrow \{0,1\}^{\ell_{out}}$, defined as $F_k(x) \Rightarrow F(k, x)$.

polynomial-time distinguishers $\mathcal{D}$, there is a negligible function η such that:

$$|Pr[\mathcal{D}^{F_k()}(1^\kappa) = 1] - Pr[\mathcal{D}^{f(\cdot)}(1^\kappa) = 1]| \leq \eta(\kappa), \qquad \text{(Eqn 2.1)}$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^{\ell_{key}}$ and the randomness of $\mathcal{D}$, and the second probability is taken over uniform choice of $f \in \mathsf{Func}_{\ell_{in},\ell_{out}}$ and the randomness of $\mathcal{D}$. Here $\mathsf{Func}_{\ell_{in},\ell_{out}}$ denotes the set of all functions with domain $\{0,1\}^{\ell_{in}}$ and codomain $\{0,1\}^{\ell_{out}}$.

2.2 Circuits

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

Definition 2.2.1 *(Circuits) A circuit is a tuple $f = (n, m, q, A, B, G)$. The parameters n, m, q are positive integers which define the number of input, output, and non-input wires respectively. Wires are indexed from 1 to $n+q$, with 1 to n being input wires, and $n+q-m+1$ to $n+q$ being output wires. A gate is identified by its outgoing wire index. For a gate $g \in [n+1, n+q]$, $A(g)$ and $B(g)$ give the left and right incoming wire indices respectively. We have $B(g) \in [1, n+q-m]$, and $A(g) \in [0, n+q-m]$; $A(g) = 0$ if g has fan-in of 1 (ie. NOT gate). In order to avoid cycles, we require that $A(g) < B(g) < g$. The gate functionality $G(g)$ is a map $G(g) : \{0,1\}^2 \mapsto \{0,1\}$.*

From the above definition, it is clear that evaluating a circuit $f = (n, m, q, A, B, G)$ comprises of executing the gate functionality of each gate $g \in [n+1, n+q]$ in increasing order, starting with a given input $x \in \{0,1\}^n$ populating the values on the input wires $w \in [1, n]$. We use $f(x)$ to denote the evaluation of input $x \in \{0,1\}^n$ on circuit f. The 'size' of f is denoted $|f|$, and is determined by computing $|f| = n + q$.

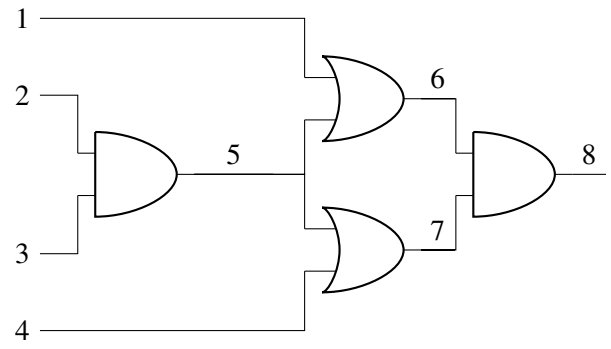

Figure FC2.1: A simple circuit with $n = 4$, $m = 1$, $q = 4$. The left incoming wires are defined as $A(5) = 2$, $A(6) = 1$, $A(7) = 4$, $A(8) = 7$, with right incoming wires defined similarly. The gate functionalities $G(5)$ and $G(8)$ consist of the truth table for AND, while $G(6)$ and $G(7)$ are the truth table for OR.

2.3 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 [17].

Definition 2.3.1 *(Formulaic Circuits) 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*[2]. *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* $G(g) : \{0, 1\}^2 \mapsto \{0, 1\}$.

[2] The injection property required here ensures that every gate in the circuit has a fan-out of one.

A formula with Boolean inputs $(x_1, x_2 \cdots x_n)$ can be expressed as a combination of those x_is and the defined gates (eg. $\wedge, \neg, \oplus$) with the appropriate bracketing. For instance, the circuit in Fig. FC2.1 can be expressed as a formula:

$$f(x_1, x_2, x_3, x_4) = ((x_2 \wedge x_3) \vee x_1) \wedge ((x_2 \wedge x_3) \vee x_4) \qquad \text{(Eqn 2.2)}$$

Observe that the circuit representation of a formula is always a tree.

2.4 Garbling Schemes

A garbling scheme $\mathcal{G}$ is characterised by a tuple of algorithms $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$. All of these algorithms are poly-time, and with the exception of Gb, are all deterministic.

The sequence of invoking these algorithms is as follows:

- $\mathsf{Gb}\,(1^\kappa, f)$ is invoked on a circuit f in order to produce a 'garbled circuit' GC, 'input encoding information' e, and 'output decoding information' d.
- $\mathsf{En}\,(x, e)$ encodes a clear input x with encoding information e in order to produce a garbled/encoded input X.
- $\mathsf{Ev}\,(\mathsf{GC}, X)$ evaluates X upon GC to produce a garbled output Y.
- $\mathsf{De}\,(Y, d)$ translates Y into a clear output y as per decoding information d.

A block diagram depicting the above sequence is provided in Fig. FC2.2.

A correctness requirement of the garbling scheme follows naturally from the above definition of its components. Informally, for a given circuit f and input x, garbling f and evaluating it on the correspondingly encoded x should produce a garbled output that decodes to the clear value $f(x)$.

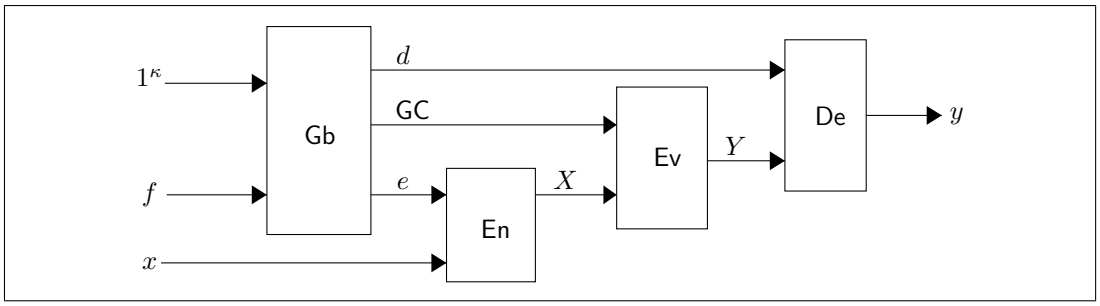

Figure FC2.2: Block diagram of garbling a circuit f, encoding an input x, and decoding the 'garbled' evaluation to retrieve $y = f(x)$.

Definition 2.4.1 *(Correctness) A garbling scheme* $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is 'correct' if and only if for all circuits* $f : \{0,1\}^n \mapsto \{0,1\}^m$ *and all inputs in its domain* $x \in \{0,1\}^n$ *where* $n \leq poly\,(1^\kappa)$*, we have that,*

$$\forall \mathsf{GC}, e, d \leftarrow \mathsf{Gb}\,(1^\kappa, f)\,,\ \mathsf{De}\,(\mathsf{Ev}\,(\mathsf{GC}, \mathsf{En}(x, e))\,, d) = f(x) \qquad \text{(Eqn 2.3)}$$

Bellare et al. formalize the security notions of a garbling scheme by means of code-based games, equivalent definitions of which are presented in the following section.

2.5 Notions of Security for Garbling Schemes

Before discussing the security notions of garbling schemes themselves, Bellare et al. note that certain 'side information' about the underlying circuit being evaluated upon may be leaked during the garbled evaluation. This is certainly the case for garbling schemes such as FreeXOR [16], which require the garbled circuit evaluator to know which gates are XOR gates. Bellare et al. formalize this additional information available to the evaluator by means of the side-information function $\Phi\,(f)$ of a circuit f. Therefore when stating that a garbling scheme achieves certain kinds of security, it is important to state with respect to which side information function Φ it does so. For instance, Yao's original construction [4] reveals only the topology of the underlying

circuit, which can conveniently be represented as $\Phi_{\mathsf{topo}}(f) = (n, m, q, A, B, 0)$ for a circuit $f = (n, m, q, A, B, G)$. However, many of the use-cases for garbled circuits permit (or even require) the function being computed to be known to all parties, indicating that the weakest side information function $\Phi_{\mathsf{circ}}(f) = f$ is usually sufficient. In addition to this, garbled circuits can be heavily optimized when the evaluator is assumed to know the circuit [16, 18, 19]. The context in which we use the definition of privacy in this thesis is to discuss how it relates to authenticity; we go on to prove that garbling with privacy with respect to even the weakest side information function (Φ_{circ}) is not possible for composable garbling schemes not achieving authenticity. Therefore for readability we do not explicitly specify the side information function; it is implicit that whenever we discuss privacy in this thesis, it is with respect to Φ_{circ}.

The notion of privacy of inputs and outputs is captured by *privacy* and *obliviousness* respectively. Informally, a private garbling scheme requires the garbled circuit, encoded input, and decoding information to be simulatable given only the clear output and description of the function. This requirement captures the intuition that the garbled evaluation itself leaks no information about the clear inputs and intermediate values in the circuit.

Definition 2.5.1 *(Privacy [17]) A garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ achieves* prv *security if there exists a PPT simulator $\mathcal{S}$ such that for every circuit $f : \{0,1\}^n \mapsto \{0,1\}^m$, $|f| \leq \mathsf{poly}(\kappa)$, and input $x \in \{0,1\}^n$, there exists no PPT adversary $\mathcal{A}(1^\kappa)$ which succeeds in distinguishing between the following distributions with probability better than $\frac{1}{2} + \eta(\kappa)$, where η is a negligible function:*

1. $\textsc{Real}(f, x)$*:* $(\mathsf{GC}, \mathsf{En}(x, e), d)$*, where* $(\mathsf{GC}, e, d) \leftarrow \mathsf{Gb}(1^\kappa, f)$
2. $\textsc{Ideal}_{\mathcal{S}}(f, f(x))$*:* $(\mathsf{GC}, X, d) \leftarrow \mathcal{S}(1^\kappa, f, f(x))$

For clarity, the above definition of privacy is represented diagrammatically as an

experiment in Fig. FC2.3 (a).

Motivated by applications where the garbled circuit evaluator is not to be given the clear output of the evaluation, Bellare et al. define obliviousness in the same spirit as privacy: given only the side information of the function, the garbled circuit and encoded input (but not the decoding information) should be simulatable. The intuition captured here is that when the decoding information is withheld, the garbled evaluation leaks no information about *any* underlying clear values; be they of the input, intermediate, or output wires of the circuit. We do not discuss obliviousness further in this thesis, as we are focussed on privacy and authenticity alone.

The final property, 'authenticity' as defined by Bellare et al. concerns the unforgeability of garbled outputs. Informally, an evaluator should not be able to derive a valid garbled output which is not the direct result of executing the Ev algorithm on the garbled circuit and encoded input which she was given.

Definition 2.5.2 *(Authenticity [17]) A garbling scheme* $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *achieves* aut *security if for all circuits* $f : \{0,1\}^n \mapsto \{0,1\}^m$*,* $|f| \leq \mathsf{poly}(\kappa)$*, and inputs* $x \in \{0,1\}^n$*, the following probability holds for every PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l} \widehat{Y} \neq \mathsf{Ev}(\mathsf{GC}, X) \\ \wedge \mathsf{De}\left(\widehat{Y}, d\right) \neq \perp \end{array} : \begin{array}{c} \widehat{Y} \leftarrow \mathcal{A}(\mathsf{GC}, X),\ X = \mathsf{En}(x, e) \\ (\mathsf{GC}, e, d) \leftarrow \mathsf{Gb}(1^\kappa, f) \end{array}\right] \leq f(\kappa) \qquad \text{(Eqn 2.4)}$$

Where f *is a negligible function. In case* $f(\kappa) = \frac{1}{2^\kappa}$ *and* $\mathcal{A}$ *is computationally unbounded, authenticity is unconditional.*

The diagram in Fig. FC2.3 (b) depicts the authenticity experiment.

Indistinguishability based notions of security. Bellare et al. [17] also define indistinguishability based notions of privacy and obliviousness. Informally, an indistinguish-

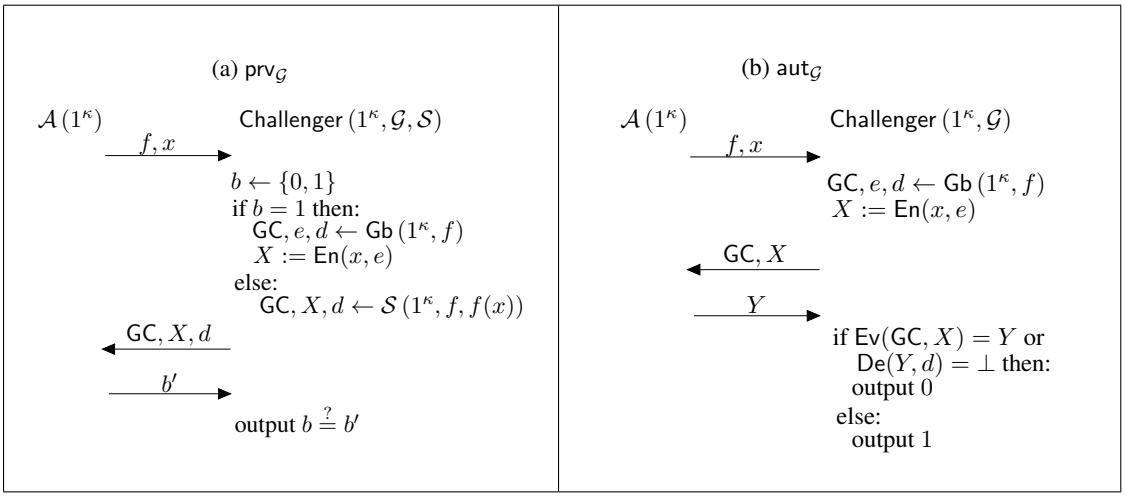

Figure FC2.3: Diagrams of experiments capturing the security requirements of a garbling scheme $\mathcal{G}$. The privacy experiment is captured in (a), and the authenticity experiment is captured in (b). The output 1 of the challenger in each experiment corresponds to the PPT adversary $\mathcal{A}$ winning the game, while output 0 denotes $\mathcal{A}$'s loss. For a private $\mathcal{G}$, the challenger in (a) must output 1 with probability not more than negligibly (in κ) greater than $\frac{1}{2}$, for every PPT $\mathcal{A}$. An authentic $\mathcal{G}$ will ensure that the challenger in (b) outputs 1 with probability negligible in κ, for every PPT $\mathcal{A}$.

ably private garbling scheme $\mathcal{G}$ ensures that no PPT adversary $\mathcal{A}$ will be able to correctly guess whether a given (GC, X, d) was produced using f_0, x_0 or f_1, x_1 (both of which are chosen by $\mathcal{A}$ subject to $f_0 = f_1$ and $f_0(x_0) = f_1(x_1)$) with probability non-negligibly greater than $\frac{1}{2}$. Indistinguishability-based obliviousness is defined along similar lines. However we do not elaborate upon, or use these definitions further in this thesis, as Bellare et al. prove in the same work that simulation-based privacy and obliviousness subsume their indistinguishability-based counterparts. In addition to this, the simulation-based definitions have arguably seen more widespread adoption in the literature that has followed [18–20].

Finally, we also consider the property of *verifiability* introduced in [1]. 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 verification that a given garbled circuit was legitimately constructed. A verifiable garbling scheme ensures that the garbled output is independent of the clear input;

ie. if $f(x_1) = f(x_2)$, then $\mathsf{Ev}(\mathsf{GC}, \mathsf{En}(x_1, e)) = \mathsf{Ev}(\mathsf{GC}, \mathsf{En}(x_2, e))$ for every honestly generated GC, e. 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 2.5.3 (Verifiability) *A verifiable garbling scheme* $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, \mathsf{Ve})$ *contains a poly-time computable function* Ve *such that there exists an expected 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,*

$$\Pr\left[\mathsf{Ext}(\mathsf{GC}, e) = \mathsf{Ev}\left(\mathsf{GC}, \mathsf{En}(e, x)\right)\right] = 1, \textit{ when } \mathsf{Ve}\left(\mathsf{GC}, f, e\right) = 1;\ \left(\mathsf{GC}, e\right) \leftarrow \mathcal{A}\left(1^{\kappa}, f\right) \tag{Eqn 2.5}$$

For completeness, for every f*,* $\forall \left(\mathsf{GC}, e, d\right) \leftarrow \mathsf{Gb}\left(1^{\kappa}, f\right)$*,* $\mathsf{Ve}\left(\mathsf{GC}, f, e\right) = 1$*.*

An unconditionally secure privacy-free garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, \mathsf{Ve})$ is correct (Def. 2.4.1), unconditionally authentic (Def. 2.5.2), and verifiable (Def. 2.5.3).

2.6 Zero knowledge protocols

We do not discuss zero knowledge (ZK) protocols formally in this thesis, and therefore only provide an informal description. Conceptualized by Goldwasser et al. [2], a zero-knowledge proof system is a protocol that allows a 'prover' to convince a 'verifier' of the validity of a statement (completeness), while revealing no information as to why the statement is true (zero-knowledge). A malicious prover should be unable to convince a verifier of a false statement, except with negligible probability (soundness). If the zero-knowledge protocol guarantees soundness only against a computationally bounded prover, it is referred to as an argument rather than a proof.

The garbled circuit approach to this problem as per the construction of Jawurek et al. [1] is informally described in Fig. FC4.14, and does not require privacy of the underlying garbling scheme.

CHAPTER 3

BACKGROUND AND RELATED WORK

The work of Bellare et al. [17] in distilling the different notions of GC security as discussed earlier is a relatively recent development. Thus far, *privacy-free* garbling schemes are the only class of special-purpose garbling that have been shown to be more efficient than general purpose garbling. Here, we briefly recall the previous results pertaining to the study of privacy-free garbling, and frame the natural orthogonal question of *authenticity-free* garbling.

3.1 Privacy-free Garbling

Further showing that the notions of privacy and authenticity as defined by Bellare et al. are conceptually separate, the work of Frederiksen et al. [21] considers what efficiency gains can be made when constructing a garbling scheme achieving only authenticity. Frederiksen et al. interpret a privacy-free garbling scheme to be a gadget which when given two input keys and ciphertext, will allow an evaluator to derive only the corresponding semantically correct output key, and no other information. While this seems like a straightforward relaxation of the standard definition of garbling, we expand the intuition of a privacy-free garbling gadget by showing that it is possible in certain cases that such a gadget allows an evaluator to entirely derive *both keys* on certain wires–a property certainly not permissible in garbling schemes achieving pri-

vacy. We constructively demonstrate that this property can be realized and leveraged in the privacy-free setting. To the best of our knowledge, no previous garbling scheme has violated the single-key invariant. Indeed, the simulation paradigm for garbled circuits introduced by Lindell and Pinkas [4] does not even define an inactive key[1] for the simulated garbled circuit.

Another point of departure from garbling with privacy is illustrated in the case of information-theoretic garbling (which is only possible for formulas). The most efficient information-theoretic garbling scheme which achieves privacy, the Gate Evaluation Secret Sharing construction of Kolesnikov [24], suffers from the key-size being dependent on the depth of the gate within the formula. Specifically, when instantiated with κ-bit output keys, a gate at depth d from the root will have key size $\mathcal{O}\left(d \cdot (\kappa + d)\right)$. However we show that this is not the case in the privacy-free setting, as our information-theoretic construction produces keys which are exactly κ-bits in length for every gate in the formula.

The work of Zahur et al. [18] defines a model of 'linear garbling' which captures most practical garbling schemes in the literature that make use of only a linear combination of symmetric-key operations, with the only non-linearity coming from a popular GC optimization called *point-and-permute* [5]. They claim a lower bound of two ciphertexts required to garble a standalone AND gate with privacy, and one ciphertext to garble such a gate in the privacy-free setting. We observe that our construction is linear as per their model, and yet is able to garble an atomic AND gate producing *no ciphertexts* whatsoever, with information-theoretic security.

Finally, we define a novel garbling technique for threshold gates in the privacy-free setting. Ball et al. [25] consider the task of garbling threshold gates natively in arithmetic circuits, and manage to do so efficiently assuming a variant of a circular correlation robust hash function [26]. Threshold/majority gates are highly non-trivial

[1] The key that is not the result of honest garbled evaluation.

to represent succinctly in formulas [27]. Our garbling scheme handles threshold gates natively in formulas, requiring no ciphertext, and only a couple of independent instances of Shamir secret sharing [28]. In addition to this, our construction immediately yields a threshold gate garbling scheme for *circuits* in the privacy-free setting, which with a standard optimization leads to improved concrete efficiency under weaker assumptions than Ball et al. .

Our garbling scheme can be used to obtain Zero-knowledge proofs (which are secure against unbounded provers, as opposed to ZK arguments which guarantee only computational soundness) for SAT in the paradigm of Jawurek et al. [1]. In addition to this, using our scheme in place of a generic privacy-free GC to obtain zero-knowledge arguments for SAT yields a concrete improvement in communication cost, as well as the qualitative benefit of using only pseudorandom generators (PRGs) which are bound to be weaker cryptographic primitives than any type of hash/key derivation function used by a circuit garbling scheme.

We discuss our privacy-free garbling scheme, prove its security, and present its extensions in Chapter 4.

3.2 Authenticity-free Garbling

While privacy-free garbling has been discussed in much detail, there has been no study yet of the orthogonal variant of garbling without authenticity. While Bellare et al. [17] do provide an example of scheme achieving privacy but not authenticity, their construction is pathological and does not offer any insights into whether it is natural to separate these two notions of security. There do exist many garbled circuit based protocols which do not rely on the authenticity property of the underlying garbling scheme; in fact, Yao's original 2PC technique [3] itself is one such protocol[2]. Even

[2] An adversarial circuit evaluator is assumed not attempt to deviate from the protocol by forging a key.

certain recent 2PC and Zero-knowledge protocols offering stronger security guarantees against malicious adversaries [29–32] who may deviate arbitrarily from the protocol, do not require authenticity of the garbling schemes that they use.

Clearly, studying techniques for garbling with privacy efficiently at the cost of authenticity is a well-motivated problem. However, we show that the current approach to garbling, which is roughly characterized by defining a garbling gadget composed with itself to garble larger units, leads to authenticity being inherently entangled with privacy. We formalize this notion of composability and prove our theorem in Chapter 5.

CHAPTER 4

A NEW APPROACH TO PRIVACY-FREE GARBLING

We first describe our approach to garbling an atomic gate; AND, XOR or NOT in the privacy-free setting, in Section 4.1.1. Our garbling scheme for AND gates (Fig. FC4.2) illustrates that it is possible for a privacy-free gate garbling gadget to violate the single-key invariant. We then prove that our gate garbling gadgets can be composed to garble formulas without any blowup in the key size in Section 4.2. We discuss the linear garbling model of Zahur et al. [18] and why our construction violates their lower bound despite fitting into their model of computation, in Section 4.3. Constructions for garbling large fan-in and threshold gates is discussed in Section 4.4, and we show to extend our construction to general circuits in Section 4.5.

4.1 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 [22, 24], our garbling scheme proceeds upwards from the output wire.

4.1.1 Garbling Individual Gates

As per Yao's paradigm of garbling circuits [3], 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.

4.1.1.1 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 [24]. 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 [24] 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.

$\mathsf{GbXOR}\left(K^0, K^1, 1^\kappa\right)$

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

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

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

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

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}

Figure FC4.1: Garbling, evaluation and verification of an XOR gate

The VeXOR routine defined in Fig. FC4.1 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.

4.1.1.2 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 of AND gates cheaper than that of XOR gates. Figure FC4.2 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 4.2. 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. FC4.2 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 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.

$\mathsf{GbAND}\left(K^0, K^1, 1^\kappa\right)$

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

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

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** $L^{b_L} \oplus R^{b_R}$

$\mathsf{VeAND}\left(L^0, L^1, R^0, R^1\right)$

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

Figure FC4.2: Garbling, evaluation and verification of an AND gate

4.1.1.3 Garbling NOT Gates.

NOT gates can be garbled for free, like in [24], 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 [24], 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.

4.1.2 Garbling an Entire Circuit

We can combine the routines defined in Fig. FC4.1 and Fig. FC4.2 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. FC4.3, Fig. FC4.4, Fig. FC4.5, Fig. FC4.4, and Fig. FC4.6 respectively.

$\mathsf{Gb}\,(f, 1^\kappa)$

– 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

Figure FC4.3: Garbling an entire circuit

We can further optimize our scheme to handle ℓ-fan-in gates with better concrete efficiency. A detailed discussion is deferred to Section 4.4. The full proof of security appears in the next section.

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

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** $\left(K_i^{x_i}||x_i\right)_{i\in[n]}$

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

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

Figure FC4.4: Encoding a clear function input and Decoding a garbled output

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

– 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}

Figure FC4.5: Evaluating a Garbled Circuit on Garbled Input

4.2 Full Proof of Security

Theorem 4.2.1 *The garbling scheme $\mathcal{G}$ is an unconditionally secure privacy-free garbling scheme.*

Correctness follows from the correctness of the garbling schemes for individual gates,

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

– 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

Figure FC4.6: Verifying a Garbled Circuit

discussed in Section 4.1.1.2. Verifiability follows from the consistency-checks of individual gates conducted in the Ve algorithm, discussed in Section 4.1.1.2.

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[1]. 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

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

wishing to compromise the authenticity of a circuit with n inputs as $\mathcal{A}_n$.

4.2.1 Single Gate Case

Lemma 4.2.2 *The garbling scheme $\mathcal{G}$ achieves unconditional authenticity as per Definition 2.5.2 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 4.2.2. This can be done by considering the garbling of AND and XOR gates, as per Fig. FC4.2 and Fig. FC4.1 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).

4.2.1.1 XOR Gate.

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

$$L^0 \oplus R^0 = L^1 \oplus R^1 = K^0, \text{ and } L^1 \oplus R^0 = L^0 \oplus R^1 = K^1 \qquad \text{(Eqn 4.1)}$$

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,

$$K^{\neg b_K} = L^{\neg b_L} \oplus R^{b_R} = L^{b_L} \oplus R^{\neg b_R} \quad \text{(Eqn 4.2)}$$

Clearly, she will be unable to forge $K^{\neg b_K}$ without guessing either $L^{\neg b_L}$ or $R^{\neg b_R}$.

4.2.1.2 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.

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.

4.2.1.3 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 4.2.2.

4.2.2 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 $\mathcal{K} = (K_i^0, K_i^1)_{i\in[1,n+q]}$.

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 4.2.2 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 4.2.2 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. FC4.7), 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,

$$\mathsf{Aut}_\mathcal{G}\left(\mathcal{A}, 1^\kappa\right) = \begin{cases} 1 & \text{if } \mathcal{A}(F, X) = Y; Y \neq \mathsf{Ev}(F, X), \mathsf{De}(Y, d) \neq \perp \\ 0 & \text{otherwise} \end{cases} \tag{Eqn 4.3}$$

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

$$\begin{aligned} \Pr\left[\mathsf{Aut}_\mathcal{G}\left(\mathcal{A}_n, 1^\kappa\right) = 1\right] = \Pr\left[\mathsf{Aut}_\mathcal{G}\left(\mathcal{A}_{n-1}, 1^\kappa\right) = 1\right] = \\ \cdots = \Pr\left[\mathsf{Aut}_\mathcal{G}\left(\mathcal{A}_2, 1^\kappa\right) = 1\right] \leq 2^{-\kappa} \end{aligned} \tag{Eqn 4.4}$$

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 4.2.1.

4.2.3 Adaptive Security

We had mentioned in an earlier section that our scheme achieves adaptive security, or $\mathsf{aut1}$ in the terminology of [33], as opposed to Definition 2.5.2 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 2.5.2.

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:

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

- $\mathcal{A}_{n-1}$ 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** $\mathcal{A}_n\left(\varnothing, X, 1^\kappa\right)$

[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.

Figure FC4.7: Constructing Adversary $\mathcal{A}_{n-1}$ given $\mathcal{A}_n$

1. The GARBLE stage accepts from $\mathcal{A}$ a circuit f, computes $(F, e, d) \leftarrow \mathsf{Gb}\left(1^\kappa, f\right)$, 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 2.5.2, which is proven not to succeed with probability better than $2^{-\kappa}$ by Theorem 4.2.1.

4.3 Breaking the Lower Bound of Zahur et al.

Zahur *et al.* [18] 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 gar-

bling 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.* [17]. The following are the bounds in the private and privacy-free settings respectively, as argued by Zahur *et al.* [18].

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.* [25] and Kempka *et al.* [34] 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.

4.3.1 Linear Garbling

We recall the formal definition of linear garbling [18], 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 [18, 21] 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. FC4.8.

– We describe here a simplified characterization of linear garbling [18] 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^{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 ciphertexts $F = \left(C_i\right)_{i\in[p]}$

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

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 $\langle \mathbf{V}_{\alpha,\beta}, \mathbf{T}\rangle$

Figure FC4.8: Form of linear garbling schemes

Claim 4.3.1 ([18]) *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, $\mathbf{K}_1 = \mathbf{L}_1 + \mathbf{R}_1 = [0\ 1\ 1]$
- $(C^i)_{i\in[p]}$ is an empty set as there are no ciphertexts required.
- Evaluation vectors $(\mathbf{V}_{\alpha,\beta})_{\alpha,\beta\in\{0,1\}}$ 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.* [18] 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.

4.3.2 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.* [18] 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 [18] 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.

4.4 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 4.4.1, then describe how to save computation in garbling and evaluating ℓ-fan-in XOR and AND gates respectively in Sections 4.4.3 and 4.4.4.

4.4.1 Threshold Gates

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

$$f_t(x_1, \cdots, x_i, \cdots, x_\ell) = \begin{cases} 1, & \text{if } \sum_{i=1}^{\ell} x_i > t \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Eqn 4.5)}$$

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.* [25], 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 [18]). 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 [28] 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 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. FC4.9. 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

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

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

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

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 $h\left(j_i\right) = K_i, \forall i \in [t+1]$.
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 $h_{K^1}\left(i\right) = K_i^1, \forall i \in [t+1]$.
2. **if** $\exists j \in [t+2, \ell]$ such that $h_{K^1}\left(j\right) \neq K_j^1$ **then return** $0, \perp, \perp$
3. Using Lagrange interpolation, we obtain the unique $(\ell-(t+1))$-degree polynomial h_{K^0}, such that $h_{K^0}\left(i\right) = K_i^0, \forall i \in [\ell-t]$.
4. **if** $\exists j \in [\ell-t+1, \ell]$ such that $h_{K^0}\left(j\right) \neq K_j^0$ **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** $1, K^0, K^1$

Figure FC4.9: Garbling, evaluation and verification of a threshold gate

an n-input formula used by $\mathcal{A}_n$, in Fig. FC4.10.

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

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

– This procedure is a modification of the adversary from Fig. FC4.7, 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** $X_i \leftarrow \{0,1\}^\kappa$

Figure FC4.10: Deleting a threshold gate to reduce $\mathcal{A}_n$ to $\mathcal{A}_{n-\ell+1}$ as per the gate deletion proof strategy

[28]. 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.

4.4.2 Embedding Threshold Gates in Circuits

The threshold gate garbling scheme in Fig. FC4.9 immediately yields a threshold gate garbling gadget to augment a circuit garbling scheme $\mathcal{G}_c$, under cryptographic assumptions no stronger than what is already required by $\mathcal{G}_c$. For instance, let H represent the cryptographic primitive required by $\mathcal{G}_c$ to instantiate encryption; this could be a

PRF [19], Key-derivation function [21], related-key or circular correlation robust hash function [18, 20]. The gadget, given the input keys $(k_i^0, k_i^1)_{i\in[\ell]}$ proceeds as follows for an ℓ-input threshold gate f_t with gate ID g:

1. Choose random keys K^0, K^1 as required by $\mathcal{G}_c$ (eg. FreeXOR compatible keys need to maintain a global offset).

2. Compute and encrypt the keys produced by the threshold gate garbling routine;

$$T_g = \left(H(g, i, 0, k_i^0) \oplus K_i^0, H(g, i, 1, k_i^1) \oplus K_i^1\right)_{i\in[\ell]} \qquad \text{(Eqn 4.6)}$$

   where $(K_i^0, K_i^1)_{i\in[\ell]} \leftarrow \mathsf{GbTHR}\,(\ell, t, K^0, K^1, 1^\kappa)$

3. Now T_g comprises the ciphertext, and K^0, K^1 the output keys for this gate.

Performance and security. The above gadget requires 2ℓ ciphertexts to be communicated. It suffices to prove that $\ell - t$ ciphertexts encrypting K_i^1s are unintelligible to an adversary attempting to forge K_i^1, and that $t + 1$ ciphertexts encrypting K_i^0 are unintelligible to an adversary attempting to forge K^0.

Optimization for concrete efficiency. We can further cut down the communication cost of this gadget by half, if we generate the curves pseudorandomly rather than uniformly at random. Specifically, the polynomial h_{K^1} in GbTHR (Fig. FC4.9) 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 $\ell + 2$.

The threshold gadget of Ball et al. [25] when embedded directly in a Boolean circuit will cost $\mathcal{O}(\log^3(\ell)/\log\log(\ell))$ ciphertexts more than ours. In addition to this, the un-

derlying cryptographic assumption for their construction is that of a circular correlation robust hash function, whereas ours can be instantiated with weaker primitives. While their construction achieves privacy in addition to authenticity, it is not immediate as to whether their scheme can be optimized in the privacy-free setting.

4.4.3 Improved ℓ-fan-in XOR

The routine to garble an individual XOR gate described in Fig. FC4.1 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 $(K_i^0, K_i^1)_{i\in[w,g]}$ to be produced by $\ell - 1$ instances of the GbXOR routine from Fig. FC4.1; starting from the root K_g^0, K_g^1 and ending at the leaves to produce $(K_i^0, K_i^1)_{i\in[w,w+\ell-1]}$. Observe that the zero and one keys on each wire differ by the same offset; ie. $\forall i \in [w, g]$:

$$K_w^0 \oplus K_w^1 = \cdots = K_i^0 \oplus K_i^1 = \cdots = K_g^0 \oplus K_g^1 \qquad \text{(Eqn 4.7)}$$

We make use of the property observed in Equation (Eqn 4.7) 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. FC4.11.

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

– We have to generate ℓ 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** $\left(K_i^0, K_i^1\right)_{i \in [1,\ell]}$

Figure FC4.11: Garbling an ℓ-fan-in XOR gate

The routine detailed in Fig. FC4.11 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. FC4.1) are directly compatible. A separate proof of authenticity is therefore not required.

As for the computation cost, the new GbXOR routine of Fig. FC4.11 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 $\ell > 3$.

4.4.4 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. FC4.2, the following relation holds:

$$K_w^0 = \cdots = K_i^0 = \cdots = K_g^0, \ \ \forall i \in [w, g] \qquad \text{(Eqn 4.8)}$$

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. FC4.12.

The only case where XOR operations are performed in the EvAND routine in Fig. FC4.12 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. FC4.2 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

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

– 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

Figure FC4.12: Evaluating an l-fan-in AND gate

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.

4.5 Garbling Circuits

In this section, we briefly discuss how to adapt our construction for generic circuits which contain multi-fan-out gates, using PRFs. Note that we do not mean for this scheme to compete with native generic circuit-garbling schemes.

4.5.1 Dealing With ℓ-fan-out Gates

We continue to garble "upwards" from the output gate(s) as per the standard paradigm of garbling for formulas. It can immediately be observed that we will run into a problem for gates with fan-out greater than one.

Consider the case where we have to garble an ℓ-fan out gate g; ie. $\exists \ell$ different gates $g_1, \cdots, g_\ell$ such that $A(g_i) = g$ or $B(g_i) = g$, $\forall i \in [1, \ell]$. We can interpret g to be the convergence of ℓ different (not necessarily disjoint) paths from the output gates.

Garbling along each of these paths will produce ℓ different pairs of zero and one keys to be assigned to g, denoted $\left(K^0_{g,i}, K^1_{g,i}\right)_{i\in[1,\ell]}$.

Kempka *et al.* [22] deal with the issue[2] by choosing the first of the ℓ pairs of keys to assign to g (ie. $K^b_g := K^b_{g,1}$), and encrypting the remaining $\ell - 1$ pairs of keys using the corresponding keys from the first pair. This way, an evaluator requiring an input key for g' from g such that $A(g') = g$ or $B(g') = g$ will be able to obtain it by decrypting the appropriate ciphertext using her key on g.

We adapt this approach to our setting, with an important tweak. Note that our violation of the single-key invariant makes it infeasible to directly assign $K^0_{g,1}$ and $K^1_{g,1}$ to be the keys for gate g. This is because while the garbling of a formula allows keys to be compromised "upward" (in a direction opposite to that of the output gates), in this case, having $K^0_{g,1}$ and $K^1_{g,1}$ compromised will cause a "sideways" leak in a generic circuit; ie. if the sub-circuit with $K^0_{g,1}, K^1_{g,1}$ as one of its input keys is structured such that leaking both $K^0_{g,1}$ and $K^1_{g,1}$ is tolerated by the original formula garbling scheme, this will directly compromise both keys on gate g, which in turn may have ramifications for other gates which have g as a parent.

Therefore, when we encounter an ℓ-fan-out gate g, we choose for it fresh random keys K^0_g, K^1_g, and use them to correspondingly encrypt $\left(K^0_{g,i}, K^1_{g,i}\right)_{i\in[1,\ell]}$. We provide an outline of such a routine to be invoked before garbling an ℓ-fan-out gate, in Fig. FC4.13. The keys produced by the Gb-ℓ-fan-out routine are supplied to a gate garbling routine (GbXOR, GbAND, GbTHR), while the ciphertexts produced are sent to the evaluator as part of the garbled circuit.

[2]They discuss how to extend their scheme for 2-fan-out gates, but this can easily be extended to ℓ-fan-out how we describe it.

Gb-ℓ-fan-out $\left(g, \left(K_i^0, K_i^1\right)_{i\in[1,\ell]}, 1^\kappa\right)$

– This routine accepts the gate index g, and ℓ pairs of keys as arguments, and returns a fresh pair of keys to act as the gate keys for g, along with 2ℓ ciphertexts. The b^{th} element of the i^{th} pair of ciphertexts (denoted $T_{i,b}$) will allow the evaluator to obtain the key corresponding to bit b for the i^{th} child of g.
– Access to a PRF $F : \{0,1\}^\kappa \times \{0,1\}^{2|g|} \mapsto \{0,1\}^\kappa$ where $|g|$ is the number of bits required to represent a gate in the circuit, is assumed.

1. Choose gate keys, $K_g^0 \leftarrow \{0,1\}^\kappa$, $K_g^1 \leftarrow \{0,1\}^\kappa$
2. Encrypt each of the ℓ pairs of keys using the corresponding gate keys, **for all** $i \in [1,\ell]$ **do**
   i. $T_{i,0} := K_{g,i}^0 \oplus F_{K_g^0}(g||i)$
   ii. $T_{i,1} := K_{g,i}^1 \oplus F_{K_g^1}(g||i)$
3. Return the gate keys and the pairs of ciphertexts, **return** $K_g^0, K_g^1, (T_{i,0}, T_{i,1})_{i\in[1,\ell]}$

Figure FC4.13: Enabling output key retrieval for an l-fan-out gate

4.5.2 Security

Rather than providing a full proof, we provide an intuition as to why the ciphertexts produced by this extension do not allow for a computationally bounded adversary to gain any non-negligible advantage in her attempt to forge a garbled output.

The authenticity of the garbling of individual gates as per this scheme follows from the authenticity of garbling individual gates by our original scheme. The authenticity of garbling entire formulaic sub-circuits within the generic circuit also follows from the authenticity of our original scheme for formulaic circuits.

The points of interest for this extension are the multi-fan-out gates. Consider an ℓ-fan-out gate g, with ciphertext $T[g]$. Observe that at no point will an evaluator hold both keys K_g^0, K_g^1 on g. This is because g is an output gate for a formulaic sub circuit (of size 1 as a border case), whose garbling is guaranteed to be authentic. The only way to efficiently obtain a key corresponding to bit b on child g_i of g, is to compute $F_{K_g^b}(g||i)$ and unmask ciphertext $T[g]_{i,b}$. Given this, along with the fact that an evaluator holds only key K_g^b on g, we have that an adversarial evaluator wishing to illegally obtain

$K_{g_i}^{\neg b}$ must decrypt $T[g]_{i,\neg b}$ without $K_g^{\neg b}$, which can not be done with non-negligible probability under a computational restriction.

Also note that should $K_{g,i}^{\neg b}$ be compromised "legitimately" due to the original garbling scheme applied to its formulaic sub-circuit, the gate key $K_g^{\neg b}$ is still protected, as are $\left\{K_{g,j}^{\neg b}\right\}$ for other gates g_j where $A(g_j) = g$ or $B(g_j) = g$. The former is represented only by $F_{K_g^{\neg b}}(g||i)$ in $T[g]_{i,\neg b}$, and the latter are encrypted independently of $K_{g,i}^{\neg b}$.

Therefore, a computationally bounded adversary attempting to forge a garbled output, is at negligible advantage by seeing the ciphertexts produced by adapting our scheme to work for multi-fan-out gates. Given that our original scheme is unconditionally authentic for formulaic circuits, it hence can be adapted to work for general circuits (by using the routine in Fig. FC4.13) while preserving authenticity against computationally bounded adversaries.

4.5.3 Performance

This extension to generic circuits adds two κ-bit ciphertexts for each gate that has a multi-fan-out parent. In the worst case, the communication cost tends to that of Yao's original scheme; four ciphertexts per gate. For general purpose privacy-free garbling, the schemes of [18] and [21] are clearly much more efficient. However, for circuits that can be interpreted as blocks of formulaic sub-circuits "stitched" together by a small number of multi-fan-out gates, this extension exploits the formulaic nature of the sub-circuits in order to provide efficient privacy-free garbling using our original scheme.

Reducing One Ciphertext per Gate. To further reduce one ciphertext per gate, we can exploit the fact that we have one degree of freedom in choosing the gate keys for AND and XOR gates in our garbling scheme. Consider a gate g' to be the i^{th} child of gate g, without loss of generality $A(g') = g$. We can first choose the gate keys K_g^0, K_g^1 uni-

formly at random for g, and then set the 1-key on the left wire of g' as $L^1 := F_{K_g^1}(g||i)$. This is effectively ensuring that the ciphertext $T[g]_{i,1} = 0$, therefore removing its need for transmission. Note that this is possible because the garbling routines for AND and XOR gates produce 1-keys for the two input wires by choosing one at random and setting the other appropriately.

Feasibility Result. We note that this extension of our scheme demonstrates the feasibility of a privacy-free garbling scheme for general circuits (relying only on PRFs) that violates the single-key invariant for some wires in the circuit.

4.6 Applications of our Construction

Our construction finds direct application in settings where privacy of garbled inputs is not required, such as Zero knowledge protocols [1, 35] and Attribute-based key exchange [36].

We informally recall the zero-knowledge protocol of Jawurek et al. in Fig. FC4.14.

Zero-knowledge proofs for SAT. We note that as our construction guarantees authenticity against an unbounded evaluator, it therefore immediately yields a simple zero-knowledge proof system for Boolean formula satisfiability when plugged into the [1] construction[3] with linear overhead in the size of the formula times the soundness parameter. Zero-knowledge proof systems guarantee that even an unbounded malicious prover can not convince a verifier that an invalid statement is true. Garbling schemes relying on cryptographic assumptions will not yield ZK proofs without additional overhead in the protocol [32], as their security guarantees do not hold when the evaluator is unbounded.

[3] Assuming that the other primitives (OT, Commitment) are instantiated with statistical security against the prover.

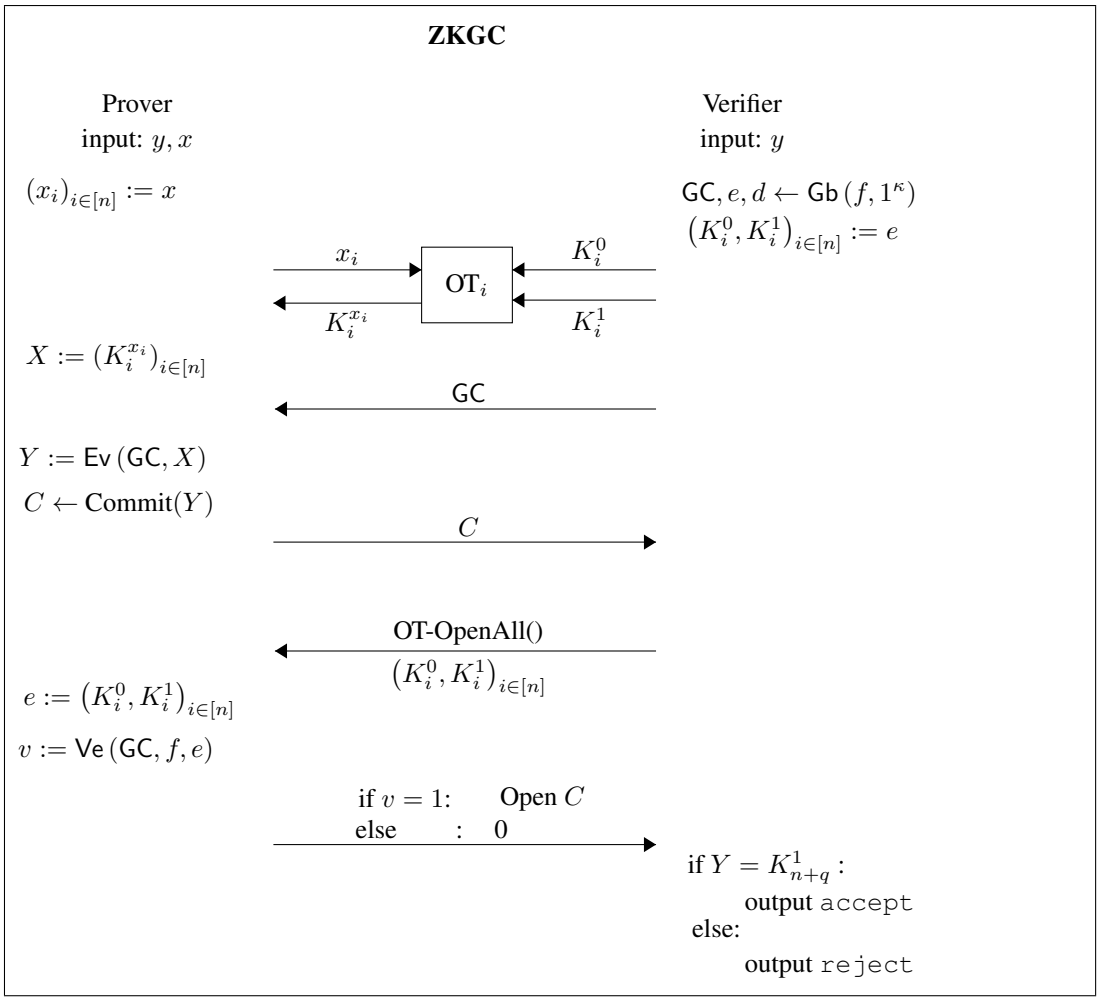

Figure FC4.14: ZKGC: Zero-knowledge from one GC [1]. Here, for an NP language L, a prover wishes to prove the statement $y \in L$, and the witness for the same is x. The circuit f realizes the witness relation checking function for the language; ie. $f(x) = 1$ iff. x is a valid witness for the statement $y \in L$. Oblivious transfer (OT) and Commitments are used as ideal primitives.

Concretely efficient ZK with weaker primitives. When security against an unbounded prover is not necessary, our construction provides the most communication and computation efficient ZK argument system for SAT yet in the [1] paradigm. Observe that with the simple PRG-based domain extension technique for OT [37] the number of public key operations required to encode a witness as garbled input becomes independent of the size of the formula. Competing privacy-free schemes [18, 21] rely on stronger cryptographic primitives (KDF/Circular correlation robust hash), and will also require

larger key sizes for the same level of security (as they are instantiated with computational security parameters). Essentially, number of ciphertexts when garbling a formula with [18,21] will be equal to the number of gates in the formula, which is also the number of inputs/leaves (which determines the communication cost for encoding an input for our garbling scheme). However, the size of each ciphertext in [18,21] will be larger than the comparable element in our garbling scheme, as ours will be instantiated with a statistical security parameter, as opposed to computational for [18,21].

CHAPTER 5

GARBLING WITHOUT AUTHENTICITY

As we have seen earlier, garbling schemes that achieve authenticity alone are well-studied. Along the lines of Frederiksen et al. [21] we ask the natural complementary question, *"can we leverage the lack of an authenticity requirement in order to construct more efficient garbling schemes?"*

"Authenticity-free" garbling schemes would find direct application in semi-honest 2PC, as well as many maliciously secure 2PC protocols in which the GC evaluator does not have to send the garbled output back to the constructor.

Unfortunately in this section we answer the above question in the negative, for most 'standard' garbling schemes. Specifically, we show that if a garbling scheme achieving privacy satisfies a notion of *composability*, then the scheme is necessarily authentic. Intuitively, any garbling scheme that does not treat output gates differently from intermediate/input gates will be composable. This definition covers state-of-the-art constructions such as those of HalfGates [18], Gueron et al. [19], and the basic Yao garbling scheme itself [4]. We show that an adversary who is able to forge a garbled output for a GC produced by a composable garbling scheme, can be used to perform multiple evaluations on a slightly larger circuit garbled by the same scheme.

5.1 Composable Garbling

Informally, a composable garbling scheme allows the 'output keys' of a previously garbled gate to be the 'input keys' to an instance of garbling another gate in the circuit. A Gb routine that directly uses the output keys of a gate as the input keys to a child gate, and does not distinguish between output and non-output gates, will make the garbling scheme composable.

We only consider projective garbling schemes which work by associating keys k_w^0, k_w^1 corresponding to semantic values 0 and 1 respectively for each input/output wire w in the circuit being garbled. Note that referring to the key corresponding to semantic 0 as k_w^0 is done for notational convenience; our proof is unaffected by point-and-permute style optimizations. We ignore cases where a garbling scheme does not achieve authenticity only because the De routine never outputs $\perp$; specifically we require that:

$$\mathsf{De}\,(\mathsf{Y}, d) \neq \perp \implies \mathsf{Y} = (\mathsf{Y}_i)_{i\in[m]} \text{ such that } \forall i \in [m],\ \mathsf{Y}_i \in \{k^0_{\mathsf{out}+i}, k^1_{\mathsf{out}+i}\} \tag{Eqn 5.1}$$

where $w \in [\mathsf{out}, \mathsf{out} + m]$ are the output wires. Also note that we use the language of [17] for circuits; the circuit itself is a directed acyclic graph, where each gate g is indexed by its outgoing wire, and its left and right incoming wires $A(g)$ and $B(g)$ are numbered such that $g > B(g) > A(g)$. Also, a circuit output wire can not be an input wire to any gate.

Definition 5.1.1 *A garbling scheme* $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is* composable *if there exists a garbling routine* $\mathsf{Gb}'_{\mathcal{G}}$ *as per Fig. FC5.1 such that the composed garbling scheme* $\mathcal{G}'_{\mathcal{G}} = \left(\mathsf{Gb}'_{\mathcal{G}}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}\right)$ *is* correct *and* private.

Our definition of composability is sufficient the purpose of our proof, while capturing most practical garbling schemes [18, 19]. Note that $\mathsf{Gb}'_{\mathcal{G}}$ does not by itself provide

For garbling scheme $\mathcal{G}$, we provide a template for a garbling routine $\mathsf{Gb}'_{\mathcal{G}}$ which uses the garbling routine $\mathsf{Gb} \in \mathcal{G}$ to garble a given circuit $\mathcal{C} : \{0,1\}^n \mapsto \{0,1\}^m$, which is subject to certain restrictions as given below. Circuit $\mathcal{C}$ is interpreted as the composition of a sub-circuit $f : \{0,1\}^{n-1} \mapsto \{0,1\}^m$ and a single 2 fan-in gate f'. The gate f' provides an output wire in circuit $\mathcal{C}$. The left and right incoming wires to f' are indexed L and R respectively. Clearly L is an output wire of f, and R is an input wire of $\mathcal{C}$. Note that given a circuit $\mathcal{C}'$ that satisfies this requirement, we can always construct $\mathcal{C}$ such that f' is indexed to be the last gate in $\mathcal{C}$, and L is indexed as the last (m^{th}) output wire of f. We assume $\mathcal{C}$ to be constructed as such.

$$\mathsf{Gb}'_{\mathcal{G}}\left(1^{\kappa}, \mathcal{C}\right)$$

1. Parse f and f' from $\mathcal{C}$, where f' is the last gate in $\mathcal{C}$.
2. Use $\mathcal{G}$ to garble f, i.e. $(\mathsf{GC}, e, d) \leftarrow \mathsf{Gb}\left(1^{\kappa}, f\right)$
3. Extract keys k_L^0, k_L^1 using $(f, \mathsf{GC}, e)^a$.
4. Choose fresh keys k_R^0, k_R^1. For garbling schemes such as FreeXOR which require a certain key structure, choose fresh input keys appropriately. Otherwise, two independent random κ-bit strings will suffice.
5. Compute $(\mathsf{GC}', e', d') = \mathsf{Gb}\left(1^{\kappa}, f'\right)$ such that $\left(e' = \left(\left(k_L^0, k_L^1\right), \left(k_R^0, k_R^1\right)\right)\right.$
6. Set $\mathsf{GC}'' = \mathsf{GC}||\mathsf{GC}'$, $e'' = e||e'[2]$ and $d'' = d[1]||d[2]||\cdots||d[m-1]||d'$.
7. **return** GC'', e'', d''

[a] This can be done by saving the required keys from Step 2

Figure FC5.1: Specification of a composing Gb routine

a template for the full garbling routines of such schemes. However, it is easy to extend this template for composable garbling to more accurately reflect garbling for general circuits by removing the constraints on the input wires of f' (ie. L and R). While not required for our proof, we provide such a template capturing gate-by-gate garbling schemes in Appendix A for completeness. The tradeoff for a more precise template is a loss of generality in the garbling techniques caputured; as an example, the work of [38] abandons the gate-by-gate approach to garbling. The composability requirement (Fig. FC5.1) for our proof is meaningful for any projective garbling scheme, which we believe will be relevant beyond current garbling techniques.

5.2 Relating Composability to Authenticity

We can show that a garbling scheme $\mathcal{G}$ whose composed garbling scheme $\mathcal{G}'$ is private and correct as per Def. 5.1.1 must necessarily be authentic. Intuitively, this is because an intermediate wire of a circuit $\mathcal{C}$ is effectively the output wire of some induced sub-circuit f. Given that the garbling scheme is composable, if it is possible to forge a missing output key for a garbling of f, it will be possible to perform multiple evaluations of an intermediate gate in a garbling of $\mathcal{C}$. Multiple evaluations can be used to derive the semantic value of some intermediate key, allowing an adversarial evaluator to distinguish between a simulated GC and a legitimately constructed GC. This is because the simulator only has access to the clear function output; for a carefully chosen circuit, even an arbitrarily complex simulation strategy will be as successful as guessing a circuit input (which is adversarially chosen).

Theorem 5.2.1 *If a* private *and* correct *garbling scheme* $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is* composable *as per Definition 5.1.1, then it is also* authentic.

Proof. Given black-box access to a an adversary $\mathcal{A}_{\mathsf{aut}}$ who is able to forge a garbled output for some f, x garbled using $\mathcal{G}$, we construct an adversary $\mathcal{A}_{\mathsf{prv}}$ who can distinguish between a legitimate and simulated garbling of some $\mathcal{C}, x||x'$ as per composed garbling scheme $\mathcal{G}'$ (as per Def. 5.1.1). For simplicity, we assume that f outputs only 1 bit; we will address the general case later.

This reduction of the authenticity of garbling scheme $\mathcal{G}$ to the privacy of composed garbling scheme $\mathcal{G}'$ is formally described in Fig. FC5.2, with a formal analysis below.

Advantage of $\mathcal{A}_{\mathsf{prv}}$. Let the advantage of $\mathcal{A}_{\mathsf{aut}}$ in correctly forging a valid missing garbled output for a GC and input produced by $\mathcal{G}$ be $h(\kappa)$ (see Def. 2.5.2). Index the REAL world $b = 0$, and IDEAL world $b = 1$. We analyze the following cases in order to deter-

Assuming black box access to an adversary $\mathcal{A}_{\mathsf{aut}}$ who can forge a garbled output for some function $f : \{0,1\}^n \mapsto \{0,1\}$ and input $x \in \{0,1\}^n$ as per $\mathcal{G}$, we construct adversary $\mathcal{A}_{\mathsf{prv}}$ who can distinguish whether a given (GC, X, d) corresponding to a larger circuit $\mathcal{C}$ as per $\mathcal{G}'$ is produced legitimately, or simulated.

$$\mathcal{A}_{\mathsf{prv}}\left(1^{\kappa}\right)$$

1. **receive** f, x from $\mathcal{A}_{\mathsf{aut}}$, compute $b = f(x)$
2. Construct $\mathcal{C}\left(y, y'\right)$ as follows[a]:
   - **if** $b = 0$ **then** $\mathcal{C}\left(y, y'\right) = f(y) \wedge y'$
   - **else** $\mathcal{C}\left(y, y'\right) = (\neg f(y)) \wedge y'$
3. Sample bit $x' \xleftarrow{R} \{0,1\}$
4. **send** $\mathcal{C}, x||x'$ as the function and input to the challenger of the privacy experiment, and **receive** GC'', X'', d'' as a response.
5. From the above response, parse $\mathsf{GC}||\mathsf{GC}' = \mathsf{GC}''$, where $\mathsf{GC} \in \mathsf{Gb}(1^{\kappa}, f)$ and $\mathsf{GC}' \in \mathsf{Gb}(1^{\kappa}, \wedge)$, as well as $X||X' = X''$ where $X \in \mathsf{En}(y, \cdot)$ and $X' \in \mathsf{En}(y', \cdot)$
6. Compute $Z = \mathsf{Ev}\left(\mathsf{GC}, X\right)$
7. **send** (GC, X) to $\mathcal{A}_{\mathsf{aut}}$ and **receive** Z' as a candidate forged output.
8. Compute $Y = \mathsf{Ev}\left(\mathsf{GC}', Z||X'\right)$ and $Y' = \mathsf{Ev}\left(\mathsf{GC}', Z'||X'\right)$
9. Check the consistency of the final garbled gate to determine whether it was garbled legitimately, as follows:
   - **if** $\mathsf{De}\left(Y', d\right) = \perp$ **then output** $guess \xleftarrow{R} \{\textsc{Real}, \textsc{Ideal}\}$
   - **else if** $(Y' = Y$ and $x' = 0)$ or $(Y' \neq Y$ and $x' = 1)$ **then output** $guess = \textsc{Real}$
   - **else output** $guess = \textsc{Ideal}$

[a] We must ensure that the output of $\mathcal{C}(y, y')$ is always 0 in order to completely hide the value y' from $\mathcal{S}$, thereby providing no clues for $\mathcal{S}$ to garble the correct input for that wire.

Figure FC5.2: Reduction of authenticity of $\mathcal{G}$ to privacy of $\mathcal{G}'$

mine how much of this advantage is translated in distinguishing a simulated (GC, X, d) from one legitimately produced by $\mathcal{G}'$:

1. **Real world:** $(\mathsf{GC}, X, d) \in \{\textsc{Real}_{\mathcal{G}'}\left(\mathcal{C}, x||x'\right)\}$.

   a) $\mathcal{A}_{\mathsf{aut}}$ *is successful in forging an output.* This case occurs with probability $h\left(\kappa\right)$. Then, $\mathcal{A}_{\mathsf{prv}}$ guesses Real and is correct with probability 1.

   b) $\mathcal{A}_{\mathsf{aut}}$ *is unsuccessful in forging an output.* This case occurs with probability $1 - h\left(\kappa\right)$. Then, either decoding or evaluation of the final gate will fail; $\mathcal{A}_{\mathsf{prv}}$

submits a random guess and is correct with probability $\frac{1}{2}$.

In this case, the adversary $\mathcal{A}_{\mathsf{prv}}$ correctly guesses that she is in the REAL world with the following probability:

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\kappa}\right)=0 \mid b=0\right] &= h\left(\kappa\right)\cdot 1+\left(1-h\left(\kappa\right)\right)\cdot\frac{1}{2} \\ &= \frac{1}{2}\cdot\left(h\left(\kappa\right)+1\right)\end{aligned} \tag{Eqn 5.2}$$

2. **IDEAL world:** $(\mathsf{GC}, X, d) \in \{\text{IDEAL}_{\mathcal{S}}\left(\mathcal{C}, 0\right)\}$.

   a. $\mathcal{A}_{\mathsf{aut}}$ *is successful in forging an output.* This case occurs with some probability $h'\left(\kappa\right)$. We will give the benefit of the doubt to $\mathcal{S}$ and assume that all keys and ciphertexts in GC are consistent, and that the decoding information is constructed correctly (if not, $\mathcal{A}_{\mathsf{prv}}$ will terminate with a random guess, winning with probability $\frac{1}{2}$). Given that GC is correct, $\mathcal{A}_{\mathsf{prv}}$ will derive the semantic value of the key she was given for input x'. Note that $\mathcal{S}$ has no information about x' whatsoever (as the function output is always 0) and hence can at best guess x' at random to provide an appropriate key. Each of the following cases occur with probability $\frac{1}{2}$:

      i) $\mathcal{S}$ guesses x' correctly, $\mathcal{A}_{\mathsf{prv}}$ outputs IDEAL with probability 0.

      ii) $\mathcal{S}$ guesses x' incorrectly, $\mathcal{A}_{\mathsf{prv}}$ outputs IDEAL with probability 1.

   b. $\mathcal{A}_{\mathsf{aut}}$ *is unsuccessful in forging an output.* This case occurs with probability $1-h'\left(\kappa\right)$. Then, either decoding or evaluation of the final gate will fail; $\mathcal{A}_{\mathsf{prv}}$ submits a random guess and is correct with probability $\frac{1}{2}$.

The adversary $\mathcal{A}_{\mathsf{prv}}$ therefore has the following probability in guessing correctly

that she is in the ideal world:

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\kappa}\right)=1 \mid b=1\right] &= h'\left(\kappa\right)\cdot\left[\frac{1}{2}\cdot 0+\frac{1}{2}\cdot 1\right]+\left(1-h'\left(\kappa\right)\right)\cdot\frac{1}{2}\\ &= \frac{1}{2}\end{aligned} \tag{Eqn 5.3}$$

We can compute the advantage of $\mathcal{A}_{\mathsf{prv}}$ in distinguishing the output distributions $\{\textsc{Real}_{\mathcal{G}'}\left(\mathcal{C}, x||x'\right)\}$ and $\{\textsc{Ideal}_{\mathcal{S}}\left(\mathcal{C}, 0\right)\}$, plugging in the probabilities from (Eqn 5.2) and (Eqn 5.3) as follows:

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\kappa}\right)=0 \mid b=0\right]-\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\kappa}\right)=0 \mid b=1\right] &= \frac{1}{2}\cdot\left(h\left(\kappa\right)+1\right)-\frac{1}{2}\\ &= \frac{1}{2}\cdot h\left(\kappa\right)\end{aligned} \tag{Eqn 5.4}$$

Given that $\mathcal{G}$ is composable, $\mathcal{G}'$ is private by definition. As per Def. 2.5.1, the advantage of $\mathcal{A}_{\mathsf{prv}}$ computed in (Eqn 5.4) must be negligible, implying that $h\left(\kappa\right)$ must be negligible for all $\mathcal{A}_{\mathsf{aut}}\left(1^{\kappa}\right)$. There can not exist a PPT adversary $\mathcal{A}_{\mathsf{aut}}$ who can successfully forge a garbled output for any GC, X produced by a composable garbling scheme $\mathcal{G}$ with non-negligible advantage as per Definition 2.5.2. Therefore, given that a garbling scheme $\mathcal{G}$ is composable, it is necessarily authentic, and this proves Theorem 5.2.1. ■

What if f has multiple wires? To handle the case where $f(x)$ has m output wires, we define $\mathcal{C}(y, y')$ such that y' is a bit vector $\left(y'_i\right)_{i\in[m]}$. Denoting f_i and $\mathcal{C}_i$ to be the i^{th} output bits of f and $\mathcal{C}$ respectively, we define $\mathcal{C}_i\left(y, y'\right) = f_i\left(y\right) \wedge y'_i$, $\forall i \in [m]$ (adjusting as per Step 2 of Fig. FC5.2). The strategy of $\mathcal{A}_{\mathsf{prv}}$ hence follows with at least the same advantage, as if $\mathcal{A}_{\mathsf{aut}}$ forges output keys on m' wires, $\mathcal{S}$ is successful only when it guesses every y'_i input to the corresponding wires, which it can do probability no greater than $2^{-m'}$.

5.3 Feasibility of Authenticity-Free Garbling

Garbling gate by gate in topological order where the output keys of one garbled gate are used as the input keys to its children, coupled with circuit output key distributions being (nearly) identical to the input key distribution of intermediate gates, is the dominant paradigm underlying most state-of-the-art garbling schemes for Boolean circuits [15, 16, 18–20]. This means that the current methods of garbling Boolean circuits privately is inherently composable, therefore making authenticity impossible to avoid.

However, we note that it is possible to have efficient authenticity-free garbling that is non-composable, assuming access to a PRF F. Consider any projective topological gate-by-gate garbling scheme $\mathcal{G}$ with the following modifications to construct $\mathcal{G}'$:

- Gb : Garble all gates until the output layer topologically as per $\mathcal{G}$; the keys on wire w are k_w^0, k_w^1 corresponding to semantic values 0 and 1, and the ciphertext for wire w is stored in $T[w]$. Let $i \in [m]$ index the output gates, while ℓ_i and r_i index the left and right incoming wires of i respectively. For all $i \in [m]$ such that i is an AND gate, set ciphertext $T[i] := F_{k_{\ell_i}^1}(i) \oplus F_{k_{r_i}^1}(i)$, zero-key $k_i^0 := 0^\kappa$ and one-key $k_i^1 := 1^\kappa$.

- Ev : Evaluate all gates until the output layer topologically as per $\mathcal{G}$; the key obtained on wire w is k_w. Let $i \in [m], \ell_i, r_i$ be defined as earlier. Evaluating output AND gate $i \in [m]$ proceeds as follows: compute $C := F_{k_{\ell_i}}(i) \oplus F_{k_{r_i}}(i)$. If $C = T[i]$ then set $k_i = 1^\kappa$, otherwise set $k_i = 0^\kappa$.

Security. The output keys for any output-layer AND gate are predictably always 0^κ and 1^κ, making $\mathcal{G}'$ clearly non-authentic. These output keys can not be reused as input keys to another gate for the same reason, making $\mathcal{G}'$ non-composable. However if $\mathcal{G}$ is private and correct, then so is $\mathcal{G}'$; leaking the semantic values of the output keys to the evalu-

ator does not compromise privacy. The privacy property of a garbling scheme requires that a PPT $\mathcal{A}$ can not (with non-negligible advantage) distinguish between an honestly constructed (GC, X, d) and such values constructed by a simulator that has access to only the clear output $f(x)$ (and not x). $\mathcal{A}$ is allowed to see the decoded output anyway, therefore the distribution of the routines of $\mathcal{G}'$ can be simulated for privacy if those of $\mathcal{G}$ can be. The Garble1 scheme of [17] achieves privacy despite leaking the semantic values of the output wires.

Performance. Most practical garbling schemes are shown to satisfy a definition of 'linearity' by Zahur et al. [18]. They go on to show that a linear garbling scheme achieving privacy requires at least two ciphertexts to garble an AND gate. However, $\mathcal{G}'$ garbles every output AND gate with just one ciphertext, implying that for any linear $\mathcal{G}$, a corresponding $\mathcal{G}'$ as defined above will necessarily produce *one less ciphertext* per output AND gate.

Lower bound of [18]. The garbling scheme $\mathcal{G}'$ when used to garble a single AND gate in isolation produces only one ciphertext, which may seem to contradict the 2-ciphertext lower bound for private garbling proven in [18]. However the Ev routine of of $\mathcal{G}'$ makes use of a comparison operation, which disqualifies it from being a linear garbling scheme.

CHAPTER 6

CONCLUSION

In this thesis we have studied the conceptual separations and relations between the authenticity and privacy security goals of a complex cryptographic primitive, ie. garbling schemes. Our study of privacy-free garbled circuits yielded a construction proving that a previously claimed lower bound for garbling an atomic AND gate in this setting does not hold. Furthermore our construction composes to garble formulas with unconditional security, in the process demonstrating another conceptual separation between garbling with and without privacy; that the key size of a gate can be independent of its depth in the latter case. However the most significant conceptual difference in garbling without privacy is brought out in the ability to leak both keys on certain wires of the circuit.

We also initiated the study of authenticity-free garbled circuits. It is possible to construct garbling schemes achieving privacy alone, that are more efficient than their authentic counterparts. However, our result indicates that any approach to authenticity-free garbling must be inherently non-composable as per our definition. We leave as an open problem the task of constructing garbling schemes achieving privacy that meaningfully sacrifice authenticity in favour of improved efficiency.

Bibliography

[1] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In Sadeghi et al. [39], pages 955–966.

[2] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[3] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[4] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[5] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[6] 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.

[7] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *EU-*

ROCRYPT 2007, volume 4515 of *LNCS*, pages 52–78. Springer, Heidelberg, May 2007.

[8] Yehuda Lindell and Ben Riva. Blazing fast 2PC in the offline/online setting with security for malicious adversaries. In Ray et al. [40], pages 579–590.

[9] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.

[10] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Paterson [41], pages 406–425.

[11] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, May 2014.

[12] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2008.

[13] Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In Cramer [42], pages 190–208.

[14] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 129–139, New York, NY, USA, 1999. ACM.

[15] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASI-*

ACRYPT 2009, volume 5912 of *LNCS*, pages 250–267. Springer, Heidelberg, December 2009.

[16] 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, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008.

[17] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012.

[18] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Oswald and Fischlin [43], pages 220–250.

[19] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In Ray et al. [40], pages 567–578.

[20] 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, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457. Springer, Heidelberg, August 2014.

[21] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In Oswald and Fischlin [43], pages 191–219.

[22] Carmen Kempka, Ryo Kikuchi, Susumu Kiyoshima, and Koutarou Suzuki. Garbling scheme for formulas with constant size of garbled gates. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 758–782. Springer, Heidelberg, November / December 2015.

[23] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2015.

[24] Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 136–155. Springer, Heidelberg, December 2005.

[25] Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 565–577, 2016.

[26] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In Cramer [42], pages 39–53.

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

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

[29] abhi shelat and Chih-Hao Shen. Fast two-party secure computation with minimal assumptions. In Sadeghi et al. [39], pages 523–534.

[30] abhi shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. In Paterson [41], pages 386–405.

[31] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 329–346. Springer, Heidelberg, March 2011.

[32] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors,

CRYPTO 2016, Part II, volume 9815 of *LNCS*, pages 397–429. Springer, Heidelberg, August 2016.

[33] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012.

[34] 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.

[35] Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 499–530. Springer, Heidelberg, August 2016.

[36] 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.

[37] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[38] Tal Malkin, Valerio Pastro, and abhi shelat. An algebraic approach to garbling, 2016. Unpublished manuscript, see `https://simons.berkeley.edu/talks/tal-malkin-2015-06-10`.

[39] Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors. *ACM CCS 13*. ACM Press, November 2013.

[40] Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors. *ACM CCS 15*. ACM Press, October 2015.

[41] Kenneth G. Paterson, editor. *EUROCRYPT 2011*, volume 6632 of *LNCS*. Springer, Heidelberg, May 2011.

[42] Ronald Cramer, editor. *TCC 2012*, volume 7194 of *LNCS*. Springer, Heidelberg, March 2012.

[43] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*. Springer, Heidelberg, April 2015.

APPENDIX A

COMPOSABLE GATE-BY-GATE GARBLING

Our template for gate-by-gate composable garbling is detailed in Fig. FC1.1.

For garbling scheme $\mathcal{G}$, we provide a template for a garbling routine $\mathsf{Gb}^\star$ which uses the garbling routine $\mathsf{Gb} \in \mathcal{G}$ to garble a given circuit $f^\star$. A circuit $f^\star$ is interpreted as the composition of a sub-circuit f and a single 2 fan-in gate f'. The gate f' provides an output wire in circuit f. The left and right incoming wires to f' are indexed L and R respectively. Note that L and R could each be any of the following: an input wire to f, an internal wire in f, or an input wire of $f^\star$ that does not touch f at all.

$$\mathsf{Gb}^\star\left(1^\kappa, f^\star\right)$$

1. Parse f and f' from $f^\star$, where f' was indexed as the last gate in f
2. Use $\mathcal{G}$ to garble f. $\mathsf{GC}, e, d \leftarrow \mathsf{Gb}\left(1^\kappa, f\right)$
3. For $w \in \{L, R\}$ such that w is a wire touching f, extract keys k_w^0, k_w^1 using (f, GC, e)[a].
4. For $w \in \{L, R\}$ such that w is a wire not touching f, choose fresh keys k_w^0, k_w^1. For garbling schemes such as FreeXOR which require a certain key structure, choose fresh input keys appropriately. Otherwise, two independent random κ-bit strings will suffice.
5. Compute $\mathsf{GC}', e', d' = \mathsf{Gb}\left(1^\kappa, f'\right)$ such that $e' = \left(\left(k_L^0, k_L^1\right), \left(k_R^0, k_R^1\right)\right)$
6. Set $\mathsf{GC}^\star = \mathsf{GC}||\mathsf{GC}'$, and initialize $e^\star = e$ and $d^\star = d||d'$.
7. The encoding and decoding information for $\mathsf{GC}^\star$ are made consistent:
   - **if** L does not touch f **then** Update $e^\star = e^\star||e'[0]$.
   - **if** R does not touch f **then** Update $e^\star = e^\star||e'[1]$.
   - For $w \in \{L, R\}$ such that w is the i^{th} output wire of f, update $d^\star = d^\star \setminus d[i]$
8. **return** $\mathsf{GC}^\star, e^\star, d^\star$

[a] This can be done by saving the required keys from Step 2

Figure FC1.1: Complete Specification of a Gate-by-gate Composing Gb Routine

Given garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$, we construct composed garbling scheme $\mathcal{G}^\star = (\mathsf{Gb}^\star, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ as per Fig. FC1.1. $\mathcal{G}$ is composable if there exists a $\mathcal{G}^\star$ as per Fig. FC1.1 which achieves privacy. Note that when the output distribution of $\mathsf{Gb}^\star$ is identical to that of Gb, $\mathcal{G}^\star$ is equivalent to $\mathcal{G}$, and hence Gb can be composed recursively to garble any poly-size circuit gate by gate. Most practical garbling schemes [18, 19] already follow this template.