Foundations of Garbled Circuits

Mihir Bellare
Dept of CS and Engineering
UC San Diego, USA

Viet Tung Hoang
Dept of Computer Science
UC Davis, USA

Phillip Rogaway
Dept of Computer Science
UC Davis, USA

ABSTRACT

Garbled circuits, a classical idea rooted in the work of Yao, have long been understood as a cryptographic *technique*, not a cryptographic *goal*. Here we cull out a primitive corresponding to this technique. We call it a *garbling scheme*. We provide a provable-security treatment for garbling schemes, endowing them with a versatile syntax and multiple security definitions. The most basic of these, *privacy*, suffices for two-party secure function evaluation (SFE) and private function evaluation (PFE). Starting from a PRF, we provide an efficient garbling scheme achieving privacy and we analyze its concrete security. We next consider *obliviousness* and *authenticity*, properties needed for private and verifiable outsourcing of computation. We extend our scheme to achieve these ends. We provide highly efficient blockcipher-based instantiations of both schemes. Our treatment of garbling schemes presages more efficient garbling, more rigorous analyses, and more modularly designed higher-level protocols.

Categories and Subject Descriptors

D.4.6 [**Software**]: Security and Protection—*cryptography*; E.3 [**Data**]: Data Encryption—*symmetric encryption*

Keywords

Garbled circuits, garbling schemes, provable security, secure function evaluation, Yao's protocol

1. INTRODUCTION

Overview. This paper is about elevating garbled circuits from a cryptographic *technique* to a cryptographic *goal*. While circuit garbling has traditionally been viewed as a method for achieving SFE (secure function evaluation) or some other cryptographic goal, we view it as an end goal in its own right, defining *garbling schemes* and formalizing several notions of security for them, these encompassing *privacy*, *authenticity*, and *obliviousness*. This enables more modular use of garbled circuits in higher-level protocols and grounds follow-on work, including the development of new and highly efficient schemes.

History. The idea of a garbled circuit is due to A. Yao, who described the technique in oral presentations [18] (p. 27) about SFE [50, 51]. The first written account of the method is by Goldreich, Micali, and Wigderson [19]. The protocol they describe, crediting Yao [50], involves associating two *tokens* to each wire of a boolean circuit, these having hidden semantics of 0 and 1. Means are then provided to propagate tokens across a gate, preserving the hidden semantics. More specifically, there's a four-row table for each gate of the circuit, each row employing public-key encryption to encrypt a pair of random strings whose xor is the token for the outgoing wire.

The term *garbled circuit* is from Beaver, Micali, and Rogaway [10], where the method was first based on a symmetric primitive. Garbled circuits took on a modern, PRF-based instantiation in work by Naor, Pinkas, and Sumner on privacy-preserving auctions [40].

Yao's idea has been enormously impactful, engendering numerous applications, implementations, and refinements. Still, there has been little definitional attention paid to garbled circuits themselves. Lindell and Pinkas [35] provide the first proof of Yao's protocol—to the extent one can say that a particular scheme is Yao's—but, even there, the authors do not formalize garbled circuits or what it means to securely create one. Instead, they prove that a particular garbled-circuit-using protocol, one based on double encryption [18], is a secure two-party SFE. Implemented SFE methods do not coincide with what's in Lindell and Pinkas [35], and absence of a good abstraction boundary makes daunting the task of providing a full proof for what's actually in optimized SFE implementations.

Scattered throughout the enormous literature dealing with garbled circuits, several papers do work to abstract out what these provide. A first set of such work begins with Feige, Kilian, and Naor [15] and is followed by [8, 13, 28, 30]. Each paper aims to modularly use garbled circuits in some intending application. To that end, they single out, definitionally, precisely what they need, usually arriving at something close to what we will later call "prv.sim security over Φ_{circ}." None of the papers pick up definitions from any other, nor does any prove that any particular construction satisfies the notion given. The conceptualization of garbling as involving a component that creates garbled circuits and another that evaluates them is found in all of these works, and in Schneider's [47, 48]. A second line of definitions begins with Ishai and Kushilevitz [25] and continues with [2, 3, 5, 6, 26, 27, 46].

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

These works define various flavors of *randomized encodings*. Their authors do see randomized encodings as a general-purpose primitive, and the definitions elegantly support a variety of theory-centered work. However, they lack the fine-grained syntax that we shall need to investigate obliviousness, authenticity, and precise measures of efficiency. Finally, in concurrent work, Kamara and Wei offer definitions to support their idea of garbling *structured* circuits [29]. See Appendix A for further discussion of selected related work.

CONTRIBUTIONS. We formalize what we call a *garbling scheme*. The notion is designed to support a burgeoning and practical area: the myriad applications of garbled circuits. Our definitions and results enable easy and widespread applications with modular, simplified, and yet more rigorous proofs of security.

Roughly said, a garbling algorithm Gb is a randomized algorithm that transforms a function $f\colon \{0,1\}^n \to \{0,1\}^m$ into a triple of functions $(F, e, d) \leftarrow \mathsf{Gb}(f)$. We require that $f = d \circ F \circ e$. The *encoding function* e turns an *initial input* $x \in \{0,1\}^n$ into a *garbled input* $X = e(x)$. Evaluating the *garbled function* F on the garbled input X gives a *garbled output* $Y = F(X)$. The *decoding function* d turns the garbled output Y into the *final output* $y = d(Y)$, which must coincide with $f(x)$. Informally, one has probabilistically factored f into $d \circ F \circ e$. Formally, it is problematic to regard Gb as operating on functions. Thus a garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is regarded as a five-tuple of algorithms, with strings d, e, f, and F interpreted as functions under the auspices of functions De, En, ev, and Ev. See Fig. 1.

Our syntactic framework is representation-independent; circuits are nowhere to be found. One can garble DFAs, OBDDs, RAMs, TMs, whatever; definitionally, this isn't even seen. See Section A, "Eclectic representations."

Of course none of this says anything about the desired security notion. We define several. The most important is *privacy*: a party acquiring (F, X, d) shouldn't learn anything impermissible beyond that which is revealed by knowing just the final output y. To formalize that which it *is* permissible to reveal, a *side-information function*, Φ, parameterizes the definition; an adversary should be able to ascertain from (F, X, d) nothing beyond $\Phi(f)$ and y. By varying Φ one can encompass the customary setting for SFE (let $\Phi(f) = f$; circuit f is not concealed) and PFE (private function evaluation) (let $\Phi(f)$ be the number of gates of f; leak just the circuit's size). We formalize privacy in multiple ways, giving an indistinguishability definition, prv.ind, and a simulation-based one, prv.sim. We show that whether or not they are equivalent depends on the side-information function Φ. For the most important ones the notions are equivalent (in general, they are not).

We provide a simple garbling scheme, Garble1, for achieving privacy. The scheme is conveniently described in terms of a *dual-key cipher* (DKC), a notion we put forward. We define a DKC's security and prove privacy for Garble1 under this assumption. Garble1 is described with uncustomary precision, including a detailed and precise definition of circuits. We show how to make a DKC from a pseudorandom function (PRF), and how to realize the PRF using a conventional blockcipher, say AES128. In this way we obtain a provably secure, blockcipher-based garbling scheme where circuit evaluation takes two AES calls per gate.

We go on to suggest a still more efficient instantiation for the dual-key cipher, one where evaluating a garbled circuit needs only *one* AES128 call per gate and all blockcipher invocations use the *same* key. This is the fastest approach now known for garbling circuits.

Beyond privacy we consider *obliviousness*: a party acquiring F and X, but not d, shouldn't learn anything about f, x, or y. As with privacy, we formalize obliviousness in different but "usually" equivalent ways. Next we explore *authenticity*: a party who learns F and X should be unable to produce a garbled output Y^* different from $F(X)$ that is deemed to be valid: $d(Y^*) \neq \perp$. Our interest in obliviousness and authenticity was sparked by Gennaro, Gentry, and Parno [17]; the notions arise in the context of private, verifiable outsourcing of computation.

We prove implications and separation among all security notions we have mentioned, painting a complete picture of definitions for this space. See Fig. 2.

We define a protocol, Garble2, to simultaneously achieve privacy, obliviousness, and authenticity. The assumption required is the same as before. The scheme is only a bit more complex than Garble1, the efficiency, only a little worse.

DISCUSSION. Once viewed as a "theoretical" approach for multiparty computation, a long line of work, beginning with Fairplay [37], has made clear that circuit garbling is now a practical technique. State-of-the-art implementations by Huang *et al.* and Kreuter *et al.* can handle complex functionalities and hundreds of millions of gates [23, 24, 32]. We aim to support such work, and applications further afield. With a protocol's garbling scheme delineated, implementations can more reasonably offer proofs for the actual scheme employed, the "messy" optimizations stripped of surrounding interaction and protocol aims. In general, an approach where the garbling scheme is conceptually separated from its use seems essential for managing complexity in this domain. As an analog, authenticated encryption took off after it was reconceptualized as a primitive, not a method formed of encryption schemes and MACs.

Garble1 and Garble2 are close to numerous other protocols (especially [40]) that incarnate Yao's idea. Given this, one might assume that, once good definitions *are* written down, proving security would be easy, based on prior work [35]. From our experience, this is not the case; the proofs we provide are not implicit in prior work.

One thing novel about our schemes is that they admit efficient AES-based instantiations whose quantitative security may be inferred via the concrete security bounds associated to our theorems. In the past, SFE schemes supported by proofs would use objects less efficiently realizable in practice [35], or, for practical realizations, would abandon proven-secure schemes and use hash-based ones, sometimes with an unproven claim that security is maintained in the random-oracle model. Given the increasing ubiquity of AES hardware support, we believe that optimized, proven, blockcipher-based schemes are a good direction.

This paper is the first of several we envision. In it we aim to instill fresh, practice-oriented foundations in an area where, historically, omitted definitions and proofs have been the norm. The current work maintains a circumscribed focus: to investigate the definitions central to the reconceptualization of garbling schemes as a *sui generis* cryptographic goal. Upcoming work will explore several directions:

- We can construct *extremely efficient* garbling schemes, like the one-call, fixed-key, AES128-based scheme we mentioned. This can be done in a way that does not preclude

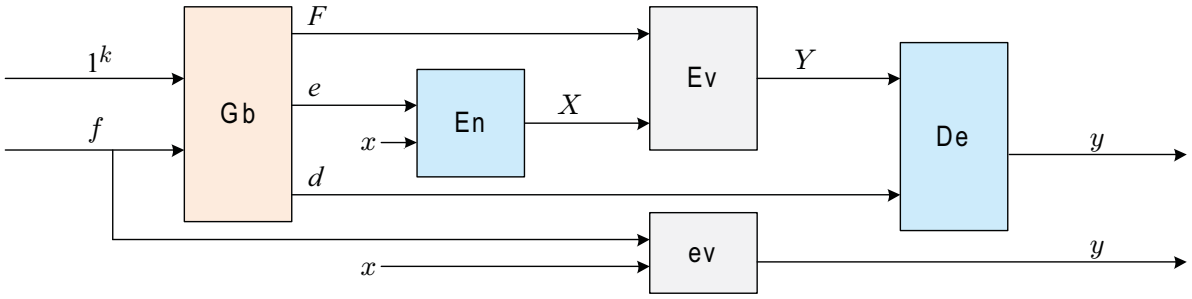

Figure 1: Components of a garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. Function Gb maps f and k to (F, e, d), strings encoding the garbled function, the encoding function, and the decoding function. Possession of e and x lets one compute the garbled input $X = \mathsf{En}(e, x)$; having F and X lets one calculate the garbled output $Y = \mathsf{Ev}(F, X)$; and knowing d and Y lets one recover the final output $y = \mathsf{De}(d, Y)$, which must equal $\mathsf{ev}(f, x)$.

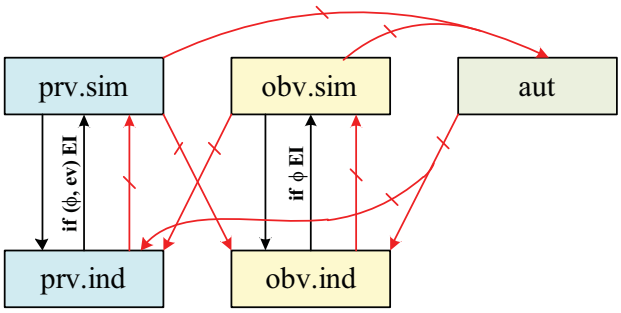

Figure 2: Relations among security notions. A solid arrow is an implication; an if-labeled arrow, a conditional implication; a hatched arrow, a separation.

the free-xor and row-elimination techniques that have proven so effective [23, 31, 44]. Proofs remain complex, even in the random-permutation model. Implementations are underway, these achieving about 15 nsec/gate.

- We can generalize security to the *adaptive* (=*dynamic*) setting. This is needed for one-time programs [20] and secure outsourcing [17]. For one flavor of adaptivity, prv1/obv1/aut1, the input x may depend on the garbled function F. For finer-grained notions, prv2/obv2/aut2, each bit of x can depend on previously acquired X_i-values. Transformations turn prv/obv/aut schemes into prv1/obv1/aut1 ones and these into prv2/obv2/aut2 ones.
- Building on the oft-described metaphor of lockboxes and keys (eg, [35] (pp. 163–164)), we can formulate garbling-scheme security using a formal treatment of dual-key enciphering. We choose to do this by absorbing the functionality of the ideal primitive into the code-based definition. Privacy, obliviousness, and authenticity become yes/no matters—no probabilities.

For all of these directions, the framework developed here serves as the needed starting point.

A thesis underlying our definitions is that they *work*—that most (though not all) applications described as using garbled circuits can be built from an arbitrary garbling scheme, instead. To date we have surveyed 20 papers containing protocols that can be recast to use a generic garbling scheme. See Fig. 3. In all cases we gain in simplicity and modularity. Applications benefit from the increased efficiency of our garbling schemes. The improvement is particularly marked in the application to KDM encryption (security with respect to key-dependent messages), where use of our abstraction leads to substantial efficiency gains over the use of the abstractions in previous work [2, 8].

Protocol	Application	Needs	Over
Y86 [18]	2-party SFE (sh)	prv	Φ_{circ}
AF90 [1]	PFE (sh)	prv	Φ_{size}
FKN94 [15]	server-aided SFE (sh)	prv	Φ_{circ}
NPS99 [40]	private auctions	prv	Φ_{circ}
KO04 [30]	2-party SFE (ma)	prv	Φ_{circ}
FAZ05 [16]	private credit checking	prv	Φ_{size}
FM06 [38]	2-party SFE (ma)	prv	Φ_{circ}
AL07 [7]	2-party SFE (covert)	prv	Φ_{circ}
LP07 [34]	2-party SFE (ma)	prv	Φ_{circ}
GKR08 [20]	one-time programs	prv2	Φ_{size}
GMS08 [21]	2-party SFE (co)	prv	Φ_{circ}
BFK+09 [9]	priv medical diag	obv	Φ_{circ}
PSS09 [41]	private credit checking	prv	Φ_{topo}
BHHI10 [8]	KDM encryption	prv	Φ_{size}
GGP10 [17]	auth outsourcing	aut1 + obv1	Φ_{circ}
HS10 [22]	2P guaranteed SFE	prv	Φ_{circ}
SS10 [46]	worry-free encryption	prv	Φ_{size}
Ap11 [2]	KDM encryption	prv	Φ_{size}
KMR11 [28]	server-aided SFE (ma)	aut + obv	Φ_{circ}
LP11 [36]	2-party SFE (ma)	prv	Φ_{circ}

Figure 3: Recasting protocols in more generic terms. so = semi-honest; co = covert; ma = malicious. All but [17] need the scheme to be projective.

2. PRELIMINARIES

NOTATION. We let $\mathbb{N}$ be the set of positive integers. A *string* is a finite sequence of bits and $\perp$ is a formal symbol that is not a string. If A is a finite set then $y \twoheadleftarrow A$ denotes selecting an element of A uniformly at random and assigning it to y. If A is an algorithm then $A(x_1, \ldots; r)$ denotes the output of A on inputs $x_1, \ldots$ and coins r, while $y \leftarrow A(x_1, \ldots)$ means we pick r uniformly at random and let $y \leftarrow A(x_1, \ldots; r)$. We let $[A(x_1, \ldots)]$ denote the set of y that have positive probability of being output by $A(x_1, \ldots)$. We

write $\text{Func}(a,b)$ for $\{f\colon\{0,1\}^a \to \{0,1\}^b\}$. Polynomial time (PT) is always measured in the length of *all* inputs, not just the first. (But random coins, when singled out as an argument to an algorithm, are never regarded as an input.) As usual, a function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every $c > 0$ there is a K such that $\varepsilon(k) < k^{-c}$ for all $k > K$.

CODE-BASED GAMES. Our definitions and proofs are expressed via code-based games [12] so we recall here the language and specify the particular conventions we use. A code-based game—see Fig. 4 for an example—consists of an INITIALIZE procedure, procedures that respond to adversary oracle queries, and a FINALIZE procedure. All procedures are optional. In an execution of game Gm with an adversary $\mathcal{A}$, the latter is given input 1^k where k is the security parameter, and the security parameter k used in the game is presumed to be the same. Procedure INITIALIZE, if present, executes first, and its output is input to the adversary, who may now invoke other procedures. Each time it makes a query, the corresponding game procedure executes, and what it returns, if anything, is the response to $\mathcal{A}$'s query. The adversary's output is the input to FINALIZE, and the output of the latter, denoted $\text{Gm}^{\mathcal{A}}(k)$, is called the output of the game. FINALIZE may be absent in which case it is understood to be the identity function, so that the output of the game is the output of the adversary. We let "$\text{Gm}^{\mathcal{A}}(k) \Rightarrow c$" denote the event that this game output takes value c and let "$\text{Gm}^{\mathcal{A}}(k)$" be shorthand for "$\text{Gm}^{\mathcal{A}}(k) \Rightarrow \mathsf{true}$." Boolean flags are assumed initialized to false and $\mathsf{BAD}(\text{Gm}^{\mathcal{A}}(k))$ is the event that the execution of game Gm with adversary A sets flag *bad* to true.

CIRCUITS. While our definitions for garbling schemes are representation-independent, the garbling schemes we specify assume a circuit-based representation. Here we specify the conventions and definitions that make this formal.

There are several reasons why it is important to cleanly define circuits (which, for many reasons, are not just DAGs). First, there are many "boundary cases" where only conventions can decide if something is or is not a valid circuit. The boundary cases matter; we have repeatedly found that degenerate or under-analyzed circuit types materially impact if a garbling scheme is correct. Beyond this, a lack of agreement on what a circuit *is* makes even informal discourse problematic. Finally, we have found that it is simply not possible to properly specify a circuit-garbling algorithm or a circuit-evaluation function, nor to carry out code-based game-playing proofs, without circuits being formalized. As an added payoff, if one establishes good conventions for circuits, then these same conventions can be used when defining a garbled circuit and its evaluation function.

A *circuit* is a 6-tuple $f = (n, m, q, A, B, G)$. Here $n \geq 2$ is the number of *inputs*, $m \geq 1$ is the number of *outputs* and $q \geq 1$ is the number of *gates*. We let $r = n+q$ be the number of *wires*. We let $\text{Inputs} = \{1, \ldots, n\}$, $\text{Wires} = \{1, \ldots, n+q\}$, $\text{OutputWires} = \{n+q-m+1, \ldots, n+q\}$, and $\text{Gates} = \{n+1, \ldots, n+q\}$. Then $A\colon \text{Gates} \to \text{Wires}\backslash\text{OutputWires}$ is a function to identify each gate's *first* incoming wire and $B\colon \text{Gates} \to \text{Wires}\backslash\text{OutputWires}$ is a function to identify each gate's *second* incoming wire. Finally $G\colon \text{Gates} \times \{0,1\}^2 \to \{0,1\}$ is a function that determines the *functionality* of each gate. We require $A(g) < B(g) < g$ for all $g \in \text{Gates}$.

The conventions above embody all of the following. Gates have two inputs, arbitrary functionality, and arbitrary fan-out. The wires are numbered 1 to $n+q$. Every non-input wire is the outgoing wire of some gate. The ith bit of input is presented along wire i. The ith bit of output is collected off wire $n+q-m+i$. The outgoing wire of each gate serves as the name of that gate. Output wires may not be input wires and may not be incoming wires to gates. No output wire may be twice used in the output. Requiring $A_g < B_g < g$ ensures that the directed graph corresponding to f is acyclic, and that no wire twice feeds a gate; the numbering of gates comprise a topological sort.

We will routinely ignore the distinction between a circuit $f = (n, m, q, A, B, G)$ as a 6-tuple and the encoding of such a 6-tuple as a string; formally, one assumes a fixed and reasonable encoding, one where $|f|$ is $O(r \log r)$ for $r = n+q$.

We define a canonical evaluation function $\mathsf{ev}_{\mathsf{circ}}$. It takes a string f and a string $x = x_1 x_2 \cdots x_n$:

```
01 proc ev_circ(f, x)
02 (n, m, q, A, B, G) ← f
03 for g ← n + 1 to n + q do
04         a ← A(g), b ← B(g), x_g ← G(x_a, x_b)
05 return x_{n+q−m+1} ... x_{n+q}
```

At line 02 we adopt the convention that any string f can be parsed as a circuit. (If f does not encode a circuit, we view it as some fixed, default circuit.) This ensures that $\mathsf{ev}_{\mathsf{circ}}$ is well-defined for all string inputs f. At line 04, x_a and x_b will always be well defined because of $A(g) < B(g) < g$. Circuit evaluation takes linear time.

We say f^- is a *topological circuit* if $f^- = (n, m, q, A, B)$ for some circuit $f = (n, m, q, A, B, G)$. Thus a topological circuit is like a conventional circuit except the functionality of the gates is unspecified. Let Topo be the function that expunges the final component of its circuit-valued argument, so $f^- = \text{Topo}(f)$ is the topological circuit underlying conventional circuit f.

3. GARBLING SCHEMES

SYNTAX. A *garbling scheme* is a five-tuple of algorithms $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. The first of these is probabilistic; the remaining algorithms are deterministic. A string f, the *original function*, describes the function $\mathsf{ev}(f, \cdot)\colon \{0,1\}^n \to \{0,1\}^m$ that we want to garble. The values $n = f.n$ and $m = f.m$ depend on f and must be easily computable from it. Specifically, fix linear-time algorithms n and m to extract $f.n = \mathsf{n}(f)$ and $f.m = \mathsf{m}(f)$. On input f and a security parameter $k \in \mathbb{N}$, algorithm Gb returns a triple of strings $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$. String e describes an *encoding function*, $\mathsf{En}(e, \cdot)$, that maps an *initial input* $x \in \{0,1\}^n$ to a *garbled input* $X = \mathsf{En}(e, x)$. String F describes a *garbled function*, $\mathsf{Ev}(F, \cdot)$, that maps each garbled input X to a *garbled output* $Y = \mathsf{Ev}(F, X)$. String d describes a *decoding function*, $\mathsf{De}(d, \cdot)$, that maps a garbled output Y to a *final output* $y = \mathsf{De}(d, Y)$.

We levy some simple requirements on garbling schemes. First, $|F|$, $|e|$, and $|d|$ may depend only on k, $f.n$, $f.m$, and $|f|$. Formally, if $f.n = f'.n$, $f.m = f'.m$, $|f| = |f'|$, $(F, e, d) \in [\mathsf{Gb}(1^k, f)]$, and $(F', e', d') \in [\mathsf{Gb}(1^k, f')]$, then $|F| = |F'|$, $|e| = |e'|$, and $|d| = |d'|$. This is the *length* condition. Second, e and d may depend only on k, $f.n$, $f.m$, $|f|$ and the random coins r of Gb. Formally, if $f.n = f'.n$, $f.m = f'.m$, $|f| = |f'|$, $(F, e, d) = \mathsf{Gb}(1^k, f; r)$, and $(F', e', d') = \mathsf{Gb}(1^k, f'; r)$, then $e = e'$ and $d = d'$. This is the *nondegeneracy* condition. Finally, if $f \in \{0,1\}^*$, $k \in \mathbb{N}$, $x \in \{0,1\}^{f.n}$,

and $(F, e, d) \in [\mathsf{Gb}(1^k, f)]$, then $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = \mathsf{ev}(f, x)$. This is the *correctness* condition.

We say that a garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is a *circuit-garbling scheme* if ev interprets f as a circuit: formally, $\mathsf{ev} = \mathsf{ev}_{\mathsf{circ}}$ for the canonical circuit-evaluation function that we defined in Section 2.

PROJECTIVE SCHEMES. A common approach in existing garbling schemes is for e to encode a list of *tokens*, one pair for each bit in $x \in \{0,1\}^n$. Encoding function $\mathsf{En}(e, \cdot)$ then uses the bits of $x = x_1 \cdots x_n$ to select from $e = (X_1^0, X_1^1, \ldots, X_n^0, X_n^1)$ the subvector $X = (X_1^{x_1}, \ldots, X_n^{x_n})$. Formally, we say that garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is *projective* if for all f, $x, x' \in \{0,1\}^{f.n}$, $k \in \mathbb{N}$, and $i \in [1..n]$, when $(F, e, d) \in [\mathsf{Gb}(1^k, f)]$, $X = \mathsf{En}(e, x)$ and $X' = \mathsf{En}(e, x')$, then $X = (X_1, \ldots, X_n)$ and $X' = (X'_1, \ldots, X'_n)$ are n vectors, $|X_i| = |X'_i|$, and $X_i = X'_i$ if x and x' have the same ith bit.

Our definitions of security do not require schemes be projective. However, this property is needed for some important applications. For example, SFE combines a projective garbling scheme and a scheme for oblivious transfer.

SIDE-INFORMATION FUNCTIONS. Privacy is rarely absolute; semantically secure encryption, for example, is allowed to reveal the length of the plaintext. Similarly, a garbled circuit might reveal the size of the circuit that was garbled, its topology (that is, the graph of how gates are connected up), or even the original circuit itself. The information that we expect to be revealed is captured by a *side-information function*, Φ, which deterministically maps f to a string $\phi = \Phi(f)$. We will parameterize our advantage notions by Φ, and in this way simultaneously define garbling schemes that may reveal a circuit's size, topology, identity, or more. We require that $f.n$ and $f.m$ be easily determined from $\phi = \Phi(f)$; formally, there must exist linear-time algorithms n' and m' that compute $f.n = \mathsf{n}'(\phi) = \mathsf{n}(f)$ and $f.m = \mathsf{m}'(\phi) = \mathsf{m}(f)$ when $\phi = \Phi(f)$. We also require that $|f|$ be easily determined from $\Phi(f)$.

Specific side-information functions are useful for circuit garbling. Side-information function Φ_{size} reveals the number of inputs, outputs, and gates of a circuit f; formally, $\Phi_{\text{size}}(f) = (n, m, q)$ for a circuit $f = (n, m, q, A, B, G)$. Side-information function Φ_{topo} reveals the topological circuit but not the functionality of each gate: $\Phi_{\text{topo}}(f) = (n, m, q, A, B)$, with notation and conventions as above. Side-information function Φ_{circ} reveals the entire circuit: $\Phi_{\text{circ}}(f) = f$.

PRIVACY. Let $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ be a garbling scheme, $k \in \mathbb{N}$ a security parameter, and Φ a side-information function. We define an indistinguishability-based notion of privacy via game $\text{PrvInd}_{\mathcal{G},\Phi}$ (top-left of Fig. 4) and a simulation-based notion of privacy via game $\text{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ (top-right of Fig. 4, where $\mathcal{S}$ is a simulator). Executing either game with an adversary requires one to specify the garbling scheme, adversary, security parameter, and side-information function. Executing game PrvSim additionally requires one to specify the algorithm $\mathcal{S}$.

Refer first to game $\text{PrvInd}_{\mathcal{G},\Phi}$. Adversary $\mathcal{A}$ gets input 1^k and must make exactly one GARBLE query. That query is answered as specified in the game, the security parameter used here being the same as the one provided to the adversary. The adversary must eventually halt, outputting a bit b', and the game's FINALIZE procedure determines if the adversary has won on this run, namely, if $b = b'$. The corresponding advantage is defined via

$$\mathbf{Adv}^{\text{prv.ind},\,\Phi}_{\mathcal{G}}(\mathcal{A}, k) = 2\Pr[\text{PrvInd}^{\mathcal{A}}_{\mathcal{G},\Phi}(k)] - 1,$$

the probability, normalized to $[0,1]$, that the adversary correctly predicts b. Protocol $\mathcal{G}$ is prv.ind secure over Φ if for every PT adversary $\mathcal{A}$ the function $\mathbf{Adv}^{\text{prv.ind},\,\Phi}_{\mathcal{G}}(\mathcal{A}, \cdot)$ is negligible.

Explaining the definition, the adversary chooses (f_0, x_0) and (f_1, x_1) such that $\Phi(f_0) = \Phi(f_1)$ and, also, $\mathsf{ev}(f_0, x_0) = \mathsf{ev}(f_1, x_1)$. The game picks challenge bit b and garbles f_b to (F, e, d). It encodes x_b as the garbled input $X = \mathsf{En}_e(x_b)$. The adversary is handed (F, X, d), which determines $y = \mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x_b))) = \mathsf{ev}(f_b, x_b)$. The adversary must guess b. In a scheme we deem secure, it should be unable to ascertain which of (f_0, x_0), (f_1, x_1) got garbled.

Next we define prv.sim security via game $\text{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ associated to garbling scheme $\mathcal{G}$, information function Φ and an algorithm $\mathcal{S}$ called a simulator. The adversary $\mathcal{B}$ is run on input 1^k and must make exactly one GARBLE query. The query is answered as specified in Fig. 4, with k being the same as the input to the adversary. The adversary must eventually output a bit, and the game's FINALIZE procedure indicates if the adversary has won—again, if the adversary correctly predicted b. The adversary's advantage is

$$\mathbf{Adv}^{\text{prv.sim},\,\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B}, k) = 2\Pr[\text{PrvSim}^{\mathcal{B}}_{\mathcal{G},\Phi,\mathcal{S}}(k)] - 1\,,$$

the probability, normalized to $[0,1]$, that the adversary wins. Protocol $\mathcal{G}$ is prv.sim secure over Φ if for every PT adversary $\mathcal{B}$ there is a PT algorithm $\mathcal{S}$ such that $\mathbf{Adv}^{\text{prv.sim},\,\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B}, k)$ is negligible.

Let us again explain. For the prv.sim notion we let the adversary choose (f, x). Either we garble it to $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$ and $X \leftarrow \mathsf{En}(e, x)$, handing the adversary (F, X, d), or else we ask the simulator to devise a "fake" (F, X, d) based solely on k, $\phi = \Phi(f)$, and $y = \mathsf{ev}(f, x)$. From this limited information the simulator must produce an (F, X, d) indistinguishable, to the adversary, from the ones produced using the actual garbling scheme.

The indistinguishability definition for garbling schemes is simpler due to the absence of the simulator, but we consider this notion "wrong" when the side-information function is such that indistinguishability is inequivalent to the simulation-based definition. See Section 4.

OBLIVIOUSNESS. Informally, a garbling scheme achieves *obliviousness* if possession of a garbled function F and garbled input X lets one compute the garbled output Y, yet (F, X) leaks nothing about f or x beyond $\Phi(f)$. Contrasting this with privacy, there the agent evaluating the garbled function *does* learn the output; here, she learns not even that, as a needed piece of information, d, is withheld. Privacy and obliviousness are both secrecy notions, and cut from the same cloth. Yet they will prove incomparable: a private scheme could divulge the output even without d; an oblivious scheme could reveal too much once d is shown.

As with privacy, we formalize two notions, obv.ind and obv.sim, via the games of Fig. 4. The formalizations consider games $\text{ObvInd}_{\mathcal{G},\Phi}$ and $\text{ObvSim}_{\mathcal{G},\Phi,\mathcal{S}}$, run with adversaries $\mathcal{A}$ and $\mathcal{B}$, respectively. As usual the adversary gets input 1^k and the security parameter used in the game is also k. The adversary makes a single call to the game's GARBLE procedure and outputs a bit b'. We define

$$\begin{aligned}
\mathbf{Adv}^{\text{obv.ind},\,\Phi}_{\mathcal{G}}(\mathcal{A}, k) &= 2\Pr[\text{ObvInd}^{\mathcal{A}}_{\mathcal{G},\Phi}(k))] - 1 \quad \text{and} \\
\mathbf{Adv}^{\text{obv.sim},\,\Phi,\,\mathcal{S}}_{\mathcal{G}}(\mathcal{B}, k) &= 2\Pr[\text{ObvSim}^{\mathcal{B}}_{\mathcal{G},\Phi,\mathcal{S}}(k)] - 1
\end{aligned}$$

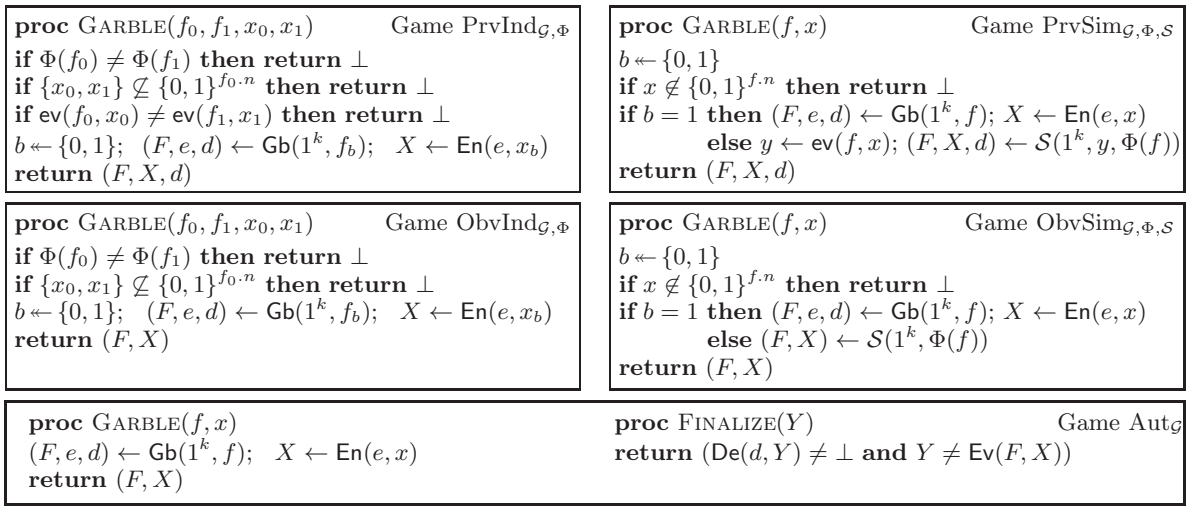

Figure 4: Games for defining the prv.ind, prv.sim, obv.ind, obv.sim, and aut security of a garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. Here $\mathcal{S}$ is a simulator, Φ is an information function and k is the security parameter input to the adversary. Procedure FINALIZE(b') of the first four games returns $(b = b')$.

as the probability, normalized to $[0, 1]$, that adversary's output is a correct guess of the underlying bit b. Protocol $\mathcal{G}$ is obv.ind secure over Φ if for every PT adversary $\mathcal{A}$, we have that $\mathbf{Adv}^{\text{obv.ind},\Phi}_{\mathcal{G}}(\mathcal{A}, k)$ is negligible. It is obv.sim secure over Φ if for every PT adversary $\mathcal{B}$ there exists a PT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\text{obv.sim},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B}, \cdot)$ is negligible.

Let us explain the difference between prv.ind and obv.ind. First, we no longer demand that $\mathsf{ev}(f, x_0) = \mathsf{ev}(f, x_1)$: the adversary may now name any (f_0, x_0) and (f_1, x_1) as long as the functions have the same side information. Second, the decoding function d is no longer provided to the adversary. The adversary must guess if (F, X) stems from garbling (f_0, x_0) or (f_1, x_1).

Similarly, the difference between prv.sim and obv.sim is two-fold. First, in the obliviousness notion the simulator is denied $y = \mathsf{ev}(f, x)$; it must create a convincing (F, X) without that. Second, the simulator no longer returns to the adversary the (simulated) decoding function d; the return value is (F, X) and not (F, X, d).

AUTHENTICITY. So far we have dealt exclusively with secrecy notions. One can formalize an authenticity property as well [17], which we do via game Aut$_{\mathcal{G}}$ of Fig. 4. Authenticity captures an adversary's inability to create from a garbled function F and its garbled input X a garbled output $Y \neq F(X)$ that will be deemed authentic.

Fix a garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$, adversary $\mathcal{A}$, and security parameter $k \in \mathbb{N}$. Run adversary $\mathcal{A}$ on input 1^k, allowing it a single call to the GARBLE procedure of the game. The adversary outputs a string Y, and, when it does, the game's FINALIZE procedure is called to decide if the adversary has won. The adversary's aut-advantage is defined as $\mathbf{Adv}^{\text{aut}}_{\mathcal{G}}(\mathcal{A}, k) = \Pr[\text{Aut}^{\mathcal{A}}_{\mathcal{G}}(k)]$. Protocol $\mathcal{G}$ is aut-secure if for all PT adversaries $\mathcal{A}$, $\mathbf{Adv}^{\text{aut}}_{\mathcal{G}}(\mathcal{A}, \cdot)$ is negligible.

SETS OF GARBLING SCHEMES. To compactly and precisely express relations between notions we will write them as containments and non-containments between sets of garbling schemes. To this end, for xxx $\in$ {prv.ind, prv.sim, obv.ind, obv.sim} we let $\mathsf{GS}(\text{xxx}, \Phi)$ be the set of all garbling schemes that are xxx-secure over Φ. Similarly, we let $\mathsf{GS}(\text{aut})$ be the set of all garbling schemes that are aut-secure.

We also let $\mathsf{GS}(\mathsf{ev})$ be the set of all garbling schemes $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ whose evaluation function is ev. This captures garbling schemes for a particular class of functions. As per our previous notation, $\mathsf{GS}(\mathsf{ev}_{\mathsf{circ}})$ now denotes the set of all circuit-garbling schemes.

4. RELATIONS

We show that prv.sim always implies prv.ind, and prv.ind implies prv.sim under certain added conditions on the side-information function. We show that the same holds for obv.ind and obv.sim, under a weaker assumption on the side-information function. The conditions on the side-information function are relatively mild. We will also justify the non-implications for the security notions compactly summarized in Fig. 2. As part of this we will show that prv.ind does not always imply prv.sim and obv.ind does not always imply obv.sim.

INVERTIBILITY OF SIDE-INFORMATION FUNCTIONS. Let Φ be a side-information function. An algorithm M is called a Φ-inverter if on input ϕ in the range of Φ it returns a preimage under Φ of that point, meaning a string f such that $\Phi(f) = \phi$. Such an inverter always exists, but it might not be efficient. We say that Φ is *efficiently invertible* if there is a polynomial-time Φ-inverter. Similarly, an algorithm M is called a (Φ, ev)-inverter if on input (ϕ, y), where $\phi = \Phi(f')$ and $y = \mathsf{ev}(f', x')$ for some f' and $x \in \{0,1\}^{f'.n}$, returns an (f, x) satisfying $\Phi(f) = \phi$ and $\mathsf{ev}(f, x) = y$. We say that (Φ, ev) is *efficiently invertible* if there is a polynomial-time (Φ, ev)-inverter.

The following theorem summarizes the invertibility attributes of the circuit-related size-information functions we defined earlier. It shows that all side-information functions $\Phi_{\text{circ}}, \Phi_{\text{topo}}$, and Φ_{size} are efficiently invertible, and that, $(\Phi_{\text{size}}, \mathsf{ev}_{\mathsf{circ}})$ and $(\Phi_{\text{topo}}, \mathsf{ev}_{\mathsf{circ}})$ are efficiently invertible.

PROPOSITION 1. *For* $\Phi \in \{\Phi_{\text{size}}, \Phi_{\text{topo}}, \Phi_{\text{circ}}\}$*, there is a linear-time inverter. For* $\Phi \in \{\Phi_{\text{size}}, \Phi_{\text{topo}}\}$ *there is a linear-time* $(\Phi, \mathsf{ev}_{\text{circ}})$*-inverter.*

In contrast, there is no efficient $(\Phi_{\text{circ}}, \mathsf{ev}_{\text{circ}})$-inverter (under a computational assumption); consider the case where f is drawn from a family implementing a one-way function.

EQUIVALENCE OF PRV.IND AND PRV.SIM. The following says that prv.sim implies prv.ind security, and conversely if (Φ, ev) is efficiently invertible. The proof is in the full paper [11].

PROPOSITION 2. *For any PT* Φ*:* (1) $\mathsf{GS}(\text{prv.sim}, \Phi) \subseteq \mathsf{GS}(\text{prv.ind}, \Phi)$ *and* (2) *If* (Φ, ev) *is efficiently invertible then* $\mathsf{GS}(\text{prv.ind}, \Phi) \cap \mathsf{GS}(\mathsf{ev}) \subseteq \mathsf{GS}(\text{prv.sim}, \Phi) \cap \mathsf{GS}(\mathsf{ev})$.

The first part says that if garbling scheme $\mathcal{G}$ is prv.sim secure over Φ then $\mathcal{G}$ is prv.ind secure over Φ. The second part says that if garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is prv.ind secure over Φ and (Φ, ev) is efficiently invertible then $\mathcal{G}$ is prv.sim secure over Φ. In the full version [11], we show that efficient invertibility of (Φ, ev) is required to prove that prv.ind implies prv.sim, so the notions are not always equivalent.

A corollary of Propositions 1 and 2 is that prv.sim and prv.ind are equivalent for circuit-garbling schemes over Φ_{topo} and Φ_{size}, which we summarize as:

COROLLARY 1. *For* $\Phi \in \{\Phi_{\text{topo}}, \Phi_{\text{size}}\}$*,* $\mathsf{GS}(\text{prv.ind}, \Phi) \cap \mathsf{GS}(\mathsf{ev}_{\text{circ}}) = \mathsf{GS}(\text{prv.sim}, \Phi) \cap \mathsf{GS}(\mathsf{ev}_{\text{circ}})$.

EQUIVALENCE OF OBV.IND AND OBV.SIM. The following says that obv.sim implies obv.ind security, and conversely if Φ is efficiently invertible. The invertibility condition is thus weaker than in the privacy case.

PROPOSITION 3. *For any PT* Φ*:* (1) $\mathsf{GS}(\text{obv.sim}, \Phi) \subseteq \mathsf{GS}(\text{obv.ind}, \Phi)$ *and* (2) *If* Φ*) is efficiently invertible then* $\mathsf{GS}(\text{obv.ind}, \Phi) \subseteq \mathsf{GS}(\text{obv.sim}, \Phi)$.

In the full version of this paper [11], we show that Φ being efficiently invertible is required to prove that obv.ind implies obv.sim. But the side-information function Φ we use is artificial; for any "reasonable" one we know, obv.ind and obv.sim will be equivalent.

Again a corollary of Propositions 1 and 3 is that obv.sim and obv.ind are equivalent for circuit-garbling schemes over side-information functions Φ_{circ}, Φ_{topo} and Φ_{size}:

COROLLARY 2. $\mathsf{GS}(\text{obv.ind}, \Phi) = \mathsf{GS}(\text{obv.sim}, \Phi)$*, for any* $\Phi \in \{\Phi_{\text{topo}}, \Phi_{\text{size}}, \Phi_{\text{circ}}\}$.

5. ACHIEVING PRIVACY

We provide a simple, privacy-achieving circuit-garbling scheme, Garble1. It is described in terms of a new primitive, a *dual-key cipher* (DKC). We will prove security of Garble1 assuming the security of its DKC. We will then show how to instantiate a DKC using a PRF. Instantiating this PRF via AES leads to an efficient garbling scheme. Differently instantiating the DKC directly with AES can give even better efficiency.

DUAL KEY CIPHERS. Before describing Garble1 we will need to specify the syntax of a DKC. These objects formalize a two-key lockbox—one where you need *both* keys to open the box. This has long been used as a metaphor to explain how garbling schemes work (e.g., [35] (pp. 163–164)), but Lindell and Pinkas also give a notion of *double-encryption security* for two-key probabilistic encryption schemes [35] (pp. 170). Dual-key ciphers provide a very different way to formalize an object sufficient to construct garbling schemes.

Formally, a *dual-key cipher* is a function $\mathbb{E}$ that associates to any $k \in \mathbb{N}$, any keys $\mathtt{A}, \mathtt{B} \in \{0,1\}^k$ and any tweak $\mathtt{T} \in \{0,1\}^{\tau(k)}$ a permutation $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}} \colon \{0,1\}^k \to \{0,1\}^k$. Let $\mathbb{D}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}} \colon \{0,1\}^k \to \{0,1\}^k$ denote the inverse of this permutation. It is required that the maps $(\mathtt{A}, \mathtt{B}, \mathtt{T}, \mathtt{P}) \mapsto \mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(\mathtt{P})$ and $(\mathtt{A}, \mathtt{B}, \mathtt{T}, \mathtt{C}) \mapsto \mathbb{D}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(\mathtt{C})$ be polynomial-time computable. We refer to τ as the tweak length of $\mathbb{E}$.

The definition above describes syntax alone. We postpone giving a security definition until we've defined Garble1.

DEFINITION OF GARBLE1. Let $\mathbb{E}$ be a dual-key cipher with tweak length τ. We associate to $\mathbb{E}$ the garbling scheme Garble1[$\mathbb{E}$] as shown in Fig. 5. Wires carry k-bit *tokens*. A token X will encode a one-bit *type*. Rather arbitrarily, the type is the final bit of the token, namely its LSB. When we write $\mathtt{T} \leftarrow g \,\|\, \mathtt{a} \,\|\, \mathtt{b}$ (line 106 and 155) where $g \in \mathbb{N}$ and $\mathtt{a}, \mathtt{b} \in \{0,1\}$, we mean that $g \bmod 2^{\tau(k)-2}$ is encoded as a $(\tau(k)-2)$-bit string and $\mathtt{a} \,\|\, \mathtt{b}$ is concatenated, yielding a $\tau(k)$-bit tweak. The ev function (lines 140–145) is precisely $\mathsf{ev}_{\text{circ}}$; the code is repeated for completeness and to make visible the commonality with Ev (lines 150–156).

To garble a circuit, we begin selecting two tokens for each wire, one of each type. One of these will represent 0—the token is said to have *semantics* of 0—while the other will represent 1. The variable X_i^b names the token of wire i with semantics (not type!) of b. Thus the encoding function e (see lines 120–123) will map $x = x_1 \cdots x_n \in \{0,1\}^n$ to $X = (X_1^{x_1}, \ldots, X_n^{x_n})$. For each wire i that is not an output wire, we select, at line 102, random tokens of opposite type, making the association between a token's type and its semantics random. For each wire i that is an output wire, we again select random tokens of opposite types, but this time the token's type *is* the token's semantics.

Lines 104–106 compute q garbled truth tables, one for each gate g. Table $P[g, \cdot, \cdot]$ has four rows, entry $\mathtt{a}, \mathtt{b}$ the row to use when the left incoming token is of type $\mathtt{a}$ and the right incoming token is of type $\mathtt{b}$. The token that gets encrypted for this row (line 106) is the token for the outgoing-wire with the correct semantics. At lines 154–155, given two tokens X_a and X_b we use their types to determine which entry of the propagation table we need to decrypt. The description of the decoding function d (line 109) is empty because no information is needed to map an output token to its semantics, the type being the semantics.

SECURITY NOTION FOR DUAL-KEY CIPHERS. We already defined the syntax of a DKC, a permutation $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}} \colon \{0,1\}^k \to \{0,1\}^k$ for each $\mathtt{A}, \mathtt{B}, \mathtt{T}$. Our definition of security will allow the adversary to select whichever of the two keys it wants to learn. We will hand it not only that key but, also, the last of the undisclosed key. (This corresponds to the type bit in runs of Garble1). We consider only nonadaptive, known-plaintext attacks. These plaintexts will be either the disclosed keys or truly random strings. We prohibit encryption cycles. During the adversary's attack, the tweaks used must be nonces—values used at most once.

More formally, the security of a DKC $\mathbb{E} \colon \{0,1\}^k \times \{0,1\}^k \times$

100 **proc** $\mathsf{Gb}(1^k, f)$
101 $(n, m, q, A, B, G) \leftarrow f$
102 **for** $i \in \{1, \ldots, n+q-m\}$ **do** $t \twoheadleftarrow \{0,1\}$, $X_i^0 \twoheadleftarrow \{0,1\}^{k-1}t$ $X_i^1 \twoheadleftarrow \{0,1\}^{k-1}\overline{t}$
103 **for** $i \in \{n+q-m+1, \ldots, n+q\}$ **do** $X_i^0 \twoheadleftarrow \{0,1\}^{k-1}0$, $X_i^1 \twoheadleftarrow \{0,1\}^{k-1}1$
104 **for** $(g, i, j) \in \{n+1, \ldots, n+q\} \times \{0,1\} \times \{0,1\}$ **do**
105 $a \leftarrow A(g)$, $b \leftarrow B(g)$
106 $\mathtt{A} \leftarrow X_a^i$, $\mathtt{a} \leftarrow \mathrm{lsb}(\mathtt{A})$, $\mathtt{B} \leftarrow X_b^j$, $\mathtt{b} \leftarrow \mathrm{lsb}(\mathtt{B})$, $\mathtt{T} \leftarrow g \,\|\, \mathtt{a} \,\|\, \mathtt{b}$, $P[g, \mathtt{a}, \mathtt{b}] \leftarrow \mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}\big(X_g^{G_g(i,j)}\big)$
107 $F \leftarrow (n, m, q, A, B, P)$
108 $e \leftarrow (X_1^0, X_1^1, \ldots, X_n^0, X_n^1)$
109 $d \leftarrow \varepsilon$
110 **return** (F, e, d)

120 **proc** $\mathsf{En}(e, x)$
121 $(X_1^0, X_1^1, \ldots, X_n^0, X_n^1) \leftarrow e$
122 $X \leftarrow (X_1^{x_1}, \ldots, X_n^{x_n})$
123 **return** X

130 **proc** $\mathsf{De}(d, Y)$
131 $(Y_1, \ldots, Y_m) \leftarrow Y$
132 **for** $i \in \{1, \ldots, m\}$ **do** $y_i \leftarrow \mathrm{lsb}(Y_i)$
133 **return** $y \leftarrow y_1 \cdots y_m$

140 **proc** $\mathsf{ev}(f, x)$
141 $(n, m, q, A, B, G) \leftarrow f$
142 **for** $g \leftarrow n+1$ **to** $n+q$ **do**
143 $a \leftarrow A(g)$, $b \leftarrow B(g)$
144 $x \leftarrow G_g(x_a, x_b)$
145 **return** $x_{n+q-m+1} \ldots x_{n+q}$

150 **proc** $\mathsf{Ev}(F, X)$
151 $(n, m, q, A, B, P) \leftarrow F$
152 **for** $g \leftarrow n+1$ **to** $n+q$ **do**
153 $a \leftarrow A(g)$, $b \leftarrow B(g)$
154 $\mathtt{A} \leftarrow X_a$, $\mathtt{a} \leftarrow \mathrm{lsb}(\mathtt{A})$, $\mathtt{B} \leftarrow X_b$, $\mathtt{b} \leftarrow \mathrm{lsb}(\mathtt{B})$
155 $\mathtt{T} \leftarrow g \,\|\, \mathtt{a} \,\|\, \mathtt{b}$, $X_g \leftarrow \mathbb{D}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}\big(P[g, \mathtt{a}, \mathtt{b}]\big)$
156 **return** $(X_{n+q-m+1}, \ldots, X_{n+q})$

Figure 5: Garbling scheme Garble1. Its components are $(\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ where ev, shown for completeness, is the canonical circuit evaluation. We assume a DKC $\mathbb{E}$ with tweak length τ and let $\mathbb{D}$ denote its inverse. At line 102, we use $\{0,1\}^{k-1}t$ and $\{0,1\}^{k-1}\overline{t}$ to refer to the sets of k-bit binary strings whose last bit is t and $\overline{t}$ respectively.

$\{0,1\}^{\tau(k)} \times \{0,1\}^k \to \{0,1\}^k$ is specified using the game of Fig. 6. The game starts by choosing a bit $b \twoheadleftarrow \{0,1\}$ and a key $K \twoheadleftarrow \{0,1\}^k$. It chooses infinitely many random strings $K_1, K_2, \ldots$ such that the last bit of K_i is i mod 2. It chooses infinitely many random strings $R_1, R_2, \ldots$. Except for the last bit of K, the key K shall be kept secret. The strings $K_1, K_2, \ldots$ are initially secret, but the adversary $\mathcal{A}$ will eventually learn them through its queries. The random strings $R_1, R_2, \ldots$, used only in the "reference game" when $b = 0$, are secret. We require that the adversary $\mathcal{A}$ be nonadaptive, that is, it prepares all queries before interrogating the DKC oracle. In each query, adversary $\mathcal{A}$ has to specify an integer i indicating that it wants to use $\{K, K_i\}$ as keys of the dual-key cipher for this query, and an integer j, indicating that it wants to encrypt the string K_j. We require that $i < j$ to avoid encryption cycles. It also specifies a boolean *pos* to indicate the position, left or right, of the secret key K. Finally, it provides a tweak $\mathtt{T}$, which must be a nonce. If $b = 1$ then the oracle returns the encryption of K_j to the adversary. If $b = 0$ then the oracle returns the encryption of R_j. When adversary $\mathcal{A}$ outputs a bit b' its advantage is $\mathbf{Adv}^{\mathrm{dkc}}_{\mathbb{E}}(\mathcal{A}, k) = 2\Pr[\mathrm{DKC}^{\mathcal{A}}(k)] - 1$. We say that $\mathbb{E}$ is a secure dual-key cipher if $\varepsilon(k) = \mathbf{Adv}^{\mathrm{dkc}}_{\mathbb{E}}(\mathcal{A}, k)$ is negligible for every nonadaptive PPT adversary $\mathcal{A}$ whose input is 1^k and the bit returned by Initialize.

Discussion. By way of further explanation, ciphertexts $\mathbb{E}^{\mathtt{T}_1}_{K,K_1}(X_1), \mathbb{E}^{\mathtt{T}_2}_{K_2,K}(X_2), \ldots$ should be indistinguishable from random strings as long as K is secret and the tweaks $T_1, T_2, \ldots$ are nonces—even if random values K_i and X_j are all disclosed. We demand that this hold even if the last bit of K is released to the adversary and the adversary can actively choose the last bit of each K_i.

proc Initialize() Game DKC
$b \twoheadleftarrow \{0,1\}$, $K \twoheadleftarrow \{0,1\}^k$, $R_1, R_2, \ldots \twoheadleftarrow \{0,1\}^k$
for $i \in \{1, 2 \ldots\}$ **do**
$K_{2i} \twoheadleftarrow \{0,1\}^{k-1}\,0$
$K_{2i-1} \twoheadleftarrow \{0,1\}^{k-1}\,1$
return $\mathrm{lsb}(K)$

proc Encrypt$(i, j, pos, \mathtt{T})$
if $used[\mathtt{T}]$ **or** $i \geq j$ **then return** $\perp$
$used[\mathtt{T}] \leftarrow \mathsf{true}$
if $pos = 1$ **then** $(\mathtt{A}, \mathtt{B}) \leftarrow (K, K_i)$ **else** $(\mathtt{A}, \mathtt{B}) \leftarrow (K_i, K)$
if $b = 1$ **then** $X \leftarrow K_j$ **else** $X \leftarrow R_j$
return $(K_i, K_j, \mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(X))$

Figure 6: Security of a DKC. Cipher $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}$ has a tweak and two keys, only one of which, K, its position chosen by the adversary, is secret. The final bit of K is disclosed. Procedure Finalize(b') returns $(b = b')$.

A subtle issue arises when the adversary happens to possess, say $\mathbb{E}^{\mathtt{T}_1}_{K_1,K}(X)$ and $\mathbb{E}^{\mathtt{T}_2}_{K_2,K}(X)$. One may be tempted to require that the two ciphertexts be indistinguishable from two independent random strings. This, however, would not allow instantiations like $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(X) = E_{\mathtt{A}}(E_{\mathtt{B}}(X))$ for an ideal cipher E. Instead, we choose a secret $Y \twoheadleftarrow \mathcal{M}$, where $\mathcal{M}$ is the message space, and demand that $\mathbb{E}^{\mathtt{T}_1}_{K_1,K}(X)$ and $\mathbb{E}^{\mathtt{T}_2}_{K_2,K}(X)$ be indistinguishable from $\mathbb{E}^{\mathtt{T}_1}_{K_1,K}(Y)$ and $\mathbb{E}^{\mathtt{T}_2}_{K_2,K}(Y)$.

The definitional intricacies for dual-key ciphers arise from wanting to require of a DKC little more than what is actually

needed to prove Garble1. Too strong a definition for DKC security and interesting instantiations will be lost.

SECURITY OF GARBLE1. Our definition of DKC security suffices to prove security for Garble1. The result is stated below and proven in the full version of this paper [11].

THEOREM 1. *Let $\mathbb{E}$ be a secure dual-key cipher. Then $\mathcal{G} = \mathrm{Garble1}[\mathbb{E}] \in \mathsf{GS}(\mathrm{prv.ind}, \Phi_{\mathrm{topo}})$.* □

The theorem is underlain by an explicit, blackbox, uniform reduction U s.t. if $\mathcal{A}(1^k)$ outputs circuits of at most r wires and fan-out at most ν, then $\mathcal{D} = U^{\mathcal{A}}$ achieves advantage $\mathbf{Adv}^{\mathrm{dkc}}_{\mathbb{E}}(\mathcal{D}, k) \geq \frac{1}{2r}\mathbf{Adv}^{\mathrm{prv.ind},\,\Phi_{\mathrm{topo}}}_{\mathcal{G}}(\mathcal{A}, k)$ and makes $Q \leq 2\nu$ oracle queries, with $\mathbf{E}[Q] < 4$. It runs in time about that of $\mathcal{A}$ plus the time for $4r$ computations of $\mathbb{E}$ on k-bit keys. The small overhead implicit in the word "about" is manifest in the proof. The above assumes that $r \leq 2^{\tau(k)-2}$. In asymptotic statements, r and ν are understood as polynomials $r(k)$ and $\nu(k)$.

We comment that Garble1 does not satisfy obliviousness or authenticity. To defeat obliviousness, an adversary can just make the query (AND, OR, 00, 11) to receive (F, X), and then evaluate $Y = \mathsf{Ev}(F, X)$, returning 1 if $\mathsf{De}(\varepsilon, Y) = 1$ and 0 otherwise. This adversary has advantage 1. To defeat authenticity, an adversary can query (OR, 11), and then output $(0^k, 0^k)$. Again it has advantage 1. We will soon describe Garble2 that satisfies obliviousness and authenticity in addition to privacy.

The primitive used by Lindell and Pinkas [35] as a basis for encryption of gate entries is a randomized, IND-CPA secure symmetric encryption scheme with an *elusive* and *efficiently verifiable* range. Dual-key ciphers, in contrast, are deterministic. Our PRF-based instantiation avoids probabilistic encryption. Besides speed it results in shorter ciphertexts for each row of each gate. The additional properties of encryption assumed by [35] are to allow the evaluator to know which gate entry is the "correct" one. Our solution via type bits (the "point-and-permute" technique, which dates to Rogaway [45]) is well known.

DUAL-KEY CIPHERS FROM A PRF. Our primary interest will be in instantiating a dual-key cipher via a PRF. Let F associate to key $\mathtt{K} \in \{0,1\}^{k-1}$ a map $\mathsf{F}_{\mathtt{K}}\colon \{0,1\}^{\tau(k)} \to \{0,1\}^k$. We require that the map $\mathtt{K}, \mathtt{T} \mapsto \mathsf{F}_{\mathtt{K}}(\mathtt{T})$ be polynomial-time computable. We refer to τ as the input length.

The prf-advantage of an adversary $\mathcal{D}$ against F is defined as $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(\mathcal{D}, k) = 2\Pr[\mathrm{PRF}^{\mathcal{D}}_{\mathsf{F}}(k)] - 1$ where game $\mathrm{PRF}_{\mathsf{F}}$ is as follows. INITIALIZE picks a random bit b and a random $(k-1)$-bit key $\mathtt{K}$. The adversary has access to procedure FN that maintains a table $\mathtt{Tbl}[\cdot]$ initially everywhere undefined. Given $\mathtt{T} \in \{0,1\}^{\tau(k)}$, the procedure returns $\mathsf{F}(\mathtt{K}, \mathtt{T})$ if $b = 1$. Otherwise, it picks and returns $\mathtt{Tbl}[\mathtt{T}] \twoheadleftarrow \{0,1\}^k$ if $\mathtt{Tbl}[\mathtt{T}] = \perp$, or returns $\mathtt{Tbl}[\mathtt{T}]$ if $\mathtt{Tbl}[\mathtt{T}] \neq \perp$. FINALIZE$(b')$ returns $(b = b')$. We say that F is PRF-secure if $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(\mathcal{D}, \cdot)$ is negligible for all polynomial-time adversaries $\mathcal{D}$.

Given a PRF F as above, we define the dual-key cipher $\mathbb{E}$ via $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(P) = \mathsf{F}_{\mathtt{A}[1:k-1]}(\mathtt{T}) \oplus \mathsf{F}_{\mathtt{B}[1:k-1]}(\mathtt{T}) \oplus P$. This dual-key cipher has tweak length τ and is denoted $\mathbb{E}[\mathsf{F}]$. During evaluation, token types are revealed, but the entire key of F remains secret.

The following result establishes that $\mathbb{E}[\mathsf{F}]$ is a good DKC when F is a good PRF. The reduction is tight and explicit. More specifically, the proof provides a blackbox reduction such that for any adversary $\mathcal{A}(1^k)$ attacking $\mathbb{E}[\mathsf{F}]$ there is an adversary $\mathcal{B}(1^k)$ attacking F for which $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(\mathcal{B}, k) = 0.5\,\mathbf{Adv}^{\mathrm{dkc}}_{\mathbb{E}[\mathsf{F}]}(\mathcal{A}, k)$. If $\mathcal{A}$ makes Q queries to ENCRYPT then $\mathcal{B}$ also makes Q queries to the PRF oracle FN. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$, where the meaning of "about" is manifest in the proof that appears in the full version of this paper [11].

THEOREM 2. *Let F be a PRF. Then $\mathbb{E}[\mathsf{F}]$ is a secure dual-key cipher.*

The instantiation of a DKC $\mathbb{E}$ by way of $\mathbb{E}[\mathsf{F}]$ is by no means the only reasonable instantiation, nor the only one that can be proven secure. We now investigate further instantiations, going all the way to a blockcipher.

DUAL-KEY CIPHERS FROM DOUBLE ENCRYPTION. We also prove the dkc-security of the instantiation $\mathbb{E}[E]$ in which $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(X) = E_{\mathtt{A}}(E_{\mathtt{B}}(X))$, with E being an ideal cipher. In the theorem below, we will show that if an adversary $\mathcal{A}$ makes Q queries to the ENCRYPT oracle, and q_E queries to E and E^{-1} then $\mathbf{Adv}^{\mathrm{dkc}}_{\mathbb{E}[E]}(\mathcal{A}, k) \leq (10Q^2 + 4Q + 8q_E)/2^k$. The above assumes that $Q + q_E \leq 2^{k-3}$.

THEOREM 3. *Let E be an ideal cipher. Then $\mathbb{E}[E]$ is a secure dual-key cipher.*

The proof is found in the full version of this paper [11].

UNWINDING THE RESULTS. One needs to be careful in combining Theorems 1 and 3 to obtain a good bound on the security of Garble1 when instantiated with a DKC made by double encryption. Let adversary $\mathcal{A}$ attack $\mathsf{Gb1}[\mathbb{E}[E]]$ and assume $\mathcal{A}(1^k)$ outputs circuits of at most $r \leq 2^{\tau(k)-2}$ wires and fan-out at most ν. Suppose it makes at most q_E queries to E and E^{-1} and that $2\nu + q_E \leq 2^{k-3}$. Then, from Theorems 1 and 3, there is a random variable $0 < Q \leq 2\nu$ such that $\mathbf{E}[Q] < 4$ and

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{prv.ind},\,\Phi_{\mathrm{topo}}}_{\mathrm{Garble1}[\mathbb{E}[E]]}(\mathcal{A}, k) \\
&\quad\leq \frac{r}{2^k} \cdot (20\mathbf{E}[Q^2] + 8\mathbf{E}[Q] + 16q_E) \\
&\quad\leq \frac{r}{2^k} \cdot (20\mathbf{E}[2\nu Q] + 8\mathbf{E}[Q] + 16q_E) \\
&\quad< 160r\nu/2^k + 32r/2^k + 16rq_E/2^k \ .
\end{aligned}$$

The bound is quite satisfactory. Above, the expectation $\mathbf{E}[Q]$ appears in the first inequality because our advantage notion satisfies the following linearity condition: if an adversary $\mathcal{A}$ behaves as adversary $\mathcal{A}_1$ with probability p, and behaves like $\mathcal{A}_2$ otherwise, then $\mathbf{Adv}^{\mathrm{prv.ind},\,\Phi_{\mathrm{topo}}}_{\mathcal{G}}(\mathcal{A}, k) = p\,\mathbf{Adv}^{\mathrm{prv.ind},\,\Phi_{\mathrm{topo}}}_{\mathcal{G}}(\mathcal{A}_1, k) + (1-p)\mathbf{Adv}^{\mathrm{prv.ind},\,\Phi_{\mathrm{topo}}}_{\mathcal{G}}(\mathcal{A}_2, k)$.

AES-BASED INSTANTIATIONS. We now consider concrete instantiations. This means we fix a value k of the security parameter and suggest ways to realize $\mathbb{E}$ on k-bit keys based on blockciphers, specifically AES. Security for these instantiations can be derived via the concrete security bounds that we stated above following Theorem 1. Different choices of instantiation lead to different tradeoffs between assumptions and efficiency. We begin with ways to instantiate F on $(k-1)$-bit keys:

- Let $\mathsf{F}_{\mathtt{K}}(\mathtt{T})$ be the first k bits of $E_{\mathtt{K}}(\mathtt{T} \,\|\, 0) \,\|\, E_{\mathtt{K}}(\mathtt{T} \,\|\, 1)$ for a blockcipher E having block length and key length of $(k-1)$; to be concrete, $E = \text{AES128}$, $k = 129$, $|\mathtt{K}| = 128$, and $\tau = |\mathtt{T}| = 127$. This construction is a good PRF under the standard assumption that E is a good PRP. With this instantiation, evaluating a garbled gate costs four AES operations.
- Let $\mathsf{F}_{\mathtt{K}}(\mathtt{T})$ be $E_{\mathtt{K}\|0}(\mathtt{T})$ for a blockcipher having a k-bit key and block size, say $E = \text{AES128}$ and $k = \tau = |\mathtt{T}| = 128$ and $|\mathtt{K}| = 127$. Assuming that E is a good PRP is not enough to prove that F is a good PRF, as zeroing out a bit of the key does not, in general, preserve PRF security [42]. Still, it seems reasonable to directly assume this F is a good PRF. Costs are halved compared to the above; now, evaluating a garbled gate requires two AES operations.

Next we suggest some further ways to make the dual-key cipher $\mathbb{E}$ directly, meaning not via a PRF. The first follows the double-encryption realization of garbled gates attributed to Yao by Goldreich [18] (which would have been understood that primitive to be probabilistic, not a blockcipher). The second method is extremely efficient—the most efficient approach now known. Implementation work is currently underway to measure the magnitude of the gain:

- Let $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(\mathtt{P}) = E_{\mathtt{A}}(E_{\mathtt{B}}(\mathtt{P}))$ (the tweak is ignored), where $E : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ is a blockcipher, say AES128. For a proof we would model E as an ideal cipher. Composition of encryption schemes is understood by many researchers to be Yao's original approach, although the earliest expositions make this seem doubtful.
- Let $\mathbb{E}^{\mathtt{T}}_{\mathtt{A},\mathtt{B}}(\mathtt{P}) = E_{\mathsf{const}}(K) \oplus K \oplus \mathtt{P}$ where $K = \mathtt{A} \oplus \mathtt{B} \oplus \mathtt{T}$ and $E = \text{AES128}$, say, and const is a fixed 128-bit string. Here $k = \tau = 128$. With this instantiation evaluating a gate costs only 1 AES operation. Even more important, all AES operations employ a single, fixed key. This allows one to take full advantage of AES-NI hardware support to get extremely high speeds. For a proof, we would model $E_{\mathsf{const}}(\cdot)$ as a random permutation π, giving the adversary access to oracles for π and its inverse.

Other one-call, fixed-key schemes are possible, for obliviousness, authenticity, and dynamic security, and adjustments to allow the free-xor and row-reduction optimizations [31, 44].

Basing garbled-circuit evaluation on AES and employing AES-NI in an implementation was also suggested by Kreuter, Shelat, and Shen [32]. They use AES-256, rekeying with gate evaluation.

6. AUTHENTICITY & OBLIVIOUSNESS

We now describe a scheme Garble2 that satisfies not only privacy but also obliviousness and authenticity. The scheme is like Garble1 except, first, the last bit of a token is always uniform, even for output wires. This will give obliviousness. Next, the string encoding the decoding function is made to list all the tokens for all the output wires, ordered to make clear which tokens have what semantics. This engenders authenticity. See Fig. 7.

Talking through some of the pseudocode, line 202 now assigns a token with random semantics to each and every wire. Lines 203–207 compute the garbled function F and encoding function e exactly as with Garble1. Line 208 now records the vector of tokens for each of the m output wires. (Recall that, under our conventions, the last m of the r total wires are the output wires, these providing the m output bits, in order.) At lines 230–235 decoding procedure De, when presented a $2m$-vector d and an m-vector Y, verifies that each component of the latter is in the corresponding set of two allowed values. If so, we determine the correct semantics for this output bit using our convention that Y_i^b has semantics b.

Scheme Garble2 simultaneously achieves privacy, obliviousness, and authenticity if instantiated in the same manner as we instantiated Garble1. This is captured by the following result; the proof appears in the full version of the paper [11]. Again, as per Corollary 2 it does not matter whether we consider ind or sim, and for simplicity we pick the former.

THEOREM 4. *Let $\mathbb{E}$ be a secure dual-key cipher. Then $\mathcal{G} = \text{Garble2}[\mathbb{E}] \in \mathsf{GS}(\text{prv.ind}, \Phi_{\text{topo}}) \cap \mathsf{GS}(\text{obv.ind}, \Phi_{\text{topo}}) \cap \mathsf{GS}(\text{aut})$.* □

Again this asymptotic claim is underlain by concrete blackbox reductions and concrete bounds as follows. There are blackbox reductions U_{xxx} for $\text{xxx} \in \{\text{prv.ind}, \text{obv.ind}, \text{aut}\}$ s.t. if $\mathcal{A}(1^k)$ outputs circuits of at most r wires and fan-out at most ν, and then $\mathcal{D} = U^{\mathcal{A}}$ achieves xxx-advantage of at least ε, then $\mathcal{D} = U^{\mathcal{A}}_{\text{xxx}}$ achieves dkc-advantage at least $\varepsilon/2r - 2^{1-k}$, makes $Q \le 2\nu$ oracle queries, with $\mathbf{E}[Q] < 4$. It runs in time about that of $\mathcal{A}$ plus the time for $4r$ computations of $\mathbb{E}$ on k-bit keys.

Acknowledgments

This research was supported under NSF grant CNS 0904380; many thanks to the NSF for their continuing support. The paper was refined and first presented when Rogaway visited the Isaac Newton Institute for Mathematical Sciences, a program co-arranged by Nigel Smart. Thanks to Shafi Goldwasser, Kenny Patterson, and Thomas Schneider for comments and pleasant interactions.

7. REFERENCES

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):1–12, 1990.

[2] B. Applebaum. Key-dependent message security: Generic amplification and completeness. *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 527–546. Springer, 2011.

[3] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[4] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[5] B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. *ICALP 2010, Part I*, volume 6198 of *LNCS*, pages 152–163. Springer, 2010.

[6] B. Applebaum, Y. Ishai, and E. Kushilevitz. How to garble arithmetic circuits. *52nd FOCS*, pages 120–129. IEEE Computer Society Press, 2011.

[7] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *TCC 2007*, volume 4392 of *LNCS*, pages 137–156. Springer, 2007.

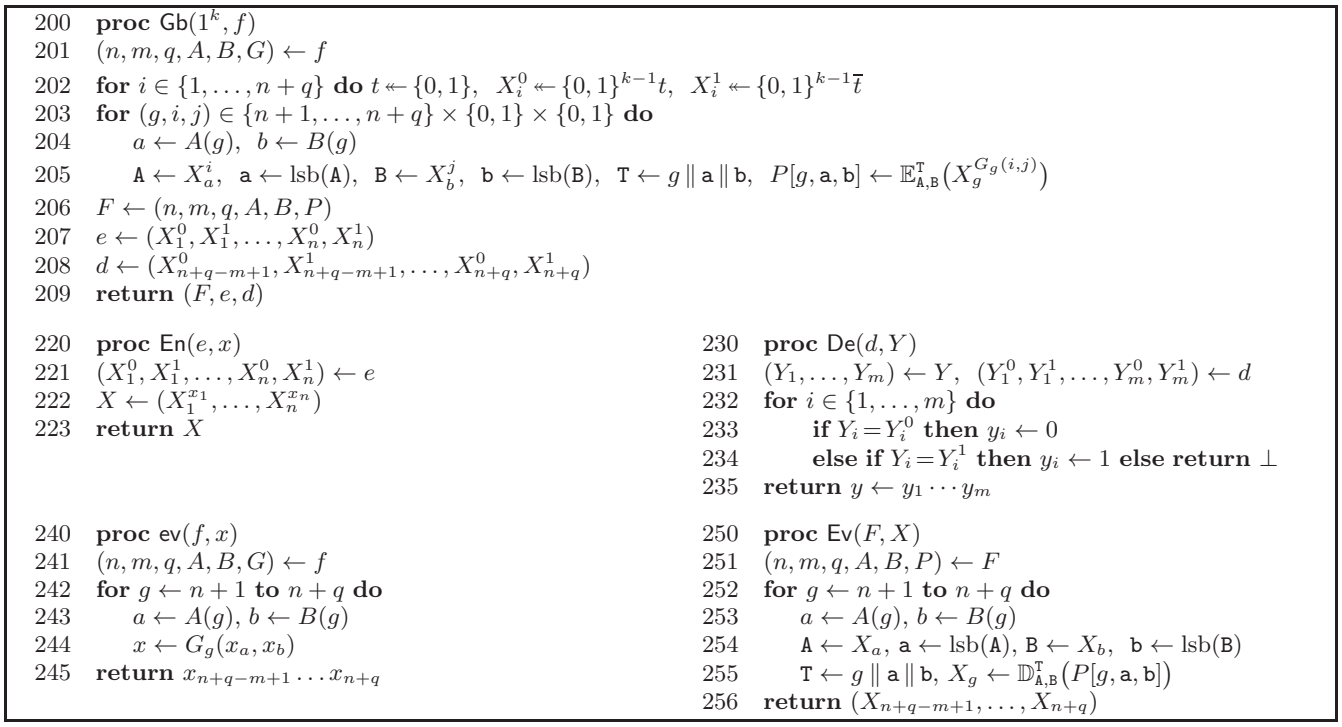

Figure 7: Garbling scheme Garble2. Its components are $(\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ where ev, shown for completeness, is canonical circuit evaluation. We assume a dual-key cipher $\mathbb{E}$ with tweak length τ and let $\mathbb{D}$ denote its inverse.

[8] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, 2010.

[9] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider. Secure evaluation of private linear branching programs with medical applications. *ESORICS 2009*, volume 5789 of *LNCS*, pages 424–439. Springer, 2009.

[10] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 503–513. ACM, 1990.

[11] M. Bellare, V. Hoang, and P. Rogaway. Foundations of garbled circuits. Cryptology ePrint Archive, Report 2012/265, 2012.

[12] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, 2006.

[13] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. *27th Intl. Colloquium on Automata, Languages, and Programming — ICALP 2000*, pages 512–523. Springer, 2000.

[14] M. Chase and S. Kamara. Structured encryption and controlled disclosure. *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 577–594. Springer, 2010.

[15] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation (extended abstract). *26th ACM STOC*, pages 554–563. ACM Press, 1994.

[16] K. Frikken, M. Atallah, and C. Zhang. Privacy-preserving credit checking. *Proceedings of the 6th ACM conference on Electronic commerce*, pages 147–154. ACM, 2005.

[17] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, 2010.

[18] O. Goldreich. Cryptography and cryptographic protocols. Manuscript, 2001.

[19] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. *19th ACM STOC*, pages 218–229. ACM Press, 1987.

[20] S. Goldwasser, Y. Kalai, and G. Rothblum. One-time programs. *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, 2008.

[21] V. Goyal, P. Mohassel, and A. Smith. Efficient two party and multi party computation against covert adversaries. *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 289–306. Springer, 2008.

[22] A. Herzberg and H. Shulman. Secure guaranteed computation. Cryptology ePrint Archive, Report 2010/449, 2010.

[23] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. *USENIX Security Symposium*, 2011.

[24] Y. Huang, C. Shen, D. Evans, J. Katz, and A. Shelat. Efficient secure computation with garbled circuits. *ICISS*, volume 7093 of *Lecture Notes in Computer Science*, pages 28–48. Springer, 2011.

[25] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. *41st FOCS*, pages 294–304. IEEE Computer Society Press, 2000.

[26] Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. *ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.

[27] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography with constant computational overhead. *40th ACM STOC*, pages 433–442. ACM Press, 2008.

[28] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. Cryptology ePrint report 2011/272, 2011.

[29] S. Kamara and L. Wei. Special-purpose garbled circuits. Unpublished manuscript.

[30] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, 2004.

[31] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.

[32] B. Kreuter, A. Shelat, and C. Shen. Billion-gate secure computation with malicious adversaries. *Proceedings of the 21th USENIX Security Symposium (USENIX 2012)*, 2012.

[33] L. Kruger, S. Jha, E. Goh, and D. Boneh. Secure function evaluation with ordered binary decision diagrams. *ACM CCS 06*, pages 410–420. ACM Press, 2006.

[34] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 52–78. Springer, 2007.

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

[36] Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *TCC 2011*, volume 6597 of *LNCS*, pages 329–346. Springer, 2011.

[37] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 20–20. USENIX Association, 2004.

[38] P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. *PKC 2006*, volume 3958 of *LNCS*, pages 458–473. Springer, 2006.

[39] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. *33rd ACM STOC*, pages 590–599. ACM Press, 2001.

[40] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. *Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139. ACM, 1999.

[41] A. Paus, A. Sadeghi, and T. Schneider. Practical secure evaluation of semi-private functions. *ACNS 09*, volume 5536 of *LNCS*, pages 89–106. Springer, 2009.

[42] K. Pietrzak. A leakage-resilient mode of operation. *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, 2009.

[43] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.

[44] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, 2009.

[45] P. Rogaway. The round complexity of secure protocols. MIT Ph.D. Thesis, 1991.

[46] A. Sahai and H. Seyalioglu. Worry-free encryption: functional encryption with public keys. *ACM CCS 10*, pages 463–472. ACM Press, 2010.

[47] T. Schneider. *Engineering Secure Two-Party Computation Protocols – Advances in Design, Optimization, and Applications of Efficient Secure Function Evaluation*. PhD thesis, Ruhr-University Bochum, Germany, February 9, 2011. http://thomaschneider.de/papers/S11Thesis.pdf.

[48] T. Schneider. *Engineering Secure Two-Party Computation Protocols*. Springer, Berlin Heidelberg, 2012.

[49] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient dna searching through oblivious automata. *ACM CCS 07*, pages 519–528. ACM Press, 2007.

[50] A. Yao. How to generate and exchange secrets. *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

[51] A. C. Yao. Protocols for secure computations. *23rd FOCS*, pages 160–164. IEEE Computer Society Press, 1982.

APPENDIX

A. RELATED WORK

We do not attempt a comprehensive review of the literature (easily a monograph-length undertaking), but elaborate on some selected prior work.

RANDOMIZED ENCODINGS. Loosely related to garbling schemes, *randomized encodings* (initially *randomized polynomials*) begin with Ishai and Kushilevitz [25] and continue, with many definitional variants, in work by Applebaum, Ishai, Kushilevitz, and others [2–6, 26, 27, 46]. The authors employ language like the following [3]: function $F(\cdot,\cdot)$ is a *randomized encoding* of $f(\cdot)$ if: (correctness) there's a PT algorithm De such that $\mathsf{De}(F(x,r)) = f(x)$ for almost all r; and (privacy) there's a PT algorithm Sim such that ensembles $F(x,\cdot)$ and $\mathsf{Sim}(f(x))$ are computationally indistinguishable. To be useful, encodings must have some extra properties, for example, that every bit of $F(x,r)$ depends on at most one bit of x, a property that has been called *decomposability* [27]. Proven realizations meeting these requirements [3, 4] do not closely resemble conventional realizations of garbled circuits [35, 40].

There is a large gap, even syntactically, between the notion just given and a garbling scheme. Above, no language is provided to speak of the algorithm that transforms f to F; in contrast, the thing doing this transformation is at the center of a garbling scheme. Likewise absent from the syntax of randomized encodings is anything to speak to the representation of functions; for garbling schemes, representations are explicit and central. Finally, the syntax, unlike that of a garbling scheme, does not separate the garbling of a function and the creation of a garbled input, and indeed there is nothing corresponding to the latter, the same input x being

fed to f or F. The minimalist syntax of randomized encodings works well for some theory-centric applications, but does not allow one to speak of obliviousness and authenticity, to investigate the low-level efficiency of different garbling schemes, and to architect schemes to useful-in-practice abstraction boundaries.

Given the variety of related definitions, let us sketch another, the *decomposable randomized encodings* defined and used by Sahai and Seyalioglu [46]. (Despite identical names, this definition is different from that above, and different again from the decomposable randomized encodings of [27], say). The object of interest can be regarded as a pair of PT algorithms $(\mathsf{En}, \mathsf{De})$ where En maps the encoding of a boolean circuit $f\colon \{0,1\}^n \to \{0,1\}^m$ to a vector of strings $(X_1^0, X_1^1, \ldots, X_m^0, X_m^1) \leftarrow \mathsf{En}(1^k, f)$ for which decoding algorithm $\mathsf{De}(X_1^{x_1}, \ldots, X_m^{x_m})$ returns $f(x_1 \cdots x_n)$. The authors demand a PPT algorithm Sim for which the ensemble of $(X_1^{x_1}, \ldots, X_m^{x_m})$ tuples induced by $\mathsf{En}(1^k, f)$ and x is computationally indistinguishable from $\mathsf{Sim}(1^k, n, |f|, f(x))$. Translating to our language, one has effectively assumed a projective scheme, a boolean circuit as input, and prv.sim security over Φ_{size}. The garbled function itself has been abstracted out of existence (in a realization, it would be dropped in the X_i^j values). Compared to a garbling scheme, one might note the lack of representation independence, granularity inadequate to speak of obliviousness, authenticity, garbled inputs, and low-level efficiency. The syntax can't handle the dynamic setting, where the adversary receives the garbled circuit before it specifies the input.

Obliviousness and authenticity. Some prior papers exploit obliviousness and authenticity of garbled circuits to achieve desired applications: private medical diagnostics [9], verifiable computation and private verifiable computation [17], and correctable verifiable computation [5]. The notions are not seen as properties of a stand-alone primitive corresponding to a garbling scheme.

In the last of the works mentioned, Applebaum, Ishai, Kushilevitz [5] describe the following generic transformations from privacy to obliviousness and to authenticity. (1) Obliviousness: instead of garbling a circuit f, let g be a circuit such that $g(x \parallel r) = f(x) \oplus r$ for every $x \in \{0,1\}^n$ and $r \in \{0,1\}^m$, where $n = f.n$ and $m = f.m$. Then, choose $r \twoheadleftarrow \{0,1\}^m$, run $(F, e, d) \leftarrow \mathsf{Gb}(g)$ and output $(F, (e, r), d)$. The garbled input corresponding to x will be $e(x \parallel r)$. (2) Authenticity: instead of garbling a circuit f, let g be a circuit such that $g(x \parallel K) = f(x) \parallel \mathrm{MAC}_K(f(x))$ for any $x \in \{0,1\}^n$ and any key K. Then, choose a random key K, run $(F, e, d) \leftarrow \mathsf{Gb}(g)$, and output $(F, (e, K), d)$. The garbled input corresponding to x will be $e(x \parallel K)$. Applied to Garble1, the transformations lead to schemes slightly (for (1)) or substantially (for (2)) less efficient that Garble2; and (2) requires a cryptographic assumption. More fundamentally, Applebaum *et al.* do not formalize any definition for the obliviousness or authenticity of a garbling scheme.

The only work that explicitly defines obliviousness and authenticity in this domain is a recent paper of Kamara, Mohassel, and Rakova [28]. Still, their syntax is designed specifically for their application; for example, a circuit's input is a pair (x_1, x_2), a garbled circuit's input is (X_1, X_2), and the encoding function takes an input x and an index $i \in \{1, 2\}$ and outputs the corresponding X_i. Their notion of obliviousness requires hiding only the input, while obv.ind and obv.sim require one to hide both the input and the function.

Obscuring topology. We are not the first to observe that conventional means to garble a circuit obscure each gate's function but not its topology. A 2002 paper of Pinkas [43] (Section 2.3) already remarks that "In this form the representation reveals nothing but the wiring of the circuit". Later, Paus, Sadeghi, and Schneider [41] use the phrase "circuit topology" to name that which is revealed by conventional garbled circuits. Nevertheless, the topology of a circuit is never formalized, and nobody ever proves that that some particular scheme reveals only the topology. We are also the first to explain the equivalence between the prv.sim and prv.ind notions relative to Φ_{topo}.

Eclectic representations. Scattered through the literature one finds computational objects other than boolean circuits that are being garbled; examples include arithmetic circuits [6], branching programs [9], circuits with lookup tables [39], DFAs [49], and ordered binary decision diagrams [33]. The range suggests, to us, that general-purpose definitions for garbling schemes ought not be tied to circuits.

Concurrent work. Concurrent work by Kamara and Wei (henceforth KW) investigates the garbling of *structured circuits* [29], a computational model they put forward resembling ordinary circuits except that gates perform operations on an arbitrary data structure. As part of this work, KW define what they too call a garbling scheme. Their syntax is similar to ours, but without the function ev. Over this syntax KW define Ind1 and Sim1 security. These notions, unlike ours, ask only for input-hiding, not function hiding. They show these definitions are equivalent for *sampleable* circuits. KW go on to give dynamic versions of their definitions, Ind2 and Sim2, and an unforgeability notion, Unf2. These definitions resemble the weaker form of the dynamic-security definitions (prv1, obv1, and aut1) mentioned in our Introduction and the subject of separate work.

Although KW speak of circuits as finitary objects described by DAGs, they appear to have in mind families of circuits, indexed by a security parameter (otherwise, we do not know how to make sense of samplability, or phrases like *polynomial size circuits*). Unlike our treatment, circuits are not provided by the adversary; security notions are with respect to a given circuit. A garbling scheme is provided in KW, but not a "conventional" one: it garbles a structured circuit and is based on a collection of *structured encryption schemes*, a notion from Chase and Kamara [14]. For the protocol to make sense with respect to the definitions given, the latter should be reinterpreted as applying to structured circuits.