Succinct Garbling Schemes from Functional Encryption through a Local Simulation Paradigm*

Prabhanjan Ananth[1] and Alex Lombardi[2]

[1] prabhanjan@csail.mit.edu, MIT
[2] alexjl@mit.edu, MIT

Abstract. We study a simulation paradigm, referred to as *local simulation*, in garbling schemes. This paradigm captures simulation proof strategies in which the simulator consists of many local simulators that generate different blocks of the garbled circuit. A useful property of such a simulation strategy is that only a few of these local simulators depend on the input, whereas the rest of the local simulators only depend on the circuit.

We formalize this notion by defining locally simulatable garbling schemes. By suitably realizing this notion, we give a new construction of succinct garbling schemes for Turing machines assuming the polynomial hardness of compact functional encryption and standard assumptions (such as either CDH or LWE). Prior constructions of succinct garbling schemes either assumed sub-exponential hardness of compact functional encryption or were designed only for small-space Turing machines.

We also show that a variant of locally simulatable garbling schemes can be used to generically obtain adaptively secure garbling schemes for circuits. All prior constructions of adaptively secure garbling that use somewhere equivocal encryption can be seen as instantiations of our construction.

1 Introduction

Garbling schemes are ubiquitous to cryptography. Their notable applications include secure computation on the web [GHV10, HLP11], constructions of functional encryption [SS10, GVW12, GKP+12], one-time programs [GKR08], delegation of computation [GGP10, AIK10], and garbled RAMs [GHL+14, GLOS15]. In fact, there are many more applications under the umbrella of randomized encodings, which are implied by garbling schemes. These applications include parallel cryptography [AIK04, AIK06], bootstrapping theorems in functional encryption and indistinguishability obfuscation [ABSV15, App14a], and key-dependent message security [BHHI10, App14b]. More recently, garbling schemes were also crucially used to solve two longstanding open problems in cryptography: achieving two-round passively secure MPC [GS17, BL18, GS18c]

* The full version of this paper is available at https://eprint.iacr.org/2018/759.

and identity-based encryption from weaker assumptions [DG17, BLSV18, DGHM18].

A garbling scheme allows for efficiently encoding a circuit C, represented by $\langle C \rangle$ (also referred to as *garbled circuit*), and separately encoding an input x, represented by $\langle x \rangle$. We require that given $\langle C \rangle$ and $\langle x \rangle$, it is possible to efficiently recover $C(x)$ and moreover, the encodings should not leak anything beyond $(C, C(x))$[3]. This notion was first introduced by Yao [Yao82, Yao86] as a technique to solve two-party secure computation (a full proof of this application was only given much later by Lindell and Pinkas [LP09]). More than three decades later, proposing new constructions of garbling schemes is still an active and fascinating research direction.

While the traditional notion of garbling schemes considers encoding circuits, this notion can be generalized for other models of computation. In particular, we consider garbling *Turing machines*; this notion is often referred to as succinct garbling schemes [BGL+15, CHJV15, KLW15]. The non-triviality in this setting is to encode both the Turing machine M and the input x in time independent of the runtime of M. In more detail, we require that the time to garble a Turing machine M should be polynomial in λ (security parameter) and $|M|$ while the time to encode an input x should be polynomial in λ and $|x|$. For decoding, we require that it should only take time polynomial in λ and t to recover $M(x)$, where t is the runtime of M on x.

Succinct garbling schemes have been used in many applications including time-lock puzzles [BGJ+16], concurrent zero-knowledge [CLP15], indistinguishability obfuscation for Turing machines [BGL+15, CHJV15, KLW15] and delegation for deterministic computations [BGL+15, CHJV15, KLW15]. In terms of constructions, the initial works of [BGL+15, CHJV15] proposed succinct garbling schemes with the caveat that the size of the garbled Turing machine grows with the maximum space taken by the Turing machine during its execution. Subsequently, Koppula et al. [KLW15] showed how to get rid of this caveat and presented a construction of succinct randomized encodings (a notion where M and x are encoded together) assuming indistinguishability obfuscation and one-way functions.

It is worth noting that the approach taken by [BGL+15] differs substantially from the approach taken by [CHJV15, KLW15] to obtain succinct randomized encodings. The construction of [BGL+15] is very simple to describe: they succinctly garble a Turing machine M (running in time at most T) by outputting an obfuscated program that on input $i \leq T$ outputs the ith garbled table of a Yao garbled circuit [Yao82, Yao86, LP09] associated to a circuit C representing M's computation. One might hope that this already yields a fully succinct garbling scheme, but the security proof of [BGL+15] requires hardwiring $O(s)$ bits of information in the obfuscated program when M requires space s, so this does not yield a fully succinct garbling scheme (which [KLW15] does achieve).

[3] In this work, we only consider the case of hiding the input x. To hide the circuit C being garbled, we can garble an universal circuit with an encryption of C hardwired inside it and produce an input encoding of x along with the decryption key.

While the final result has an undesirable dependence on s, the [BGL+15] approach has the advantage of relying only on obfuscation for circuits of input length $\log(T) = O(\log(\lambda))$ and hence can be proved secure assuming the existence of polynomially secure functional encryption [AJ15, BV15, LZ17, LT17]. The approach of [CHJV15, KLW15] does not share this property, and indeed there is currently no known construction of fully succinct garbling from (poly-secure) FE. In general, there are a few primitives (such as trapdoor permutations and non-interactive key exchange) known to follow from FE [GPS16, GS16, GPSZ17, LZ17] while many others (such as NIZK [SW14, BP15], deniable encryption [SW14], and long output secure function evaluation [HW15]) we know only how to construct from IO (see [LZ17] for a more detailed discussion). One of our main goals is to understand whether constructing succinct garbling schemes requires the full power of IO in this sense.

Rather intriguingly, the progress on succinct randomized encodings followed a similar pattern to progress on the problem of constructing adaptively secure circuit garbling schemes. There is a simple transformation [BHR12] from selectively secure garbling schemes to adaptively secure garbling schemes in which the online complexity (that is, the size of the input encoding) grows with the circuit size. Subsequent to [BHR12], the work of [HJO+16] showed how to achieve adaptive schemes with online complexity that only depends on the width w of the circuit (from one-way functions) or depth d of the circuit (from $2^{-O(d)}$-secure one-way functions). Following [HJO+16], the works of [JW16, JSW17] present additional constructions of adaptive circuit garbling schemes. Finally, a beautiful work of Garg and Srinivasan [GS18a] showed how to achieve adaptive garbling schemes with online complexity $|x| + \mathrm{poly}(\log(|C|, \lambda)$ assuming either the computational Diffie-Hellman (CDH) or learning with errors (LWE) assumption.

We note that the measure of width complexity in the case of circuits is related to the measure of space complexity in the case of Turing machines. Indeed, we can transform a Turing machine M that requires space s on inputs of length n into a circuit of width $O(n + s)$; similarly, a circuit of width w can be simulated by a Turing machine which takes space at most $O(w)$. Moreover, there are actually major similarities between the *security proofs* of [HJO+16] (for their width-dependent adaptive garbling scheme) and [BGL+15] (for their space-dependent succinct garbling scheme). At a high level, both require opening up the [LP09] proof of security for Yao's garbling scheme and make use of the fact that security is argued by a gate-by-gate hybrid argument.

These similarities present the possibility of transporting some of the techniques from the adaptive garbling literature in order to construct new and improved succinct garbling schemes. In particular, we ask: can the ideas from [GS18a] be used to construct succinct garbling?

1.1 Our Contributions

We give a new construction of succinct garbling schemes using the ideas of [GS18a]. Unlike the work of [KLW15][4] based on sub-exponentially secure compact functional encryption, our construction is based on polynomially secure compact functional encryption and polynomially secure CDH/LWE. As an added advantage, our construction is conceptually simpler. Instead of using IO/FE to compress a Yao garbled circuit as in [BGL+15], we compress an appropriately modified [GS18a] garbled circuit.

To prove security, we identify a property, termed as *local simulation*, of selectively secure garbling schemes for circuits that when combined with other tools yields succinct garbling schemes. To describe this property, we first recall the security experiment of garbling schemes. To prove that a given garbling scheme is secure, one needs to exhibit a simulator with the following property: given just the circuit C and the output $C(x)$, it can output a simulated garbled circuit and input encoding that is indistinguishable from an honest garbled circuit and input encoding. Typically this indistinguishability is shown by a sequence of hybrids: in every step, a hybrid simulator is defined to take an input C and x produces the simulated garbling and input encoding. The first hybrid defines the honest garbling of C and the honest encoding of x, while the final hybrid defines the simulated distribution. At a bare minimum, our notion of local simulation captures a class of such hybrid arguments wherein the simulation of garbled circuit is divided into blocks and in every hybrid, only a *small* L_{sim}-sized subset of blocks are simulated using C and x while the rest are simulated only using C. We observe that this seemingly artificial property is already satisfied by current known schemes [Yao86, GS18a].

To make the local simulation notion useful for applications, we need to consider strengthenings of this notion. We formalize the above informal description of local simulation and call this **weak** local simulation; correspondingly the garbling scheme will be called a weak locally simulatable garbling scheme (weak LSGS). We consider two strengthenings: (i) **strong** locally simulatable garbling schemes (strong LSGS) and (ii) **semi-adaptive** locally simulatable garbling schemes (semi-adaptive LSGS). Both the notions of semi-adaptive LSGS and strong LSGS imply weak LSGS and will be parameterized by $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$, where L_{inp} refers to the online complexity of the garbling scheme.

We now state our results on succinct garbling.

Succinct Garbling. We prove the following theorem.

[4] We note that [KLW15] construct succinct randomized encodings scheme and not garbling schemes. However, their construction can be adapted to get succinct garbling schemes

Theorem 1. *(Main Theorem) Assuming single-key compact*[5] *public-key functional encryption for circuits*[6] *and X, where $X \in \{$Computational Diffie-Hellman, Factoring, Learning with Errors$\}$, there exists a succinct garbling scheme for Turing machines.*

Previous constructions of succinct garbling schemes were based on indistinguishability obfuscation[7] (implied by *sub-exponentially* secure compact functional encryption) and one-way functions [KLW15]. This is the first work to show the feasibility of succinct garbling schemes from falsifiable assumptions. Moreover, [KLW15] is significantly more involved whereas our construction is conceptually simpler. We note that several works subsequent to [KLW15] use their construction to achieve various primitives including garbled RAM [CH16, CCC+16, CCHR16, ACC+16], constrained PRFs for Turing machines [DKW16], indistinguishability obfuscation for Turing machines with constant overhead [AJS17a], patchable indistinguishability obfuscation [AJS17b, GP17] and so on. We hope that our simpler construction will correspondingly yield simpler presentation of these applications as well.

One new consequence of the above theorem is that we obtain collusion-resistant functional encryption for Turing machines from collusion-resistant functional encryption for circuits and standard assumptions; this follows from [AS16].

We prove Theorem 1 in two steps. First, we prove the following proposition.

Proposition 1 (Informal). *Assuming strong $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$-LSGS and compact functional encryption for circuits, there exists a succinct garbling scheme in which the complexity of garbling a Turing machine M is $\mathrm{poly}(\lambda, |M|, L_{\mathsf{sim}})$ and the complexity of encoding x is $L_{\mathsf{inp}}(\lambda, |x|, m)$, where m is the output length of M.*

Once we prove the above proposition, we show how to instantiate strong LSGS from laconic oblivious transfer[8] to obtain our result.

Proposition 2 (Informal). *Assuming laconic oblivious transfer, there exists a strong $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$-LSGS with $L_{\mathsf{sim}} = \mathrm{poly}(\lambda)$.*

[5] From prior works [BV15, AJS15], we can replace compact public-key FE with collusion-resistant FE in the theorem statement.

[6] A public-key functional encryption scheme is a public-key encryption scheme with the additional key generation procedure that takes as input circuit C and produces a functional key for C that can be used to decrypt an encryption of x to obtain $C(x)$. A **compact** functional encryption is a functional encryption scheme where the complexity to encrypt a message x is a fixed polynomial in $(\lambda, |x|)$ and in particular, the encryption complexity grows only with $\log(|C|)$. A functional encryption scheme is a **single-key scheme** if it satisfies $\{\mathsf{PK}, \mathsf{Enc}(\mathsf{PK}, x_0), sk_C\} \cong_c \{\mathsf{PK}, \mathsf{Enc}(\mathsf{PK}, x_1), sk_C\}$ for an adversarially chosen C and x and specifically, the adversary is only issued a single key in the security experiment.

[7] See the full version for a formal definition.

[8] We actually use the existentially equivalent notion of *appendable laconic OT*, which we define in the full version.

Since laconic oblivious transfer can be instantiated from CDH, factoring, LWE and other assumptions [CDG+17, DG17, BLSV18, DGHM18], this proves Theorem 1. In addition, we note (in full version) that laconic OT[9] can be constructed from IO and one-way functions; combined with the above propositions, this says that our succinct garbling scheme can also be instantiated from IO and OWFs alone (giving an alternative construction to [KLW15]).
We note that the garbling scheme of Yao [Yao86] also yields a strong $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$-LSGS with L_{sim} proportional to the width of the circuit being garbled. Combining this with Proposition 1, we get a succinct garbling scheme for small space Turing machines; this is essentially the same scheme as that of [BGL+15].

Adaptive circuit garbling. Next, we show how to construct adaptive circuit garbling schemes using our notion of (semi-adaptive) LSGS. First, we recall the definition of adaptive circuit garbling schemes. In the adaptive security experiment, an adversary can submit the circuit C and the input x in any order; specifically, it can choose the input as a function of the garbled circuit or vice versa. We show,

Theorem 2 (Informal). *Assuming semi-adaptive $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$-LSGS and one-way functions, there exists an adaptively secure circuit garbling scheme with online complexity $L_{\mathsf{inp}} + \mathrm{poly}(\lambda, L_{\mathsf{sim}})$.*

This theorem can be seen as an abstraction of what the somewhere equivocal encryption-based technique of [HJO+16] can accomplish. For example, the semi-adaptive LSGS can be instantiated from laconic oblivious transfer, recovering the result of [GS18a]. The theorem below follows from a previous work [GS18a].

Theorem 3 ([GS18a]). *Assuming laconic oblivious transfer, there exists a semi-adaptive $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$-LSGS scheme with online complexity $L_{\mathsf{inp}}(\lambda, n, m) = n + m + \mathrm{poly}(\lambda)$ and $L_{\mathsf{sim}} = \mathrm{poly}(\lambda)$, where n and m denote the input and output lengths for the circuit.*

We note that Yao's garbling scheme is also a semi-adaptive $(L_{\mathsf{sim}}, L_{\mathsf{inp}})$-LSGS with L_{sim} being proportional to the width of the circuit and thus, combining the above two theorems we get an adaptively secure circuit garbling scheme with the online complexity proportional to the width of the circuit. This construction is essentially the same as the width-based construction of [HJO+16], with a more modular security proof.
We summarise the results in Figure 1.

1.2 Concurrent Work

In concurrent and independent work, Garg and Srinivasan [GS18b] give a construction of succinct randomized encodings from IO (and laconic OT)

[9] We can only achieve laconic OT satisfying selective security, which suffices for Proposition 2.

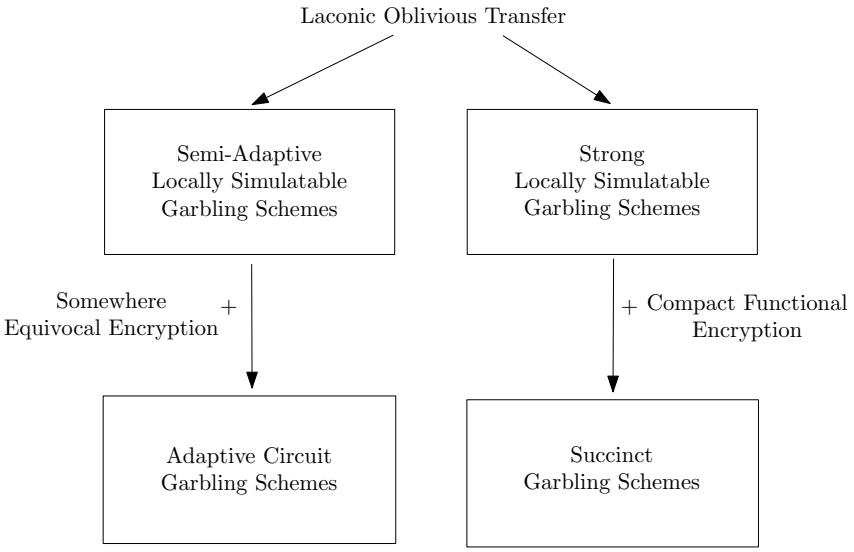

Fig. 1. Summary of results.

that implicitly relies on only IO for logarithmic length inputs, and hence polynomially-secure functional encryption. While their work is phrased differently from this work (in particular, they give a direct construction without considering the abstraction of local simulatability), the basic SRE constructions are essentially the same.

1.3 Technical Overview

We first recall the garbling scheme of Yao [Yao86] and describe an overview of its security proof. Yao's scheme will serve as a starting point to understanding the definition of locally simulatable garbling schemes.

Yao's Garbling Scheme [Yao86]. Consider a boolean circuit $C : \{0,1\}^\ell \to \{0,1\}$ comprising only of NAND gates. For ease of presentation, we assume that C is layered such that all gates that are at the same distance from the output gate belong to the same layer. Moreover, every intermediate wire in the circuit connects two gates in adjacent layers.
The first step in the garbling of a circuit C is to generate two wire keys K_w^0 and K_w^1 for every wire w in the circuit. Next, associate with every gate G a garbled table consisting of four entries $(\mathsf{CT}_{00}, \mathsf{CT}_{01}, \mathsf{CT}_{10}, \mathsf{CT}_{11})$. For $b_0, b_1 \in \{0,1\}$, $\mathsf{CT}_{b_0 b_1}$ is an encryption of $K_{w_c}^{\mathrm{NAND}(b_0,b_1)}$ under the two keys[10] $K_{w_a}^{b_0}$ and $K_{w_b}^{b_1}$. Wires w_a and w_b are input wires of G and w_c is the output wire of G. Finally, permute the garbled table $(\mathsf{CT}_{00}, \mathsf{CT}_{01}, \mathsf{CT}_{10}, \mathsf{CT}_{11})$.

[10] There are many ways of realizing an encryption scheme under two different secret keys. One convenient method is to secret share the message and encrypt the two shares using the two keys.

The garbling of C consists of permuted garbled tables associated with every gate in the circuit. The input encoding of x consists of keys $K_{w_i}^{x_i}$, where w_i is the i^{th} input wire of C and x_i is the i^{th} bit of x. Also part of the input encoding is a translation table that maps 0 to $K_{w_{\text{out}}}^0$ and 1 to $K_{w_{\text{out}}}^1$, where w_{out} is the output wire of C.

Selective Security of Yao's Garbling Scheme: To show that Yao's garbling scheme is secure we need to demonstrate a probabilistic polynomial time simulator Sim that given $(C, C(x))$ (and in particular, x is not given) outputs a simulated garbling of C and a simulated input encoding. Sim is defined as follows: every wire w is only associated with a single key K_w. Associated with every gate G is a garbled table consisting of $(\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_3, \mathsf{CT}_4)$, where: for a randomly picked index $i^* \in [4]$, (i) CT_{i^*} is an encryption of K_{w_c} under keys K_{w_a} and K_{w_b}, (ii) for $i \neq i^*$, CT_i is an encryption of 0 under two randomly chosen secret keys (and in particular these two keys are not used anywhere). The simulated garbling of C consists of the simulated garbled tables associated with every gate in the circuit. The input encoding consists of the keys $\{K_w\}$ for every input wire w. In addition, it consists of the translation table that maps $C(x)$ to $K_{w_{\text{out}}}$ and maps $\overline{C(x)}$ to $K'_{w_{\text{out}}}$, where $K'_{w_{\text{out}}}$ is generated afresh.

The indistinguishability of the output of Sim from an honestly generated garbled circuit and input encoding can be argued by a hybrid argument explicitly described in [HJO+16]. This hybrid argument will be associated with a sequence of intermediate simulators $\mathsf{Sim}_1, \ldots, \mathsf{Sim}_q$. Except Sim_q, all the other simulators take as input circuit and the input; (C, x). The final simulator Sim_q takes as input $(C, C(x))$. Sim_1 computes the garbling of C and the input encoding of x as dictated by the scheme. The final intermediate simulator Sim_q is identical to Sim.

The i^{th} intermediate simulator Sim_i works as follows: for every wire w such that w is the output wire of a j^{th} layer for $j \geq i$, sample two keys K_w^0 and K_w^1. For any other wire w, sample a single wire key K_w. The simulator consists of two components:

- **Input-Dependent Simulation.** This component takes as input (C, x) and simulates all the garbled gates in the i^{th} layer of C. For every gate G (with input wires w_a, w_b and output wire w_c) in the i^{th} layer, generate a garbled table $(\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_3, \mathsf{CT}_4)$, where for a randomly picked index $i^* \in [4]$, (i) CT_{i^*} is an encryption of $K_{w_c}^{\mathsf{val}(w_c)}$ under keys K_{w_a} and K_{w_b}, (ii) for $i \neq i^*$, CT_i is an encryption of 0 under two randomly chosen secret keys (and in particular these two keys are not used anywhere). Here, $\mathsf{val}(w_c)$ denotes the value assigned to the wire w_c during the evaluation of C on x.
- **Input-Independent Simulation.** This component only takes as input C and simulates the garbled gates in all the layers except the i^{th} layer. There are two cases:
  - for a gate G in the j^{th} layer, for $j < i$, the simulation of the garbled gate for G is performed according to Sim.
  - for a gate G in the j^{th} layer, for $j > i$, the garbled gate for G is generated according to the scheme.

Once the computational indistinguishability of Sim_i and Sim_{i+1} is shown for every i, the security of the scheme follows.

Complexity of Input-Dependent Simulation. Observe that the output length of the input-dependent simulation component of every simulator Sim_i is only proportional to the width of the circuit (in other words, the maximum length of any layer in C). This observation has been crucially exploited in two lines of work:

- The work of [HJO+16] introduced the powerful tool of somewhere equivocal encryption (SEE) and showed how to combine it with the garbling scheme of Yao to obtain adaptive garbling schemes with online complexity that grows with the width of the circuit. Informally, somewhere equivocal encryption is used in conjunction with the above proof of security for Yao's scheme: in each step of a hybrid argument, the input-dependent simulated gates are equivocated in the online phase of the adaptive security game. Since the number of input-dependent simulated gates is bounded by the width w of the circuit, the online complexity of this garbling scheme is proportional to w. Alternative proof strategies for Yao's garbling scheme can be used instead of our sketch above to obtain, for example, the depth-based result of [HJO+16].
- The work of [BGL+15] showed how to combine indistinguishability obfuscation for circuits and the garbling scheme of Yao to obtain a succinct garbling scheme for small-space Turing machines. To garble a Turing machine M that has worst-case runtime T, they construct an obfuscation of a circuit that takes as input an index i and outputs the garbled table corresponding to the ith gate of C[11]. Security is argued by sequentially invoking the simulators $(\mathsf{Sim}_1, \ldots, \mathsf{Sim}_q)$ of Yao's garbling scheme. Hardwiring the entire simulator's output in the obfuscated circuit would ruin the encoding complexity of the succinct garbling scheme. However, it turns out that security can be argued when only the *input-dependent* simulation component is hardwired. This is exactly the reason why the encoding complexity of this succinct garbling scheme grows with the maximum space complexity of the Turing machines.

Locally Simulatable Garbling Schemes. We introduce the notion of a locally simulatable garbling scheme as an abstraction that connects the above proofs of adaptive security and succinctness for garbling schemes. We give a brief overview of the security property associated with a locally simulatable garbling scheme. The security property is parameterized by an integer L_{sim} and a sequence of simulators $(\mathsf{Sim}_1, \ldots, \mathsf{Sim}_q)$ for some polynomial q. Every simulator Sim_i consists of an input-dependent component and an input-independent component.

[11] Their actual scheme instead outputs an entire layer of garbled tables at once, but this variant has the same efficiency and security proof.

– The input-dependent component of Sim_i takes as input circuit C and input x to be simulated. We require that this component of Sim_i is of size at most $L_{\mathsf{sim}} \cdot \mathrm{poly}(\lambda)$ for some fixed polynomial poly.
– The input-independent component of Sim_i takes as input only the circuit C.

We require that the output distribution of Sim_1 is computationally indistinguishable from an honest generated garbling of C and honestly generated encoding of x. The output distributions of Sim_i and Sim_{i+1} are required to be computationally indistinguishable. Finally, we require that the final simulator Sim_q does not have any input-dependent component and in particular, Sim_q can simulate the garbled circuit and input encoding on input $(C, C(x))$. We refer to this security property as *weak* local simulation security.

Note that Yao's garbling scheme, using the security proof given in our outline, is a particular instantiation of a locally simulatable garbling scheme with L_{sim} set to be the width of the circuit being garbled. The depth-based analysis of Yao's garbling scheme given in [HJO+16] can also be seen as an instantiation of weak local simulation, albeit with $q = 2^{O(d)}$ hybrids.

While the above security property captures the essence of local simulation, it does not suffice for either the application of adaptively secure garbling schemes or the application of succinct garbling schemes. To get around this, we strengthen the security definition in two ways, resulting in notions of **semi-adaptive** locally simulatable garbling schemes and **strong** locally simulatable garbling schemes.

Succinct Garbling from Strong LSGS. We define the notion of a strong locally simulatable garbling scheme and use it as an intermediate tool to construct a succinct garbling scheme. To motivate our definition, we will consider a candidate succinct garbling scheme (and proof strategy) from IO and a weak LSGS, and see what additional properties are required from the LSGS.

Generalizing the approach of [BGL+15], our candidate succinct garbling scheme is as follows: garbling a Turing machine M with a runtime bound T consists of computing an indistinguishability obfuscation of a circuit $H_{M,T,\mathsf{MSK}}$ with hardwired values M, T and a master secret key MSK. This circuit takes as input an index $i \leq T$, constructs the i^{th} gate of C, where C is the circuit representing T steps of M's computation on inputs of length n, and then outputs a garbling of this gate computed with respect to MSK. Encoding x consists of computing the input encoding of x with respect to the LSGS. Decoding proceeds by evaluating the obfuscated circuit on all indices ranging from 1 to T to obtain the different gate encodings. These encodings are then decoded to obtain the result.

We are already implicitly assuming some properties of the underlying LSGS in order for the above construction to make any sense at all. Specifically,

– Our candidate implicitly assumes that a garbling of C is computed in a gate-by-gate fashion. To enable this, we introduce the notion of a local encoding of an LSGS, which guaratees that a garbling of C

consists of components that are each computed in time independent of $|C|$; in particular, it must be computable from a small amount of information about C. In fact, we further require that this information about C is efficiently computable from M. In the case of Yao, this amounts to saying that an individual gate of C can be computed very efficiently from M.

- A priori, the master secret key MSK could be as large as $|C| = \text{poly}(T)$. Strictly speaking, this means that the above candidate is not succinct. To overcome this, we think of $\mathsf{MSK} = (sk_1, \ldots, sk_N)$ and define a *local key generation* procedure that takes as input an index j and only generates the local secret key sk_j. Then, the program H in our scheme takes as input an index i, determines the keys sk_j that are necessary to encode the i^{th} component of C, and then computes the i^{th} garbled component.
- To identify the subset of keys to be locally generated for the i^{th} component, we define a *key list generation* procedure that takes as input i and outputs a list L_i. This allows us to compress the potentially large MSK using a pseudorandom function key.
- The size of the input encoding of the succinct garbling candidate is exactly the same as the online complexity of the underlying strong LSGS scheme. Thus, in order for our scheme to be succinct, the online complexity of the underling LSGS scheme will have to be independent of T.

By carefully defining the above notions, we can guarantee that the program H is sufficiently small (polynomial in λ) so that the candidate garbling scheme is succinct. What remains is to prove security in a way such that programs H' that are obfuscated in the security proof are also small. This is the most subtle step; in particular, this is the step where [BGL+15] is limited to achieving succinctness that depends on the space of the Turing machine.

To prove the security of the above scheme, a naive approach would be to hardwire the entire simulated garbled circuit inside the obfuscation of $H_{M,T,\mathsf{MSK}}$; however, this would violate succinctness. Instead, we want to leverage *local simulation* in the following way: in each of a sequence of hybrid circuits $(H_1, \ldots, H_q)$, only hardwire the *input-dependent* components of Sim_i, and instead include the code of the input-independent components of Sim_i (which naively contains all of MSK) inside H_i. We would then hope to argue using some combination of IO security and LSGS security that adjacent hybrid programs in this sequence are indistinguishable.

If the size of the input-dependent portion is small, meaning polynomial in λ, then we can hope to achieve succinctness using this proof strategy. This approach again implicitly assumes properties of the LSGS; namely, that the input-independent local simulators each require only a small portion of the master secret key (just as in the honest garbling case). This is required so that the hybrid circuit H_i is still small.

Unfortunately, the security argument above is flawed. The problem is that information about the master secret key MSK is contained within the obfuscated program $\tilde{H}$, so it is unclear how to argue that the input-dependent components of Sim_i and Sim_{i+1} (i.e. the components that are

hardwired) are indistinguishable. Indeed, if the above strategy is not carefully implemented (e.g. if the program H_i actually reveals the entire MSK), they will be distinguishable.

To circumvent these issues, we require that the input-dependent portion of the garbled circuit output by Sim_i is indistinguishable from the corresponding input-dependent portion of the garbled circuit output by Sim_{i+1} *even in the presence of* $\{sk_j\}_{j\in S}$*, where* S *consists of all indices accessed by the input-independent portion of the garbled circuit.* In fact, we define a stronger property that allows the adversary to choose the keys $\{sk_j\}_{j\in S}$.

In order to complete the hybrid argument, our proof strategy then works in two steps: first switch the input-dependent components of the simulated circuit from Sim_i to Sim_{i+1} (using the above strong LSGS security), and then switch the input-independent components from Sim_i to Sim_{i+1}. Since we are actually including the code of these input-independent simulators within the obfuscated circuit, we must require that the input-independent components of Sim_i and Sim_{i+1} are functionally equivalent to invoke IO security.

To summarize, a strong LSGS must satisfy two main properties in order for the security of our succinct garbling scheme to be proved:

- The input-dependent components of Sim_i and Sim_{i+1} must be indistinguishable *even given* all of the local secret keys necessary to compute the input-independent components of Sim_i.
- The *algorithms* computing the input-independent components of Sim_i and Sim_{i+1} must be functionally equivalent.

Indeed, the security proof of [BGL+15] can be retroactively seen as invoking the above properties of Yao's garbling scheme. For completeness, we sketch a proof (see the full version) that Yao's garbling scheme satisfies this definition with L_{sim} proportional to the width of the circuit.

Constructing Strong LSGS from Laconic OT. In order to complete the proof of Theorem 1, we show that the garbling scheme of [GS18a] can be adapted to satisfy our strong LSGS notion with $L_{\mathsf{sim}} = \mathrm{poly}(\lambda)$. We begin by giving a high level description of the [GS18a] garbling scheme:

- An encoding of an input x consists of (1) a somewhere equivocal encryption secret key, (2) a one-time pad encryption $r \oplus x$ of x, (3) a hash value $h_0 = H(r \oplus x || 0^{|C|-n})$ of an initial memory state for the computation, (4) a signature on h_0, and (5) the one-time pads corresponding to each output gate. The hash function H is associated to a laconic OT scheme (we omit a discussion of laconic OT from this overview).
- An encoding of a circuit C consists of $s = |C|$ "garbled programs" maintaining the following invariant: after executing i such programs, the evaluator will have obtained a one-time pad encryption of the first $n + i$ gates of C evaluated on the input x along with a hash of this one-time padded state and a signature on this hash value. The garbled programs are then jointly encrypted using a somewhere equivocal encryption scheme.

– Simulation security is argued by a sequence of hybrid simulators; in a hybrid simulator, each garbled program is either computed via an input-independent simulator or an input-dependent simulator, and moreover only poly(λ) garbled programs require input-dependent simulation. To prove adaptive security, the input-dependent simulated gates are equivocated as part of the input encoding.

We interpret the scheme of [GS18a] – after removing the somewhere equivocal encryption layer – as a LSGS by thinking of each garbled program above as one component of the LSGS. Indeed, we show that each garbled program in the [GS18a] scheme only requires a small amount of the garbling secret key and that the input-dependent components of Sim_i and Sim_{i+1} are indistinguishable even in the presence of adversarially chosen secret keys used for the other components. In fact, all but one of the properties of a strong LSGS as defined earlier can be demonstrated to hold for the [GS18a] scheme without modification.

The only problem with using the [GS18a] scheme as a strong LSGS is that computation of the initial hash value $H(r \oplus x||0^{|C|-n})$ requires $O(|C|)$ time. Naively, this means that computing even the input encoding would take $O(|C|)$ time, but [GS18a] note that if H is computed via a Merkle tree, the computation of $H(0^{|C|-n})$ can be delegated to the garbled circuit and only $H(r \oplus x)$ need be computed during the input encoding. However, computing $H(0^{|C|-n})$ *cannot* be done locally (i.e. distributed in pieces to local components of the garbled circuit), which violates the local encoding property of a strong LSGS.

To circumvent this problem, we modify the [GS18a] scheme so that the initial hash value $h_0 = H(r \oplus x)$ is a hash of only an n-bit string, and we redesign the garbled programs so that each step updates the one-time padded computation state by *appending* the next value. Instantiating this corresponds to a new notion of *appendable laconic OT*, which we define and construct generically from laconic OT. The local simulators for our new scheme remain essentially the same, and our previous security proof carries over to this modified version. We note that the same modification could be made to the [GS18a] adaptive garbled circuit construction, with the advantage that the more complicated notion of *updatable laconic OT* is not required, and hence the [GS18a] scheme can be somewhat simplified.

Combining this construction of strong LSGS from laconic OT with our construction of succinct garbling from FE and strong LSGS, we obtain Theorem 1.

Adaptive Garbling from Semi-Adaptive LSGS. In order to construct adaptive garbling schemes, it turns out that the notion of strong LSGS does not capture the essence of the adaptive security proof. We define a notion of semi-adaptive LSGS and show that a semi-adaptive LSGS can be used to construct adaptive circuit garbling schemes. We define the notion below.

The semi-adaptive security property is associated with a sequence of simulators $(\mathsf{Sim}_1, \ldots, \mathsf{Sim}_q)$ for some polynomial $q = q(\lambda)$. As before, the output of Sim_i consists of an input-dependent component and an input-independent component. However, in this security definition, we allow

the adversary to choose the input *after* he receives the input-*independent* component of the garbled circuit from the challenger. In particular, the adversary can choose the instance as a function of the input-independent component.

Our transformation from semi-adaptive LSGS to adaptive garbling is inspired by the work of [HJO+16]. In particular, our transformation abstracts out the usage of somewhere equivocal encryption in this and other prior works. In this transformation, the size of the input-dependent component (i.e. L_{sim}) determines the size of the secret key in a somewhere equivocal encryption scheme, and hence plays a role in determining the online complexity of the adaptive garbling scheme. The online complexity of the resulting adaptively secure garbling scheme is the sum of $\mathrm{poly}(\lambda, L_{\mathsf{sim}})$ and the online complexity L_{inp} of the semi-adaptive LSGS. This can be used to recover the result of [GS18a] (as well as that of [HJO+16]).

Acknowledgements

We thank Huijia Lin and Vinod Vaikuntanathan for useful discussions. We also thank the anonymous reviewers for their helpful feedback.

References

ABSV15. Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Annual Cryptology Conference*, pages 657–677. Springer, 2015.

ACC+16. Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating ram computations with adaptive soundness and privacy. In *Theory of Cryptography Conference*, pages 3–30. Springer, 2016.

AIK04. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175, 2004.

AIK06. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

AIK10. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *International Colloquium on Automata, Languages, and Programming*, pages 152–163. Springer, 2010.

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Annual Cryptology Conference*, pages 308–326. Springer, 2015.

AJS15. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Eprint*, 730:2015, 2015.

AJS17a. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation for turing machines: Constant overhead and amortization. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 252–279, 2017.

AJS17b. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Patchable indistinguishability obfuscation: io for evolving software. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–155. Springer, 2017.

App14a. Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 162–172. Springer, 2014.

App14b. Benny Applebaum. Key-dependent message security: Generic amplification and completeness. *Journal of Cryptology*, 27(3):429–451, 2014.

AS16. Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. In *Theory of Cryptography Conference*, pages 125–153. Springer, 2016.

BGJ+16. Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 345–356. ACM, 2016.

BGL+15. Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Siddartha Telang. Succinct randomized encodings and their applications. In *STOC*, 2015.

BHHI10. Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 423–444. Springer, 2010.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796. ACM, 2012.

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 500–532. Springer, 2018.

BLSV18. Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Advances in Cryptology - EUROCRYPT 2018*, 2018.

BP15. Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 401–427. Springer, 2015.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 171–190. IEEE, 2015.

CCC+16. Yu-Chi Chen, Sherman SM Chow, Kai-Min Chung, Russell WF Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Cryptography for parallel ram from indistinguishability obfuscation. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 179–190. ACM, 2016.

CCHR16. Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Adaptive succinct garbled ram or: How to delegate your database. In *Theory of Cryptography Conference*, pages 61–90. Springer, 2016.

CDG+17. Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *Annual International Cryptology Conference*, pages 33–65. Springer, 2017.

CH16. Ran Canetti and Justin Holmgren. Fully succinct garbled ram. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 169–178. ACM, 2016.

CHJV15. Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. In *STOC*, 2015.

CLP15. Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In *Annual Cryptology Conference*, pages 287–307. Springer, 2015.

DG17. Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Annual International Cryptology Conference*, pages 537–569. Springer, 2017.

DGHM18. Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In *IACR International Workshop on Public Key Cryptography*. Springer, 2018.

DKW16. Apoorvaa Deshpande, Venkata Koppula, and Brent Waters. Constrained pseudorandom functions for unconstrained inputs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 124–153. Springer, 2016.

GGP10. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Annual Cryptology Conference*, pages 465–482. Springer, 2010.

GHL+14. Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled ram revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 405–422. Springer, 2014.

GHV10. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In *Annual Cryptology Conference*, pages 155–172. Springer, 2010.

GKP+12. Shafi Goldwasser, Yael Tauman Kalai, Raluca A Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. *IACR Cryptology ePrint Archive*, 2012:733, 2012.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Annual International Cryptology Conference*, pages 39–56. Springer, 2008.

GLOS15. Sanjam Garg, Steve Lu, Rafail Ostrovsky, and Alessandra Scafuro. Garbled ram from one-way functions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 449–458. ACM, 2015.

GP17. Sanjam Garg and Omkant Pandey. Incremental program obfuscation. In *Annual International Cryptology Conference*, pages 193–223. Springer, 2017.

GPS16. Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *Annual Cryptology Conference*, pages 579–604. Springer, 2016.

GPSZ17. Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 156–181. Springer, 2017.

GS16. Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In *Theory of Cryptography Conference*, pages 419–442. Springer, 2016.

GS17. Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round mpc from bilinear maps. *FOCS 2017*, 2017.

GS18a. Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–565. Springer, 2018.

GS18b. Sanjam Garg and Akshayaram Srinivasan. A simple construction of io for turing machines. In *Theory of Cryptography Conference*. Springer, 2018.

GS18c. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 468–499. Springer, 2018.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multiparty computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179, 2012.

HJO+16. Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In *CRYPTO*, 2016.

HLP11. Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Annual Cryptology Conference*, pages 132–150. Springer, 2011.

HW15. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 163–172. ACM, 2015.

JSW17. Zahra Jafargholi, Alessandra Scafuro, and Daniel Wichs. Adaptively indistinguishable garbled circuits. In *Theory of Cryptography Conference*, pages 40–71. Springer, 2017.

JW16. Zahra Jafargholi and Daniel Wichs. Adaptive security of yao's garbled circuits. In *TCC*, 2016.

KLW15. Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, 2015.

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

LT17. Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Annual International Cryptology Conference*, pages 630–660. Springer, 2017.

LZ17. Qipeng Liu and Mark Zhandry. Exploding obfuscation: A framework for building applications of obfuscation from polynomial hardness. *IACR Cryptology ePrint Archive*, 2017:209, 2017.

SS10. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 463–472. ACM, 2010.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

Yao82. Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.