The Landscape of Reusable Garbling

Anasuya Acharya[1], Carmit Hazay[2], and Rahul Satish[3]

[1] Aarhus University acharya@cs.au.dk
[2] Bar-Ilan University carmit.hazay@biu.ac.il
[3] IT University of Copenhagen rahs@itu.dk

Abstract. Reusability is a recurring theme in cryptography, appearing in various contexts where a one-time setup produces an encoded program that can be applied to multiple inputs. Prominent examples include indistinguishability obfuscation (iO), functional encryption (FE), laconic function evaluation (LFE), homomorphic secret-sharing (HSS), and function secret-sharing (FSS), each offering different trade-offs in efficiency and functionality. A particularly clean setting for reusability arises in garbling schemes: a garbler publishes a garbled circuit that can be evaluated on multiple inputs chosen by an evaluator. While one-time garbling has become a central and widely applicable primitive, its reusable variant has received comparatively little attention, typically studied only as a consequence of FE.

In this work, we revisit the foundations of reusable garbling and develop a framework that clarifies its relationship to other reusable primitives. We first show that reusable garbling is equivalent to a single-key private-key variant of FE, capturing exactly the guarantees required for reusability and isolating it as a primitive in its own right. This equivalence further implies a black-box separation between reusable garbling and public-key FE, establishing that reusability can be realized entirely within the private-key setting without invoking public-key mechanisms. Building on this perspective, we demonstrate direct constructions from several inherently reusable primitives, including LFE, iO, HSS, and FSS, broadening the foundations of reusable garbling and revealing how reusability naturally emerges across diverse cryptographic paradigms.

1 Introduction

Modern cryptographic applications often involve evaluating computations on data that must remain hidden. For instance, hospitals wish to run analytics on patient records without exposing them; companies train and deploy machine learning models on private user inputs; and cloud services execute outsourced computations where confidentiality is essential. A range of cryptographic primitives has been developed to enable such tasks, each with its own trade-offs in efficiency and expressiveness. One prominent example is fully homomorphic encryption (FHE), which allows arbitrary computation directly on encrypted data. A client can encrypt its inputs, publish them to a server that can run arbitrary computations without ever learning the underlying plaintexts. The client later decrypts the output to recover the result. Notably, the asymptotic complexity of this approach is optimal for most natural computations and depends only on the input and output size, making it especially appealing when the same encrypted inputs are reused across multiple computations.

Garbling schemes offer a complementary model, where the function itself is fixed in advance and compiled into an encoded representation that can later be used with encoded inputs. This approach is especially appealing when the computation is predetermined, such as in outsourced cloud services, secure machine learning, or privacy-preserving use of specialized hardware, because the function needs to be set up only once and can then be securely applied to multiple, potentially adaptively chosen, inputs.

Introduced in Yao's seminal work [73] and later formalized as a cryptographic primitive in [15], garbling schemes enable private evaluation of functions: given a function f and an input x, they produce encodings that allow an evaluator to compute $f(x)$ without learning anything beyond the output. Beyond serving as a core technique for constant-round secure computation [14, 34, 46, 59, 63], garbling has found wide applications across cryptography, including computing on encrypted data [29, 72], verifiable computing [7, 36], parallel cryptography [5, 6], software protection [15, 39, 43], code obfuscation [4, 30], functional encryption [40, 69], and key-dependent message security [3, 13].

Conceptually, a garbling scheme separates the "offline" step of encoding the function from the "online" step of encoding inputs. Early work, [44, 53, 54, 56, 63, 65, 68, 74] to name a few, focused

on reducing the size of garbled circuits, establishing optimizations based on symmetric-key related assumptions. More recent advances, such as Homomorphic Secret Sharing (HSS), have pushed these limits further, achieving constant-size encodings per gate and, in some cases, even succinct garbling; see [11, 31, 32, 47, 48, 60–62] for a partial list.

Despite this progress, traditional garbling schemes share a fundamental limitation: they are one-time use. Each garbled circuit can only be evaluated once. To rerun the same computation, the garbler must repeat the entire garbling procedure and provide new encodings along with the fresh garbling, incurring overhead proportional to the circuit size each time. This limitation implies that the communication complexity with each new computation grows with the function representation. This is not desirable in many of the scenarios described earlier, where the same function is evaluated repeatedly. Ideally, one would like to garble the function once and thereafter, for every new computation, only generate encodings for the fresh input and recover the output, with cost depending only on the input and output sizes rather than the full circuit. This paradigm captures the same spirit as the offline/online paradigm in secure multi-party computation, where heavy preprocessing enables lightweight online interaction.

This limitation naturally leads to the notion of reusable garbling. Unlike the one-time setting, a reusable scheme must protect a hidden state (possibly large) across multiple uses and remain secure even when inputs are chosen adaptively[4]. These requirements already suggest a demand for strong cryptographic objects, and early works reflect this demand. To the best of our knowledge, the idea first appeared implicitly in [41], which proposed a scheme that captured only the authenticity aspect of reusability, without providing input privacy. Shortly after, [38] gave the first formal definition of reusable garbling with full privacy guarantees and established its feasibility. Their construction proceeded by building a succinct 1-key functional encryption (FE)[5] scheme from a specialized form of attribute-based encryption (ABE)[6] together with FHE, and then showing how such a succinct 1-key FE scheme implies reusable garbling.

Since [38], all subsequent constructions of reusable garbling [17, 18, 20, 45] have followed the same route: derive it through succinct 1-key FE. A long line of works [2, 20–22, 38, 40–42, 51, 55, 57, 58, 69, 71] improved the efficiency, expressiveness, and security guarantees of FE, advancing the state of reusable garbling. This perspective has been powerful, showing that FE is sufficient for reusable garbling and tightly linking its study to other central primitives. In particular, FE itself has been realized from combinations of FHE and ABE, and, in later works, from indistinguishability obfuscation ($i\mathcal{O}$) and laconic function evaluation (LFE). These connections firmly establish feasibility under strong assumptions.

However, existing literature on reusable garbling exhibits two key limitations. Firstly, all known constructions are fundamentally routed through FE. While there exists a clean and simple black-box construction from FE to reusable garbling, this is effectively the only direct instantiation known; every other construction merely routes through it. Secondly, the underlying FE constructions themselves rely on heavy, non–black-box public-key cryptography, using primitives such as FHE, ABE, or $i\mathcal{O}$, which embed FHE decryption procedures or circuit representations within larger cryptographic transformations. For instance, the construction of 1-key FE from ABE and FHE in [38] involves computing the FHE decryption algorithm within a one-time secure garbled circuit, making non-black-box use of FHE. Another example would be the iO-based instantiation in [18] whose construction is non-black-box in (several variants of) iO, succinct randomized encodings, and one-time adaptively secure garbling schemes. As a result, even though the reduction from FE to reusable garbling is black-box, the overall construction is ultimately non–black-box in its public-key building blocks.

1.1 Our Contributions

Given that reusable garbling has been studied almost exclusively through FE and not examined in depth as a stand-alone primitive, we highlight two complementary directions that merit further

[4] Weaker notions such as selective security have been defined, but adaptive security is typically the meaningful guarantee for applications.

[5] Functional encryption (FE) is an encryption paradigm where a secret key for a function f allows decrypting a ciphertext of x to learn only $f(x)$, and nothing else. In the 1-key setting, the scheme supports just a single chosen function f, and succinctness requires ciphertexts to scale only with the input/output length (or depth of the function), not the full circuit size.

[6] ABE is a special case of FE where functions correspond to policies on attributes.

investigation. First, we revisit the relationship between FE and reusable garbling. We show that a succinct single-key *private-key* variant of FE captures precisely the security guarantees required for reusable garbling. In particular, we prove that these two notions are, in fact, equivalent.

Secondly, we ask whether reusable garbling can be realized from cryptographic primitives beyond public-key FE, ideally through direct, black-box constructions. To this end, we broaden the perspective and show that several inherently reusable primitives, such as LFE [66], iO, Functional Secret Sharing (FSS) [24], and Homomorphic Secret Sharing (HSS) [25], can serve as black-box primitives to construct reusable garbling. This expands the known landscape of constructions and illustrates that reusability can arise naturally from a range of cryptographic primitives without routing through public-key FE. They are summarized in Table 1. These results collectively broaden the theoretical foundations of reusable garbling and provide a unified view of how reusability manifests across various cryptographic paradigms.

Reference	Black-Box Building-Block	Non-Black-Box Building-Block[7]	Hardness Assumption	Garbling Size	Encoding Size	Encoding Time
[38]	ABE	selectiveGS FHE	LWE	$\lvert f\rvert$	$\lvert \mathsf{ct}_x\rvert$	$o(\lvert\mathbf{C}\rvert)$
[66]	non-succinct FE	LFE	LWE	$\lvert \mathsf{sk}_f\rvert + \lvert\mathsf{crs}\rvert$	$\lvert \mathsf{ct}_x\rvert$	$o(\lvert\mathbf{C}\rvert)$
Theorem 2	LFE	–	LWE	$\lvert\mathsf{crs}\rvert$	$\lvert \mathsf{ct}_x\rvert$	$o(\lvert\mathbf{C}\rvert)$
[18]	1-Key FE	PPRF, xiO, iO adaptiveGS	DLin, LPN sparse LPN	$o(\lvert f\rvert)$[8]	$\mathsf{poly}(\mathsf{n},\mathsf{m},\kappa)$	$o(\lvert\mathbf{C}\rvert)$
Theorem 3	iO	PPRF	DLin, LPN sparse LPN	$\lvert\mathbb{P}_{\mathcal{C}}\rvert$[9]	$\mathsf{n}+\mathsf{m}$	$O(F)$[10]
Theorem 4	iO (NC^1)	PPRF, adaptiveGS[12]	DLin, LPN sparse LPN	$\lvert\mathbb{P}_{\mathsf{Garble}}\rvert$[11]	$\kappa\mathsf{n}+\kappa\mathsf{m}$	$o(\lvert\mathbf{C}\rvert)$
Theorem 5	FSS[13]	PRF	LWE/LPN DDH/DCR	$\lvert k_f\rvert + \mathsf{poly}(\kappa)$	$\mathsf{m}+\mathsf{n}+$ $\mathsf{polylog}(\kappa)$	$o(\lvert\mathbf{C}\rvert)$
Theorem 6	HSS	–	LWE/LPN DDH/DCR	$\lvert\mathsf{ek}\rvert$	$\mathsf{m}+\lvert\mathsf{ct}_x\rvert$	$O(\lvert\mathbf{C}\rvert)$

Table 1. Table summarizing our constructions for reusable garbling schemes, their complexities, and how they relate to the state-of-the-art in prior work. Let $\mathcal{F}$ be the function family of polynomial-time circuits $\mathbf{C}$ that the garbling scheme supports. We segregate each object used in the constructions as either a 'black-box building-block' or a 'non-black-box building-block' under the respective columns. In the above, n, m, and κ denote the circuit input size, circuit output size, and the security parameter, respectively. Constructions with encoding time $o(|\mathbf{C}|)$ have sub-linear encoding in circuit size, and those with $O(|\mathbf{C}|)$ are only non-trivial when $\mathsf{n},\mathsf{m} = O(|\mathbf{C}|)$. Further, ct_x denotes a ciphertext encrypting x; sk_f, the function secret-key (in FE); ek, the evaluation key (in HSS); k_f, the function secret-share (in FSS); and crs denotes the CRS (in LFE).

Equivalence of Reusable Garbling and (Private-Key) FE. Prior work [38] has established that succinct *public-key* single-key FE implies reusable garbling through a clean black-box transformation. However, the other direction, whether reusable garbling itself can yield FE, has not been explored. Moreover, it is unclear whether we need the full power of FE. We strengthen this connection by showing that the equivalence in fact holds in the private-key setting: reusable garbling with independent input encoding and succinct single-key private-key FE are *equivalent*. Details of the construction and proofs are presented in Section 3.1.

[7] All of our constructions are only non-black-box in cryptographic objects that can be realized using minicrypt assumptions.

[8] This garbling scheme achieves a notion of succinctness wherein both the garbling time and size are sub-linear in function size.

[9] $|\mathbb{P}_{\mathcal{C}}|$ size of program obfuscating the circuit $\mathbf{C}$ to be garbled.

[10] $O(F)$ refers to the time required to evaluate the PPRF F.

[11] $|\mathbb{P}_{\mathsf{Garble}}|$ is the size of the obfuscated one-time garbling algorithm hard-coded with the circuit to be garbled.

[12] We require an adaptively secure one-time garbling scheme that has independent input encoding (Definition 7)

[13] We also require that the FSS has asymmetric share sizes and evaluation with one of the key shares must be sub-linear in the time required to compute the function.

Theorem 1 (Informal) *There exist reusable garbling schemes with independent input encoding for a function family $\mathcal{F}$ if and only if there exist succinct single-key private-key functional encryption schemes for $\mathcal{F}$.*

From Private-Key FE to Reusable Garbling. We first show that a succinct single-key private-key FE scheme can be used, in a black-box way, to construct a reusable garbling scheme. In light of [9], which implies a black-box separation between private-key FE and *any* public-key primitive, including FE, our work implies a black-box separation between reusable garbling schemes and public-key FE. Prior work [19] derives public-key encryption from private-key FE, and under additional assumptions, also iO using non-black-box techniques, and therefore does not contradict known black-box separations relevant to our setting.

From Reusable Garbling to Private-Key FE. In addition to the above, we show that any reusable garbling scheme whose garbling algorithm can be decomposed into two subroutines, one that generates the input encodings and another that produces the reusable garbling, implies a succinct single-key private-key FE scheme, based only on black-box public-key primitives. This establishes private-key FE as the minimal abstraction that precisely captures the guarantees required for reusable garbling. As a consequence, all reusable garbling constructions we present in Table 1 also yield corresponding private-key FE schemes, including function-hiding variants (from iO and FSS) and succinct-key variants (from LFE). In particular, our constructions via FSS instantiate FE for point functions and function-hiding inner-product FE under standard assumptions such as DDH and DCR, a class of schemes with applications to biometric authentication, nearest-neighbour search, and order-revealing encryption [52].

Reusable Garbling from Other Reusable Cryptographic Objects. We further study how reusable garbling schemes can be built from other cryptographic objects that allow secure function evaluation and are inherently reusable, like LFE, iO, FSS, and HSS. We present several constructions of reusable garbling schemes that make *black-box* use of the underlying public-key building blocks. We decouple these public-key objects from other, more lightweight, private-key building-blocks that some of our constructions make non-black-box use of. We start by revisiting objects such as LFE and iO that have been used in prior reusable garbling constructions, and show that they can be greatly simplified. We then show that reusable garbling can be constructed directly and in a black-box manner from both FSS and HSS, establishing new implications

Reusable Garbling from LFE. Laconic function evaluation (LFE) [66] allows compressing a large function into a short digest using a Common Reference String (crs) - a function-independent public parameter generated during setup. An evaluator uses the crs and digest to encrypt an input x, enabling the garbler to decrypt exactly $f(x)$ using a secret state. Crucially, the digest size is sub-linear in the function size. While prior work [66] constructs reusable garbling from LFE via non-succinct 1-key FE, this indirect approach relies on non-black-box techniques and additional public-key primitives.

In contrast, we adopt a direct approach and present a black-box construction of reusable garbling from LFE that uses no additional cryptographic primitives. We observe similarities between LFE and 1-key FE, which allow us to map the algorithms in an LFE scheme to those of a reusable garbling scheme. This resulting garbling scheme supports the same function family as the LFE and produces garbling sizes *sub-linear* in the function size (linear only in the depth of the function [1]) and input encoding sizes *linear* in the size of function inputs, outputs, and security parameter. In comparison, the construction in [66] yields a garbling size linear in function size and input encoding sizes polynomial in the size of function inputs, outputs, and security parameter.

Theorem 2 (Informal) *Assuming an LFE scheme for $\mathcal{F}$, there exists a reusable garbling scheme for $\mathcal{F}$ with garbling size exactly equal to the size of the public* crs *generated by the LFE setup algorithm.*

Our proofs rely on the fact that LFE is secure when the same digest is used to encrypt multiple function inputs that are adaptively chosen by an adversary, given the crs. Details can be found in Section 3.2. Implicit in this result is a direct construction of succinct private-key one-key FE from LFE, where the algorithms of the LFE scheme can be mapped onto those of the FE scheme. Such an FE scheme is succinct in both its ciphertext and function secret key size, and this asymptotically

improves over all prior work. Furthermore, our construction naturally supports scenarios where the evaluator generates the digest. If the underlying LFE is function-hiding, the garbler remains oblivious to the evaluated circuit, a unique feature closely related to the notion of oblivious garbling [10].

Reusable Garbling from iO. We then revisit the iO-based approach of [18]. Since iO is inherently reusable, an obfuscated program can be evaluated on arbitrarily many inputs, and it is a natural candidate for reusable garbling. [18] combined non-succinct iO with adaptive[14] one-time garbling, and through a sequence of transformations (to succinct iO, then to functional encryption, and finally to reusable garbling) established feasibility. Each step, however, adds assumptions, with reusable garbling appearing only as a by-product of FE. In contrast, we show how to use iO to build reusable garbling schemes in two ways:

– First, we present a construction using the direct approach from iO for general circuits (and puncturable PRFs) to a reusable garbling scheme that foregoes even the need to use a one-time garbling scheme.

Theorem 3 (Informal) *Let* $\mathcal{C}_{\mathsf{PRF}}$ *be the family of circuits containing members of any circuit class* $\mathcal{C}$ *composed with PRFs* F, F'*. Assuming* iO *for* $\mathcal{C}_{\mathsf{PRF}}$*, there exists a function-hiding reusable garbling scheme for* $\mathcal{C}$ *with garbling size bounded by the size of the obfuscated program that evaluates the composed circuit in* $\mathcal{C}_{\mathsf{PRF}}$*.*

– Next, we show that using the indirect approach allows us to make black-box use of a weaker iO primitive. That is, we use non-succinct iO and adaptive one-time garbling (captured in NC^1 and achievable from one-way functions) to yield reusable garbling for circuits, thereby establishing a direct link to iO (for NC^1) and still avoiding all intermediate steps (through randomized encoding and FE) used in prior work.

Theorem 4 (Informal) *Assuming puncturable PRFs, adaptively secure one-time garbling in* NC^1 *for* $\mathcal{C}$*, and* iO *for the garbling algorithm, there exists a function-hiding reusable garbling scheme for* $\mathcal{C}$ *with garbling size bounded by the size of the obfuscated program, which in this case obfuscates the one-time garbling algorithm with the circuit* $\mathbf{C}$ *hardcoded.*

An additional feature of these garbling schemes is that they are function-hiding. Note also that, with Theorem 1, this imply constructions of private-key succinct 1-key FE, where the function-hiding property carries over to the FE's function secret key. Details for both garbling constructions can be found in Section 4.

Reusable Garbling from FSS. Function Secret Sharing [24] is a cryptographic object that allows one to convert a secret function f into two shares that each reveal no information about the function itself. Each of these shares can be used along with a function input x to later evaluate to additive shares of the function output $f(x)$. On the theme of function-hiding reusable garbling schemes, we observe that many of the techniques used in the direct approach construction from iO can be used to build reusable garbling from FSS. We therefore present such a construction from FSS and PRFs using the direct approach, which uses FSS in a black-box way. This construction requires garbling time and size linear in the function size, and input encoding sizes are linear only in the function input and output sizes.

Theorem 5 (Informal) *Let* $\mathcal{F}_{\mathsf{PRF}}$ *be the family of functions containing members of any function class* $\mathcal{F}$ *composed with a PRF* F*. Assuming FSS for* $\mathcal{F}_{\mathsf{PRF}}$*, there exists a function-hiding reusable garbling scheme for* $\mathcal{F}$ *with garbling size scaling with the FSS key share size.*

Depending on how the FSS is instantiated, the input encoding time for this scheme differs:

– With *asymmetric FSS* (where one share is sub-linear in the function size), the resulting scheme achieves sub-linear input encoding times, meaningfully supporting the full function family of the underlying FSS.

[14] In selective security, the adversary must commit to all of its input queries before seeing the reusable garbling, whereas in adaptive security, the adversary can choose inputs adaptively after observing the reusable garbling and the results of prior queries. Adaptive security is therefore strictly stronger and typically harder to achieve.

– With *traditional FSS* (with equal share sizes), input encoding requires time linear in the function size. This remains meaningful specifically for functions computable in $\mathcal{O}(\mathsf{n}+\mathsf{m})$ time.

Details for this construction can be found in Section 5.1. By instantiating the underlying traditional FSS scheme with known constructions from standard cryptographic assumptions, our theorem directly yields several feasibility results. For instance, FSS for general circuits can be based on LWE [26, 33], while FSS for classes like NC^1 can be based on DCR [27] /DDH [25], and for log log-depth circuits on LPN [23]. Consequently, we obtain the following corollaries:

Corollary 1 (Informal) *Assuming LWE, there exists a function-hiding reusable garbling scheme with garbling size linear in the function size, for any function* $f : \{0,1\}^{\mathsf{n}} \mapsto \{0,1\}^{\mathsf{m}} \in \mathsf{P}/\mathsf{poly}$ *computable in* $O(\mathsf{n}+\mathsf{m})$ *time.*

Corollary 2 (Informal) *Assuming DCR, there exists a function-hiding reusable garbling scheme with garbling size linear in the function size, for any function* $f : \{0,1\}^{\mathsf{n}} \mapsto \{0,1\}^{\mathsf{m}} \in \mathsf{NC}^1$ *computable in* $O(\mathsf{n}+\mathsf{m})$ *time.*

Corollary 3 (Informal) *Assuming LPN, there exists a function-hiding reusable garbling scheme with garbling size linear in the function size, for any function* $f : \{0,1\}^{\mathsf{n}} \mapsto \{0,1\}^{\mathsf{m}}$ *represented by* $\mathsf{loglog}\ \mathsf{n}$*-depth circuits computable in* $O(\mathsf{n}+\mathsf{m})$ *time.*

Corollary 4 (Informal) *Assuming DDH, there exists a function-hiding reusable garbling scheme with* $1/\mathsf{poly}$ *correctness error and garbling size linear in the function size, for any function* $f : \{0,1\}^{\mathsf{n}} \mapsto \{0,1\}^{\mathsf{m}} \in \mathsf{NC}^1$ *computable in* $O(\mathsf{n}+\mathsf{m})$ *time.*

In light of recent work (see [24, 26] for a few examples) that constructs Distributed Point Functions (DPFs) from FSS, our results also enable constructing reusable garbling schemes for point functions. Therefore, we also have the following corollary:

Corollary 5 (Informal) *Assuming OWFs, there exists a function-hiding reusable garbling scheme for point functions with garbling size linear in the function size.*

Reusable Garbling from HSS. Homomorphic secret-sharing (HSS) [25] is the dual of FSS. It is a primitive that allows one to convert a secret input x into two shares, each of which reveals no information, which in turn can be used along with a function f to later evaluate to additive shares of the function output $f(x)$. In Section 5.2, we present a black-box construction of reusable garbling from adaptively secure HSS. This construction uses no additional building-blocks.

Theorem 6 (Informal) *Assuming an adaptively secure HSS scheme for any function class* $\mathcal{F} \in \mathsf{NC}^1$ *computable in* $O(\mathsf{n}+\mathsf{m})$ *time, there exists a reusable garbling scheme for* $\mathcal{F}$ *with garbling size equal to the HSS evaluation key size.*

While this scheme achieves garbling size and time sub-linear in the function size, its input encoding requires time $\mathcal{O}(\mathsf{n}+\mathsf{m})$. It is therefore meaningful specifically for functions computable within this same $\mathcal{O}(\mathsf{n}+\mathsf{m})$ time bound. We note, however, that known HSS constructions from standard assumptions (such as LWE, DDH, and DCR) currently achieve only selective security to the best of our knowledge. Consequently, fully instantiating this adaptively secure reusable garbling scheme from standard assumptions remains an interesting open problem.

Remark 1 (selectively secure reusable garbling schemes). In [45], the authors define a weaker selectively secure notion of reusable garbling, and show how to construct it using LFE, through FE. Although our constructions achieve the adaptively secure reusable garbling, we could use selectively secure variants of the building-blocks, such as HSS, private-key FE, and LFE, and one-time secure garbling schemes, underlying our constructions to achieve selectively secure reusable garbling schemes.

2 Preliminaries

General Notation. We say that a function f is *negligible*, denoted by $\mathsf{negl}(\cdot)$, if for all constants $c \geq 0$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$. We denote by κ, the computational security

parameter. We define the sets $[n] = \{1, 2, \cdots, n\}$ and $[0, n] = \{0, 1, \cdots, n\}$ for any $n \in \mathbb{N}$. Letters in bold-face denote vectors, e.g. $\mathbf{x}$ denotes a vector of elements where $\mathbf{x}[i]$ denotes the ith index of the vector, and $|\mathbf{x}|$ represents the number of elements in vector $\mathbf{x}$. Any vector $\mathbf{x}$ can thus be also represented as $\{\mathbf{x}[i]\}_{i\in[|\mathbf{x}|]}$. For any $i < j \in |\mathbf{x}|$ we denote by $\mathbf{x}[i:j]$ the substring of $\mathbf{x}$ from the ith to the jth element.

Circuits. We define a boolean circuit $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ as the following tuple: $(\mathsf{n}, \mathsf{m}, \mathsf{q}, \{A_g, B_g, C_g, f_g\}_{g\in[\mathsf{q}]})$, where $f_g \in \{\mathsf{and}, \mathsf{xor}\}$ represents the functionality of the gate, $\mathsf{n} \geq 2$ is the number of inputs, $\mathsf{m} \geq 1$ is the number of outputs, and $\mathsf{q} \geq 1$ is the number of gates. A_g, B_g denote the input wires of the gate g while C_g denotes the output wire. For a given gate $g \in \mathbf{C}$, the function $\mathsf{GateInputs}(g)$ returns the indices of all the input wires, while the function $\mathsf{GateOutputs}(g)$ outputs the index of the output wire. Additionally, the functions $\mathsf{CircInputs}(\mathbf{C}), \mathsf{CircOutputs}(\mathbf{C})$ outputs the list of all input and output gates of the circuit $\mathbf{C}$ respectively.

Definition 1 *Two ensembles of probability distributions $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are* computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every PPT distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that for every $\kappa \in \mathbb{N}$,*

$$\left| \Pr_{x\leftarrow X_\kappa}[D(1^\kappa, x) = 1] - \Pr_{y\leftarrow Y_\kappa}[D(1^\kappa, y) = 1] \right| < \epsilon(\kappa).$$

Pseudorandom Functions (PRF). We restate the definition of PRFs from [37].

Definition 2 (Pseudorandom Functions) *An l-bit function ensemble $\mathcal{F} = \{F_\kappa : \{0,1\}^\kappa \times \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}\}_{\kappa\in\mathbb{N}}$ is called pseudorandom if for all probabilistic poly-time adversary A, and all sufficiently large κ, we have:*

$$\left|\Pr[\mathsf{A}^{F_\kappa(k;\cdot)}(1^\kappa) = 1] - \Pr[\mathsf{A}^{R_\kappa(\cdot)}(1^\kappa) = 1]\right| \leq \mathsf{negl}(\kappa)$$

where $k \leftarrow \{0,1\}^\kappa$ and $R = \{R_\kappa\}_{\kappa\in\mathbb{N}}$ is the uniform l-bit function ensemble.

Definition 3 (Puncturable Pseudo-Random Functions) *Let $n(\cdot)$ and $m(\cdot)$ be polynomial functions. A Puncturable family of PRFs $\mathcal{F} = \{F_\kappa : \{0,1\}^\kappa \times \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}\}_{\kappa\in\mathbb{N}}$ is given by a triple of uniform PPT machines $(\mathsf{Key}_F, \mathsf{Puncture}_F, \mathsf{Eval}_F)$ satisfying the following syntax:*

- $\mathsf{Key}_F(1^\kappa)$ *takes as input 1^κ and outputs a key $k \leftarrow \{0,1\}^\kappa$ uniformly at random.*
- $\mathsf{Puncture}_F(k, x)$ *takes as input a key $k \in \{0,1\}^\kappa$ and an input $x \in \{0,1\}^{n(\kappa)}$ and outputs a punctured key $k_x \in \{0,1\}^\kappa$.*
- $\mathsf{Eval}_F(k, x)$ *takes as input a key (possibly punctured) $k \in \{0,1\}^\kappa$ and an input $x \in \{0,1\}^{n(\kappa)}$ and outputs $y \in \{0,1\}^{m(\kappa)}$*

We require the following correctness and security properties:

- ***Correctness:*** *For all $\kappa \in \mathbb{N}$, all $x^* \in \{0,1\}^{n(\kappa)}$, all $x \in \{0,1\}^{n(\kappa)} \setminus \{x^*\}$, all keys $k \in \{0,1\}^\kappa$, and a punctured key $k_{x^*} = \mathsf{Puncture}(k, x^*)$, we have that, $\mathsf{Eval}_F(k_{x^*}, x) = \mathsf{Eval}_F(k, x)$.*
- ***Pseudorandom at punctured point:*** *For every pair of PPT adversaries $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$, consider the following setup:*
  1. *$(x^*, \mathsf{State}_{\mathsf{A}_1}) \leftarrow \mathsf{A}_1(1^\kappa)$.*
  2. *$k \leftarrow \mathsf{Key}_F(1^\kappa)$ and $k_{x^*} \leftarrow \mathsf{Puncture}_F(k, x^*)$*
  3. *$y_0 := \mathsf{Eval}_F(k, x^*)$ and $y_1 \leftarrow \{0,1\}^{m(\kappa)}$.*

  Given this setup, we require the advantage of the PPT adversary A:

$$\mathsf{Adv}^{\mathsf{PPRF}}_{\mathsf{A}}(\kappa) := \Big|(\Pr[\mathsf{A}_2(\mathsf{State}_{\mathsf{A}_1}, k_{x^*}, y_0) = 1] - \Pr[\mathsf{A}_2(\mathsf{State}_{\mathsf{A}_1}, k_{x^*}, y_1) = 1]) - \frac{1}{2}\Big| \leq \mathsf{negl}(\kappa)$$

for some negligible function $\mathsf{negl}(\cdot)$.

For ease of notation, we write $\mathsf{Eval}_F(k, x)$ as $F_k(x)$.

Garbling Schemes. Bellare, Hoang and Rogaway in [15] formally defined garbling schemes as a cryptographic primitive. We use this definition, and restate it below.

Definition 4 (Garbling Scheme [15]) *A garbling scheme* GS *is a tuple of four polynomial-time algorithms* $\mathsf{GS} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ *with the following syntax:*

- $(\mathbf{G}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{Garble}(1^\kappa, \mathbf{C})$*: on input a security parameter (in unary) and a circuit* $\mathbf{C}$*, returns a garbling* $\mathbf{G}$*, input encoding function* $\mathbf{e}$*, and output decoding function* $\mathbf{d}$*.*
- $\mathbf{X} = \mathsf{Encode}(\mathbf{e}, \mathbf{x})$*: returns the encoding* $\mathbf{X}$ *for function input* $\mathbf{x}$*.*
- $\mathbf{Y} = \mathsf{Eval}(\mathbf{G}, \mathbf{X})$*: returns the output labels* $\mathbf{Y}$ *by evaluating* $\mathbf{G}$ *on* $\mathbf{X}$*.*
- $\{\perp, \mathbf{y}\} = \mathsf{Decode}(\mathbf{Y}, \mathbf{d})$*: returns either the failure symbol* $\perp$ *or a value* $\mathbf{y} = f(\mathbf{x})$*.*

These algorithms must satisfy the following properties:

- Correctness*: For every circuit* $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ *and input* $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$*,*

$$\Pr[\mathbf{y} = \mathbf{C}(\mathbf{x}) : (\mathbf{G}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{Garble}(\mathbf{C}),\ \mathbf{X} = \mathsf{Encode}(\mathbf{e}, \mathbf{x}),\ \mathbf{Y} = \mathsf{Eval}(\mathbf{G}, \mathbf{X}),\ \mathbf{y} = \mathsf{Decode}(\mathbf{d}, \mathbf{Y})] = 1$$

- Non-degeneracy*: For any pair* $\mathbf{C}_0, \mathbf{C}_1 : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ *with* q *gates,*

$$\{\mathbf{e}_0, \mathbf{d}_0\}_{(\mathbf{G}_0, \mathbf{e}_0, \mathbf{d}_0) \leftarrow \mathsf{Garble}(\mathbf{C}_0)} \equiv \{\mathbf{e}_1, \mathbf{d}_1\}_{(\mathbf{G}_1, \mathbf{e}_1, \mathbf{d}_1) \leftarrow \mathsf{Garble}(\mathbf{C}_1)}$$

- Projective Encoding*: For all* $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$*, all inputs* $\mathbf{x}, \mathbf{x}' \in \{0,1\}^{\mathsf{n}}$*, and* $i \in [\mathsf{n}]$*, when* $(\mathbf{G}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{Garble}(1^\kappa, \mathbf{C})$*,* $\mathbf{X} = \mathsf{Encode}(\mathbf{e}, \mathbf{x})$ *and* $\mathbf{X}' = \mathsf{Encode}(\mathbf{e}, \mathbf{x}')$*, we have that* $\mathbf{X} = (\mathbf{X_1}, \cdots, \mathbf{X_n})$ *and* $\mathbf{X}' = (\mathbf{X}'_\mathbf{1}, \cdots, \mathbf{X}'_\mathbf{n})$ *are* n *vectors such that* $|\mathbf{X_i}| = |\mathbf{X}'_\mathbf{i}|$ *and* $\mathbf{X_i} = \mathbf{X}'_\mathbf{i}$ *if* $\mathbf{x}[i] = \mathbf{x}'[i]$*.*

Garbling schemes are additionally defined to have a privacy property. For our purposes, we consider *adaptively* private garbling schemes [8] satisfying the definition given below.

Definition 5 (Adaptive Privacy by Simulation) *Let* $\mathsf{n}(\cdot)$ *and* $\mathsf{m}(\cdot)$ *be polynomial functions. Let* $\mathsf{GS} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ *be a garbling scheme (Definition 4) and* κ *a computational security parameter. We say that* GS *satisfies adaptive privacy by simulation if there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{off}}, \mathsf{Sim}_{\mathsf{on}})$ *such that any PPT adversary* A *has advantage* $< \mathsf{negl}(\kappa)$ *in the following security game:*

- ***Offline Phase:***
  - A *chooses a circuit* $\mathbf{C} : \{0,1\}^{\mathsf{n}(\kappa)} \to \{0,1\}^{\mathsf{m}(\kappa)}$ *and gives this to* C
  - C *samples a bit* $b \leftarrow \{0,1\}$
  - *if* $b = 0$*,* C *computes* $(\tilde{\mathbf{C}}, \mathbf{e}, \mathbf{d}) \leftarrow \mathsf{Garble}(1^\kappa, \mathbf{C})$ *and gives* $\tilde{\mathbf{C}}$ *to* A
  - *if* $b = 1$*,* C *computes* $(\tilde{\mathbf{C}}, \mathsf{aux}) \leftarrow \mathsf{Sim}_{\mathsf{off}}(1^\kappa, \Phi(\mathbf{C}))$ *and gives* $\tilde{\mathbf{C}}$ *to* A
- ***Online Phase:***
  - A *chooses* $\mathbf{x} \in \{0,1\}^{\mathsf{n}(\kappa)}$ *and gives it to* C
  - *if* $b = 0$*,* C *computes* $\tilde{\mathbf{X}} = \mathsf{Encode}(\mathbf{e}, \mathbf{x})$ *and gives* $(\tilde{\mathbf{X}}, \mathbf{d})$ *to* A
  - *if* $b = 1$*,* C *computes* $(\tilde{\mathbf{X}}, \mathbf{d}) \leftarrow \mathsf{Sim}_{\mathsf{on}}(\mathsf{aux}, \mathbf{C}(\mathbf{x}))$ *and gives* $(\tilde{\mathbf{X}}, \mathbf{d})$ *to* A
  - A *outputs a bit* b'

Here the advantage of the PPT adversary A *is*

$$\mathsf{Adv}_{\mathsf{A}}^{\mathsf{Priv\text{-}Sim}}(\kappa) := \left| \Pr[b' = b \mid b' \leftarrow \langle \mathsf{A}(1^\kappa), \mathsf{C}(1^\kappa) \rangle] - \frac{1}{2} \right| < \mathsf{negl}(\kappa)$$

for some negligible function $\mathsf{negl}(\cdot)$*.*

Reusability for Garbling Schemes. We restate the definition of reusable garbling schemes from [38]. We generalise this definition to account for any circuit leakage $\Phi(\mathbf{C})$ instead of just leaking $1^{|\mathbf{C}|}$. Looking ahead, we require this for some of our schemes achieves as they only give input privacy for the corresponding reusable garbling construction.

Definition 6 (Reusable Garbling for Circuits) *A Reusable Garbling Scheme* RGb *for a family of circuits* $\mathcal{C} = \{\mathbf{C}_n\}_{n\in\mathbb{N}}$*, where* $\mathbf{C}_n$ *is a set of boolean circuits taking as input* n *bits, is a tuple of PPT algorithms* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *with the following syntax:*

- $(\Gamma, \mathsf{sk}) \leftarrow \mathsf{Garble}(1^\kappa, \mathbf{C})$*: on input a security parameter* κ *in unary and a circuit* $\mathbf{C}$*, returns a garbled circuit* Γ *and input encoding function* sk*.*
- $\mathbf{X} = \mathsf{Encode}(\mathsf{sk}, \mathbf{x})$*: returns the encoding* $\mathbf{X}$ *for function input* $\mathbf{x}$*.*
- $\mathbf{C}(\mathbf{x}) = \mathsf{Eval}(\Gamma, \mathbf{X})$*: returns the output* $\mathbf{C}(\mathbf{x})$ *by evaluating* Γ *on* $\mathbf{X}$*.*

These algorithms must satisfy the following properties:

- Correctness*: For any polynomial* $n(\cdot)$ *for all sufficiently large security parameteres* κ*, for* $n = n(\kappa)$*, for all circuits* $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ *and input* $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$*,*

$$\Pr[\mathbf{y} = \mathbf{C}(\mathbf{x}) : (\Gamma, \mathsf{sk}) \leftarrow \mathsf{Garble}(\mathbf{C}),\ \mathbf{X} = \mathsf{Encode}(\mathsf{sk}, \mathbf{x}),\ \mathbf{y} = \mathsf{Eval}(\Gamma, \mathbf{X})] = 1$$

- Efficiency: *There exists a polynomial* $p = p(\kappa, \mathsf{n})$ *such that for all input sizes* n*, security parameters* κ*, for all boolean circuits* $\mathbf{C}$ *with* n *bits of input, for all* $\mathbf{x} \in \{0,1\}^{\mathsf{n}}$*,*

$$\Pr[(\Gamma, \mathsf{sk}) \leftarrow \mathsf{Garble}(1^\kappa, \mathbf{C}) : |\mathsf{sk}| \leq p(\kappa, \mathsf{n}) \wedge \mathsf{runtime}(\mathsf{Encode}(\mathsf{sk}, x)) \leq p(\kappa, \mathsf{n})] = 1$$

- Security: *Let* RGb *be a garbling scheme for a family of circuits* $\mathcal{C} = \{\mathbf{C}_i\}_{i\in\mathbb{N}}$*. For a pair of PPT algorithms* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *and a PPT simulator* $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$*, consider the following two experiments:*

$\mathsf{Exp}^{\mathsf{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)$:	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)$:
1: $(\mathbf{C}, \mathsf{State}_\mathsf{A}) \leftarrow \mathsf{A}_1(1^\kappa)$	1: $(\mathbf{C}, \mathsf{State}_\mathsf{A}) \leftarrow \mathsf{A}_1(1^\kappa)$
2: $(\mathsf{gsk}, \Gamma) \leftarrow \mathsf{Garble}(1^\kappa, \mathbf{C})$	2: $(\tilde{\Gamma}, \mathsf{State}_\mathsf{Sim}) \leftarrow \mathcal{S}_1(1^\kappa, \Phi(\mathbf{C}))$
3: $\alpha \leftarrow \mathsf{A}_2^{\mathsf{Encode}(\mathsf{gsk},\cdot)}(\mathbf{C}, \Gamma, \mathsf{State}_\mathsf{A})$	3: $\alpha \leftarrow \mathsf{A}_2^{\mathcal{O}(\cdot,\mathbf{C})[[\mathsf{State}_\mathsf{Sim}]]}(\mathbf{C}, \tilde{\Gamma}, \mathsf{State}_\mathsf{A})$
4: Output α	4: Output α

In the above, $\mathcal{O}(\cdot, \mathbf{C})[[\mathsf{State}_\mathsf{Sim}]]$ *is an oracle that on input* x *from* A_2*, runs* $\mathcal{S}_2$ *with inputs* $\mathbf{C}(x), 1^{|x|}$*, and the latest state of* Sim*; it returns the output of* $\mathcal{S}_2$ *(storing the new simulator state for the next invocation). We say that the garbling scheme* RGb *is input-private with reusability if there exists a PPT simulator* Sim *such that for all pairs of PPT adversaries* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$*, the following two distributions are computationally indistinguishable:*

$$\{\mathsf{Exp}^{real}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{Exp}^{ideal}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$$

We adapt the definition of garbling schemes with independent input encoding (see [18]) below to adaptive garbling schemes. Looking ahead, we show that this notion of reusable garbling implies private-key FE.

Definition 7 (Garbling with Independent Input Encoding) *Any garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *for a deterministic circuit class* $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ *has independent input encoding if the following holds: For every* $\kappa \in \mathbb{N}$*, and every* $\mathbf{C} \in \mathcal{C}_\kappa$*,*

- *The algorithm* Garble *on input* $(1^\kappa, \mathbf{C})$ *invokes first* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\kappa, \mathsf{par})$ *and then* $\Gamma \leftarrow \mathsf{Garble}(\mathsf{sk}, \mathbf{C})$*, where* Gen *and* Garble *are all PPT algorithms.*
- *The security condition holds w.r.t. a simulator* $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ *that given an input* $(1^\lambda, 1^{|x|}, 1^{|\mathbf{C}|}, \mathbf{C}(x))$ *invokes* $(\mathsf{State}_{\mathcal{S}_1}) \leftarrow \mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathsf{par})$*, then* $(\Gamma, \mathsf{State}_{\mathcal{S}_2}) \leftarrow \mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathcal{S}_1}, \Phi(\mathbf{C}))$ *and* $\mathbf{X} \leftarrow \mathsf{Sim}.\mathcal{S}_3(\mathsf{State}_{\mathcal{S}_2}, \mathbf{C}(\mathbf{x}), |x|)$ *where* $\mathsf{Sim}.\mathcal{S}_1$*,* $\mathsf{Sim}.\mathcal{S}_2$ *and* $\mathsf{Sim}.\mathcal{S}_3$ *runs in time* $\mathsf{poly}(\kappa, 1^{|x|})$*,* $\mathsf{poly}(\kappa, |\mathbf{C}|)$*, and* $\mathsf{poly}(\kappa, |x|)$ *respectively.*

1-Key Succinct Public-Key Functional Encryption. We now provide a definition for (single-key) succinct public-key functional encryption below as stated in [38].

Definition 8 ((1-key) Public-Key FE) *A public-key functional encryption scheme* FE *for a class of functions* $\mathcal{F} = \{\mathcal{F}_i\}_{i\in\mathbb{N}}$ *represented as boolean circuits with* n*-bit inputs is a tuple of four algorithms* $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *such that:*

- $\mathsf{FE.Setup}(1^\kappa)$ *takes as input the security parameter and outputs a master public key* fmpk *and a master secret key* fmsk*.*
- $\mathsf{FE.KeyGen}(\mathsf{fmsk}, f)$ *takes as input the master secret key* fmsk *and a function* $f \in \mathcal{F}$ *to output a key* fsk_f*.*
- $\mathsf{FE.Enc}(\mathsf{fmpk}, x)$ *takes as input the master public key* fmpk *and input* $x \in \{0,1\}^*$ *and outputs a ciphertext* c*.*
- $\mathsf{FE.Dec}(\mathsf{fsk}_f, c)$ *takes as input the secret key* fsk_f *corresponding to function* f*, and ciphertext* c *and outputs* y*.*

We want such a scheme to hold the following properties:

- ***Correctness:*** *For any polynomial* $p(.)$ *and for every sufficiently large security parameter* κ*, for* $n = p(\kappa)$ *for all* $f \in \mathcal{F}_n$ *and all* $x \in \{0,1\}^n$*, we have the following to be negligible:*

$$1 - \Pr\left[\, y = f(x) \;\middle|\; \begin{array}{r} (\mathsf{fmpk}, \mathsf{fmsk}) \leftarrow \mathsf{FE.Setup}(1^\kappa); \\ \mathsf{fsk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, f); \\ c \leftarrow \mathsf{FE.Enc}(\mathsf{fmpk}, x); \\ y := \mathsf{FE.Dec}(\mathsf{fsk}_f, c) \end{array} \right]$$

- FULL-SIM ***Security****: For every ppt adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *and a ppt simulator* $\mathsf{Sim_{FE}}$*, consider the following two experiments:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\kappa):}$	$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\kappa):}$
1: $(\mathsf{fmpk}, \mathsf{fmsk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$	1: $(\mathsf{fmpk}, \mathsf{fmsk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(f, \mathsf{State}_{\mathsf{A}_1}) \leftarrow \mathsf{A}_1(\mathsf{fmpk})$	2:$(f, \mathsf{State}_{\mathsf{A}}) \leftarrow \mathsf{A}_1(\mathsf{fmpk})$
3: $\mathsf{fsk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, f)$	3: $\mathsf{fsk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, f)$
4: $(x, \mathsf{State}_{\mathsf{A}_2}) \leftarrow \mathsf{A}_2(\mathsf{State}_{\mathsf{A}_1}, \mathsf{fsk}_f)$	4: $(x, \mathsf{State}_{\mathsf{A}_2}) \leftarrow \mathsf{A}_2(\mathsf{State}_{\mathsf{A}_1}, \mathsf{fsk}_f)$
5: $\mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fmpk}, x)$	5: $\mathsf{ct}' \leftarrow \mathsf{Sim_{FE}}(\mathsf{fmpk}, \mathsf{fsk}_f, f, f(x), 1^{\lvert x\rvert})$
6: Output $(\mathsf{State}_{\mathsf{A}_2}, \mathsf{ct})$	6: Output $(\mathsf{State}_{\mathsf{A}_2}, \mathsf{ct}')$

The scheme is said to be (single-key) FULL-SIM *secure, if there exists a ppt simulator* $\mathsf{Sim_{FE}}$ *such that for all pairs of ppt adversaries* $(\mathsf{A}_1, \mathsf{A}_2)$*, the outcomes of the two experiments are computationally indistinguishable:*

$$\{\mathsf{Exp}^{real}_{\mathsf{FE},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{Exp}^{ideal}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$$

- SEL-SIM ***Security****: The same as* FULL-SIM *security, but modify the game so that the first step consists of* A *specifying the challenge input* x *given only the security parameter.*

Remark 2. We say that a 1-key public-key FE scheme FE (Definition 8) is succinct if the size of the output of $\mathsf{FE.Enc}$ is sub-linear in function size. More precisely, the ciphertext size $|\mathsf{ct_{FE}}| \leq \mathsf{poly}(\mathsf{n}, \mathsf{m}, \mathsf{d}(f), \kappa)$ where, for a f, n and m are the input and output length respectively, κ is a security parameter and $\mathsf{d}(f)$ is the depth of a circuit $\mathbf{C}$ representing f.

Reusable Garbling using 1-Key Succinct Public-Key FE. In [38], they showed how to construct a reusable garbling scheme using any 1-key succinct FE scheme. We restate the construction in Appendix A. We refer to their result below.

Fact 1 *If there exists a (1-key) succinct Public-Key FE scheme (Definition 8) with master secret key size* $|\mathsf{fmsk}|$*, master public key size* $|\mathsf{fmpk}|$*, function secret key size* $|\mathsf{fsk}_f|$ *and ciphertext size* $|\mathsf{ct}_x|$*, then there exists a reusable garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *(Definition 6) with size of the garbled circuit* $|\Gamma| = |\mathsf{fsk}_f|$ *and input encoding size* $|\mathsf{ct}_x|$*.*

Laconic Function Evaluation. Laconic Function Evaluation (LFE) [66] is a primitive that enables a party holding a function to compute a short, reusable digest from it. Given this digest and an input, another party can produce a succinct response that allows recovery of the function's output on that input, without revealing anything beyond the result. We restate the definition from [66] for LFE below:

Definition 9 (Laconic Function Evaluation for Circuits) *A laconic function evaluation (LFE) scheme for a class of circuits* $\mathcal{C}$ *consists of the following four PPT algorithms* $(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{crsGen}(1^\kappa, \mathsf{par})$ *takes as input the security parameter* 1^κ *and circuit parameters* par *and outputs a uniformly random common random string* crs *of appropriate length.*
- $\mathsf{Compress}(\mathsf{crs}, \mathbf{C})$ *is a deterministic algorithm that takes as input the common random string* crs *and a circuit* $\mathbf{C} \in \mathcal{C}$ *and outputs a digest* $\mathsf{digest}_{\mathbf{C}}$*.*
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathbf{C}}, x)$ *takes as input the common random string* crs*, a digest* $\mathsf{digest}_{\mathbf{C}}$ *and a message* x *and outputs a ciphertext* ct_x*.*
- $\mathsf{Dec}(\mathsf{crs}, \mathbf{C}, \mathsf{ct})$ *akes as input the common random string* crs*, a circuit* $\mathbf{C} \in \mathcal{C}$*, and a ciphertext* ct_y *and outputs a message* y*.*

We require the following properties from the above algorithms:

- ***Correctness:*** *We require that for all* κ, par *and* $\mathbf{C} \in \mathcal{C}$ *with* $\mathbf{C}.\mathsf{par} = \mathsf{par}$*:*

$$\Pr\left[y = \mathbf{C}(x) \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\kappa, \mathsf{par}) \\ \mathsf{digest}_{\mathbf{C}} \leftarrow \mathsf{Compress}(\mathsf{crs}, \mathbf{C}) \\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathbf{C}}, x) \\ y \leftarrow \mathsf{Dec}(\mathsf{crs}, \mathbf{C}, \mathsf{ct}_x) \end{array} \right] = 1.$$

- ***Efficiency:*** *For the scheme to be non-trivial, we require that the size of* $\mathsf{crs}, \mathsf{digest}_{\mathbf{C}}, \mathsf{ct}$ *and the run-time of* Enc *should be at least smaller than the circuit size of* $\mathbf{C}$*.*
- ***Security:*** *We require that there exists a PPT simulator* Sim *such that for all stateful PPT adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$*, we have:*

$$\left| \Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathsf{LFE}}(1^\kappa) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{LFE}}(1^\kappa)] \right| \leq \mathsf{negl}(\kappa)$$

for the experiments $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{LFE}}(1^\kappa)$ *and* $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{LFE}}(1^\kappa)$ *defined below:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{LFE}}(1^\kappa):}$	$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{LFE}}(1^\kappa):}$
0: $\mathsf{par} \leftarrow \mathsf{A}_1(1^\kappa)$	0: $\mathsf{par} \leftarrow \mathsf{A}_1(1^\kappa)$
1: $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\kappa, \mathsf{par})$	1: $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\kappa, \mathsf{par})$
2: $(\mathsf{State}_{\mathsf{A}_1}, \mathbf{C} \in \mathcal{C}) \leftarrow \mathsf{A}_1(\mathsf{crs}) : \mathbf{C}.\mathsf{par} = \mathsf{par}$	2: $(\mathsf{State}_{\mathsf{A}_1}, \mathbf{C} \in \mathcal{C}) \leftarrow \mathsf{A}_1(\mathsf{crs}) : \mathbf{C}.\mathsf{par} = \mathsf{par}$
3: $\mathsf{digest}_{\mathbf{C}} \leftarrow \mathsf{Compress}(\mathsf{crs}, \mathbf{C})$	3: $\mathsf{digest}_{\mathbf{C}} \leftarrow \mathsf{Compress}(\mathsf{crs}, \mathbf{C})$
4: $\alpha \leftarrow \mathsf{A}_2^{\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathbf{C}}, \cdot)}(\mathsf{State}_{\mathsf{A}_1}, \mathsf{crs})$	4: $\alpha \leftarrow \mathsf{A}_2^{\mathsf{Sim}(\mathsf{crs}, \mathbf{C}, \mathsf{digest}_{\mathbf{C}}, \mathbf{C}(\cdot))}(\mathsf{State}_{\mathsf{A}_1}, \mathsf{crs})$
5: Output α	5: Output α

[66] defines a single-message adaptive query based LFE definition, and argues informally that one could extend this to support multi-message adaptive queries. This is because the ideal experiment simulator is given a correctly generated crs as an input rather than being able to sample it itself. This guarantees composability. Given several ciphertexts ct_i encrypting various inputs x_i under the same or different digests $\mathsf{digest}_{\mathbf{C}}$, all using the same crs, all of them can be simulated simultaneously and security follows via a simple hybrid argument where they can be switched from real to simulated one by one. In our constructions we therefore directly define and use the multi-message variant of LFE implied in [66].

Besides the instantiations presented in [66], recently, Abram et al. [1] constructed a rate-1, adaptively secure LFE for general circuits under the standard LWE assumption. The encoding size in their scheme scales with the input length, output length, and circuit depth. Looking ahead, we present a black-box construction of reusable garbling using only such an LFE scheme.

Succinct FE from LFE. In [66], they showed how to generically construct any 1-key FE scheme from an LFE scheme, using an underlying non-succinct FE scheme, that is achievable from PKE [40, 69]. We restate the construction in Appendix A and refer to their result below.

Fact 2 *If there exists a non-succinct FE scheme with with master secret key size* $\left|\mathsf{fmsk}'\right|$*, master public-key size* $\left|\mathsf{fmpk}'\right|$*, function private-key size* $\left|\mathsf{fsk}'_f\right|$ *and ciphertext size* $|\mathsf{ct}'_x|$ *(Definition 8), and a laconic function evaluation scheme with crs size* $|\mathsf{crs}|$*, digest size* $|\mathsf{digest}_{\mathbf{C}}|$ *for a circuit* $\mathbf{C}$*, ciphertext size* $|\mathsf{ct}_{\mathsf{LFE}}|$ *(Definition 9), then there exists a (1-key) succinct Private-Key FE scheme with* $|\mathsf{fmsk}| = |\mathsf{crs}| + \left|\mathsf{fmsk}'\right|, |\mathsf{fmpk}| = \left|\mathsf{fmpk}'\right|, |\mathsf{sk}_{\mathbf{C}}| = \left|\mathsf{fsk}'_{\mathbf{C}_{\mathsf{LFE}}}\right| + |\mathsf{crs}| + |\mathbf{C}|, |\mathsf{ct}_x| = |\mathsf{ct}'_x|$.

Indistinguishability Obfuscation. Indistinguishability Obfuscation (iO) [12, 70] guarantees that the obfuscations of two circuits are computationally indistinguishable as long as they are functionally equivalent, i.e., the output of both circuits are the same on every input.

Definition 10 (Indistinguishability obfuscator (iO) for Circuits) *A uniform PPT algorithm* iO *is called an obfuscator for polynomial-sized circuits if,*

- Completeness: *Let* $\mathsf{n}(\cdot)$ *and* $\mathsf{m}(\cdot)$ *be polynomial functions. For every* $\kappa \in \mathbb{N}$*, every polynomial-sized circuit* $\mathbf{C}_\kappa : \{0,1\}^{\mathsf{n}(\kappa)} \to \{0,1\}^{\mathsf{m}(\kappa)}$*, and every input* $x \in \{0,1\}^{\mathsf{n}(\kappa)}$*, we have that*

$$\Pr_{\tilde{\mathbf{C}} \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_\kappa)}[\tilde{\mathbf{C}}(x) = \mathbf{C}_\kappa(x)] = 1.$$

- Security: *For an adversary* A*, consider the following interactive game:*
  1. A *outputs (the description of) two circuits* $(\mathbf{C}_0, \mathbf{C}_1)$*,*
  2. A *receives an obfuscation* $\mathsf{iO}(\mathbf{C}_b)$ *of* $\mathbf{C}_b$ *(for uniformly chosen* $b \leftarrow \{0,1\}$*),*
  3. A *outputs a guess bit* b'*.*

  We say that A *is valid if it initially only outputs pairs* $(\mathbf{C}_0, \mathbf{C}_1)$ *such that* $\mathbf{C}_0$ *and* $\mathbf{C}_1$ *are functionally equivalent, and have the same size, input length, and output length. For security, we require that for every valid PPT (in* κ*)* A*, the advantage*

$$\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{A}}(\kappa) := \left| \Pr[b = b' \text{ in the above game with } \mathsf{A}] - \frac{1}{2} \right| \leq \mathsf{negl}(\kappa)$$

  for some negligible function $\mathsf{negl}(\cdot)$*.is negligible, where the probability is taken over* b*, the randomness of* iO*, and of* A*.*

iO has been shown to exist relying on three assumptions, namely, the Decisional Linear (DLin) assumption on symmetric bilinear maps, Learning Parity with Noise (LPN) over large fields, and constant-local PRGs or sparse LPN (see [49, 50, 67]). Looking ahead, we present two constructions of reusable garbling that use iO in a black-box way: one that requires iO for $\mathsf{P/poly}$, and one from iO for NC^1 and adaptively secure one-time garbling.

Functional Secret Sharing. Functional Secret Sharing (FSS) [24] is a primitive in which a function is split into shares across multiple parties. Given an input, each party evaluates its function share on the input, and the resulting partial outputs can be combined to reconstruct the value of the original function on that input. We restate the definition from [27] for the case of two-party function secret sharing.

Definition 11 (Function Secret Sharing) *A function secret sharing (FSS) scheme for a function class $\mathcal{F}$ consists of the following two PPT algorithms* $(\mathsf{Gen}, \mathsf{Eval})$*:*

- $\mathsf{Gen}(1^\kappa, f)$ *outputs a pair of keys* (k_0, k_1) *and a correction word* CW *upon receiving the security parameter and function* f*.*
- $\mathsf{Eval}(b, k_b, CW, x)$ *outputs the corresponding share of* $f(x)$ *upon receiving the party index* b*, key* k_b *and input* x*. We ignore* CW *whenever it is implicit.*

We define $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ *to be a secure FSS scheme if it satisfies the following properties:*

- ***Correctness:*** *For all functions* $f \in \mathcal{F}$*, and all* $x \in D_f$*:*

$$\Pr[\mathsf{Eval}(0, k_0, x) - \mathsf{Eval}(1, k_1, x) = f(x) : (k_0, k_1) \leftarrow \mathsf{Gen}(1^\kappa, f)] \geq 1 - \mathsf{negl}(\kappa)$$

  where D_f *is the domain of* f*.*
- ***Security:*** *Consider the following security experiment for an adversary* A *corrupting the party indexed* $z \in \{0,1\}$*:*
  1. *Adversary* A *outputs* $(f_0, f_1) \leftarrow \mathsf{A}(1^\kappa, \mathcal{F})$*.*
  2. *The challenger samples* $b \leftarrow \{0,1\}$*, computes* $(k_0, k_1, CW) \leftarrow \mathsf{Gen}(1^\kappa, f_b)$*.*
  3. *The adversary* A *outputs* $b' \leftarrow (k_z, CW)$*.*

  Let $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{FSS}}(1^\kappa)$ *be the advantage of* A *in guessing* b *as follows:*

$$\mathsf{Adv}_{\mathsf{A}}^{\mathsf{FSS}}(1^\kappa) := |\Pr[b = b'] - 1/2| \,.$$

  We say $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ *is a secure FSS scheme if* $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{FSS}}$ *is negligible for every* $b, z \in \{0,1\}$ *and every PPT adversary* A*.*

Looking ahead, we present a construction of reusable garbling using FSS and PRFs that makes black-box use of FSS.

Homomorphic Secret-Sharing. We adapt the definition of 2-party public-key homomorphic secret-sharing (HSS) from [25] to allow for adaptive queries from the adversary for inputs chosen after it is given an evaluation key.

Definition 12 (Homomorphic Secret Sharing) *Let* $\mathcal{C}$ *be a family of circuits with input space* $\{0,1\}^{\mathsf{n}}$ *and output space* $\{0,1\}^{\mathsf{m}}$*. A (2-party,* $\frac{1}{\mathsf{poly}}$*-error)* ***Homomorphic Secret-Sharing*** *(HSS) scheme for* $\mathcal{C}$*, consists of PPT algorithms* $(\mathsf{HSS.Gen}, \mathsf{HSS.Enc}, \mathsf{HSS.Eval})$ *with the following syntax:*

- $\mathsf{HSS.Gen}(1^\kappa; r)$ *: On input a security parameter* 1^κ*, the randomized key generation algorithm outputs a public key* pk *and a pair of evaluation keys* $(\mathsf{ek}_0, \mathsf{ek}_1)$*.*
- $\mathsf{HSS.Enc}(\mathsf{pk}, x)$ *: On input a public key* pk *and a secret* $x \in \{0,1\}^{\mathsf{n}}$*, the encryption algorithm outputs a ciphertext* ct_x*.*
- $\mathsf{HSS.Eval}(b, \mathsf{ek}_b, \mathsf{ct}_x, \mathbf{C}, \delta)$ *: on input a party index* $b \in \{0,1\}$*, the corresponding evaluation key* ek_b*, ciphertext* ct_x *encrypting* x*, a circuit* $\mathbf{C} \in \mathcal{C}$*, error bound* $\delta > 0$*, the homomorphic evaluation algorithm outputs* $y_b \in \{0,1\}^{\mathsf{m}}$*, constituting party* b*'s share of an output* $y \in \{0,1\}^m$*.*

The algorithms $(\mathsf{HSS.Gen}, \mathsf{HSS.Enc}, \mathsf{HSS.Eval})$ *should satisfy the following correctness and security requirements:*

- ***Correctness:*** *For every polynomial* p *there exists a negligible* ν *such that for every positive integer* κ*, input* $\mathbf{x} \in \{0,1\}^n$*, circuit* $\mathbf{C} \in \mathcal{C}$ *with input length* n*, error bound* $\delta > 0$*, where* $\frac{1}{\delta} \leq p(\kappa)$*, we have*

$$\Pr\Big[\begin{array}{r} (\mathsf{pk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow \mathsf{HSS.Gen}(1^\kappa) \,; \\ \mathsf{ct}_x \leftarrow \mathsf{HSS.Enc}(\mathsf{pk}, x) \,; \\ \forall b \in \{0,1\}, y_b \leftarrow \mathsf{HSS.Eval}(b, \mathsf{ek}_b, \mathsf{ct}_x, \mathbf{C}, \delta) \,; \\ P(\mathbf{x}) = y_0 + y_1 \end{array} \Big] \geq 1 - \delta - \nu(\kappa)$$

– ***Security:*** *For $b \in \{0,1\}$, we require that there exists a PPT simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all stateful PPT adversary A and $x \in \{0,1\}^{\mathsf{n}}$, we have:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathsf{HSS}}(1^\kappa) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{HSS}}(1^\kappa)]\right| \leq \mathsf{negl}(\kappa)$$

for the experiments $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{HSS}}(1^\kappa)$ and $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{HSS}}(1^\kappa)$ defined below:

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{HSS}}(1^\kappa):}$	$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{HSS}}(1^\kappa):}$
1: $(\mathsf{pk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow \mathsf{HSS.Gen}(1^\kappa)$	1: $(\mathsf{State}_{\mathcal{S}_1}, \mathsf{pk}, \mathsf{ek}_b) \leftarrow \mathcal{S}_1(1^\kappa, b)$
2: For $i \in [T]$:	4: For $i \in [T]$:
$x_i \leftarrow \mathsf{A}(\mathsf{ek}_b, \{\mathsf{ct}_j\}_{j\in[i-1]})$	$x_i \leftarrow \mathsf{A}(\mathsf{ek}_b, \{\mathsf{ct}_j\}_{j\in[i-1]})$
$\mathsf{ct}_i \leftarrow \mathsf{HSS.Enc}(\mathsf{pk}, x_i)$	$\mathsf{ct}_i \leftarrow \mathcal{S}_2(\mathsf{State}_{\mathcal{S}_1})$
5: Output $\mathsf{A}(\mathsf{ct}_1, \cdots, \mathsf{ct}_T)$	5: Output $\mathsf{A}(\mathsf{ct}_1, \cdots, \mathsf{ct}_T)$

There exist HSS schemes for circuits assuming LWE and for RMS programs (that captures NC^1 computation) from hardness assumptions like LPN, DDH and DCR. Looking ahead, we present a black-box construction of reusable garbling using only HSS.

3 Reusable Garbling and FE

In this section we explore relations between reusable garbling and functional encryption. It has been established in prior work [38] that (1-key) succinct public-key FE (Definition 8) implies reusable garbling (Fact 1). In the same work, [38] showed that a public-key garbling notion (where the input encoding information need not be kept private from the adversary) would imply this public-key FE definition, and also that unfortunately such a garbling notion doesn't exist. We show a similar two-way implication that a *secret-key* notion of 1-key FE is, in fact, equivalent to reusable garbling schemes (Definition 6).

In the rest of this section, in Section 3.1 we formalize the notion of private-key succinct 1-key FE (Definition 13) and prove its equivalence to reusable garbling schemes. Towards this, we show that secret-key FE is sufficient to obtain reusable garbling. Then we show that reusable garbling schemes (with independent input encoding – Definition 7) also black-box implies this notion of 1-key FE.

In Section 3.2, we go on to explore laconic function evaluation (LFE – Definition 9), a notion closely related to FE, as a reusable primitive, and present a construction of reusable garbling from LFE whose main ideas closely follow that of the construction from 1-key FE. This construction serves as a starting point to show that, unlike how it has been done in all prior work, not all constructions for reusable garbling must route through 1-key FE.

3.1 Secret-Key FE and Reusable Garbing

In this section we establish the equivalence of the following cryptographic objects: a reusable garbling scheme (Definition 6) with independent input encoding (Definition 7); and a private-key succinct 1-key FE (Definition 13). Formally, we prove the following theorem:

Theorem 1 (Equivalence of Reusable Garbling and Private-Key FE) *There exist reusable garbling schemes $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ (Definition 6) with independent input encoding (Definition 7) for the function family $\mathcal{F}$ if and only if there exist private-key succinct 1-key functional encryption schemes $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ (Definition 13) for the same function family.*

In the ensuing text, we start by formalizing the notion of a private-key succinct 1-key FE in Definition 13 below. Then towards proving this theorem, we first show through Lemma 1 that such a 1-key FE can be used in a black-box way to construct a reusable garbling scheme. Finally, through Lemma 2 we show that any reusable garbling scheme with independent input encoding can be used in a black-box way to construct private-key succinct 1-key FE.

1-Key Succinct Private-Key Functional Encryption. We now provide a simulation variant definition for (single-key) succinct private-key functional encryption below. The definition follows the same spirit as in the previous work (for example, see [16, 21, 28, 64]), but modifies it to allow for simulation security for the 1-key selectively-secure variant of private-key FE.

Definition 13 ((1-key) Private-Key FE) *A private-key functional encryption scheme* FE *for a class of functions* $\mathcal{F} = \{\mathcal{F}_i\}_{i\in\mathbb{N}}$ *represented as boolean circuits with* n*-bit inputs is a tuple of four algorithms* $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *such that:*

- $\mathsf{FE.Setup}(1^\kappa, \mathsf{par})$ *takes as input the security parameter, and function class parameters* par *and outputs a master secret key* $(\mathsf{par}, \mathsf{fmsk})$*.*
- $\mathsf{FE.KeyGen}(\mathsf{fmsk}, f)$ *takes as input the master secret key* fmsk *and a function* $f \in \mathcal{F}$ *to output a key* fsk_f*.*
- $\mathsf{FE.Enc}(\mathsf{fmsk}, x)$ *takes as input the master secret key* fmsk *and input* $x \in \{0,1\}^*$ *and outputs a ciphertext* c*.*
- $\mathsf{FE.Dec}(\mathsf{fsk}_f, c)$ *takes as input the secret key* fsk_f *corresponding to function* f*, and ciphertext* c *and outputs* y*.*

We want such a scheme to hold the following properties:

- ***Correctness:*** *For any polynomial* $p(.)$ *and for every sufficiently large security parameter* κ*, for* $n = p(\kappa)$ *for all* $f \in \mathcal{F}_n$ *and all* $x \in \{0,1\}^n$*, we have the following to be negligible:*

$$1 - \Pr\left[\, y = f(x) \;\middle|\; \begin{array}{l} \mathsf{fmsk} \leftarrow \mathsf{FE.Setup}(1^\kappa, \mathsf{par}); \\ \mathsf{fsk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, f); \\ c \leftarrow \mathsf{FE.Enc}(\mathsf{fmsk}, x); \\ y := \mathsf{FE.Dec}(\mathsf{fsk}_f, c) \end{array} \right]$$

- $\mathsf{FULL\text{-}SIM}$ ***Security****: For every ppt adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *and a ppt simulator* $\mathsf{Sim}_{\mathsf{FE}} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$*, consider the following two experiments:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\kappa):}$	$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\kappa):}$
1: $\mathsf{fmsk} \leftarrow \mathsf{FE.Setup}(1^\kappa, \mathsf{par})$	1: $\mathsf{State}_{\mathcal{S}_1} \leftarrow \mathcal{S}_1(1^\kappa, \mathsf{par})$
2: $(f, \mathsf{State}_{\mathsf{A}_1}) \leftarrow \mathsf{A}_1(\mathsf{par})$	2: $(f, \mathsf{State}_{\mathsf{A}_1}) \leftarrow \mathsf{A}_1(\mathsf{par})$
3: $\mathsf{fsk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, f)$	3: $(\mathsf{State}_{\mathcal{S}_2}, \mathsf{fsk}_f) \leftarrow \mathcal{S}_2(\mathsf{State}_{\mathcal{S}_1}, f)$
4: $\alpha \leftarrow \mathsf{A}_2^{\mathsf{FE.Enc}(\mathsf{fmsk},\cdot)}(\mathsf{State}_{\mathsf{A}_1}, \mathsf{fsk}_f)$	4: $\alpha \leftarrow \mathsf{A}_2^{\mathcal{O}(\cdot,f)[\![\mathsf{State}_{\mathcal{S}_2}]\!]}(\mathsf{State}_{\mathsf{A}_1}, \mathsf{fsk}_f)$
5: Output α	5: Output α

In the above, $\mathcal{O}(\cdot, f)[\![\mathsf{State}_{\mathsf{A}_1}]\!]$ *is an oracle that on input* x *from* A_2*, runs* $\mathcal{S}_3$ *with inputs* $\mathsf{State}_{\mathcal{S}_2}, f(x), 1^{|x|}$*, and the latest state of* $\mathsf{Sim}_{\mathsf{FE}}$*; it returns the output of* $\mathcal{S}_3$ *(storing the new simulator state for the next invocation). The scheme is said to be (single-key)* $\mathsf{FULL\text{-}SIM}$ *secure, if there exists a ppt simulator* $\mathsf{Sim}_{\mathsf{FE}}$ *such that for all pairs of ppt adversaries* $(\mathsf{A}_1, \mathsf{A}_2)$*, the outcomes of the two experiments are computationally indistinguishable:*

$$\{\mathsf{Exp}^{real}_{\mathsf{FE},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{Exp}^{ideal}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$$

Reusable Garbling from (one-key) private-key FE. Through the construction in Algorithm 1 and Lemma 1 given below, we show that reusable garbling schemes are implied by private-key succinct 1-key FE.

Lemma 1 (Secret-key FE implies Reusable Garbling) *Assuming the existence of a (1-key) succinct secret-key FE scheme (Definition 13) with master secret key size* $|\mathsf{fmsk}|$*, function secret-key size* $|\mathsf{fsk}_f|$*, and ciphertext size* $|\mathsf{ct}_x|$*, there exists a reusable garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *(Definition 6) with size of the garbled gate* $|\Gamma| = |\mathsf{fsk}_f|$ *and input encoding size* $|\mathsf{ct}_x|$*.*

Proof. We describe the proof for Lemma 1 below.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a succinct 1-key private-key FE scheme (Definition 13). Then the reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ works as follows:

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C})$:
1. Generate FE key $\mathsf{fmsk} \leftarrow \mathsf{FE.Setup}(1^\kappa, \mathbf{C}.\mathsf{par})$.
2. $\mathsf{fsk}_{\mathbf{C}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, \mathbf{C})$.
3. Output $\Gamma = \mathsf{fsk}_{\mathbf{C}}$ and $\mathsf{gsk} = \mathsf{fmsk}$.

$\mathsf{RGb.Encode}(\mathsf{gsk}, \mathbf{x}_j)$:
1. Parse $\mathsf{gsk} = \mathsf{fmsk}$.
2. Compute $\mathsf{ct}_x \leftarrow \mathsf{FE.Enc}(\mathsf{fmsk}, \mathbf{x}_j)$.
3. Output $\mathbf{X}_j := \mathsf{ct}_x$.

$\mathsf{RGb.Eval}(\Gamma, \mathbf{X}_j)$:
1. Parse $\Gamma = \mathsf{fsk}_{\mathbf{C}}$ and $\mathbf{X}_j := \mathsf{ct}_x$.
2. Compute and output $\mathsf{FE.Dec}(\mathsf{fsk}_{\mathbf{C}}, \mathsf{ct}_x)$.

Figure 1: Reusable Garbling from (one-key) private key FE

Simulation. Recall that in the security game described for Reusable Garbling in Definition 6, the offline phase requires a simulator that can produce a simulated garbled circuit. For a 1-key FE scheme FE (Definition 13), let $\mathsf{Sim}_{\mathsf{FE}} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ be the PPT simulator. The the simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for the reusable garbling scheme invoke these as subroutines. The garbling simulator first invokes $\mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_1$ and $\mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_2$ in sequence to generate the simulated garbling. Then during input encoding phase, the state output by $\mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_2$ is then used by $\mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_3$ for each invocation and its output is treated the input encoding. This simulation strategy is detailed in Algorithm 2.

.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. For a 1-key FE scheme FE (Definition 13), let $\mathsf{Sim}_{\mathsf{FE}} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ be the PPT simulator. Then the simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for the reusable garbling scheme (Algorithm 1) is as given below:

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathbf{C})$:
1. Compute $(\mathsf{State}_{\mathcal{S}_1}) \leftarrow \mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_1(1^\kappa, \mathbf{C}.\mathsf{par})$.
2. Compute $(\mathsf{State}_{\mathcal{S}_2}, \mathsf{fsk}_{\mathbf{C}}) \leftarrow \mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_2(\mathsf{State}_{\mathcal{S}_1}, \mathbf{C})$.
3. Output $\Gamma = \mathsf{fsk}_{\mathbf{C}}$ and $\mathsf{State}_{\mathsf{Sim}} = \mathsf{State}_{\mathcal{S}_2}$.

$\mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{Sim}} = \mathsf{State}_{\mathcal{S}_2}$.
3. Compute $\mathsf{ct}' \leftarrow \mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_3(\mathsf{State}_{\mathcal{S}_2}, \mathbf{C}(\mathbf{x}_j), 1^{|\mathbf{x}_j|})$.
4. Output $\mathbf{X}_j = \mathsf{ct}'$ and increment j by 1.

Figure 2: Reusable Garbling Simulation for Construction in Figure 1

It only remains to show that with respect to the simulator in Algorithm 2 above, the distributions $\{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ and $\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ (Definition 6) are indistinguishable, given that the underlying FE scheme is secure as in Definition 13.

Claim. If FE is a secret-key functional encryption scheme as in Definition 13, then

$$\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}.$$

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the real and simulated experiments with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for FE (in Definition 13) as follows:

- A_1 receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A_1 outputs $(\mathbf{C}, \mathsf{State}_{\mathsf{A}_1} = (\mathbf{C}.\mathsf{par}))$.

- A_2 receives from the challenger in the FE game the function secret key $\mathsf{fsk}_{\mathbf{C}}$.
- A_2 sends $\mathsf{fsk}_{\mathbf{C}}$ as the garbling to $\mathcal{D}$.
- On receiving input $\mathbf{x}_j$ from $\mathcal{D}$, A_2 uses the oracle access in the FE game to get encodings ct'_j and outputs it to $\mathcal{D}$.
- $\mathcal{D}$ outputs b' to A_2.
- A_2 outputs b'.

Note that the view of $\mathcal{D}$ matches the real garbling view if the challenger in the FE game used the oracle for $\mathsf{FE.Enc}$ and the simulated garbling view if the oracle used $\mathsf{Sim}_{\mathsf{FE}}$. Thus, if $\mathcal{D}$ can distinguish between these two views, it can be used by A to distinguish between the real and simulated FE encodings with the same advantage, which is non-negligible by assumption. However, since FE satisfies the security requirement in Definition 13, no such A exists, and therefore no such $\mathcal{D}$ exists.

Succinct (One-Key) Private-Key FE from Reusable Garbling. Through the construction in Algorithm 3 and Lemma 2 given below, we now show how any reusable garbling can be used to describe a private key FE scheme, thus showing an equivalence between private-key FE schemes and reusable garbling constructions.

Let $\mathsf{RGb} = (\mathsf{Setup}, \mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ be a reusable garbling scheme with independent input encoding (Definition 7). Then the succinct 1-key private-key FE scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ works as follows:

$\mathsf{FE.Setup}(1^\kappa, \mathsf{par})$:
1. Obtain $\mathsf{sk} \leftarrow \mathsf{RGb.Gen}(1^\kappa, \mathsf{par})$.
2. Output $\mathsf{fmsk} = \mathsf{sk}$.

$\mathsf{FE.KeyGen}(\mathsf{fmsk}, f)$:
1. Parse $\mathsf{fmsk} = \mathsf{sk}$.
2. Obtain $\Gamma \leftarrow \mathsf{RGb.Garble}(\mathsf{sk}, f)$.
3. Output $\mathsf{sk}_f := \Gamma$.

$\mathsf{FE.Enc}(\mathsf{fmsk}, x)$:
1. Parse $\mathsf{fmsk} = \mathsf{sk}$.
2. Obtain $X \leftarrow \mathsf{RGb.Encode}(\mathsf{sk}, x)$.
3. Output $\mathsf{ct}_x := X$.

$\mathsf{FE.Dec}(\mathsf{sk}_f, \mathsf{ct}_x)$:
1. Parse $\mathsf{sk}_f = \Gamma, \mathsf{ct}_x = X$.
2. Output $y = \mathsf{RGb.Eval}(\Gamma, X)$.

Figure 3: Secret-key FE from Reusable Garbling

Lemma 2 (Reusable Garbling implies Secret-Key FE) *Assuming the existence of a reusable garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *(Definition 6) with independent input encoding (Definition 7) with size of the garbled gate* $|\Gamma|$ *and input encoding size* $|X|$*, then there exists a (1-key) succinct secret-key FE scheme (Definition 13) with* $|\mathsf{fmsk}| = |X|, |\mathsf{sk}_f| = |\Gamma|, |\mathsf{ct}_x| = |X|$.

Proof. We describe the proof for Lemma 2 below.

Simulation. For a reusable garbling scheme RGb (Definition 6) with independent input encoding (Definition 7), let $\mathsf{Sim}_{\mathsf{RGb}} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ be the PPT simulator. We define the PPT simulation algorithms $\mathsf{Sim}_{\mathsf{FE}} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ for the 1-key FE scheme to simply invoke the respective algorithms of $\mathsf{Sim}_{\mathsf{RGb}}$. That is, for each $i \in [3]$, $\mathsf{Sim}_{\mathsf{FE}}.\mathcal{S}_i$ invokes $\mathsf{Sim}_{\mathsf{RGb}}.\mathcal{S}_i$. This simulation strategy is formalized in Algorithm 4 below.

We now show that the FE scheme is secure given that the underlying reusable garbling scheme is secure under Definitions 6 and 7.

Claim. If RGb is a reusable garbling scheme with independent input encoding (Definition 6 and Definition 7), then

$$\{\mathsf{Exp}^{\mathrm{real}}_{\mathsf{FE},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{Exp}^{\mathrm{ideal}}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$$

.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. For a reusable garbling scheme RGb (Definition 6) with independent input encoding (Definition 7), let $\mathsf{Sim}_{\mathsf{RGb}} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ be the PPT simulator. Then the simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ for the 1-key FE scheme (Algorithm 3) is as given below:

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathsf{par})$:
1. Compute $(\mathsf{State}_{\mathcal{S}_1}^{\mathsf{RGb}}) \leftarrow \mathsf{Sim}_{\mathsf{RGb}}.\mathcal{S}_1(1^\kappa, \mathsf{par})$.
2. Output $\mathsf{State}_{\mathcal{S}_1} = \mathsf{State}_{\mathcal{S}_1}^{\mathsf{RGb}}$.

$\mathsf{Sim}.\mathcal{S}_2(\mathcal{S}_1, \mathbf{C})$:
1. Parse $\mathsf{State}_{\mathcal{S}_1} = \mathsf{State}_{\mathcal{S}_1}^{\mathsf{RGb}}$.
2. Compute $(\mathsf{State}_{\mathcal{S}_2}^{\mathsf{RGb}}, \Gamma) \leftarrow \mathsf{Sim}_{\mathsf{RGb}}.\mathcal{S}_2(\mathsf{State}_{\mathcal{S}_1}^{\mathsf{RGb}}, \mathbf{C})$.
3. Output $\mathsf{fsk}_{\mathbf{C}} = \Gamma$ and $\mathsf{State}_{\mathcal{S}_2} = \mathsf{State}_{\mathcal{S}_2}^{\mathsf{RGb}}$.

$\mathsf{Sim}.\mathcal{S}_3(\mathsf{State}_{\mathcal{S}_2}, \mathbf{C}(\mathbf{x}_j), |\mathbf{x}_j|)$:
1. Parse $\mathsf{State}_{\mathcal{S}_2} = \mathsf{State}_{\mathcal{S}_2}^{\mathsf{RGb}}$.
2. Compute $\mathbf{X} \leftarrow \mathsf{Sim}_{\mathsf{RGb}}.\mathcal{S}_3(\mathsf{State}_{\mathcal{S}_2}^{\mathsf{RGb}}, \mathbf{C}(\mathbf{x}), |\mathbf{x}_j|)$.
3. Output $\mathsf{ct} = \mathbf{X_j}$.

Figure 4: Private-Key Functional Encryption Simulation for Construction in Figure 3

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the real and simulated experiments with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for RGb (in Definition 6,7) as follows:

- A_1 receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A_1 outputs $(\mathbf{C}, \mathsf{State}_{\mathsf{A}_1} = (\mathbf{C}.\mathsf{par}))$.
- A_2 receives from the challenger in the RGb game, the value Γ as the garbling.
- A_2 forwards $\mathsf{fsk}_{\mathbf{C}} = \Gamma$ to $\mathcal{D}$.
- On receiving input $\mathbf{x}_j$ from $\mathcal{D}$, A_2 uses the oracle access in the RGb game to get encodings $\mathbf{X}_j$ and outputs $\mathsf{ct}'_j = \mathbf{X}_j$ to $\mathcal{D}$.
- $\mathcal{D}$ outputs b' to A_2.
- A_2 outputs b'.

Note that the view of $\mathcal{D}$ matches that of the real experiment for the FE view if the challenger in the RGb game used real garbling and the simulated FE execution if the challenger used the simulated garbling. Thus, if $\mathcal{D}$ can distinguish between these two views, it can be used by A to distinguish between the real and simulated garbling with the same advantage, which is non-negligible by assumption. However, since RGb satisfies the security requirement in Definition 6, 7, no such A exists, and therefore no such $\mathcal{D}$ exists.

3.2 Reusable Garbling and LFE

In this section we present a direct transformation of any laconic function evaluation scheme (LFE – Definition 9) to a reusable garbling scheme (Definition 6) supporting the same function family. This construction follows the direct approach and is black-box in LFE. The high-level idea follows exactly like the transformation from public-key succinct 1-key FE to reusable garbling (Fact 1) and essentially amounts to a re-naming of the algorithms in LFE to those of the reusable garbling scheme. This construction is detailed in Algorithm 32.

Theorem 2 (LFE implies Reusable Garbling) *Assuming the existence of an LFE scheme (Definition 9) for function family $\mathcal{F}$ with crs size $|\mathsf{crs}|$, digest size $|\mathsf{digest}_{\mathbf{C}}|$ for a circuit $\mathbf{C}$, and ciphertext size $|\mathsf{ct}_{\mathsf{LFE}}|$, there exists a reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ (Definition 6) supporting $\mathcal{F}$ with size of the garbled circuit $|\Gamma| = |\mathsf{crs}|$ and input encoding size $|\mathsf{ct}_x|$.*

Proof. We describe the proof for Theorem 2 below.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $\mathsf{LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ be an LFE scheme (Definition 9). Then the reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ works as follows:

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C})$:
1. Obtain $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\kappa)$.
2. Obtain $\mathsf{digest}_{\mathbf{C}} \leftarrow \mathsf{Compress}(\mathsf{crs}, \mathbf{C})$.
3. Output $\Gamma = (\mathsf{crs})$, $\mathsf{gsk} = (\mathsf{crs}, \mathsf{digest}_{\mathbf{C}})$.

$\mathsf{RGb.Encode}(\mathsf{gsk}, x)$:
1. Parse $\mathsf{gsk} = (\mathsf{crs}, \mathsf{digest}_{\mathbf{C}})$.
2. Obtain $\mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathbf{C}}, x)$.
3. Output ct_x.

$\mathsf{RGb.Eval}(\Gamma, \mathsf{ct}_x)$:
1. Parse $\Gamma = \mathsf{crs}$.
2. Obtain $y = \mathsf{Dec}(\mathsf{crs}, \mathbf{C}, \mathsf{ct}_x)$.
3. Output y.

Figure 5: Reusable Garbling from LFE

Simulation. For an LFE scheme LFE (Definition 9), let $\mathsf{Sim}_{\mathsf{LFE}}$ be the PPT simulator. Recall that in the security game described for Reusable Garbling in Definition 6, the offline phase requires a simulator that can produce a simulated garbled circuit. In our construction, the garbled circuit is the crs for the underlying LFE, just like in the real experiment. The simulation only deviates from the real execution in the input encoding algorithm where the LFE simulator is used to output ciphertexts that are the input encoding. This simulation strategy is detailed in Algorithm 6.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. For an LFE scheme LFE (Definition 9), let $\mathsf{Sim}_{\mathsf{LFE}}$ be the PPT simulator. Then the simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for the reusable garbling scheme (Algorithm 5) is as given below:

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathbf{C}_f)$:
1. Compute $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\kappa)$.
2. Compute $\mathsf{digest}_{\mathbf{C}} \leftarrow \mathsf{Compress}(\mathsf{crs}, \mathbf{C})$.
3. Output $\Gamma = (\mathsf{crs})$ and $\mathsf{State}_{\mathsf{Sim}} = (\mathsf{crs}, \mathsf{digest}_{\mathbf{C}})$.

$\mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{Sim}} = (\mathsf{crs}, \mathsf{digest}_{\mathbf{C}})$.
3. Compute $\mathsf{ct}_j \leftarrow \mathsf{Sim}_{\mathsf{LFE}}(\mathsf{crs}, \mathsf{digest}_{\mathbf{C}}, \mathbf{C}(x))$.
4. Output $\mathbf{X}_j = \mathsf{ct}_j$ and increment j by 1.

Figure 6: Reusable Garbling Simulation for Construction in Figure 5

It only remains to show that with respect to the simulator in Algorithm 6 above, the distributions $\{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ and $\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ (Definition 6) of interactions with a PPT adversary A are indistinguishable, given that the underlying LFE scheme is secure (Definition 9).

Claim. If LFE is a laconic function evaluation scheme as in Definition 9, then

$$\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{\mathrm{c}}{\approx} \{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}.$$

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the real and simulated experiments with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}_{\mathsf{LFE}}$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for LFE (in Definition 9) as follows:

- $\mathsf{A}_{\mathsf{LFE}}$ receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- $\mathsf{A}_{\mathsf{LFE}}$ computes $\mathsf{par} = \mathbf{C}.\mathsf{par}$.
- $\mathsf{A}_{\mathsf{LFE}}$ outputs par to the challenger in the LFE game.

- $\mathsf{A}_{\mathsf{LFE}}$ receives crs from the challenger in the LFE game, that it sends to $\mathcal{D}$.
- $\mathsf{A}_{\mathsf{LFE}}$ outputs $\mathbf{C}$ to the challenger in the LFE game.
- For every input $\mathbf{x_i}$ sent by $\mathcal{D}$, $\mathsf{A}_{\mathsf{LFE}}$ forwards $\mathbf{x}_i$ to the challenger in the LFE game. It receives ct_i from the challenger and forwards it to $\mathcal{D}$.
- $\mathcal{D}$ outputs b' to $\mathsf{A}_{\mathsf{LFE}}$.
- $\mathsf{A}_{\mathsf{LFE}}$ outputs b'.

Note that the view of $\mathcal{D}$ matches the real garbling view if the challenger in the LFE game used Enc to compute ciphertexts and the simulated garbling view if it used $\mathsf{Sim}_{\mathsf{LFE}}$. Thus, if $\mathcal{D}$ can distinguish between these two views, it can be used by $\mathsf{A}_{\mathsf{LFE}}$ to distinguish between the real and simulated LFE encodings with the same advantage, which is non-negligible by assumption. However, since LFE satisfies the security requirement in Definition 9, no such $\mathsf{A}_{\mathsf{LFE}}$ exists, and therefore no such $\mathcal{D}$ exists.

4 Reusable Garbling and iO

In this section we present two constructions of reusable garbling schemes that make black-box use of indistinguishability obfuscation (iO – Definition 10).

- Our first construction, detailed in Section 4.1, follows the direct approach of constructing reusable garbling: we obfuscate the function to be garbled where its inputs and outputs are respectively unmasked and masked using outputs of a puncturable PRF (Definition 3), and use this obfuscated program as the reusable garbling. This construction makes non-black-box use of the PPRF.
- Our next construction follows the indirect approach of constructing reusable garbling schemes. It makes non-black-box use of PPRFs and a one-time adaptively secure garbling scheme (Definition 4) but has the advantage of requiring iO for a more restricted class of functions. The high-level approach here is to obfuscate the one-time *garbling algorithm* that is hard-coded with the function to be garbled. Details for this can be found in Section 4.2.

Both of these constructions of reusable garbling are function hiding and we explain these constructions in greater detail in the ensuing text.

4.1 A Direct Reusable Garbling Construction from iO

Here, we describe a generic transformation from iO to a reusable garbing scheme, assuming one-way functions. This construction makes non-black-box use of puncturable PRFs (PPRF – Definition 3). The high-level idea, as mentioned before, is to simply obfuscate the function that needs to be garbled, after masking its input and output. More precisely, for a circuit class $\mathcal{C}$ and appropriate PPRFs F, F', we define a circuit class $\mathcal{C}_{\mathsf{PRF},\mathcal{C}}$ as follows:

$$\mathcal{C}_{\mathsf{PRF},\mathcal{C}} = \{f'_{k_1,k_2}(x,i) = \mathbf{C}(x - F_{k_1}(i)) - F'_{k_2}(i)\}_{\mathbf{C}\in\mathcal{C}} \tag{1}$$

for a random choice of $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$. Our reusable garbled circuit for $\mathbf{C}$ would be the obfuscated program of the corresponding circuit in $\mathcal{C}_{\mathsf{PRF},\mathcal{C}}$. The complete construction is detailed in Algorithm 7.

Theorem 3 (iO implies Reusable Garbling) *Assuming the existence of an indistinguishability obfuscator (Definition 10) supporting $\mathcal{C}_{\mathsf{PRF},\mathcal{C}}$ with program size $\left|\mathbb{P}_{\mathcal{C}_{\mathsf{PRF},\mathcal{C}}}\right|$ and a puncturable pseudorandom functions F, F' (Definition 3), there exists a reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ (Definition 6) for the circuit class $\mathcal{C}$ with size of the garbled circuit $|\Gamma| = \left|\mathbb{P}_{\mathcal{C}_{\mathsf{PRF},\mathcal{C}}}\right|$ and input encoding size $|x|$, where x is the input to the circuit.*

Proof. We describe the proof for Theorem 3 below.

Simulation. Recall that in the security game described for Reusable Garbling in Definition 6, the offline phase requires a simulator that can produce a simulated garbled circuit. In our construction, the garbled circuit is the obfuscated program for circuit computing f'_{k_1,k_2} determined by f and the pseudo-randomness of F_{k_1} and F'_{k_2}. In the simulator, we would sample random functions r_1, r_2 instead to derive the randomness required and replace the masked input x with this random input. We would

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{n}}$ and $F' : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{m}}$ be puncturable pseudorandom functions (Definition 3). Let $\mathbf{C}_f : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let iO be an indistinguishability obfuscator (Definition 10).

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C}_f)$:
1. Sample PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$.
2. Define function $f'_{k_1,k_2}(x,i)$ as in Equation 1.
3. Compute $\mathbb{P}_{f'}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^\kappa, f'_{k_1,k_2}(\cdot,\cdot))$.
4. Output $\Gamma = \mathbb{P}_{f'}$ and $\mathsf{sk} = (k_1, k_2)$.

$\mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{sk} = k$.
3. Set $R_j^1 = F_{k_1}(j) + \mathbf{x}_j$, $R_j^2 = F'_{k_2}(j)$.
4. Output $\mathbf{X}_j = (R_j^1, R_j^2)$.

$\mathsf{RGb.Eval}(\Gamma, \mathbf{X}_j)$:
1. Parse $\Gamma = \mathbb{P}_{f'}$ and $\mathbf{X}_j = (R_j^1, R_j^2)$.
2. Compute $\mathbf{y}_j = \mathbb{P}_{f'}(R_j^1, j) + R_j^2$.
3. Output $\mathbf{y}_j$.

Figure 7: Reusable Garbling for a Boolean Circuit $\mathbf{C}$ from iO

then encode the input in the online phase in such a way that evaluation yields the function output as required.

More precisely, for a circuit class $\mathcal{C}$ and appropriate random functions r_1, r_2, we define a circuit class $\mathcal{C}_{\mathsf{Rand},\mathcal{C}}$ as follows:

$$\mathcal{C}_{\mathsf{Rand},\mathcal{C}} = \{f'_{r_1,r_2}(x,i) = \mathbf{C}(r_1(i)) - r_2(i)\}_{\mathbf{C}\in\mathcal{C}} \tag{2}$$

for a random choice of functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$. Our simulated garbled circuit for $\mathbf{C}$ would be the obfuscated program of the corresponding circuit in $\mathcal{C}_{\mathsf{Rand},\mathcal{C}}$. The simulator is detailed in Figure 8.

Let $\mathcal{D}_{\mathsf{n}}$ (resp. $\mathcal{D}_{\mathsf{m}}$) be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$ (resp. $\{0,1\}^{\mathsf{m}}$). Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let iO be an indistinguishability obfuscator (Definition 10). Let $\mathbf{C}_f : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit for a function f, where n is the number of inputs and m is the number of outputs.

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathbf{C}_f)$:
1. Sample random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
2. Define function $f'_{r_1,r_2}(x,i)$ as in Equation 2.
3. Compute $\mathbb{P}_{\mathsf{Sim}}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_{f'_{r_1,r_2}})$.
4. Output $\Gamma = \mathbb{P}$ and $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$.

$\mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}_f(\mathbf{x}_j))$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$.
3. Compute $R_j^1 = r_1(j), R_j^2 = \mathbf{C}_f(\mathbf{x}_j) - \mathbf{C}_f(r_1(j)) - r_2(j)$.
4. Output $\mathbf{X}_j = (R_j^1, R_j^2)$ and increment j by 1.

Figure 8: Reusable Garbling Simulation for Construction in Figure 7

Hybrid argument. Let T be an upper-bound on the running time of the PPT adversary A. We now describe a set of $T+1$ hybrid experiments H_i for $0 \le i \le T$, where $\mathsf{H}_0 = \{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ is the ideal execution and $\mathsf{H}_T = \{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ is the real execution. For the ith hybrid H_i, for the first i reuses, the adversary's view is the same as in the real garbling, and the simulated garbling beyond

that. In the online phase, we give out the labels corresponding to the real or simulated garbling, based on the reuse id.

More precisely, for a circuit class $\mathcal{C}$, appropriate random functions r_1, r_2 and appropriate PPRFs F, F', we define a circuit class $\mathcal{C}_{\mathsf{Hyb},\mathcal{C}}$ as follows:

$$\mathcal{C}_{\mathsf{Hyb},\mathcal{C}} = \left\{ f''_{k_1,k_2,r_1,r_2,i}(x,j) = \begin{cases} f'_{k_1,k_2}(x,j)(\text{See Eqn. 1}) & j \le i \\ f'_{r_1,r_2}(x,j)(\text{See Eqn. 2}) & j > i \end{cases} \right\}_{\mathbf{C}\in\mathcal{C}} \tag{3}$$

for a random choice of functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$ and keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$. For the rest of this section, we skip the subscripts r_1, r_2, k_1, k_2 and use the shorthand notation $f''_{\mathsf{Hyb},i}$ to denote a function from the above function family in Equation 3 and the corresponding circuit is referred to as $\mathbf{C}_{f'',\mathsf{Hyb},i}$. The offline garbled circuit in the experiment would be for $\mathbf{C}$ would be the obfuscated program of the corresponding circuit in $\mathcal{C}_{\mathsf{Hyb},\mathcal{C}}$. In the online phase, we give out the labels corresponding to the real or simulated garbling, based on the reuse id. The hybrid experiment is detailed in Figure 9.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{n}}$ and $F' : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{m}}$ be puncturable pseudorandom functions (Definition 3). Let $\mathcal{D}_{\mathsf{n}}$ (resp. $\mathcal{D}_{\mathsf{m}}$) be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$ (resp. $\{0,1\}^{\mathsf{m}}$). Let iO be an indistinguishability obfuscator (Definition 10).

Offline Phase:

1. Accept a circuit $\mathbf{C}_f$ from adversary A_1.
2. Sample PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$.
3. Sample random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
4. Define function $f''_{\mathsf{Hyb},i}$ as in Equation 3.
5. Compute $\mathbb{P}_{\mathsf{Hyb}}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_{f'',\mathsf{Hyb},i})$.
6. Output $\Gamma = \mathbb{P}_{\mathsf{Hyb}}$ and $\mathsf{State}_{\mathsf{Sim}} = (k_1, k_2, r_1, r_2)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{H}_i} = (k_1, k_2, f(0))$.
3. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
   - if $j \ge i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
4. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 9: Hybrid Experiment H_i for Construction in Figure 7

Claim. For each $i \in [T] \cup \{0\}$, hybrids $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_{i+1}$.

Proof. To show that each consecutive hybrids H_i and H_{i+1} are indistinguishable, we describe the sub-hybrids $\mathsf{H}_i^{\mathsf{iO}}$, $\mathsf{H}_i^{\mathsf{Rand}}$, H_i^{F}, and $\mathsf{H}_i^{F'}$. Then, we show that for any given i, we have the following sequence of indistinguishability:

$$\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_i^{\mathsf{iO}} \equiv \mathsf{H}_i^{\mathsf{Rand}} \overset{c}{\approx} \mathsf{H}_i^{F} \overset{c}{\approx} \mathsf{H}_i^{F'} \overset{c}{\approx} \mathsf{H}_{i+1}.$$

Hybrid $\mathsf{H}_i^{\mathsf{iO}}$*:* For this sub-hybrid $\mathsf{H}_i^{\mathsf{iO}}$, we use a function that additionally contains a hardwired values $R = (R_1, R_2)$ that is used to randomize the input and output when $j = i$, instead of computing it using the PRFs or random functions. More precisely, for a circuit class $\mathcal{C}$ and appropriate random functions r_1, r_2 and appropriate PPRFs F, F', we define a circuit class $\mathcal{C}_{\mathsf{Hyb}_{\mathsf{iO}},\mathcal{C}}$ as follows:

$$\mathcal{C}_{\mathsf{Hyb}_{\mathsf{iO}},\mathcal{C}} = \left\{ f''_{k_1,k_2,r_1,r_2,R_1,R_2,i}(x,j) = \begin{cases} f'_{k_1,k_2}(x,j)(\text{See Eqn. 1}) & j < i \\ \mathbf{C}(R_1) - R_2 & j = i \\ f'_{r_1,r_2}(x,j)(\text{See Eqn. 2}) & j > i \end{cases} \right\}_{\mathbf{C}\in\mathcal{C}} \tag{4}$$

for a random choice of functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$ and keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$. The text in blue indicates the changes made to the function from the one used for Hybrid H_i. For the

rest of this section, we skip the subscripts r_1, r_2, k_1, k_2 and use the shorthand notation $f''_{\mathsf{Hyb}_{\mathsf{iO}},R_1,R_2,i}$ to denote a function from the above function family in Equation 4 and the corresponding circuit as $\mathbf{C}_{f'',\mathsf{Hyb}_{\mathsf{iO}},R_1,R_2,i}$. In this hybrid, we just program the randomness used to the same as in f'_{r_1,r_2} but then precompute the same at $j = i$ and hardwire it instead. We describe the hybrid experiment in 10.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $F : \{0,1\}^{\kappa} \times [2^{\kappa}] \to \{0,1\}^{\mathsf{n}}$ and $F' : \{0,1\}^{\kappa} \times [2^{\kappa}] \to \{0,1\}^{\mathsf{m}}$ be puncturable pseudorandom functions (Definition 3). Let $\mathcal{D}_{\mathsf{n}}$ (resp. $\mathcal{D}_{\mathsf{m}}$) be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$ (resp. $\{0,1\}^{\mathsf{m}}$). Let iO be an indistinguishability obfuscator (Definition 10).

Offline Phase:

1. Accept a circuit $\mathbf{C}_f$ from adversary A_1.
2. Sample PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^{\kappa}), k_2 \leftarrow \mathsf{Key}_{F'}(1^{\kappa})$.
3. Sample random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
4. Set $R_1 = r_1(i), R_2 = r_2(i)$.
5. Obfuscate $\mathbb{P}_{\mathsf{Hyb},i}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^{\kappa}, \mathbf{C}_{f'',\mathsf{Hyb}_{\mathsf{iO}},R_1,R_2,i})$.
6. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i^{\mathsf{iO}}} = (k_1, k_2, r_1, r_2, R_1, R_2)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{H}_i^{\mathsf{iO}}} = (k_1, k_2, r_1, r_2, R_1, R_2)$.
3. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
   - if $j = i$, Set $\mathbf{X}_j = (R_1, \mathbf{C}_f(\mathbf{x_j}) - C_f(R_1) - R_2)$.
   - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
4. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 10: Hybrid Experiment $\mathsf{H}_i^{\mathsf{iO}}$ for Construction in Figure 7

Claim. If iO is an obfuscator as in Definition 10, then Hybrids $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_i^{\mathsf{iO}}$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments H_i and $\mathsf{H}_i^{\mathsf{iO}}$ with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary A that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for iO (in Definition 10) as follows:

- A receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A samples PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^{\kappa}), k_2 \leftarrow \mathsf{Key}_{F'}(1^{\kappa})$.
- A samples random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
- A sets $R_1 = r_1(i), R_2 = r_2(i)$.
- A outputs two circuits $\mathbf{C}_0 = \mathbf{C}_{f'',\mathsf{Hyb},i}$ (Figure 9),$\mathbf{C}_1 = \mathbf{C}_{f'',R_1,R_2,i}$ (Figure 10) to the challenger in the iO security game.
  Functional Equivalence: Note that the two programs are the same for the cases $j < i$ and $j > i$. The code only differs for the case $j = i$, where $\mathbf{C}_{f'',\mathsf{Hyb},i}$ computes randomness using r_1, r_2 on the fly and $\mathbf{C}_{f'',\mathsf{Hyb}_{\mathsf{iO}},R_1,R_2,i}$ has hardwired randomness computed from the same functions for $j = i$. Therefore, the output of the program is the same in both cases. Hence, the two circuits are functionally equivalent.
- A receives from the challenger an obfuscated program $\mathbb{H}$.
- A outputs $\mathbb{H}$ to $\mathcal{D}$. Looking ahead, we note that if $\mathbb{H}$ is an obfuscation of $\mathbf{C}_0$, then the view generated by this experiment for $\mathcal{D}$ would be the same as H_i in Figure 9. If $\mathbb{H}$ is an obfuscation of $\mathbf{C}_1$, the view would be as $\mathsf{H}_i^{\mathsf{iO}}$ in Figure 10.
- A then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
     - if $j = i$, Set $\mathbf{X}_j = (R_1, \mathbf{C}_f(\mathbf{x_j}) - f(0) - R_2)$.
     - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2, f''(0))$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.

- output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1

3. Receive a bit b from $\mathcal{D}$

– A outputs bit $b' = b$.

Note that $\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{A}}$ in the above game is the same as that of the distinguisher $\mathcal{D}$, which is non-negligible. However, since iO satisfies the security requirement in Definition 10, no such A exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}^{\mathsf{iO}}_i$.

Hybrid $\mathsf{H}^{\mathsf{Rand}}_i$*:* For this sub-hybrid $\mathsf{H}^{\mathsf{Rand}}_i$, we move from computing the function on a random input, to the masked input. We also change the output accordingly and adjust the decodings to maintain correctness. While doing so, we claim perfect indistinguishability from the view of the adversary in $\mathsf{Hyb}^{\mathsf{iO}}_i$, as we are only changing random values with one-time padded function input and output, and this is perfectly indistinguishable. More precisely, for a circuit class $\mathcal{C}$ and appropriate random functions r_1, r_2 and appropriate PPRFs F, F', we define a circuit class $\mathcal{C}_{\mathsf{Hyb},\mathcal{C},i}$ as follows:

$$\mathcal{C}_{\mathsf{Hyb},\mathcal{C},i} = \left\{ f''_{k_1,k_2,r_1,r_2,R_1,R_2,i}(x,j) = \begin{cases} f'_{k_1,k_2}(x,j)(\text{See Eqn. 1}) & j < i \\ \mathbf{C}(x - R_1) - R_2 & j = i \\ f'_{r_1,r_2}(x,j)(\text{See Eqn. 2}) & j > i \end{cases} \right\}_{\mathbf{C}\in\mathcal{C}} \tag{5}$$

for a random choice of functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$ and keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$. The text in blue indicates the changes made to the function from the one used for Hybrid H_i. For the rest of this section, we skip the subscripts r_1, r_2, k_1, k_2 and use the shorthand notation $f''_{\mathsf{Hyb},R_1,R_2,i}$ to denote a function from the above function family in Equation 5 and the corresponding circuit as $\mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i}$. In this hybrid, we just program the randomness used to the same as in f'_{r_1,r_2} but then precompute the same at $j = i$ and hardwire it instead. We describe the hybrid experiment in 11.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{n}}$ and $F' : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{m}}$ be puncturable pseudorandom functions (Definition 3). Let $\mathcal{D}_{\mathsf{n}}$ (resp. $\mathcal{D}_{\mathsf{m}}$) be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$ (resp. $\{0,1\}^{\mathsf{m}}$). Let iO be an indistinguishability obfuscator (Definition 10).

Offline Phase:

1. Accept a circuit $\mathbf{C}_f$ from adversary A_1.
2. Sample PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$.
3. Sample random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
4. Set $R_1 = r_1(i), R_2 = r_2(i)$.
5. Obfuscate $\mathbb{P}_{\mathsf{Hyb},i}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i})$.
6. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}^{\mathsf{iO}}_i} = (k_1, k_2, r_1, r_2, R_1, R_2)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{H}^{\mathsf{iO}}_i} = (k_1, k_2, r_1, r_2, R_1, R_2)$.
3. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
   - if $j = i$, Set $\mathbf{X}_j = (x + R_1, R_2)$.
   - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
4. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 11: Hybrid Experiment $\mathsf{H}^{\mathsf{Rand}}_i$ for Construction in Figure 7

Claim. Hybrids $\mathsf{H}^{\mathsf{iO}}_i \equiv \mathsf{H}^{\mathsf{Rand}}_i$.

Proof. We replaced the function evaluated at a random value and masked with a random value to a function evaluated at one-time padded inputs and one-time padded with random values as output.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $F : \{0,1\}^{\kappa} \times [2^{\kappa}] \to \{0,1\}^{\mathsf{n}}$ and $F' : \{0,1\}^{\kappa} \times [2^{\kappa}] \to \{0,1\}^{\mathsf{m}}$ be puncturable pseudorandom functions (Definition 3). Let $\mathcal{D}_{\mathsf{n}}$ (resp. $\mathcal{D}_{\mathsf{m}}$) be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$ (resp. $\{0,1\}^{\mathsf{m}}$). Let iO be an indistinguishability obfuscator (Definition 10).

Offline Phase:

1. Accept a circuit $\mathbf{C}_f$ from adversary A_1.
2. Sample PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^{\kappa}), k_2 \leftarrow \mathsf{Key}_{F'}(1^{\kappa})$.
3. Sample random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
4. Set $R_1 = F_{k_1}(i), R_2 = r_2(i)$.
5. Obfuscate $\mathbb{P}_{\mathsf{Hyb},i}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^{\kappa}, \mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i})$.
6. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i^{\mathsf{iO}}} = (k_1, k_2, r_1, r_2, R_1, R_2)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{H}_i^{\mathsf{iO}}} = (k_1, k_2, r_1, r_2, R_1, R_2)$.
3. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
   - if $j = i$, Set $\mathbf{X}_j = (x + R_1, R_2)$.
   - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
4. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 12: Hybrid Experiment H_i^F for Construction in Figure 7

Hybrid H_i^F*:* In this hybrid, we switch the programmed randomness R_1, which was generated using random function r_1 at i for $\mathsf{H}_i^{\mathsf{Rand}}$, with PRRF randomness by setting $R_1 = F_{k_1}(i)$, and using function r_2 for the randomness R_2. We describe the hybrid experiment in 12.

Claim. If F is a Puncturable PRF as in Definition 3, then hybrids $\mathsf{H}_i^{\mathsf{Rand}} \overset{c}{\approx} \mathsf{H}_i^F$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments $\mathsf{H}_i^{\mathsf{Rand}}$ and H_i^F with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}^F = (\mathsf{A}_1^F, \mathsf{A}_2^F)$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the PPRF game for F (in Definition 3) as follows:

- A_1^F receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A_1^F outputs $(x^* = i, \mathsf{State}_{\mathsf{A}_1^F} = \mathbf{C})$ to the PPRF challenger.
- A_2^F receives as part of the setup described in Definition 3, the following: $(\mathsf{State}_{\mathsf{A}_1^F} = \mathbf{C}, k_{1,i}, y)$, where either $y = F_{k_1}(i)$ or a uniformly random string in $\mathcal{R}_{\mathsf{n}}$ based on the PPRF game setup.
- A_2^F sets PRF key for F as $k_1 = k_{1,i}$.
- A_2^F sets PRF key $k_1 = k_{1,i}$ and samples PRF key $k_2 \leftarrow \mathsf{Key}_{F'}(1^{\kappa})$.
- A_2^F samples random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
- A_2^F sets $R_1 = y, R_2 = r_2(i)$.
- A_2^F computes $\mathbb{P}_{\mathsf{Hyb},i}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^{\kappa}, \mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i})$
- A_2^F outputs $\mathbb{P}_{\mathsf{Hyb},i}$ to $\mathcal{D}$. Looking ahead, we note that if y was a truly random string, then the view of $\mathcal{D}$ would be identical to an adversary's view in experiment $\mathsf{H}_i^{\mathsf{Rand}}$ (Figure 11) and if y was pseudorandom, the view of $\mathcal{D}$ would be identical to an adversary's view in H_i^F (Figure 12).
- A_2^F then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
     - if $j = i$, Set $\mathbf{X}_j = (x + R_1, R_2)$.
     - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
     - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1.
  3. Receive a bit b from $\mathcal{D}$
- A_2^F outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the experiments only differ at $j = i$. As discussed earlier, at $j = i$, notice that the randomness at i is either truly random, or pseudorandom, based on the bit b chosen in

the PPRF game for F. Thus, if $\mathcal{D}$ can distinguish between these two views, that can be used by A^F to distinguish between the a pseudorandom function output and a random function output in the PPRF game with the same advantage, which is non-negligible by assumption. However, since F satisfies the security requirement in Definition 3, no such A^F exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_i^{\mathsf{Rand}} \overset{c}{\approx} \mathsf{H}_i^F$.

Hybrid $\mathsf{H}_i^{F'}$*:* In this hybrid, we also switch the programmed randomness R_2, which was generated using random function r_2 at i for H_i^F, with PRRF randomness by setting $R_2 = F_{k_2}(i)$. We describe the hybrid experiment in 13.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{n}}$ and $F' : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{m}}$ be puncturable pseudorandom functions (Definition 3). Let $\mathcal{D}_{\mathsf{n}}$ (resp. $\mathcal{D}_{\mathsf{m}}$) be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$ (resp. $\{0,1\}^{\mathsf{m}}$). Let iO be an indistinguishability obfuscator (Definition 10).

Offline Phase:

1. Accept a circuit $\mathbf{C}_f$ from adversary A_1.
2. Sample PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$.
3. Sample random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
4. Set $R_1 = F_{k_1}(i), R_2 = F_{k_2}(i)$.
5. Obfuscate $\mathbb{P}_{\mathsf{Hyb},i}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i})$.
6. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i^{\mathsf{iO}}} = (k_1, k_2, r_1, r_2, R_1, R_2)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{State}_{\mathsf{H}_i^{\mathsf{iO}}} = (k_1, k_2, r_1, r_2, R_1, R_2)$.
3. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
   - if $j = i$, Set $\mathbf{X}_j = (x + R_1, R_2)$.
   - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
4. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 13: Hybrid Experiment $\mathsf{H}_i^{F'}$ for Construction in Figure 7

Claim. If F is a Puncturable PRF as in Definition 3, then hybrids $\mathsf{H}_i^F \overset{c}{\approx} \mathsf{H}_i^{F'}$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments H_i^F and $\mathsf{H}_i^{F'}$ with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}^{F'} = (\mathsf{A}_1^{F'}, \mathsf{A}_2^{F'})$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the PPRF game for F' (in Definition 3) as follows:

- $\mathsf{A}_1^{F'}$ receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- $\mathsf{A}_1^{F'}$ outputs $(x^* = i, \mathsf{State}_{\mathsf{A}_1^{F'}} = \mathbf{C})$ to the PPRF challenger.
- $\mathsf{A}_2^{F'}$ receives as part of the setup described in Definition 3, the following: $(\mathsf{State}_{\mathsf{A}_1^F} = \mathbf{C}, k_{1,i}, y)$, where either $y = F_{k_1}(i)$ or a uniformly random string in $\mathcal{R}_{\mathsf{n}}$ based on the PPRF game setup.
- $\mathsf{A}_2^{F'}$ sets PRF key for F as $k_2 = k_{2,i}$.
- $\mathsf{A}_2^{F'}$ sets PRF key $k_2 = k_{2,i}$ and samples PRF key $k_1 \leftarrow \mathsf{Key}_F(1^\kappa)$.
- $\mathsf{A}_2^{F'}$ samples random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
- $\mathsf{A}_2^{F'}$ sets $R_1 = F_{k_1}(i), R_2 = y$.
- $\mathsf{A}_2^{F'}$ computes $\mathbb{P}_{\mathsf{Hyb},i}(\cdot,\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i})$
- $\mathsf{A}_2^{F'}$ outputs $\mathbb{P}_{\mathsf{Hyb},i}$ to $\mathcal{D}$. Looking ahead, we note that if y was a truly random string, then the view of $\mathcal{D}$ would be identical to an adversary's view in experiment $\mathsf{H}_i^{\mathsf{Rand}}$ (Figure 11) and if y was pseudorandom, the view of $\mathcal{D}$ would be identical to an adversary's view in H_i^F (Figure 12).
- $\mathsf{A}_2^{F'}$ then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,

- if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
- if $j = i$, Set $\mathbf{X}_j = (x + R_1, R_2)$.
- if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
- output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1.

3. Receive a bit b from $\mathcal{D}$

– $\mathsf{A}_2^{F'}$ outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the experiments only differ at $j = i$. As discussed earlier, at $j = i$, notice that the randomness at i is either truly random, or pseudorandom, based on the bit b chosen in the PPRF game for F'. Thus, if $\mathcal{D}$ can distinguish between these two views, that can be used by $\mathsf{A}^{F'}$ to distinguish between the a pseudorandom function output and a random function output in the PPRF game with the same advantage, which is non-negligible by assumption. However, since F' satisfies the security requirement in Definition 3, no such $\mathsf{A}^{F'}$ exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_i^F \overset{c}{\approx} \mathsf{H}_i^{F'}$.

Claim. If iO is an obfuscator as in Definition 10, then Hybrids $\mathsf{H}_i^{F'} \overset{c}{\approx} \mathsf{H}_{i+1}$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments $\mathsf{H}_i^{F'}$ and H_{i+1} with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary A that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for iO (in Definition 10) as follows:

– A receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
– A samples PRF keys $k_1 \leftarrow \mathsf{Key}_F(1^\kappa), k_2 \leftarrow \mathsf{Key}_{F'}(1^\kappa)$.
– A samples random functions $r_1(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}, r_2(\cdot) \leftarrow \mathcal{D}_{\mathsf{m}}$.
– A sets $R_1 = F_{k_1}(i), R_2 = F_{k_2}(i)$.
– A outputs two circuits $\mathbf{C}_0 = \mathbf{C}_{f'',\mathsf{Hyb},R_1,R_2,i}$ (Figure 13),$\mathbf{C}_1 = \mathbf{C}_{f'',\mathsf{Hyb},i+1}$ (Figure 9) to the challenger in the iO security game.
Functional Equivalence: Note that the two programs are the same for the cases $j < i$ and $j > i$. The code only differs for the case $j = i$, where $\mathbf{C}_{f'',\mathsf{Hyb},i+1}$ computes randomness using PRFs $F.F'$ on the fly and $\mathbf{C}_{f'',\mathsf{Hyb}_{\mathsf{iO}},R_1,R_2,i}$ has hardwired randomness computed from the same PRFs for $j = i$. Therefore, the output of the program is the same in both cases. Hence, the two circuits are functionally equivalent.
– A receives from the challenger an obfuscated program $\mathbb{H}$.
– A outputs $\mathbb{H}$ to $\mathcal{D}$. Looking ahead, we note that if $\mathbb{H}$ is an obfuscation of $\mathbf{C}_0$, then the view generated by this experiment for $\mathcal{D}$ would be the same as $\mathsf{H}_i^{\mathbb{F}'}$ in Figure 13. If $\mathbb{H}$ is an obfuscation of $\mathbf{C}_1$, the view would be as H_{i+1} in Figure 9.
– A then runs the online phase of the experiment as follows:
1. maintain internal state j, initialized at $j = 1$
2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
   - if $j < i$, use $\mathsf{sk} = (k_1, k_2)$ and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$.
   - if $j = i$, Set $\mathbf{X}_j = (R_1, \mathbf{C}_f(\mathbf{x_j}) - f(0) - R_2)$.
   - if $j > i$, use $\mathsf{State}_{\mathsf{Sim}} = (r_1, r_2, f''(0))$ and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$.
   - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1
3. Receive a bit b from $\mathcal{D}$

– A outputs bit $b' = b$.

Note that $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{iO}}$ in the above game is the same as that of the distinguisher $\mathcal{D}$, which is non-negligible. However, since iO satisfies the security requirement in Definition 10, no such A exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_i^{\mathsf{iO}}$.

With the above sequence of hybrid arguments, we haves shown that $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_{i+1}$.

4.2 Reusable Garbling from iO for NC^1

In this section we present a construction of a reusable garbling scheme from indistinguishability obfuscation (iO – Definition 10), any adaptively secure garbling scheme for circuits (Definition 4) and puncturable pseudo-random functions (PPRF – Definition 3). Our approach deviates from all prior known constructions of reusable garbling and results in a scheme where the input encoding size is

linear only in the circuit input and output sizes. In particular, it is independent of all other function size parameters, including the depth of the circuit representation [18, 38]. Further, while there exist prior works [18, 35] that construct reusable garbling using iO, their constructions either start from a Turing machine representation of the function (not circuit), or is more complex (requiring additional public-key primitives and indirect calls to obfuscators, resulting in much larger garbling size), and would incur higher concrete costs.

The high-level idea in our reusable garbling scheme is for the garbler to obfuscate the garbling program $\mathsf{RGb}_{k,\mathbf{C}}(\cdot)$ (described in Figure 14) of a one-time adaptively secure garbling scheme and send this obfuscated program $\mathbb{P}(\cdot)$ to the evaluator. The evaluator can then invoke this for multiple reuses, creating a fresh one-time secure garbled circuit on its end every time, and then evaluating it. The role of the PPRF within the obfuscated program is to compactly represent the randomness generation for each one-time garbling, so that the size of the obfuscated program remains succinct, independent of the number of re-uses. This reusable garbling scheme is formalized in Figure 15.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a puncturable pseudorandom function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ (Definition 5). Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs.

Program $\mathsf{RGb}_{k,\mathbf{C}}(j)$:
1. Compute $r_j \leftarrow F_k(j)$
2. Compute $(\hat{\mathbf{G}}, e, d) := \mathsf{GS.Garble}(\mathbf{C}; r_j)$
3. Output $\hat{\mathbf{G}}$

Figure 14: Garbling Program $\mathsf{RGb}_{k,\mathbf{C}}(j)$ for a boolean circuit $\mathbf{C}$ with reuse id j

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a puncturable pseudorandom function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ (Definition 5). Let iO be an indistinguishability obfuscator (Definition 10). Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{RGb}_{k,\mathbf{C}}(.)$ be the garbling program as described in Figure 14.

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C})$:
1. Sample PRF key $k \leftarrow \mathsf{Key}_F(1^\kappa)$.
2. Obfuscate $\mathbb{P}(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{RGb}_{k,\mathbf{C}}(\cdot))$.
3. Output $\Gamma = \mathbb{P}$ and $\mathsf{sk} = k$.

$\mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$:
1. Maintain internal state j, initialized at $j = 1$.
2. Compute $r_j \leftarrow F_k(j)$.
3. Compute $(\hat{\mathbf{G}}, e, d) := \mathsf{GS.Garble}(\mathbf{C}; r_j)$.
4. Compute $\mathbf{X}'_j = \mathsf{GS.Encode}(e, \mathbf{x}_j)$.
5. Output $\mathbf{X}_j = \mathbf{X}'_j \cup \mathbf{d}_j$ and increment j by 1.

$\mathsf{RGb.Eval}(\Gamma, \mathbf{X}_j)$:
1. Parse $\Gamma = \mathbb{P}(\cdot)$ and $\mathbf{X}_j = \mathbf{X}'_j \cup \mathbf{d}_j$.
2. Compute $\mathbb{P}(j) = \hat{\mathbf{G}}$.
3. Compute $\mathbf{Y_j} := \mathsf{GS.Eval}(\hat{\mathbf{G}}, \mathbf{X}'_j)$.
4. Output $\mathbf{y_j} := \mathsf{GS.Decode}(\mathbf{d}_j, \mathbf{Y_j})$.

Figure 15: Reusable Garbling for a Boolean Circuit $\mathbf{C}$ from iO

Theorem 4 *Assuming the existence of an adaptively secure garbling scheme* $\mathsf{GS} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ *for polynomial sized-circuit family* $\mathcal{C} = \{\mathbf{C}_n\}_{n\in\mathbb{N}}$ *(Definition 5), an obfuscator* iO *for any polynomial sized-circuit family (Definition 10), and a puncturable pseudorandom function F (Definition 3), there is an input-private Reusable Garbling scheme(Definition 6).*

Proof. We describe the proof for Theorem 4 below.

Simulation. Recall that in the security game described for Reusable Garbling in Definition 6, the offline phase requires a simulator that can produce a simulated garbled circuit. In our construction, the garble circuit is an obfuscation of the garbling algorithm. Therefore, for the simulation in the offline phase of the reusable garbling experiment, we compute an obfuscation of the offline simulation program (Figure 16) for the given adaptively secure garbling scheme. Specifically, we construct a simulator by obfuscating the offline simulator algorithm $\mathsf{Sim}_{\mathsf{off}}(\mathbf{C}, \cdot)$ of the one-time garbling in the offline phase, and use the simulated encoding information to compute encodings of the given inputs in the online phase. This simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for reusable garbling is detailed in Figure 17.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a puncturable pseudorandom function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let iO be an indistinguishability obfuscator (Definition 10). Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{GS}.\mathsf{Sim}_{\mathsf{off}}(\mathbf{C}, \cdot)$ be the offline simulator for the adaptive garbling scheme GS, whose randomness space is also $\mathcal{R}_{\mathsf{GS}}$.

Program $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(j)$:
1. Compute $r_j \leftarrow F_k(j)$.
2. Compute $(\hat{\mathbf{G}}, \mathsf{aux}_j) := \mathsf{GS}.\mathsf{Sim}_{\mathsf{off}}(\mathbf{C}, r_j)$.
3. Output $\hat{\mathbf{G}}$.

Figure 16: Simulator $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(j)$ for a Boolean Circuit $\mathbf{C}$ with reuse id j

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a puncturable pseudorandom function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let iO be an indistinguishability obfuscator (Definition 10). Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be the simulator program as described in Figure 16. Let $\mathsf{GS}.\mathsf{Sim}_{\mathsf{off}}(\mathbf{C}, \cdot)$ and $\mathsf{GS}.\mathsf{Sim}_{\mathsf{on}}(\mathsf{aux}, \mathbf{C}(x))$ be the online and offline simulators for the adaptive garbling scheme GS, whose randomness space is also $\mathcal{R}_{\mathsf{GS}}$.

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathbf{C})$:
1. Sample PRF key $k \leftarrow \mathsf{Key}(1^\kappa)$.
2. Obfuscate $\mathbb{P}(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot))$.
3. Output $\Gamma = \mathbb{P}$ and $\mathsf{State}_{\mathsf{Sim}} = k$.

$\mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parsing $\mathsf{State}_{\mathsf{Sim}} = k$, compute $r_j \leftarrow F_k(j)$.
3. Compute $(\hat{\mathbf{G}}, \mathsf{aux}_j) := \mathsf{GS}.\mathsf{Sim}_{\mathsf{off}}(\mathbf{C}, r_j)$.
4. Compute $\mathbf{X}_j = \mathsf{GS}.\mathsf{Sim}_{\mathsf{on}}(\mathsf{aux}_j, \mathbf{C}(x_j))$.
5. Output $\mathbf{X}_j$ and increment j by 1.

Figure 17: Reusable Garbling Simulation for a Boolean Circuit $\mathbf{C}$

Hybrid argument. Let T be an upper-bound on the running time of the PPT adversary A. We now describe a set of T hybrid experiments H_i for $0 \le i \le T$, where $\mathsf{H}_0 = \{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ is the ideal execution and $\mathsf{H}_T = \{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ is the real execution. For the ith hybrid H_i, we use a program that, for the first i reuses, outputs the same thing as the real garbling program $\mathsf{Garble}_{k,\mathbf{C}}$, and outputs the output of the simulated program $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(j)$ beyond that. The input encoding in the online phase also reflects this change. This program is formalized in Figure 18 and the modified hybrid experiment in Figure 19.

Claim. For each $i \in [T] \cup \{0\}$, hybrids $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_{i+1}$.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{Garble}_{k,\mathbf{C}}(j)$ and $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be the programs as desribed in Figure 14 and Figure 16 respectively.

Program $\mathsf{H}^i_{k,\mathbf{C}}(j)$:
1. if $j < i$, compute $\mathbf{G} = \mathsf{Garble}_{k,\mathbf{C}}(j)$
2. if $j \geq i$, compute $\mathbf{G} = \mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(j)$
3. output $\mathbf{G}$

Figure 18: Hybrid Program $\mathsf{H}^i_{k,\mathbf{C}}(j)$ for a Boolean Circuit $\mathbf{C}$ with reuse id j

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a Puncturable Pseudorandom Function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling (and simulation) randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{H}^i_{k,\mathbf{C}}(\cdot)$ be the program given in Figure 18. Let $\mathsf{Garble}_{k,\mathbf{C}}(j)$ and $\mathsf{Sim}_{\mathsf{on},k,j}(\cdot)$ be the programs as described in Figure 14 and Figure 16 respectively.

Offline Phase:
1. Accept a circuit $\mathbf{C}$ from adversary A_1.
2. Sample PRF key $k \leftarrow \mathsf{Key}(1^\kappa)$.
3. Obfuscate $\mathbb{H}^i(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k,\mathbf{C}}(\cdot))$.
4. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i} = k$.

Online Phase:
1. Maintain internal state j, initialized at $j = 1$.
2. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$.
   - if $j \geq i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
3. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 19: Hybrid Experiment H_i for a Boolean Circuit $\mathbf{C}$

Proof. To show that each consecutive hybrids H_i and H_{i+1} are indistinguishable, we describe the sub-hybrids $\mathsf{H}_i^{\mathsf{iO}}$, H_i^F, $\tilde{\mathsf{H}}_i^F$ and $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$. Then, we show that for any given i, we have the following sequence of indistinguishability:

$$\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_i^{\mathsf{iO}} \overset{c}{\approx} \mathsf{H}_i^F \overset{c}{\approx} \tilde{\mathsf{H}}_i^F \overset{c}{\approx} \tilde{\mathsf{H}}_i^{\mathsf{iO}} \overset{c}{\approx} \mathsf{H}_{i+1}.$$

For this sequence of sub-hybrids $\mathsf{H}_i^{\mathsf{iO}}$ and H_i^F (resp. $\tilde{\mathsf{H}}_i^F$ and $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$), we describe a program $\mathsf{H}^i_{k,\mathbf{C},\hat{G}}(\cdot)$ that additionally contains a hardcoded simulated garbling (resp. real garbling) $\hat{G}$ that is used as the output garbling for $j = i$, instead of computing $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(i)$ (resp. $\mathsf{Garble}_{k,\mathbf{C}}(i)$) as done in H_i (resp. H_{i+1}). We describe the program in Figure 20, and highlight the differences from the program in Hybrid H_i in blue.

Let $\mathsf{Garble}_{k,\mathbf{C}}(\cdot)$ and $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be programs as given in Figure 15 and Figure 17 respectively.

$\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(j)$:
1. If $j < i$, compute $\mathbf{G} = \mathsf{Garble}_{k,\mathbf{C}}(j)$.
2. If $j > i$, compute $\mathbf{G} = \mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(j)$.
3. If $j = i$, output $\hat{\mathbf{G}}$, else output $\mathbf{G}$.

Figure 20: Hybrid Program $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(j)$ for a Boolean Circuit $\mathbf{C}$ with reuse id j

Hybrid $\mathsf{H}_i^{\mathsf{iO}}$: We describe the hybrid experiment $\mathsf{H}_i^{\mathsf{iO}}$ in Figure 21. For this experiment, we use the program $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(.)$ to hardcode a simulated circuit for $j = i$ instead of computing $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(i)$.

Claim. If iO is an obfuscator as in Definition 10, then Hybrids $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}_i^{\mathsf{iO}}$.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a Puncturable Pseudorandom Function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(\cdot)$ be the program given in Figure 20. Let $\mathsf{Garble}_{k,\mathbf{C}}(j)$ and $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be the programs as described in Figure 14 and Figure 16 respectively.

Offline Phase:
1. Accept a circuit $\mathbf{C}$ from adversary A_1.
2. Sample PRF key $k \leftarrow \mathsf{Key}_F(1^\kappa)$.
3. Set $\mathbf{G}_i := \mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(i)$.
4. Obfuscate $\mathbb{H}^i(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k,\mathbf{C},\mathbf{G}_i}(\cdot))$.
5. output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i} = k$.

Online Phase:
1. maintain internal state j, initialized at $j = 1$.
2. on input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$.
   - if $j \geq i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
3. output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 21: Hybrid Experiment $\mathsf{H}^{\mathsf{iO}}_i$ for a Boolean Circuit $\mathbf{C}$

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments H_i and $\mathsf{H}^{\mathsf{iO}}_i$ with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary A that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for iO (in Definition 10) as follows:

- A receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A samples a PRF key $k \leftarrow \{0,1\}^\kappa$.
- A sets $\mathbf{G}_i := \mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(i)$.
- A outputs two circuits $\mathbf{C}_0 = \mathsf{H}^i_{k,\mathbf{C}}$ (Figure 18),$\mathbf{C}_1 = \mathsf{H}^i_{k,\mathbf{C},\mathbf{G}_i}$ (Figure 20) to the challenger in the iO security game.
  Functional Equivalence: Note that the two programs are the same for the cases $j < i$ and $j > i$. The code only differs for the case $j = i$, where the program in $\mathsf{H}^i_{k,\mathbf{C}}$ computes a garbling and the program $\mathsf{H}^i_{k,\mathbf{C},\mathbf{G}}(\cdot)$ hardcodes a garbled circuit $\mathbf{G}$. However, the output of the program for the case $j = i$ is the same in both cases, as in the latter, $\mathbf{G}$ is just hardcoded to be the same value as in the earlier. Hence, the two circuits are functionally equivalent.
- A receives from the challenger an obfuscated program $\mathbb{H}$.
- A outputs $\mathbb{H}$ to $\mathcal{D}$. Looking ahead, we note that if $\mathbb{H}$ is an obfuscation of $\mathbf{C}_0$, then the view generated by this experiment for $\mathcal{D}$ would be the same as H_i in Figure 19. If $\mathbb{H}$ is an obfuscation of $\mathbf{C}_1$, the view would be as $\mathsf{H}^{\mathsf{iO}}_i$ in Figure 21.
- A then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$
     - if $j \geq i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$
     - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1
  3. Receive a bit b from $\mathcal{D}$
- A outputs bit $b' = b$.

Note that $\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{A}}$ in the above game is the same as that of the distinguisher $\mathcal{D}$, which is non-negligible. However, since iO satisfies the security requirement in Definition 10, no such A exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_i \overset{c}{\approx} \mathsf{H}^{\mathsf{iO}}_i$.

Hybrid H^F_i*:* We describe the hybrid experiment H^F_i in Figure 22. Recollect that in the experiment $\mathsf{H}^{\mathsf{iO}}_i$, we hardcoded a simulated garbling generated using pseudo-randomness for the instance $j = i$. For this experiment, we use the program $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(.)$ to hardcode a simulated circuit computed from a truly random string r_i instead of using a PRF to generate the randomness for the instance $j = i$.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a Puncturable Pseudorandom Function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling (and simulation) randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{H}^i_{k,\mathbf{C},\mathbf{G}}(\cdot)$ be the program given in Figure 20. Let $\mathsf{Garble}_{k,\mathbf{C}}(j)$ and $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be the programs as described in Figure 14 and Figure 16 respectively. Let $\mathsf{GS.Sim}_{\mathsf{off},k}(\mathbf{C},\cdot)$ and $\mathsf{GS.Sim}_{\mathsf{on}}(\mathsf{aux}, \mathbf{C}(x))$ be the online and offline simulators for the adaptive garbling scheme GS.

Offline Phase:

1. Accept a circuit $\mathbf{C}$ from adversary A_1.
2. Sample PRF key $k \leftarrow \mathsf{Key}(1^\kappa)$ and $r_i \leftarrow \mathcal{R}_{\mathsf{GS}}$.
3. Set $(\mathbf{G}_i, \mathsf{aux}_i) := \mathsf{GS.Sim}_{\mathsf{off}}(\mathbf{C}, r_i)$.
4. Obfuscate $\mathbb{H}^i(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k,\mathbf{C},\mathbf{G}_i}(\cdot))$.
5. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i} = (k, \mathsf{aux}_i)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$.
   - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
   - if $j = i$, use aux_i to compute $\mathbf{X}_j := \mathsf{GS.Sim}_{\mathsf{on}}(\mathsf{aux}_i, \mathbf{C}(\mathbf{x}_j))$.
3. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 22: Hybrid Experiment H^F_i for a Boolean Circuit $\mathbf{C}$

Claim. If F is a Puncturable PRF as in Definition 3, then hybrids $\mathsf{H}^{\mathsf{iO}}_i \overset{c}{\approx} \mathsf{H}^F_i$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments $\mathsf{H}^{\mathsf{iO}}_i$ and H^F_i with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}^F = (\mathsf{A}^F_1, \mathsf{A}^F_2)$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the PPRF game for F (in Definition 3) as follows:

- A^F_1 receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A^F_1 outputs $(x^* = i, \mathsf{State}_{\mathsf{A}^F_1} = \mathbf{C})$ to the PPRF challenger.
- A^F_2 receives as part of the setup described in Definition 3, the following: $(\mathsf{State}_{\mathsf{A}^F_1} = \mathbf{C}, k_i, y)$, where either $y = F_k(i)$ or a uniformly random string in $\mathcal{R}_{\mathsf{GS}}$ based on the PPRF game setup.
- A^F_2 computes $(\mathbf{G}_i, \mathsf{aux}_i) := \mathsf{GS.Sim}_{\mathsf{off}}(\mathbf{C}, y)$.
- A^F_2 computes $\mathbb{H}^i(.) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k_i,\mathbf{C},G_i})$.
- A^F_2 outputs $\mathbb{H}^i$ to $\mathcal{D}$. Looking ahead, we note that if y was a truly random string, then the view of $\mathcal{D}$ would be identical to an adversary's view in experiment H^F_i (Figure 22) and if y was pseudorandom, the view of $\mathcal{D}$ would be identical to an adversary's view in $\mathsf{H}^{\mathsf{iO}}_i$ (Figure 21)
- A^F_2 then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$.
     - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
     - if $j = i$, use aux_i to compute $\mathbf{X}_j := \mathsf{GS.Sim}_{\mathsf{on}}(\mathsf{aux}_i, \mathbf{C}(\mathbf{x}_j))$.
     - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1.
  3. Receive a bit b from $\mathcal{D}$
- A^F_2 outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the experiments only differ at $j = i$. As discussed earlier, at $j = i$, notice that the programmed garbling at i is either a simulated garbling with randomness derived from a truly random string, or a simulated garbling with randomness derived from a pseudorandom string, based on the bit b chosen in the PPRF game for F. Thus, if $\mathcal{D}$ can distinguish between these two views, that can be used by A^F to distinguish between the a pseudorandom function output and a random function output in the PPRF game with the same advantage, which is non-negligible by assumption. However, since F satisfies the security requirement in Definition 3, no such A^F exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}^{\mathsf{iO}}_i \overset{c}{\approx} \mathsf{H}^F_i$.

Hybrid $\tilde{\mathsf{H}}_i^F$*:* We describe the hybrid experiment $\tilde{\mathsf{H}}_i^F$ in Figure 23. For this experiment, we use the program $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(.)$ to hardcode a real garbled circuit computed from a truly random string r_i instead of a simulated garbled circuit computed from a truly random string r_i for the instance $j = i$.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a Puncturable Pseudorandom Function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling (and simulation) randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{H}^i_{k,\mathbf{C},\mathbf{G}}(\cdot)$ be the program given in Figure 20. Let $\mathsf{Garble}_{k,\mathbf{C}}(j)$ and $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be the programs as described in Figure 14 and Figure 16 respectively. Let $\mathsf{GS}.\mathsf{Sim}_{\mathsf{off},k}(\mathbf{C},\cdot)$ be the offline simulator for the adaptive garbling scheme GS.

Offline Phase:

1. Accept a circuit $\mathbf{C}$ from adversary A_1.
2. Sample PRF key $k \leftarrow \mathsf{Key}(1^\kappa)$ and $r_i \leftarrow \mathcal{R}_{\mathsf{GS}}$.
3. Set $(\mathbf{G}_i, e_i, d_i) := \mathsf{GS}.\mathsf{Garble}(\mathbf{C}, r_i)$.
4. Obfuscate $\mathbb{H}^i(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k,\mathbf{C},\mathbf{G}_i}(\cdot))$.
5. Output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i} = (k, e_i, d_i)$.

Online Phase:

1. Maintain internal state j, initialized at $j = 1$.
2. On input $\mathbf{x}_j$ from adversary A_2,
   - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb}.\mathsf{Encode}_j(k, \mathbf{x}_j)$.
   - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
   - if $j = i$, use e_i and d_i to compute $\mathbf{X}_j := \mathsf{GS}.\mathsf{Encode}(e_i, \mathbf{x}_j) \cup d_i$.
3. Output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 23: Hybrid Experiment $\tilde{\mathsf{H}}_i^F$ for a Boolean Circuit $\mathbf{C}$

Claim. If $\mathsf{GS} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ is an adaptively secure garbling scheme for polynomial sized-circuit family $\mathcal{C} = \{\mathbf{C}_n\}_{n\in\mathbb{N}}$ as in Definition 5, then hybrids $\mathsf{H}_i^F \overset{c}{\approx} \tilde{\mathsf{H}}_i^F$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments H_i^F and $\tilde{\mathsf{H}}_i^F$ with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary A that can use $\mathcal{D}$ in a black-box way, to distinguish between the real and simulated execution in the interactive game for GS (in Definition 4) as follows:

- A receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A sends $\mathbf{C}$ to C, the challenger in the GS adaptive privacy game.
- A receives from C, the garbling $\mathbf{G}^*$.
- A samples a PRF key $k \leftarrow \{0,1\}^\kappa$.
- A computes $\mathbb{H}^i(.) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k,\mathbf{C},\mathbf{G}^*})$.
- A outputs $\mathbb{H}^i$ to $\mathcal{D}$. Looking ahead, we note that if $\mathbf{G}^*$ was a real garbling (C had sampled $b = 0$), then $\mathsf{H}^i_{k,\mathbf{C},\mathbf{G}^*}$ outputs the same as in the experiment $\tilde{\mathsf{H}}_i^F$ (in Figure 23). Similarly, if $\mathbf{G}^*$ was the output of the simulation (C had sampled $b = 1$), then $\mathsf{H}^i_{k,\mathbf{C},\mathbf{G}^*}$ outputs the same as in experiment H_i^F (in Figure 22).
- A then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb}.\mathsf{Encode}_j(k, \mathbf{x}_j)$.
     - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
     - if $j = i$, A gives $\mathbf{x}_j$ to C, obtains $(\mathbf{X}, d)$ from C and sets $\mathbf{X}_j = \mathbf{X} \cup d$.
     - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1
  3. Receive a bit b from $\mathcal{D}$
- A outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the experiments only differ at $j = i$. At $j = i$, notice that the obfuscated program $\mathbb{H}^i$ outputs either the real garbling or the simulated garbling based on the bit b chosen by

C in the $\mathsf{Priv\text{-}Sim}$ game for GS. Thus, if $\mathcal{D}$ can distinguish between these two views, that can be used by A to distinguish between the real and simulated garbling in the $\mathsf{Priv\text{-}Sim}$ game with the same advantage, which is non-negligible by assumption. However, since GS satisfies the security requirement in Definition 4, no such A exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_i^F \overset{c}{\approx} \tilde{\mathsf{H}}_i^F$.

Hybrid $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$*:* We describe the hybrid experiment $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$ in Figure 24. For this experiment, we use the program $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(.)$ to hardcode a real garbled circuit computed from a pseudorandom function output rather than a truly random string for the instance $j = i$.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \mathcal{R}_{\mathsf{GS}}$ be a Puncturable Pseudorandom Function (Definition 3), where $\mathcal{R}_{\mathsf{GS}}$ is the space of garbling (and simulation) randomness needed for the adaptive garbling scheme $\mathsf{GS} := (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$. Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit, where n is the number of inputs and m is the number of outputs. Let $\mathsf{H}^i_{k,\mathbf{C},\hat{\mathbf{G}}}(\cdot)$ be the program given in Figure 20. Let $\mathsf{Garble}_{k,\mathbf{C}}(j)$ and $\mathsf{Sim}_{\mathsf{off},k,\mathbf{C}}(\cdot)$ be the programs as described in Figure 14 and Figure 16 respectively.

Offline Phase:

1. Accept a circuit $\mathbf{C}$ from adversary A_1.
2. Sample PRF key $k \leftarrow \mathsf{Key}(1^\kappa)$.
3. Set $\mathbf{G}_i := \mathsf{Garble}_{k,\mathbf{C}}(i)$.
4. Obfuscate $\mathbb{H}^i(\cdot) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k,\mathbf{C},\mathbf{G}_i}(\cdot))$.
5. output $\Gamma = \mathbb{H}^i$ and $\mathsf{State}_{\mathsf{H}_i} = k$.

Online Phase:

1. maintain internal state j, initialized at $j = 1$.
2. on input $\mathbf{x}_j$ from adversary A_2,
   - if $j \leq i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$.
   - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
3. output $\mathbf{X_j}$ to the adversary and increment j by 1.

Figure 24: Hybrid Experiment $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$ for a Boolean Circuit $\mathbf{C}$

Claim. If F is a Puncturable PRF as in Definition 3, then hybrids $\tilde{\mathsf{H}}_i^F \overset{c}{\approx} \tilde{\mathsf{H}}_i^{\mathsf{iO}}$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments $\tilde{\mathsf{H}}_i^F$ and $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$ with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}^F = (\mathsf{A}_1^F, \mathsf{A}_2^F)$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the PPRF game for F (in Definition 3) as follows:

- A_1^F receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- A_1^F outputs $(x^* = i, \mathsf{State}_{\mathsf{A}_1^F} = \mathbf{C})$ to the PPRF challenger.
- A_2^F receives as part of the setup described in Definition 3, the following: $(\mathsf{State}_{\mathsf{A}_1^F} = \mathbf{C}, k_i, y)$, where either $y = F_k(i)$ or a uniformly random string in $\mathcal{R}_{\mathsf{GS}}$ based on the PPRF game setup.
- A_2^F computes $(\mathbf{G}_i, e_i, d_i) := \mathsf{GS.Garble}(\mathbf{C}, y)$.
- A_2^F computes $\mathbb{H}^i(.) \leftarrow \mathsf{iO}(1^\kappa, \mathsf{H}^i_{k_i,\mathbf{C},G_i})$.
- A_2^F outputs $\mathbb{H}^i$ to $\mathcal{D}$. Looking ahead, we note that if y was a truly random string, then the view of $\mathcal{D}$ would be identical to an adversary's view in experiment $\tilde{\mathsf{H}}_i^F$ (Figure 23) and if y was pseudorandom, the view of $\mathcal{D}$ would be identical to an adversary's view in $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$ (Figure 24)
- A_2^F then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j < i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$.
     - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$.
     - if $j = i$, use e_i and d_i to compute $\mathbf{X}_j := \mathsf{GS.Encode}(e_i, \mathbf{x}_j) \cup d_i$.
     - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1.
  3. Receive a bit b from $\mathcal{D}$

– A_2^F outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the experiments only differ at $j = i$. As discussed earlier, at $j = i$, notice that the programmed garbling at i is either a simulated garbling with randomness derived from a truly random string, or a simulated garbling with randomness derived from a pseudorandom string, based on the bit b chosen in the PPRF game for F. Thus, if $\mathcal{D}$ can distinguish between these two views, that can be used by A^F to distinguish between the a pseudorandom function output and a random function output in the PPRF game with the same advantage, which is non-negligible by assumption. However, since F satisfies the security requirement in Definition 3, no such A^F exists, and therefore no such $\mathcal{D}$ exists. Thus, $\tilde{\mathsf{H}}_i^{\mathsf{iO}} \overset{c}{\approx} \tilde{\mathsf{H}}_i^F$.

Claim. If iO is an obfuscator as in Definition 10, then Hybrids $\tilde{\mathsf{H}}_i^{\mathsf{iO}} \overset{c}{\approx} \mathsf{H}_{i+1}$.

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments $\tilde{\mathsf{H}}_i^{\mathsf{iO}}$ and H_{i+1} with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary A that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive game for iO (in Definition 10) as follows:

– A receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
– A samples a PRF key $k \leftarrow \{0,1\}^\kappa$.
– A outputs two circuits $\mathbf{C}_0 = \tilde{\mathsf{H}}^i_{k,\mathbf{C},\mathbf{G}}$ (Figure 20) and $\mathbf{C}_1 = \mathsf{H}^{i+1}_{k,\mathbf{C}}$ (Figure 18) to the challenger in the iO game.
  Functional Equivalence: Note that the two programs are the same for the cases $j < i$ and $j > i$. The code only differs for the case $j = i$. However, the output of the program for the case $j = i$ is the same in both cases, as in the latter, it is just hardcoded to be the same value as in the earlier. Hence, the two circuits are functionally equivalent.
– A receives from the challenger an obfuscation $\mathbb{H}$.
– A outputs $\mathbb{H}$ to $\mathcal{D}$. Looking ahead, we note that if $\mathbb{H}$ is an obfuscation of $\mathbf{C}_0$, then the view generated by this experiment for $\mathcal{D}$ would be the same as in Figure 19. If $\mathbb{H}$ is an obfuscation of $\mathbf{C}_1$, the view would be as in Figure 24.
– A then runs the online phase of the experiment as follows:
  1. maintain internal state j, initialized at $j = 1$
  2. on input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - if $j \leq i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{RGb.Encode}_j(k, \mathbf{x}_j)$
     - if $j > i$, use PRF key k and internal state j to invoke $\mathbf{X}_j = \mathsf{Sim}.\mathcal{S}_{2,j}(k, \mathbf{C}(\mathbf{x}_j))$
     - output $\mathbf{X_j}$ to $\mathcal{D}$ and increment j by 1
  3. Receive a bit b from $\mathcal{D}$
– A outputs bit $b' = b$.

Note that $\mathsf{Adv}^{\mathsf{iO}}_{\mathsf{A}}$ in the above game is the same as that of the distinguisher $\mathcal{D}$, which is non-negligible. However, since iO satisfies the security requirement in Definition 10, no such A exists, and therefore no such $\mathcal{D}$ exists. Thus, $\tilde{\mathsf{H}}_i^{\mathsf{iO}} \overset{c}{\approx} \mathsf{H}_{i+1}$.

This concludes the proof.

Remark 3. We note that the above construction runs $\mathsf{GS.Garble}$ inside $\mathsf{RGb.Encode}$ which makes the encoding run in time $\mathcal{O}(|\mathbf{C}|)$. This is not necessary, and one could use any garbling scheme that has independent input encoding as stated in Definition 7. This would allow the encoding to just run the encoding algorithm with the generated secret key, allowing for encoding time proportional to the input and output length of the circuit.

5 Reusable Garbling from Secret-Sharing Primitives

In this section we study the relations between reusable garbling (Definition 6) and cryptographic primitives like function secret-sharing (FSS – Definition 11) and homomorphic secret-sharing (HSS – Definition 12). In doing so, we present two constructions of reusable garbling schemes:

– Our first construction, detailed in Section 5.1, uses FSS in a black-box way to construct a reusable garbling scheme, with the help of pseudorandom functions (PRFs – Definition 2).
– Our next construction, detailed in Section 5.2, constructs reusable garbling directly from HSS.

Both our constructions follow the direct approach for constructing reusable garbling schemes, are simple and black-box in the underlying secret-sharing scheme.

5.1 Reusable Garbling from FSS

In this section, we present a transformation from any function secret-sharing scheme (FSS – Definition 11) to a reusable garbling scheme (Definition 6), with the help of a pseudorandom function (PRF – Definition 2). The high-level idea for the construction is to secret-share the function to be garbled, send one share as the reusable garbling and use the other as input encoding information. The resulting reusable garbling scheme therefore requires input encoding time proportional to the size of one function secret-share. So, our garbling scheme is non-trivial when:

- FSS produces asymmetric function secret-shares (one function secret-share is sub-linear in the function size while the other is (super-)linear), non-trivially supporting the complete function family that FSS would support.
- FSS is instantiated generically (with function secret-shares of equal size and linearly proportional to function size), where the family of functions has input or output size is linear in the function description size.

FSS with asymmetric share sizes. In our construction, we evaluate the FSS with respect to one of the key-shares as a part of the input encoding for the garbling scheme. To ensure that the encoding time is sublinear in the function computation time, we require that one of the key-shares is sublinear in the function size, and the FSS evaluation with respect to that share to run in time sub-linear in the time required to compute the function. While we do not know of prior work that instantiates such a variant of FSS, the definition of FSS itself does not dis-allow it. Further, we conjecture that instantiations of such FSS scheme could be based on two-server private information retrieval (PIR) schemes based on randomized truth-tables and linearly homomorphic encryption schemes.

Our construction of reusable garbling from FSS is detailed in Algorithm 25. More precisely, this is a transformation from FSS to a function hiding reusable garbling scheme such that any FSS scheme supporting the function family $\mathcal{F}_{\mathsf{PRF}}$ can be used to construct a reusable garbling scheme supporting $\mathcal{F}$, where

$$\mathcal{F}_{\mathsf{PRF}} = \{f'_k(x,i) = f(x - F_k(i))\}_{f\in\mathcal{F}} \tag{6}$$

and F is a PRF with input space $[2^\kappa]$, key space $\{0,1\}^\kappa$ and output space equal to the input space of each $f \in \mathcal{F}$.

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{n}}$ be a pseudorandom function (Definition 2). Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n is the number of inputs and m is the number of outputs. Let $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ be an FSS scheme (Definition 11). The reusable garbling algorithms are defined as follows:

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C})$:
1. Sample PRF key $k \leftarrow \{0,1\}^\kappa$.
2. Define function f'_k as $f'_k(x,i) = f(x - F_k(i))$.
3. Compute $(k_0, k_1, CW) \leftarrow \mathsf{Gen}(1^\kappa, f'_k)$.
4. Output $\Gamma = (k_1, CW)$ and $\mathsf{sk} = (k, k_0, CW)$.

$\mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{sk} = (k, k_0, CW)$.
3. Set $R_j = F_k(j) + \mathbf{x}_j$.
4. Compute $[f(\mathbf{x})]_0 = \mathsf{Eval}(0, k_0, CW, (R_j, j))$.
5. Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$.

$\mathsf{RGb.Eval}(\Gamma, \mathbf{X}_j)$:
1. Parse $\Gamma = (k_1, CW)$ and $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$.
2. Compute $[f(\mathbf{x}_j)]_1 \leftarrow \mathsf{Eval}(1, k_1, CW, (R_j, j))$.
3. Output $\mathbf{y}_j = [f(\mathbf{x}_j)]_0 - [f(\mathbf{x}_j)]_1$.

Figure 25: Reusable Garbling for a Boolean Circuit $\mathbf{C}$ from FSS

Theorem 5 (FSS implies Reusable Garbling) *Let F be a PRF (Definition 2), and for function family $\mathcal{F}$, let $\mathcal{F}_{\mathsf{PRF}}$ be as described in Equation* (6)*. Assuming the existence of an FSS scheme (Definition 11) supporting function family $\mathcal{F}_{\mathsf{PRF}}$, there exists a reusable garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *(Definition 6) for function family $\mathcal{F}$ with size of the garbled circuit $|\Gamma| = |k_0| + \mathsf{poly}(\kappa)$ and input encoding size $\mathsf{m} + \mathsf{n} + \mathsf{polylog}(\kappa)$, where m is the size of the circuit output, and n is the size of the circuit input.*

Proof. We describe the proof for Theorem 5 below.

Simulation. Recall that in the security game described for Reusable Garbling in Definition 6, the offline phase requires a simulator that can produce a simulated garbled circuit. Let F be a PRF (Definition 2). In our construction, the garbled circuit is the key share k_1 and the correction word CW for the function f'_k determined by f and the pseudo-randomness of F_k (where $k \leftarrow \{0,1\}^\kappa$). In the simulator, we would sample a random function r instead, to derive the randomness required and replace the masked input x with this random input. We would then encode the input in the online phase in such a way that evaluation yields the function output as required. This simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for reusable garbling is detailed in Algorithm 26.

Let $\mathcal{D}_{\mathsf{n}}$ be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$. Let $\mathbf{C}_f : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit for a function f, where n is the number of inputs and m is the number of outputs. Let $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ be an FSS scheme (Definition 11). Then the simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for reusable garbling scheme (Algorithm 25) is as given below:

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathbf{C}_f)$:
1. Sample a random function $r(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}$
2. Define function f' as $f'(r(i), i) = f(r(i))$
3. Compute $(k_0, k_1, CW) \leftarrow \mathsf{Gen}(1^\kappa, f')$
4. Output $\Gamma = (k_1, CW)$ and $\mathsf{State}_{\mathsf{Sim}} = (r, k_1, CW)$

$\mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$:
1. Maintain internal state j, initialized at $j = 1$
2. Parse $\mathsf{State}_{\mathsf{Sim}} = (r, k_1, CW)$
3. Compute $R_j = r(j)$
4. Compute $[f(\mathbf{x}_j)]_1 = \mathsf{Eval}(1, k_1, CW, (R_j, j))$
5. Set $[f(\mathbf{x}_j)]_0 := C(\mathbf{x}_j) + [f(\mathbf{x}_j)]_1$
6. Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$ and increment j by 1

Figure 26: Reusable Garbling Simulation for Construction in Figure 25

Hybrid Argument. We show that the adversary's view in ideal execution using the above described simulator, $\mathsf{H}_0 = \{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$, and in the real execution, $\mathsf{H}_3 = \{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$, are indistinguishable via two intermediate hybrid experiments $\mathsf{H}_1, \mathsf{H}_2$ as shown in Algorithm 27 and Algorithm 28.

Hybrid H_1*:* In this hybrid experiment, we use the input $\mathbf{x}_j$ from the adversary and set the input of f' in the simulation as $\mathbf{x}_j + r(j)$. The input encoding is altered to match. The only difference in the simulated execution and this hybrid is the involvement of the input $\mathbf{x}_j$ to compute the input of f'. However the views produced by both experiments is identically distributed, owing to each $r(j)$ acting as a one-time-pad for $\mathbf{x}_j$.

Hybrid H_2*:* In this hybrid, we replace the random function from the previous hybrid with an actual PRF F for a randomly sampled key $k \leftarrow \{0,1\}^\kappa$. We can show this and H_1 are indistinguishable by a reduction to PRF security. Lastly, we reduce to the security of FSS to show that the output of this hybrid experiment is computationally indistinguishable from the real execution.

Claim. If F is a PRF as in Definition 2, then $\mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_2$.

Let $\mathcal{D}_{\mathsf{n}}$ be the uniform distribution over all functions with range $\{0,1\}^{\mathsf{n}}$. Let $\mathbf{C}_f : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit for a function f, where n is the number of inputs and m is the number of outputs. Let $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ be an FSS scheme (Definition 11). Then the hybrid experiment H_1 behaves as follows:

Offline Phase:
1. Sample a random function $r(\cdot) \leftarrow \mathcal{D}_{\mathsf{n}}$
2. Define function f' as $f'(r(i), i) = f(r(i))$
3. Compute $(k_0, k_1, CW) \leftarrow \mathsf{Gen}(1^\kappa, f')$
4. Output $\Gamma = (k_1, CW)$ and $\mathsf{State}_{\mathsf{Sim}} = (r, k_1, CW)$

Online Phase:
1. Maintain internal state j, initialized at $j = 1$
2. Parse $\mathsf{State}_{\mathsf{Sim}} = (r, k_1, CW)$
3. On input $\mathbf{x}_j$ from the adversary, compute $R_j = \mathbf{x}_j + r(j)$
4. Compute $[f(\mathbf{x}_j)]_1 = \mathsf{Eval}(1, k_1, CW, (R_j, j))$
5. Set $[f(\mathbf{x}_j)]_0 := \mathbf{C}_f(\mathbf{x}_j) + [f(\mathbf{x}_j)]_1$
6. Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$ and increment j by 1

Figure 27: Hybrid Experiment H_1 for Construction in Figure 25

Let $F : \{0,1\}^\kappa \times [2^\kappa] \to \{0,1\}^{\mathsf{n}}$ be a pseudorandom function (Definition 2). Let $\mathbf{C}_f : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit for a function f, where n is the number of inputs and m is the number of outputs. Let $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ be an FSS scheme (Definition 11). Then the hybrid experiment H_2 behaves as follows:

Offline Phase:
1. Sample a key $k \leftarrow \{0,1\}^\kappa$
2. Define function f' as $f'(F_k(i), i) = f(F_k(i))$
3. Compute $(k_0, k_1, CW) \leftarrow \mathsf{Gen}(1^\kappa, f')$
4. Output $\Gamma = (k_1, CW)$ and $\mathsf{State}_{\mathsf{Sim}} = (k, k_1, CW)$

Online Phase:
1. Maintain internal state j, initialized at $j = 1$
2. Parse $\mathsf{State}_{\mathsf{Sim}} = (k, k_1, CW)$
3. On input $\mathbf{x}_j$ from the adversary, compute $R_j = \mathbf{x}_j + F_k(j)$
4. Compute $[f(\mathbf{x}_j)]_1 = \mathsf{Eval}(1, k_1, CW, (R_j, j))$
5. Set $[f(\mathbf{x}_j)]_0 := \mathbf{C}_f(\mathbf{x}_j) + [f(\mathbf{x}_j)]_1$
6. Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$ and increment j by 1

Figure 28: Hybrid Experiment H_2 for Construction in Figure 25

Proof. Assume for the sake of contradiction that there exists a distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments H_1 and H_2 with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary A^F that can use $\mathcal{D}$ in a black-box way, to distinguish in the PRF game for F (in Definition 2) as follows:

- A^F has access to oracle $\mathcal{O}$ that is either $F_k(\cdot)$ or random function $r(\cdot)$.
- A^F receives from $\mathcal{D}$ a circuit $\mathbf{C}$ representing a function f.
- A^F defines function $f'(y, i) = f(y)$.
- A^F computes $(k_0, k_1, CW) \leftarrow \mathsf{Gen}(1^\kappa, f')$.
- A^F output $\Gamma = (k_1, CW)$ to $\mathcal{D}$.
- A^F then runs the online phase of the experiment as follows:
  1. Maintain internal state j, initialized at $j = 1$.
  2. On input $\mathbf{x}_j$ from adversary $\mathcal{D}$, A^F does the following:
     - Query $\mathcal{O}$ on j to obtain $\mathcal{O}(j)$.
     - Compute $R_j = \mathbf{x}_j + \mathcal{O}(j)$
     - Compute $[f(\mathbf{x}_j)]_1 = \mathsf{Eval}(1, k_1, CW, (R_j, j))$
     - Set $[f(\mathbf{x}_j)]_0 := \mathbf{C}_f(\mathbf{x}_j) + [f(\mathbf{x}_j)]_1$
     - Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$ and increment j by 1.
  3. Note that if $\mathcal{O} = F_k$, then the view of $\mathcal{D}$ is identical to H_2. Otherwise, it is identical to that of H_1.

– A^F receives a bit b from $\mathcal{D}$ and outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the hybrid experiments are the same except that the randomness in the online phase is derived from a truly random string, or from a pseudorandom string, based on the oracle chosen in the PRF game for F. Thus, if $\mathcal{D}$ can distinguish between these two views, it can be used by A^F to distinguish between the a pseudorandom function output and a random function output in the PRF game with the same advantage, which is non-negligible by assumption. However, since F satisfies the security requirement in Definition 2, no such A^F exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_2$.

Claim. If FSS is an FSS scheme as in Definition 11, then $\mathsf{H}_2 \overset{c}{\approx} \mathsf{H}_3$.

Proof. Assume for the sake of contradiction that there exists a distinguisher $\mathcal{D}$ that can distinguish between the views generated in the experiments H_2 and H_3 with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}^{\mathsf{FSS}}$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the FSS game for FSS (in Definition 11) for $z = 1$ as follows:

– $\mathsf{A}^{\mathsf{FSS}}$ receives from $\mathcal{D}$ a circuit $\mathbf{C}$ representing a function f.
– $\mathsf{A}^{\mathsf{FSS}}$ samples a PRF key $k \leftarrow \{0,1\}^\kappa$.
– $\mathsf{A}^{\mathsf{FSS}}$ defines functions $f_1(X, i) = f(X)$, $f_2(X, i) = f(X - F_k(i))$ and sends it to the challenger in the FSS game.
– $\mathsf{A}^{\mathsf{FSS}}$ recevies from the challenger (k_1, CW).
– $\mathsf{A}^{\mathsf{FSS}}$ outputs $\Gamma = (k_1, CW)$ to $\mathcal{D}$.
– $\mathsf{A}^{\mathsf{FSS}}$ then runs the online phase of the experiment as follows:
  1. Maintain internal state j, initialized at $j = 1$.
  2. On input $\mathbf{x}_j$ from adversary $\mathcal{D}$,
     - Compute $R_j = \mathbf{x}_j + F_k(j)$
     - Compute $[f(\mathbf{x}_j)]_1 = \mathsf{Eval}(1, k_1, CW, (R_j, j))$
     - Set $[f(\mathbf{x}_j)]_0 := \mathbf{C}_f(\mathbf{x}_j) + [f(\mathbf{x}_j)]_1$
     - Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, R_j, j)$ and increment j by 1
– $\mathsf{A}^{\mathsf{FSS}}$ receives a bit b from $\mathcal{D}$ and outputs bit $b' = b$.

The view of $\mathcal{D}$ in both the hybrid experiments are the same except that in Hybrid H_2, the function is computed on a random input, while in the real execution the actual input is unmasked and the function f is computed using it. Both executions yield an output share that, when used for reconstruction, give the correct function output expected. Thus, if $\mathcal{D}$ can distinguish between these two views, that can be used by $\mathsf{A}^{\mathsf{FSS}}$ to distinguish between the case where function f_1 was chosen and the case where f_2 was chosen in the FSS game with the same advantage, which is non-negligible by assumption. However, since FSS satisfies the security requirement in Definition 11, no such $\mathsf{A}^{\mathsf{FSS}}$ exists, and therefore no such $\mathcal{D}$ exists. Thus, $\mathsf{H}_2 \overset{c}{\approx} \mathsf{H}_3$.

Corollary 4. *Let F be a PRF (Definition 2), and for function family $\mathcal{F}$, let $\mathcal{F}_{\mathsf{PRF}}$ be as described in Equation* (6)*. Assuming the existence of an FSS scheme (Definition 11) supporting function family $\mathcal{F}_{\mathsf{PRF}}$, with share sizes $|k_0|, |k_1|$ where $|k_0| \ll |k_1|$, there exists a reusable garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *(Definition 6) for $\mathcal{F}$.*

Corollary 5. *Let F be a PRF (Definition 2), and for function family $\mathcal{F}$ where either $\forall f \in \mathcal{F}, |f| = O(\mathsf{n})$, or $\forall f \in \mathcal{F}, |f| = O(\mathsf{m})$, let $\mathcal{F}_{\mathsf{PRF}}$ be as described in Equation* (6)*. Assuming the existence of an FSS scheme (Definition 11) supporting function family $\mathcal{F}_{\mathsf{PRF}}$, there exists a reusable garbling scheme* $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *(Definition 6) for $\mathcal{F}$.*

5.2 Reusable Garbling from HSS

In this section we present a generic transformation from any homomorphic secret-sharing scheme (HSS – Definition 12) to a reusable garbling scheme (Definition 6). The resulting reusable garbling scheme requires input encoding time proportional to function running time (or, at the very least, needs to read the function description). Therefore the scheme is non-trivial only for the circuit family where the function description size is comparable to the input or output size. A notable feature of this construction, however, is that the garbling time (and the size of the reusable garbling) is sub-linear in the description of the function being garbled.

Adaptively Secure HSS. For our construction, we require that the underlying HSS scheme is adaptively secure. Traditionally, HSS schemes have simulators that outputs a public key, one evaluation key and a set of simulated ciphertexts. In contrast, in our definition, we require the simulator to output only the public key and evaluation key, and allow the adversary to ask for ciphertexts corresponding to it's choice of inputs in an online phase. We leave checking if the existing constructions satisfy this stronger security definition as an open problem.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit representing the function f, where n and m are the number of inputs and outputs respectively. Let $\mathsf{HSS} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval})$ be an HSS scheme (Definition 12). The reusable garbling algorithms are defined as follows:

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C})$:
1. Compute $(\mathsf{pk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow \mathsf{HSS.Gen}(1^\kappa, r)$.
2. Output $\Gamma = (\mathsf{ek}_1)$ and $\mathsf{sk} = (\mathsf{pk}, \mathsf{ek}_0)$.

$\mathsf{RGb.Encode}_j(\mathsf{sk}, \mathbf{x}_j)$:
1. Maintain internal state j, initialized at $j = 1$.
2. Parse $\mathsf{sk} = (\mathsf{pk}, \mathsf{ek}_0)$.
3. Compute $\mathsf{ct}_{x_j} \leftarrow \mathsf{HSS.Enc}(\mathsf{pk}, \mathbf{x}_j)$.
4. Compute $[f(\mathbf{x}_j)]_0 \leftarrow \mathsf{HSS.Eval}(0, \mathsf{ek}_0, \mathsf{ct}_{x_j}, \mathbf{C}, \delta)$.
5. Output $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, \mathsf{ct}_{x_j})$.

$\mathsf{RGb.Eval}(\Gamma, \mathbf{X}_j)$:
1. Parse $\Gamma = (\mathsf{ek}_1)$ and $\mathbf{X}_j = ([f(\mathbf{x}_j)]_0, \mathsf{ct}_{x_j})$.
2. Compute $[f(\mathbf{x}_j)]_1 \leftarrow \mathsf{HSS.Eval}(1, \mathsf{ek}_1, \mathsf{ct}_{x_j}, \mathbf{C}, \delta)$.
3. Output $\mathbf{y}_j = [f(\mathbf{x}_j)]_0 - [f(\mathbf{x}_j)]_1$.

Figure 29: Reusable Garbling for a Boolean Circuit $\mathbf{C}$ from HSS

Theorem 6 (HSS implies Reusable Garbling) *Assuming the existence of an HSS scheme (Definition 12) with evaluation key size $|\mathsf{ek}|$ and encoding size $|\mathsf{ct}_x|$ for a circuit family $\mathcal{C}$ whose members have description of size $\mathcal{O}(\mathsf{n}+\mathsf{m})$ for n, m denoting the size of inputs and outputs to the circuit, there exists a reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ (Definition 6) for the same class of circuits $\mathcal{C}$ with size of the garbled circuit $|\Gamma| = |\mathsf{ek}|$ and input encoding size $\mathsf{m} + |\mathsf{ct}_x|$.*

Proof. We describe the proof for Theorem 6 below.

Simulation. Recall that in the security game described for Reusable Garbling in Definition 6, the offline phase requires a simulator that can produce a simulated garbled circuit. In our construction, the garbled circuit is the key share ek_1 which can be simulated using the HSS simulator. In the online phase, we need to create corresponding ciphertexts and function output shares. We use the fact that the HSS is additive, to create the output share from the function output and HSS evaluation. This simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for reusable garbling is detailed in Algorithm 30.
It only remains to show that with respect to the simulator in Algorithm 30 above, the distributions $\{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ and $\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ (Definition 6) of interactions with any PPT adversary A are indistinguishable, given that the underlying HSS scheme is secure (Definition 12).

Claim. If HSS is a homomorphic secret-sharing scheme as in Definition 12, then

$$\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathsf{RGb},\mathsf{A},\mathsf{Sim}}(1^\kappa)\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathrm{Exp}^{\mathrm{real}}_{\mathsf{RGb},\mathsf{A}}(1^\kappa)\}_{\kappa\in\mathbb{N}}.$$

Proof. Assume for the sake of contradiction that there exists an distinguisher $\mathcal{D}$ that can distinguish between the views generated in the real and simulated experiments with non-negligible advantage $\epsilon(\kappa)$. We now show that there exists an adversary $\mathsf{A}_{\mathsf{HSS}}$ that can use $\mathcal{D}$ in a black-box way, to distinguish in the interactive security game for HSS (in Definition 12) as follows:

- $\mathsf{A}_{\mathsf{HSS}}$ receives from $\mathcal{D}$ a circuit $\mathbf{C}$.
- $\mathsf{A}_{\mathsf{HSS}}$ receives ek_1 from the challenger in the HSS game.

Let $\mathbf{C} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{m}}$ be a boolean circuit for a function f, where n is the number of inputs and m is the number of outputs. For an HSS scheme HSS (Definition 12), let $\mathsf{Sim}_{\mathsf{HSS}} = (\mathcal{S}_1, \mathcal{S}_2)$ be the PPT simulator. Then the simulator $\mathsf{Sim} = (\mathcal{S}_1, \mathcal{S}_2)$ for reusable garbling scheme (Algorithm 29) is as given below:

$\mathsf{Sim}.\mathcal{S}_1(1^\kappa, \mathbf{C}_f)$:

1. Compute $(\mathsf{State}_{\mathcal{S}_1}, \mathsf{pk}, \mathsf{ek}_1) \leftarrow \mathsf{Sim}_{\mathsf{HSS}}.\mathcal{S}_1(1^\kappa)$
2. Output $\Gamma = (\mathsf{ek}_1)$ and $\mathsf{State}_{\mathsf{Sim}} = \mathsf{State}_{\mathcal{S}_1}$

$\mathsf{Sim}.\mathcal{S}_2(\mathsf{State}_{\mathsf{Sim}}, \mathbf{C}(\mathbf{x}_j))$:

1. Maintain internal state j, initialized at $j = 1$
2. Parse $\mathsf{State}_{\mathsf{Sim}} = \mathsf{State}_{\mathcal{S}_1}$
3. Compute $\mathsf{ct}_j \leftarrow \mathsf{Sim}_{\mathsf{HSS}}.\mathcal{S}_2(\mathsf{State}_{\mathcal{S}_1})$
4. Compute $[\mathbf{C}(\mathbf{x}_j)]_1 = \mathsf{HSS.Eval}(1, \mathsf{ek}_1, \mathsf{ct}_j, \mathbf{C}, \delta)$
5. Set $[\mathbf{C}(\mathbf{x}_j)]_0 := \mathbf{C}(\mathbf{x}_j) - [\mathbf{C}(\mathbf{x}_j)]_1$
6. Output $\mathbf{X}_j = (\mathsf{ct}_j, [\mathbf{C}(\mathbf{x}_j)]_0)$ and increment j by 1

Figure 30: Reusable Garbling Simulation for Construction in Algorithm 29

- $\mathsf{A}_{\mathsf{HSS}}$ outputs ek_1 as the reusable garbling to $\mathcal{D}$.
- For every input $\mathbf{x_i}$ sent by $\mathcal{D}$ to $\mathsf{A}_{\mathsf{HSS}}$, it is forwarded to the challenger in the HSS game. $\mathsf{A}_{\mathsf{HSS}}$ receives ct_i from the challenger and forwards it to $\mathcal{D}$.
- Finally $\mathsf{A}_{\mathsf{HSS}}$ outputs whatever $\mathcal{D}$ outputs.

Note that the view of $\mathcal{D}$ matches the real garbling view if the challenger in the HSS game used $\mathsf{HSS.Gen}$ to generate the evaluation key and $\mathsf{HSS.Enc}$ to compute ciphertexts. If the challenger used the algorithms in $\mathsf{Sim}_{\mathsf{HSS}}$, the resulting view matches the simulated garbling view. Thus, if $\mathcal{D}$ can distinguish between these two views, it can be used by $\mathsf{A}_{\mathsf{HSS}}$ to distinguish between the real and simulated HSS shares with the same advantage, which is non-negligible by assumption. However, since HSS satisfies the security requirement in Definition 12, no such $\mathsf{A}_{\mathsf{HSS}}$ exists, and therefore no such $\mathcal{D}$ exists.

Remark 6. While we show how to construct reusable garbling for certain function classes above using public-key HSS, one could easily switch to secret-key HSS. This is because the adversary only has oracle access to the encoding algorithm (that includes the public key pk in our construction) and therefore this could easily be replaced by a secret key sk.

Acknowledgements

Anasuya Acharya was supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 101124977 (DECRYPSYS). Carmit Hazay was supported by ISF grant No. 607/25. Rahul Satish was supported by the Independent Research Fund Denmark (IRFD) grant number 0165-00079B.

References

1. Abram, D., Malavolta, G., Roy, L.: Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor hashing for all circuits. In: Koucký, M., Bansal, N. (eds.) 57th ACM STOC. pp. 1875–1886. ACM Press (Jun 2025)
2. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 3–35. Springer, Cham (Aug 2017)
3. Applebaum, B.: Key-dependent message security: Generic amplification and completeness. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 527–546. Springer, Berlin, Heidelberg (May 2011)
4. Applebaum, B.: Cryptography in Constant Parallel Time. ISC, Springer (2014)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: 45th FOCS. pp. 166–175. IEEE Computer Society Press (Oct 2004)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Comput. Complex. 15(2), 115–162 (2006)
7. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 152–163. Springer, Berlin, Heidelberg (Jul 2010)

8. Applebaum, B., Ishai, Y., Kushilevitz, E., Waters, B.: Encoding functions with constant online rate or how to compress garbled circuits keys. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 166–184. Springer, Berlin, Heidelberg (Aug 2013)
9. Asharov, G., Segev, G.: Limits on the power of indistinguishability obfuscation and functional encryption. In: Guruswami, V. (ed.) 56th FOCS. pp. 191–209. IEEE Computer Society Press (Oct 2015)
10. Ashur, T., Hazay, C., Satish, R.: Oblivious garbling and its applications. Cryptology ePrint Archive, Paper 2026/1132 (2026), https://eprint.iacr.org/2026/1132
11. Ball, M., Li, H., Lin, H., Liu, T.: New ways to garble arithmetic circuits. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 3–34. Springer, Cham (Apr 2023)
12. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM 59(2) (May 2012)
13. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Berlin, Heidelberg (May / Jun 2010)
14. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. pp. 503–513. ACM Press (May 1990)
15. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 784–796. ACM Press (Oct 2012)
16. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 13. LNCS, vol. 8257, pp. 218–234. Springer, Cham (Nov 2013)
17. Bitansky, N., Garg, R.: Succinct randomized encodings from laconic function evaluation, faster and simpler. Cryptology ePrint Archive, Report 2024/1815 (2024), https://eprint.iacr.org/2024/1815
18. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 439–448. ACM Press (Jun 2015)
19. Bitansky, N., Nishimaki, R., Passelègue, A., Wichs, D.: From cryptomania to obfustopia through secret-key functional encryption. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 391–418. Springer, Berlin, Heidelberg (Oct / Nov 2016)
20. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Berlin, Heidelberg (May 2014)
21. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Berlin, Heidelberg (Mar 2011)
22. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Berlin, Heidelberg (Feb 2007)
23. Boyle, E., Couteau, G., Meyer, P.: Sublinear-communication secure multiparty computation does not require FHE. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 159–189. Springer, Cham (Apr 2023)
24. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 337–367. Springer, Berlin, Heidelberg (Apr 2015)
25. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 509–539. Springer, Berlin, Heidelberg (Aug 2016)
26. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1292–1303. ACM Press (Oct 2016)
27. Boyle, E., Kohl, L., Li, Z., Scholl, P.: Direct FSS constructions for branching programs and more from PRGs with encoded-output homomorphism. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part VI. LNCS, vol. 15489, pp. 266–298. Springer, Singapore (Dec 2024)
28. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Berlin, Heidelberg (Mar 2015)
29. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-round secure computation and secure autonomous mobile agents. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 512–523. Springer, Berlin, Heidelberg (Jul 2000)
30. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Berlin, Heidelberg (May 2001)
31. Couteau, G., Hazay, C., Hegde, A., Kumar, N.: Breaking the $1/\lambda$-rate barrier for arithmetic garbling. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VI. LNCS, vol. 15606, pp. 182–213. Springer, Cham (May 2025)

32. Couteau, G., Hazay, C., Hegde, A., Kumar, N.: $\omega(1/\lambda)$-rate boolean garbling scheme from generic groups. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part IV. LNCS, vol. 16003, pp. 489–521. Springer, Cham (Aug 2025)
33. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 93–122. Springer, Berlin, Heidelberg (Aug 2016)
34. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: 26th ACM STOC. pp. 554–563. ACM Press (May 1994)
35. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013)
36. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Berlin, Heidelberg (Aug 2010)
37. Goldreich, O.: Foundations of Cryptography: Volume 1. Cambridge University Press, USA (2006)
38. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press (Jun 2013)
39. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Berlin, Heidelberg (Aug 2008)
40. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Berlin, Heidelberg (Aug 2012)
41. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 545–554. ACM Press (Jun 2013)
42. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Berlin, Heidelberg (Aug 2015)
43. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Berlin, Heidelberg (Feb 2010)
44. Heath, D.: Efficient arithmetic in garbled circuits. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part V. LNCS, vol. 14655, pp. 3–31. Springer, Cham (May 2024)
45. Hsieh, Y.C., Lin, H., Luo, J.: Attribute-based encryption for circuits of unbounded depth from lattices. In: 64th FOCS. pp. 415–434. IEEE Computer Society Press (Nov 2023)
46. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS. pp. 294–304. IEEE Computer Society Press (Nov 2000)
47. Ishai, Y., Li, H., Lin, H.: Succinct homomorphic MACs from groups and applications. Cryptology ePrint Archive, Paper 2024/2073 (2024), `https://eprint.iacr.org/2024/2073`
48. Ishai, Y., Li, H., Lin, H.: A unified framework for succinct garbling from homomorphic secret sharing. Cryptology ePrint Archive, Report 2025/442 (2025), `https://eprint.iacr.org/2025/442`
49. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Khuller, S., Williams, V.V. (eds.) 53rd ACM STOC. pp. 60–73. ACM Press (Jun 2021)
50. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 670–699. Springer, Cham (May / Jun 2022)
51. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Berlin, Heidelberg (Apr 2008)
52. Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 544–562. Springer, Cham (Sep 2018)
53. Kolesnikov, V., Mohassel, P., Rosulek, M.: FleXOR: Flexible garbling for XOR gates that beats free-XOR. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 440–457. Springer, Berlin, Heidelberg (Aug 2014)
54. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Berlin, Heidelberg (Jul 2008)
55. Li, H., Lin, H., Luo, J.: ABE for circuits with constant-size secret keys and adaptive security. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 680–710. Springer, Cham (Nov 2022)

56. Li, H., Liu, T.: How to garble mixed circuits that combine boolean and arithmetic computations. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 331–360. Springer, Cham (May 2024)
57. Lin, H., Luo, J.: Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 247–277. Springer, Cham (May 2020)
58. Lin, H., Luo, J.: Succinct and adaptively secure ABE for ABP from k-Lin. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 437–466. Springer, Cham (Dec 2020)
59. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology 22(2), 161–188 (Apr 2009)
60. Liu, H., Wang, X., Yang, K., Yu, Y.: BitGC: Garbled circuits with 1 bit per gate. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VII. LNCS, vol. 15607, pp. 437–466. Springer, Cham (May 2025)
61. Meyer, P., Orlandi, C., Roy, L., Scholl, P.: Rate-1 arithmetic garbling from homomorphic secret sharing. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024, Part IV. LNCS, vol. 15367, pp. 71–97. Springer, Cham (Dec 2024)
62. Meyer, P., Orlandi, C., Roy, L., Scholl, P.: Silent circuit relinearisation: Sublinear-size (boolean and arithmetic) garbled circuits from DCR. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part IV. LNCS, vol. 16003, pp. 426–458. Springer, Cham (Aug 2025)
63. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: ACM-EC. pp. 129–139 (1999)
64. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010), `https://eprint.iacr.org/2010/556`
65. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Berlin, Heidelberg (Dec 2009)
66. Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: Thorup, M. (ed.) 59th FOCS. pp. 859–870. IEEE Computer Society Press (Oct 2018)
67. Ragavan, S., Vafa, N., Vaikuntanathan, V.: Indistinguishability obfuscation from bilinear maps and LPN variants. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024, Part IV. LNCS, vol. 15367, pp. 3–36. Springer, Cham (Dec 2024)
68. Rosulek, M., Roy, L.: Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 94–124. Springer, Cham, Virtual Event (Aug 2021)
69. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 463–472. ACM Press (Oct 2010)
70. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press (May / Jun 2014)
71. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Berlin, Heidelberg (May 2005)
72. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC1. In: 40th FOCS. pp. 554–567. IEEE Computer Society Press (Oct 1999)
73. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)
74. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 220–250. Springer, Berlin, Heidelberg (Apr 2015)

A Additional Preliminaries

A.1 Reusable Garbling from 1-Key Succinct Public-Key FE.

In [38], they showed how to construct a reusable garbling scheme using any 1-key succinct FE scheme. We restate the construction below in Figure 31.

Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a succinct 1-key public-key FE scheme (Definition 8). Then the reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ works as follows:

$\mathsf{RGb.Garble}(1^\kappa, \mathbf{C})$:

1. Generate FE keys $(\mathsf{fmpk}, \mathsf{fmsk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$.
2. $\mathsf{fsk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{fmsk}, \mathbf{C})$
3. Output $\mathsf{gsk} = \mathsf{fmpk}$ as the secret key, and $\Gamma = \mathsf{fsk}_f$ as the garbling of $\mathbf{C}$.

$\mathsf{RGb.Encode}(\mathsf{gsk}, x)$: Output $c_x \leftarrow \mathsf{FE.Enc}(\mathsf{gsk}, x)$.

$\mathsf{RGb.Eval}(\Gamma, c_x)$: Output $\mathsf{FE.Dec}(\Gamma, c_x)$.

Figure 31: Reusable Garbling from 1-Key FE

A.2 Succinct FE from LFE and Non-Succint FE

In [66], they showed how to generically construct any 1-key FE scheme from an LFE scheme, using an underlying non-succinct FE scheme, that is achievable from PKE [40,69]. We restate their construction in Figure 32 below.

Let $\mathsf{FE}' = (\mathsf{FE}'.\mathsf{Setup}, \mathsf{FE}'.\mathsf{KeyGen}, \mathsf{FE}'.\mathsf{Enc}, \mathsf{FE}'.\mathsf{Dec})$ be a non-succinct public-key FE scheme (Definition 8). Let $\mathsf{LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ be an LFE scheme (Definition 9). Let $\mathbf{C}_{\mathsf{LFE}}$ be a circuit that on input (x, r) outputs $\mathsf{Enc}(\mathsf{LFE.crs}, \mathsf{LFE.digest}_{\mathbf{C}}, x; r)$ where $\mathsf{digest}_{\mathbf{C}} = \mathsf{LFE.Compress}(\mathsf{LFE.crs}, \mathbf{C})$. Then the reusable garbling scheme $\mathsf{RGb} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ works as follows:

$\mathsf{FE.Setup}(1^\kappa)$:

1. Obtain $\mathsf{LFE.crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa)$.
2. Obtain $(\mathsf{FE}'.\mathsf{fmpk}', \mathsf{FE}'.\mathsf{fmsk}') \leftarrow \mathsf{FE}'.\mathsf{Setup}(1^\kappa)$.
3. Output $\mathsf{fmpk} = \mathsf{FE}'.\mathsf{fmpk}'$ and $\mathsf{fmsk} = (\mathsf{LFE.crs}, \mathsf{FE}'.\mathsf{fmsk}')$.

$\mathsf{FE.KeyGen}(\mathsf{fmsk}, C)$:

1. Obtain $\mathsf{FE}'.\mathsf{fsk}'_{\mathbf{C}_{\mathsf{LFE}}} \leftarrow \mathsf{FE}'.\mathsf{KeyGen}(\mathbf{C}_{\mathsf{LFE}})$.
2. Output $\mathsf{sk}_{\mathbf{C}} := (\mathsf{FE}'.\mathsf{fsk}'_{\mathbf{C}_{\mathsf{LFE}}}, \mathsf{LFE.crs}, \mathbf{C})$.

$\mathsf{FE.Enc}(\mathsf{fmpk}, x)$:

1. Obtain $\mathsf{ct}_x \leftarrow \mathsf{FE}'.\mathsf{Enc}(\mathsf{FE}'.\mathsf{fmpk}', (x, r))$.
2. Output ct_x.

$\mathsf{FE.Dec}(\mathsf{sk}_f, \mathsf{ct}_x)$:

1. Obtain $\mathsf{ct}_{\mathsf{LFE}} = \mathsf{FE}'.\mathsf{Dec}(\mathsf{FE}'.\mathsf{sk}'_{\mathbf{C}_{\mathsf{LFE}}}, \mathsf{ct}_x)$.
2. Obtain $y = \mathsf{LFE.Dec}(\mathsf{LFE.crs}, \mathbf{C}, \mathsf{ct}_{\mathsf{LFE}})$

Figure 32: Succinct 1-key FE from LFE