Adaptive Distributional Security for Garbling Schemes with $\mathcal{O}(|x|)$ Online Complexity

Estuardo Alpírez Bock[1], Chris Brzuska[2], Pihla Karanko[2], Sabine Oechsner[3], and Kirthivaasan Puniamurthy[2]

[1] Xiphera, Finland, estuardo.alpirezbock@xiphera.com
[2] Aalto University, Finland, [first name].[last name]@aalto.fi
[3] University of Edinburgh, UK, s.oechsner@ed.ac.uk

Abstract. Garbling schemes allow to garble a circuit C and an input x such that $C(x)$ can be computed while hiding both C and x. In the context of adaptive security, an adversary specifies the input to the circuit *after* seeing the garbled circuit, so that one can pre-process the garbling of C and later only garble the input x in the *online phase*. Since the online phase may be time-critical, it is an interesting question how much information needs to be transmitted in this phase and ideally, this should be close to $|x|$. Unfortunately, Applebaum, Ishai, Kushilevitz, and Waters (AIKW, *CRYPTO 2013*) show that for some circuits, specifically PRGs, achieving online complexity close to $|x|$ is impossible with simulation-based security, and Hubáček and Wichs (HW, *ITCS 2015*) show that online complexity of maliciously secure 2-party computation needs to grow with the incompressibility entropy of the function. We thus seek to understand under which circumstances optimal online complexity is feasible despite these strong lower bounds.
Our starting point is the observation that lower bounds (only) concern *cryptographic* circuits and that, when an embedded secret is not known to the adversary (distinguisher), then the lower bound techniques do not seem to apply. Our main contribution is *distributional simulation-based security* (DSIM), a framework for capturing weaker, yet meaningful simulation-based (adaptive) security which does not seem to suffer from impossibility results akin to AIKW. We show that DSIM can be used to prove security of a distributed symmetric encryption protocol built around garbling. We also establish a bootstrapping result from DSIM-security for NC^0 circuits to DSIM-security for arbitrary polynomial-size circuits while preserving their online complexity.

1 Introduction

The garbled circuits approach to secure two-party computation goes back to a seminal work of Yao [Yao82a] and allows two parties to evaluate a circuit C on their private inputs. Bellare, Hoang and Rogaway (BHR) [BHR12b] suggest to abstract out the central building block behind this approach into a *garbling*

scheme. A garbling scheme allows a party (called garbler) to *garble* a circuit C and input x into $\tilde{C}$ and $\tilde{x}$ such that another party (called evaluator) can derive $C(x)$ from $\tilde{C}$ and $\tilde{x}$. Selective (simulation-based) security of a garbling scheme is defined as the comparison between real and simulated garbling: A garbling scheme is secure if no efficient adversary can distinguish a real garbling from the output of a simulator that is given only $C(x)$ and some leakage $\Phi(C)$ on C.

BHR further point out [BHR12a] that in some settings, a stronger adaptive security notion is needed, in which the adversary chooses the input x adaptively after seeing the garbled circuit $\tilde{C}$. In particular, now the simulator works in two stages and has to produce $\tilde{C}$ first given only $\Phi(C)$, and produces $\tilde{x}$ only after seeing $C(x)$. The second *adaptive* phase is often referred to as the *online phase*, and $|\tilde{x}|$ as the online complexity.

In the selective setting, $|\tilde{x}|$ can be linear in the size of x and independent of any other circuit dimension. For example, $|\tilde{x}| = \lambda|x|$ in the case of Yao's garbling scheme and many of its variants, where λ denotes the security parameter. Ideally, the online complexity of *adaptively* secure garbling schemes would also match this bound. Unfortunately, Applebaum, Ishai, Kushilevitz and Waters (AIKW [AIKW13]) show that adaptively secure garbling schemes for pseudorandom generators (PRGs) require online complexity of at least the output size $|C(x)|$, and indeed, adaptively secure constructions for general circuits match this bound [BHR12a,AS16,HJO+16,JW16,GS18,JO20].

Circumventing the AIKW lower bound. Approaches to circumvent the AIKW lower bound include complexity leveraging over all possible values of x (with a superpolynomial-time simulator) and proving security in the programmable random oracle model [BHR12a]. Another line of work shows how to construct adaptively secure garbling schemes with low online complexity ($|\tilde{x}| = O(|x|)$) for restricted circuit classes [JSW17,KKP21,AS16] which is contained in the class of all efficiently invertible circuits and, therefore, unfortunately exclude garbling most cryptographic functions with adaptive simulation-security.

Relaxing garbling scheme security. In this work, we explore a new approach which seems suitable for *cryptographic* circuits. Concretely, we identify a relaxed (simulation-based) garbling scheme security notion that is not affected by the same lower bound and ask which circuit classes can be securely garbled with respect to it. When garbling *cryptographic* circuits, some inputs are typically generated *honestly* and *secretly*, and hence, an (efficient) adversary does not know them - an indication that the standard definition of adaptive simulation-based security might be too strong.

1.1 Summary of contributions

Generalizing adaptive garbling scheme security. Our main contribution is a security framework for garbling schemes, called *distributional simulation-based security (DSIM)* which relaxes the existing simulation-based adaptive security

notion. The idea is to model realistic restrictions on the knowledge of adversary and simulator such as partially hiding the circuit (or inputs) that is garbled from the adversary. DSIM security bypasses the AIKW lower bound and our generalization of it as they are specific to the adversary's and simulator's (lack of) knowledge in the adaptive simulatability game.

We then show relations between DSIM and existing adaptive security notions, in particular that adaptive simulatability implies DSIM when suitably restricting DSIM parameters. We further show how use a DSIM-secure garbling scheme to turn an authenticated encryption scheme (AE) into a two-party distributed encryption protocol. The AE scheme needs to be additionally secure under linear related-key (RK) attacks. PRFs secure under RK attacks can be achieved, e.g., based on key-homormorphic PRFs [BLMR13].

Bootstrapping distributional security. Moreover, we show a bootstrapping result for circuits which are *output indistinguishable*, i.e. for which there exists an efficient circuit sampler producing C such that $C(x)$ and $C(0^{|x|})$ are computationally indistinguishable for arbitrary adversarially chosen inputs x.

Theorem 1 (Informal). *Assume that garbling scheme* Gb *is DSIM-secure for garbling the class of output indistinguishable* NC^0 *circuits with online complexity* α *and assume IND-CPA secure symmetric encryption exists. Then, there exists a garbling scheme that is DSIM-secure for garbling the class of all polynomial size output indistinguishable circuits with online complexity* α*.*

Note that the bootstrapping result preserves the online complexity of the garbling scheme, and hence it suffices to construct a DSIM-secure garbling scheme for the class of output indistinguishable NC^0 circuits with online complexity $\mathcal{O}(|x|)$ to obtain a DSIM-secure garbling scheme for the class *all* output indistinguishable circuits with online complexity $\mathcal{O}(|x|)$. The construction of such a garbling scheme remains a very interesting open problem.

Further results about standard simulation-based security To complement our main contribution, we also include two smaller results within the context of the standard adaptive security notion. We generalize the AIKW lower bound and provide small improvements regarding the online complexity for garbling NC^1 circuits.

Bounding online complexity in terms of (pseudo-)entropy. AIKW show that garbling a pseudorandom generator (PRG) with adaptive security requires the online complexity of the garbling scheme to grow with the output size $|C(x)|$. Using a similar, more general idea, Hubáček and Wichs (HW [HW15]) show that the online complexity of maliciously secure 2-party computation is lower bounded by the ('worst-case') Yao incompressibility entropy [Yao82b,BSW03] of the computed function. As a corollary, HW gives an analogous lower bound for the online complexity of adaptively secure garbling schemes: Consider a special

case of 2PC protocol where one of the parties does not have any input, now we can realize the maliciously secure two-party computation protocol as simply one party first sending the garbled function (offline) and then sending the garbled input (online). Now the online complexity equals the length of the garbled input, so HW lower-bound tells that in an adaptively secure garbling scheme, the online complexity is lower bounded by the ('worst-case') Yao incompressibility entropy of the function output.

We (Section 7) prove the HW corollary for adaptively secure garbling schemes directly and additionally we show how to extend the corollary to *average case* Yao incompressibility entropy [TVZ05]. The lower bounds include cryptographic functions such as PRGs, pseudorandom functions, and encryption schemes, all of which (when using suitable parameters) have higher incompressibility entropy than their input size. Note that the more standard notion of HILL pseudo-entropy (indistinguishabililty from a distribution with k bits of Shannon entropy [HILL99] or min-entropy [BSW03]), if high, also implies high (average case or worst case, depending on the HILL pseudo-entropy version[4]) incompressibility entropy, but the converse is not necessarily true [HLR07,HMS23], whence HW and we state our results in terms of incompressibility entropy.

Theorem (Informal). *Let $C \leftarrow\!\$\ \mathsf{Sam}_C(1^\lambda)$ be a distribution over circuits and $x \leftarrow\!\$\ \mathsf{Sam}_x(C, 1^\lambda)$ be a distribution over inputs, depending on C. Then if Gb is a SIM_Φ-secure garbling scheme, its online complexity $|\tilde{x}|$ is greater or equal to the worst case (average case) Yao incompressibility-entropy of $C(x)$ (minus a small constant) conditioned on $\Phi(C)$, where Φ is some function on the circuit.*

Here Φ is a function that depends on the flavour of SIM security we want to capture. Typically, Φ is just the topology of the circuit C, which would not contribute to the entropy, but it can also be a less trivial function.

Garbling schemes with (almost) optimal online complexity. Our final result revisits the question of constructing garbling schemes with (almost) optimal online complexity under existing security notions. To circumvent the AIKW lower bound, various works considered the weaker notion of *indistinguishability-based* adaptive security [BHR12a]. (For brevity, we will call this notion adaptive indistinguishability or adaptive IND security.) Table 1 summarizes the existing results, the circuit class they apply to, their online complexity and security assumptions. Note that any circuit with constant treewidth can be simulated in NC^1 [JS14], and that there exist NC^1 circuits that do not have low treewidth, e.g. Goldreich's PRG [Gol11] is in $\mathrm{NC}^0 \subseteq \mathrm{NC}^1$ [App14] but has high expansion and hence high treewidth. In fact, low treewidth already implies invertibility [Bod88,Fre90,BK08][5].

Concretely, we revisit the adaptively indistinguishable garbling scheme by Jafargholi, Scafuro and Wichs (JSW) [JSW17] for garbling NC^1 circuits. The

[4] See [Kar23] for a survey on the different notions of HILL pseudo-entropy and Yao incompressibility entropy and their implications.

[5] See Appendix A for further discussion.

	Circuit class	online complexity	assumption
Ananth & Sahai [AS16]	Arbitrary	$\mathsf{poly}(\lambda, \lvert x\rvert)$	iO + OWF
Kamath, Klein & Pietrzak [KKP21]	Constant tw	$\lambda\lvert x\rvert$	OWF
Jafargholi, Scafuro & Wichs [JSW17]	NC^1	$2\lambda\lvert x\rvert + 2\lambda^2 d$	OWF
this work	NC^1	$2\lambda\lvert x\rvert$	OWF

Table 1: Overview of adaptively indistinguishable garbling schemes with online complexity linear in the input size $|x|$, by circuit class for which security is known. d denotes the circuit depth and λ the security parameter.

JSW construction has an online complexity that includes a small linear overhead in the *circuit depth*. We show how to modify the construction to remove this overhead.

Theorem (Informal). *Assuming the existence of one-way functions, there exists an adaptively indistinguishable garbling scheme for NC^1 circuits with online complexity $2\lambda|x|$.*

Jafargholi, Scafuro and Wichs further observed that an adaptively indistinguishable garbling scheme (IND) is also adaptively simulatable when restricting the garbling scheme to *efficiently invertible functions*. Adaptive IND security ensures that garblings of (C_0, x_0) and (C_1, x_1) are indistinguishable, even when x_0 and x_1 are chosen adaptively, as long as $C_0(x_0) = C_1(x_1)$. Since this equality requirement is rather restrictive, it is not obvious how to use IND security for cryptographic circuits in general. However, IND secure garbling should be useful to use with all techniques that are compatible with indistinguishability obfuscation (iO)as well, because iO requires that for all x, $C_0(x) = C_1(x)$ which implies the condition $C_0(x_0) = C_1(x_1)$ for $x = x_0 = x_1$. Therefore, IND security should be useful, e.g., for puncturable PRFs. We show the following:

Corollary (Informal). *Assuming the existence of one-way functions, there exists an adaptively simulatable garbling scheme for efficiently invertible NC^1 circuits with online complexity $2\lambda|x|$.*

1.2 Outline

In Section 2, we provide a technical overview over our main results. Section 3 provides background on garbling schemes and cryptographic primitives. Section 4 introduces our notion of distributional simulation-based security (DSIM). We show how to use the notion in Section 5 on the example of distributed symmetric encryption and prove a bootstrapping results for DSIM-secure garbling schemes in Section 6. Our direct proof of the HW incompressibility-based lower bound for garbling schemes is presented in Section 7. Finally, we show how to garble efficiently invertible NC^1 circuits with online complexity $2\lambda|x|$ based on one-way functions in Sections 8.

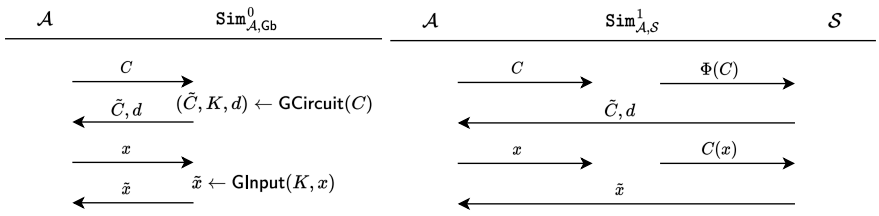


Fig. 1: Simulation-based security games $\mathtt{Sim}^0_{\mathcal{A},\mathsf{Gb}}$ and $\mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}$ for garbling scheme Gb, adversary $\mathcal{A}$ and simulator $\mathcal{S}$, and circuit leakage function Φ.

2 Technical overview

This section provides an overview of our core contribution, a relaxed simulation-based security notion for garbling schemes.

2.1 Definining distributional simulation-based (DSIM) security

A quick recap of the existing adaptive simulation-based security notion [BHR12a]: Figure 1 shows the real and ideal security games as interaction between adversary $\mathcal{A}$, the security game, and a simulator $\mathcal{S}$. In the ideal game $\mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}$, $\mathcal{A}$ chooses (and hence knows) the circuit C and input x to be garbled, while the simulator has to simulate given leakage $\Phi(C)$ and output $C(x)$ only. Intuitively, this simultaneously captures strong privacy guarantees on the circuit and input.

This security notion gives the adversary the power to choose and learn the full circuit C *and* the input x to be garbled. The AIKW lower bound and our generalization effectively exploit an attack vector that is specific to this security notion: Since the adversary knows both C and x, they can compute the output $C(x)$ themselves and then compare it to the result of the garbled circuit evaluation. A simulator on the other hand is asked to simulate the garbled input given only $C(x)$. As AIKW and HW show, this simulation can simply not succeed if $C(x)|\Phi(C)$ produces too much (pseudo-)entropy. However, the main purpose of using a garbling scheme is to hide at least some information about either C or x, else one could evaluate $C(x)$ in the clear. For example when garbling a cryptographic function such as an encryption function $enc(k, \cdot)$ for secret key k, the key is typically sampled as part of the outer protocol. Hence only the garbler knows k. Thus when defining and proving security against a malicious evaluator, it suffices to prove security against an adversary (i.e. the evaluator) who does not know k and thus does not know the entire circuit C. As it suffices to consider a weaker adversary, we introduce two modifications to games $\mathtt{Sim}^0_{\mathcal{A},\mathsf{Gb}}$ and $\mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}$:

Modification 1: Partially hiding the circuit from $\mathcal{A}$. Instead of $\mathcal{A}$ choosing C, the game is parameterized by a sampler Sam whose description is known to the adversary, leading to a *distributional* definition style. The sampler outputs a circuit C and circuit leakage lkg_C. The latter is given to both adversary and simulator

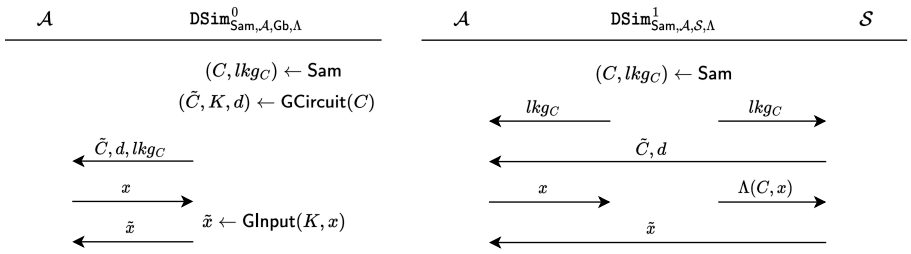


Fig. 2: Distributional simulation-based security games $\mathtt{DSim}^0_{\mathsf{Sam},\mathcal{A},\mathsf{Gb},\Lambda}$ and $\mathtt{DSim}^1_{\mathsf{Sam},\mathcal{A},\mathcal{S},\Lambda}$ for garbling scheme Gb, sampler Sam, adversary $\mathcal{A}$, simulator $\mathcal{S}$, and simulator leakage Λ.

and captures the partial information that $\mathcal{A}$ receives about C. An adversary may for instance be allowed to choose and learn which *class* of functions is garbled, e.g. the particular encryption scheme, but not the random choice of key. If on the other hand Sam samples from a distribution containing a single fixed circuit, then the adversary's knowledge is the same as in the existing security notion. We further augment the leakage so that it outputs a PPT oracle $\mathcal{O}$ which depends on the (secret) state of Sam. Oracle $\mathcal{O}$ models, e.g., encryption queries which depend on a secret key k that is also used in a circuit. Allowing such oracles supports composability, as we will see in Section 5, and it is similar to the interface which the ideal functionality exposes to the simulator in Canetti's universal composability framework [Can01]. For example, in our case this allows us to prove meaningful security of garbling an \$-RK-AE secure encryption scheme; here the \$-RK-AE security allows the adversary to make multiple queries to the encryption oracle, which would be impossible to model without our "free-form" oracle $\mathcal{O}$ that can contain the encryption oracle in this example use-case.

Modification 2: Relaxing the consistency requirement on $\mathcal{S}$. The relaxation to partial adversarial knowledge of C then makes it also natural to relax simulation requirements: If the partial knowledge an adversary gains about C and x does not allow to infer $C(x)$ but only some distribution of a possible output, then simulation may be with respect to this distribution instead of a particular output. We capture this observation through a new leakage function $\Lambda(C, x)$ which as before, is applied to obtain the simulator's input in the game's online phase. This leakage could be $C(x)$ (as in SIM), or empty (the other extreme) or anything in between, depending on what models the situation best (the less information we give to the simulator, the stronger the security guarantee by DSIM). In our \$-RK-AE encryption example $\Lambda(C, x) = \emptyset$, since as long as the key remains unknown to the adversary, a uniformly random output is indistinguishable from $enc(k, x)$ for any x. So, the simulator can model the output as a uniformly random string, without knowing the actual output. Interestingly, this gives the security notion a semi-adaptive flavour as the simulator does not depend on the input x at all.

DSIM security games. Figure 2 shows a slightly simplified version of our new security games as interaction between adversary $\mathcal{A}$, the security game, and a simulator $\mathcal{S}$. Note that, while consistency is relaxed in the security definition, the garbling scheme is still expected to provide the usual correctness guarantees.

Related work Our distributional security notion for garbling schemes is inspired by some existing techniques for sampling inputs in security games.

Distributional security notions. A distributional approach is the standard way to define security of deterministic public-key encryption schemes [BBO07]. In this context, an adversary that is allowed to choose the message m to be encrypted could simply re-encrypt m under the public key and compare with the challenge ciphertext, leading to a trivial attack. As discussed above, the AIKW lower bound is derived from a similar attack where the adversary can compare $C(x)$ to the output of the simulation. In both cases, the solution is to limit the adversary's knowledge, and define and prove security only for the scenario where the adversary knows the input *distribution*.

Another context where distributional security notions have been used before is zero knowledge [Gol93,DNRS99,JKKR17,Khu21]. Distributional zero knowledge and related notions relax the zero knowledge property by choosing the statement, i.e. the verifier's input to the computation, from an efficiently samplable distribution instead of universally quantifying over it. In the context of garbling schemes on the other hand, existing notions give the adversary the power to choose *all* inputs to the computation, and our relaxation to distributional security allows to hide some of the inputs from the adversary while maintaining a relaxed form of adversarial control over them.

Circuits samplers. Our definitional style uses a circuit sampler which is reminiscent of the treatment of Universal Computational Extractors (UCE) as abstraction of keyed hash functions by Bellare, Hoang and Keelveedhi [BHK13]. When using the DSIM definition with respect to the class of *output indistinguishable* circuits, one can view the DSIM notion as a notion of garbling schemes for *probabilistic functionalities* which is somewhat similar to the concept of probabilistic indistinguishability obfuscation (piO), a notion put forward by Canetti, Lin, Tessaro and Vaikuntanathan [CLTV15], where one should not be able to distinguish obfuscations of two computationally indistinguishable distributions $\mathcal{D}_0$ and $\mathcal{D}_1$. Our DSIM variant is different from piO in that DSIM is simulation-based and, since we study garbling schemes, the adversary only gets to see a *single* sample of the distribution.

2.2 Application: Distributed symmetric encryption

A distributed symmetric encryption protocol (DSE) allows multiple servers to jointly perform symmetric encryption of a message, such that each server holds

only a *share* of the key. The notion was formally introduced by Agrawal, Mohassel, Mukherjee and Rindal (AMMR [AMMR18]) and a construction for an arbitrary number of parties based on distributed PRFs was proposed.

Distributed symmetric encryption for arbitrary encryption schemes. For efficiency reasons, existing constructions have focused on distributed version of special-purpose encryption schemes [Muk20,ADL+22]. AMMR point out that while DSE can in principle be achieved from general-purpose MPC (which would allow to distribute any existing encryption scheme), this approach would be prohibitively expensive. Nevertheless, such a construction would be preferable when the encryption scheme to be used in an application is already fixed. In this case, efficiency can be improved in other ways, e.g. through a message-independent preprocessing phase.

In Section 5, we show a construction for the two-server case based on a garbling scheme. The idea is to split encryption into two phases: First (and possibly in advance), the encryption circuit is garbled. Then in the online phase, only the message needs to be garbled. This preprocessing strategy means that the garbling scheme needs some flavour of adaptive security and we show that our new distributional simulation-based security notion suffices.

Remark. Encryption is typically length-preserving, so one could use a garbling scheme whose online complexity is proportional to its output length also. However, if the same message was, say, encrypted under t keys into t ciphertexts, then we would have a length-expanding circuit whose pseudo-entropy grows with t. For simplicity of exposition, we will focus on the single-key version.

We show security in a simplified and more restricted model in comparison to AMMR: We assume a two-server setting where only server S_2 makes encryption queries. Server S_2 acts as evaluator in the garbling scheme, and we focus on showing security against a corrupt S_2 as this is the more difficult case. Decryption is assumed to be performed by a trusted party. Finally, we restrict the adversary to multiple indirect encryption queries but only a single direct encryption query in the terminology of AMMR.

Protocol overview. For two servers S_1 and S_2, the construction works as follows. Assume a symmetric authenticated encryption scheme se, a garbling scheme Gb with DSIM security with respect to output indistinguishable samplers such as authenticated encryption, and ideal oblivious transfer. In the basic construction, each server S_i holds an additive share k_i of the symmetric encryption key $k = k_1 \oplus k_2$. To encrypt a message m under se and k and randomness $r = r_1 \oplus r_2$, the servers act as follows: Server S_1 samples r_1 and garbles $\mathsf{enc}(k_1 \oplus \cdot, \cdot; r_1 \oplus \cdot)$. Both servers then run an OT protocol to garble the missing inputs k_2, m, and r_2. We show that the AE security of the symmetric encryption scheme se still holds against a malicious server S_2. We need to assume related-key security of the AE against linear functions since the malicious server can choose an arbitrary key k_1 to be xored on k_2.

For the case of encrypting the message under t different keys $k^1, \ldots, k^t$, the output of the circuit would consist of t ciphertexts. To compress the size of S_2's input, the key shares k_2^j can be computed by a PRG as $k_2^j = \mathsf{PRG}(s^j)$, and similarly for S_2's randomness. Then in the online phase, we only need to garble the two PRG seeds and the message, both of which are *independent* in size of t and hence the output size. See Section 5 for details on the single-key version of the protocol.

2.3 Bootstrapping $\mathbf{NC^0}$ to polynomial-size circuits

In the realm of obfuscation, *bootstrapping* is an established technique to turn positive results for a *limited* class of circuits into a positive result for arbitrary polynomial-size circuits. The bootstrapping techniques in the context of obfuscation follow the randomized encoding approach by Applebaum, Ishai, Kushilevitz [AIK04], see e.g. [GGH+13,Agr19] for examples. Assume we can obfuscate low-depth circuits. Instead of directly obfuscating a given high-depth circuit $C(x)$ for inputs x, a randomized encoding function is applied to the circuit. The result $\tilde{C}(x;r)$ is a new circuit that outputs a randomized encoding of $C(x)$. $\tilde{C}(x;r)$ uses additional randomness r and is chosen such that for random r, $\tilde{C}(x;r)$ does not leak more information than $y = C(x)$. If the encoding function itself has low depth, then $\tilde{C}(x;r)$ can now be obfuscated, and y can be recovered from $\tilde{C}(x;r)$ via a (high-depth, public) decoding function Dec.

Since randomized encodings and garbling schemes are different abstractions of the same underlying idea, it is natural to ask if the randomized encodings bootstrapping technique can be used to bootstrap adaptive security of garbling schemes from low-depth circuits to arbitrary polynomial-size circuits. That is, instead of directly garbling a circuit $C(x)$, we could try to (outer) garble the (inner) garbling function $\tilde{C}(x;r) := [\mathsf{GCircuit}(C(\cdot);r), \mathsf{GInput}(x)]$ with explicit hardcoded randomness r. Now, if the inner garbling scheme produces a low depth circuit $\tilde{C}$, the outer garbling scheme only needs to be secure for low-depth circuits, and, the inner garbling scheme only needs to be selectively secure, not adaptively. Now, the inner garbling scheme $\tilde{C}$ can be simply, e.g., Yao's garbling scheme, which is useful for two reasons: (1) Yao can be proven selectively secure assuming it uses $-IND-CPA symmetric encryption [LP09,BHR12b,BO23] and (2) Yao's garbling can be implemented in constant depth (see Lemma 1 for details).

However, proving SIM security in this setting is infeasible for the following reason. When we try to reduce the SIM security of the combined (garbling of garbling) garbling scheme to the SIM security of the outer garbling scheme, the adversarial evaluator chooses the circuit to be garbled, in this case, the inner garbling function $\tilde{C}(.;r)$ *with the hardcoded randomness* r. Adversary knowing r trivially compromises the security of the inner garbling.

Luckily, our DSIM security notion precisely allows to circumvent the randomness issue by dividing the adversary into two parts that do not share a state: first part samples the circuit from a suitable *output indistinguishable* distribution (e.g. $-IND-CPA secure encryption with a uniformly random key, see

Definition 8 for how exactly we define the natural property of output indistinguishability) and the second adversary interacts with the garbler. Hence, we can prove Theorem 1.

3 Preliminaries

All algorithms take as input the security parameter 1^λ. We write it explicitly for some algorithms, but leave it implicit for most algorithms. $a \leftarrow \mathsf{A}(x)$ assigns the result of an execution of the deterministic algorithm A on input x to variable a. $a \leftarrow\!\!\$\ \mathsf{A}(x)$ denotes the execution of a *randomized* algorithm. $a||b$ denotes the concatenation of two bit strings a and b. $\mathsf{A}_2 \circ \mathsf{A}_1$ denotes the composition of two algorithms in the form $\mathsf{A}_2(\mathsf{A}_1(.))$. We often write $\mathcal{A}^{\mathsf{O}_1,\mathsf{O}_2}$ for an adversary accessing oracle O_1 and O_2 and sometimes write $\mathcal{A}^{\mathsf{O}_1}_{\mathsf{O}_2}$ for conciseness. We sometimes use the notation $\mathcal{A}^{\mathsf{O}_1,..,\mathsf{O}_t} \Rightarrow \mathtt{G}^b$ inspired from state-separating proofs [BDF+18] to say that the adversary $\mathcal{A}$ is the main procedure who has access to oracles $\mathsf{O}_1, .., \mathsf{O}_t$ of a distinguishing game $\mathtt{G}^b$. $\Pr\left[1 = \mathcal{A}^{\mathsf{O}_1,..,\mathsf{O}_t} \Rightarrow \mathtt{G}^b\right]$ then refers to the probability that the adversary $\mathcal{A}$, after interacting with the oracles $\mathsf{O}_1, .., \mathsf{O}_t$ of $\mathtt{G}^b$ returns 1. We like this notation, because it makes both the name of the game and the adversary's oracle explicit, while the standard oracle subscript notation $\mathcal{A}^{\mathsf{O}_1,..,\mathsf{O}_t}$ only contains the oracles (but not the name of the game), and the experiment notation $\mathtt{G}^b_{\mathcal{A}}$ only contains the name of the game (but not the oracles).

3.1 Cryptographic primitives

For a symmetric encryption scheme se, \$-IND-CPA-security captures that ciphertexts are indistinguishable from random strings of the same length. The related-key authenticated encryption security game \$-RK-AE provides decryption queries in addition and allows adversarially chosen linear offsets on the key. Related-key security has been introduced by Bellare and Kohno [BK03].

$\underline{\$\text{-}\mathtt{IND\text{-}CPA}^b_{\mathsf{se}}(1^\lambda)}$	$\underline{\$\text{-}\mathtt{RK\text{-}AE}^b_{\mathsf{se}}(1^\lambda)}$	
$\underline{\mathsf{ENC}(m)}$	$\underline{\mathsf{ENC}(m, \Delta)}$	$\underline{\mathsf{DEC}(c, \Delta)}$
if $k = \bot$	**if** $k = \bot$	**if** $b = 1$
$\quad k \leftarrow\!\!\$\ \{0,1\}^\lambda$	$\quad k \leftarrow\!\!\$\ \{0,1\}^\lambda$	$\quad$ **return** $\bot$
$c \leftarrow\!\!\$\ \mathsf{enc}(k, m)$	$c \leftarrow\!\!\$\ \mathsf{enc}(k \oplus \Delta, m)$	**if** $k = \bot$
	$S \leftarrow S \cup \{c\}$	$\quad k \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $b = 1$	**if** $b = 1$	**assert** $c \notin S$
$\quad c \leftarrow\!\!\$\ \{0,1\}^{\lvert c\rvert}$	$\quad c \leftarrow\!\!\$\ \{0,1\}^{\lvert c\rvert}$	$m \leftarrow \mathsf{dec}(k \oplus \Delta, c)$
return c	**return** c	**return** m

Fig. 3: Games $\$\text{-}\mathtt{IND\text{-}CPA}^b_{\mathsf{se}}(1^\lambda)$ and $\$\text{-}\mathtt{RK\text{-}AE}^b_{\mathsf{se}}(1^\lambda)$.

Definition 1 (\$-IND-CPA and \$-RK-AE Security). *A symmetric encryption scheme* $\mathsf{se} = (\mathsf{enc}, \mathsf{dec})$ *is indistinguishable under chosen plaintext attacks (\$-IND-CPA) if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\$\text{-}\mathtt{IND\text{-}CPA}}_{\mathsf{se},\mathcal{A}}(1^\lambda) :=$

$$\left|\Pr\left[1 = \mathcal{A}(1^\lambda) \overset{\mathsf{ENC}}{\Rightarrow} \$\text{-}\mathtt{IND\text{-}CPA}^0(1^\lambda)\right] - \Pr\left[1 = \mathcal{A}(1^\lambda) \overset{\mathsf{ENC}}{\Rightarrow} \$\text{-}\mathtt{IND\text{-}CPA}^1(1^\lambda)\right]\right|$$

is negligible in λ. se *is authenticated encryption under linear related key attacks if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathsf{Adv}^{\$\text{-RK-AE}}_{\mathsf{se},\mathcal{A}}(1^\lambda) :=$

$$\left|\Pr\left[1 = \mathcal{A}(1^\lambda)^{\ \mathsf{ENC},\mathsf{DEC}} \longrightarrow \$\text{-RK-AE}^0(1^\lambda)\right] - \Pr\left[1 = \mathcal{A}(1^\lambda)^{\ \mathsf{ENC},\mathsf{DEC}} \longrightarrow \$\text{-RK-AE}^1(1^\lambda)\right]\right|$$

is negligible in λ.

3.2 Garbling schemes

Definition 2 (Garbling scheme). *A Garbling scheme* Gb *consists of three PPT algorithms* $(\mathsf{GCircuit}, \mathsf{GInput}, \mathsf{GEval})$ *with the following syntax:*

- $(\tilde{C}, K, d) \leftarrow\!\!\$\ \mathsf{GCircuit}(C)$: *the garbling algorithm takes as input a circuit $C : \{0,1\}^n \to \{0,1\}^m$, and outputs a garbled circuit $\tilde{C}$, keys K and output decoding information d. m and n are polynomials in λ.*
- $\tilde{x} \leftarrow\!\!\$\ \mathsf{GInput}(K, x)$: *the input garbling algorithm takes an input $x \in \{0,1\}^n$ and the keys K and outputs an encoding of the input $\tilde{x}$.*
- $y \leftarrow \mathsf{GEval}(\tilde{C}, \tilde{x}, d)$: *the evaluation algorithm takes as input the garbled circuit, a garbled input, the output encoding information d and returns a value $y \in \{0,1\}^m$, where m is the output length of C.*

Correctness. *of the garbling scheme holds if for any λ, any circuit C and any input $x \in \{0,1\}^n$ we have*

$$\Pr\left[C(x) = \mathsf{GEval}(\tilde{C}, \tilde{x}, d)\right] = 1 - \mathsf{negl}(\lambda),$$

where $(\tilde{C}, K, d) \leftarrow\!\!\$\ \mathsf{GCircuit}(C)$, $\tilde{x} \leftarrow\!\!\$\ \mathsf{GInput}(K, x)$.

We first define *selective* simulation security, where the adversary chooses circuit C and input x at the same time. We define adaptive garbling scheme security in Section 4, where we compare it to the DSIM definition, which we propose.

$\mathtt{SelSim}^0_{\mathcal{A},\mathsf{Gb}}(1^\lambda)$	$\mathtt{SelSim}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)$
$(C, x, \mathrm{st}) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$	$(C, x, \mathrm{st}) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$(\tilde{C}, K, d) \leftarrow\!\!\$\ \mathsf{GCircuit}(C)$	$(\tilde{C}, \tilde{x}, d) \leftarrow\!\!\$\ \mathcal{S}(\Phi(C), C(x))$
$\tilde{x} \leftarrow\!\!\$\ \mathsf{GInput}(K, x)$	$b' \leftarrow\!\!\$\ \mathcal{A}(\tilde{C}, \tilde{x}, d, \mathrm{st})$
$b' \leftarrow\!\!\$\ \mathcal{A}(\tilde{C}, \tilde{x}, d, \mathrm{st})$	**return** b'
return b'	

Fig. 4: Experiments $\mathtt{SelSim}^0_{\mathcal{A},\mathsf{Gb}}$ and $\mathtt{SelSim}^1_{\mathcal{A},\mathcal{S}}$.

Definition 3 (SelSim security). *Let Φ be a leakage function. We say that* Gb *is* ***selectively simulation secure*** *(SelSim) if for any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that*

$$\mathsf{Adv}^{\mathsf{SelSim}}_{\mathcal{A},\mathcal{S},\mathsf{Gb}}(1^\lambda) := \left|\Pr\left[1 = \mathtt{SelSim}^0_{\mathcal{A},\mathcal{S},\mathsf{Gb}}\right] - \Pr\left[1 = \mathtt{SelSim}^1_{\mathcal{A},\mathcal{S}}\right]\right| = \mathsf{negl}(\lambda),$$

where experiments $\mathtt{SelSim}^0_{\mathcal{A},\mathsf{Gb}}$ and $\mathtt{SelSim}^1_{\mathcal{A},\mathcal{S}}$ are specified in Figure 4.

4 Distributional simulation-based security (DSIM)

In this section, we first define the simulation-based and indistinguishability flavours of adaptive security and then present our new definition of adaptive distributional simulation-based security (DSIM) for garbling schemes.

$\underline{\mathtt{Sim}^0_{\mathcal{A},\mathsf{Gb}}(1^\lambda)}$
$(C, \mathrm{st}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
$(\tilde{C}, K, d) \leftarrow\!\$\ \mathsf{GCircuit}(C)$
$(x, \mathrm{st}') \leftarrow\!\$\ \mathcal{A}(\tilde{C}, d, \mathrm{st})$
$\tilde{x} \leftarrow\!\$\ \mathsf{GInput}(K, x)$
$b' \leftarrow\!\$\ \mathcal{A}(\tilde{x}, \mathrm{st}')$
return b'

$\underline{\mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)}$
$(C, \mathrm{st}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
$(\tilde{C}, d, \mathrm{st}_S) \leftarrow\!\$\ \mathcal{S}(\Phi(C))$
$(x, \mathrm{st}') \leftarrow\!\$\ \mathcal{A}(\tilde{C}, d, \mathrm{st})$
$\tilde{x} \leftarrow\!\$\ \mathcal{S}(C(x), \mathrm{st}_S)$
$b' \leftarrow\!\$\ \mathcal{A}(\tilde{x}, \mathrm{st}')$
return b'

$\underline{\mathtt{Ind}^b_{\mathcal{A},\mathsf{Gb}}(1^\lambda)}$
$(C_0, C_1, \mathrm{st}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
$(\tilde{C}, K, d) \leftarrow\!\$\ \mathsf{GCircuit}(C_b)$
$(x_0, x_1, \mathrm{st}') \leftarrow\!\$\ \mathcal{A}(\tilde{C}, d, \mathrm{st})$
$\tilde{x} \leftarrow\!\$\ \mathsf{GInput}(K, x_b)$
$b' \leftarrow\!\$\ \mathcal{A}(\tilde{x}, \mathrm{st}')$
return b'

$\underline{\mathtt{DSim}^0_{\mathsf{Sam},\mathcal{A},\mathsf{Gb},\Lambda,\mathsf{Filter}}(1^\lambda)}$
$(C, \mathrm{lkg}_C, \mathcal{O}) \leftarrow\!\$\ \mathsf{Sam}(1^\lambda)$
$(\tilde{C}, K, d) \leftarrow\!\$\ \mathsf{GCircuit}(C)$
$(x, \mathrm{st}_\mathcal{A}) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_C)$
$\tilde{x} \leftarrow\!\$\ \mathsf{GInput}(K, x)$
$b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_{C(x)}}(\tilde{x}, \mathrm{st}_\mathcal{A})$
return b'

$\underline{\mathtt{DSim}^1_{\mathsf{Sam},\mathcal{A},\mathcal{S},\Lambda,\mathsf{Filter}}(1^\lambda)}$
$(C, \mathrm{lkg}_C, \mathcal{O}) \leftarrow\!\$\ \mathsf{Sam}(1^\lambda)$
$(\tilde{C}, d, \mathrm{st}_S) \leftarrow\!\$\ \mathcal{S}^{\mathcal{O}}(\mathsf{Filter}(\mathrm{lkg}_C))$
$(x, \mathrm{st}_\mathcal{A}) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_C)$
$\tilde{x} \leftarrow\!\$\ \mathcal{S}^{\mathcal{O}}(\Lambda(\mathrm{st}_\mathcal{A}), \mathrm{st}_S)$
$b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_{\tilde{C}(\tilde{x})}}(\tilde{x}, \mathrm{st}_\mathcal{A})$
return b'

Fig. 5: Experiments $\mathtt{Ind}^b_{\mathcal{A},\mathsf{Gb}}$, $\mathtt{Sim}^0_{\mathcal{A},\mathsf{Gb}}$, $\mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}$, $\mathtt{DSim}^0_{\mathsf{Sam},\mathcal{A},\mathsf{Gb},\Lambda,\mathsf{Filter}}(1^\lambda)$ and $\mathtt{DSim}^1_{\mathsf{Sam},\mathcal{A},\mathcal{S},\Lambda,\mathsf{Filter}}(1^\lambda)$.

Definition 4 (Adaptive SIM security). *Let Φ be a leakage function. We say that* Gb *is adaptively SIM$_\Phi$-secure if for any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that*

$$\mathsf{Adv}^{\mathsf{Sim}}_{\mathcal{A},\mathcal{S},\mathsf{Gb}}(1^\lambda) := \left|\Pr\left[1 = \mathtt{Sim}^0_{\mathcal{A},\mathcal{S},\mathsf{Gb}}\right] - \Pr\left[1 = \mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}\right]\right| = \mathsf{negl}(\lambda),$$

where experiments $\mathtt{Sim}^0_{\mathcal{A},\mathsf{Gb}}$ and $\mathtt{Sim}^1_{\mathcal{A},\mathcal{S}}$ are specified in Figure 5.

Definition 5 (Adaptive IND security). *We say that* Gb *is* ***adaptively IND-secure*** *if for any PPT adversary $\mathcal{A}$ which queries circuits C_0 and C_1 with equal input length and inputs x_0 and x_1 such that $C_0(x_0) = C_1(x_1)$ and $\Phi(C_0) = \Phi(C_1)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that:*

$$\mathsf{Adv}^{\mathsf{Ind}}_{\mathcal{A},\mathsf{Gb}}(1^\lambda) := \left|\Pr\left[1 = \mathtt{Ind}^0_{\mathcal{A},\mathsf{Gb}}\right] - \Pr\left[1 = \mathtt{Ind}^1_{\mathcal{A},\mathsf{Gb}}\right]\right| = \mathsf{negl}(\lambda),$$

where the experiment $\mathtt{Ind}^b_{\mathcal{A},\mathsf{Gb}}$ is specified in Figure 5.

Definition 6 (Sampler classes). *A class of samplers $\mathcal{C}$ is a set of PPT adversaries* Sam *such that* $(C, lkg_C, \mathcal{O}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$.

Definition 7 (Adaptive distributional SIM security (DSIM)). *Let $\mathcal{C}$ be a sampler class and Λ be a leakage-function. Garbling scheme* Gb *is* $DSim_{\Lambda,\mathsf{Filter}}[\mathcal{C}]$*-secure if for any PPT* $\mathsf{Sam} \in \mathcal{C}$ *and any PPT $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that* $\mathsf{Adv}^{\mathsf{DSim}}_{\mathsf{Sam},\mathcal{A},\mathcal{S},\mathsf{Gb}}(1^\lambda) :=$

$$\left|\Pr\left[1 = \mathtt{DSim}^0_{\mathsf{Sam},\mathcal{A},\mathsf{Gb},\Lambda,\mathsf{Filter}}(1^\lambda)\right] - \Pr\left[1 = \mathtt{DSim}^1_{\mathsf{Sam},\mathcal{A},\mathcal{S},\Lambda,\mathsf{Filter}}(1^\lambda)\right]\right|$$

is negligible, where Figure 5 defines the experiments $\mathtt{DSim}^0_{\mathsf{Sam},\mathcal{A},\mathsf{Gb},\Lambda,\mathsf{Filter}}(1^\lambda)$ *and* $\mathtt{DSim}^1_{\mathsf{Sam},\mathcal{A},\mathcal{S},\Lambda,\mathsf{Filter}}(1^\lambda)$.

In the above DSIM definition lkg_C could be e.g. the topology of the circuit C (to match the SIM definition). Alternatively, if we are garbling e.g. an encryption scheme with the key hardcoded in the circuit C, then lkg_C could be the circuit C without the hardcoded random key. $\mathcal{O}$ is a (possibly stateful) PPT oracle which might be, for example, an encryption oracle which depends on the key k embedded into C, $\mathcal{O}_{C(x)}$ might additionally depend on $C(x)$, e.g., when forbidding to send $C(x)$ to a decryption oracle, see Section 5 for an example. See Section 2.1 for more intuition behind the different parameters.

In the DSIM definition, the grey parameters (Λ, Filter) are optional and can be ignored (i.e. Filter can be thought of as identity and Λ as empty) when reading this paper, that is, they are not needed for understanding the bootstrapping proof or DENC example. They are included just as an example on how to extend the definition and to draw connection to SIM definition.

Remark on the optional parameters. W.l.o.g. we consider that $\mathrm{st}_\mathcal{A}$ contains variables for lkg_C and x, and, if lkg_C allows to compute C, then we also assume that w.l.o.g., lkg_C and $\mathrm{st}_\mathcal{A}$ contain a variable for C which contains the correct value. This will later allow us to discuss specific leakage functions.

In the special case when the sampler leaks $\mathrm{lkg}_C := C$ and $\Lambda(\mathrm{st}_\mathcal{A}) = C(x)$, SIM implies DSIM, because the adversary has slightly less information in DSIM than in SIM (since it does not know the randomness used for sampling), and additionally, the simulator $\mathcal{S}$ is stronger in DSIM.

Theorem 2 (SIM implies DSIM). *Let Φ be a polynomial-time computable leakage function. If a garbling scheme* Gb *is SIM_Φ-secure, then* Gb *is* $DSim_{\Lambda,\mathsf{Filter}}[\mathcal{C}]$*-secure, where $\mathcal{C} = \{PPT\ \mathsf{Sam} : (C, lkg_C := C) \leftarrow\!\!\$\ \mathsf{Sam}\}$, $\Lambda(st_\mathcal{A}) = C(x)$ and* $\mathsf{Filter}(lkg_C)$ *is such that $\Phi(lkg_C)$ can be computed given* $\mathsf{Filter}(lkg_C)$.

Proof. Let Gb be a SIM_Φ-secure garbling scheme. Assume towards contradiction that there is $\mathsf{Sam} \in \mathcal{C}$ and PPT $\mathcal{A}$ s.t. for all PPT $\mathcal{S}$ $\mathsf{Adv}^{\mathsf{DSim}}_{\mathsf{Sam},\mathcal{A},\mathcal{S},\mathsf{Gb}}(1^\lambda)$ is non-negligible. Define the first stage of the SIM-adversary as $\mathcal{A}'(1^\lambda) := \mathsf{Sam}(1^\lambda)$ and $\mathcal{A}'$ is $\mathcal{A}$ for all further stages. Let $\mathcal{S}'$ be the simulator ensured by SIM-security such that $\mathsf{Adv}^{\mathsf{Sim}}_{\mathcal{A}',\mathcal{S}',\mathsf{Gb}}(1^\lambda)$ is negligible.

Now, define the first stage simulator as $\mathcal{S}(\mathsf{Filter}(\mathrm{lkg}_C)) = \mathcal{S}'(\Phi(\mathrm{lkg}_C))$ and the second stage simulator as $\mathcal{S}(\cdot,\cdot) := \mathcal{S}'(\cdot,\cdot)$ and $\mathcal{A}'$ is $\mathcal{A}$ for all other inputs. Now $\mathsf{Adv}^{\mathsf{Sim}}_{\mathcal{A}',\mathcal{S}',\mathsf{Gb}}(1^\lambda) = \mathsf{Adv}^{\mathsf{DSim}}_{\mathsf{Sam},\mathcal{A},\mathcal{S},\mathsf{Gb}}(1^\lambda)$, and we reached a contradiction. □

We can also recover the other direction, DSIM implies SIM, if we choose lkg_C to be the sampler's full state and choose $\mathsf{Filter}(\mathrm{lkg}_C) = \Phi(C)$ and $\Lambda(\mathrm{st}_{\mathcal{A}}) = C(x)$.

Note however, that the core difference between SIM and DSIM is the idea of *sampler classes* which *restrict* the information lkg which is passed from Sam to $\mathcal{A}$[6]. Recall that this restriction models that in garbling scheme application such as private function evaluation, the party choosing the input might have *some* information about the function to be evaluated, but not *all* information—otherwise one party could simply send its function to the other party. Of particular interest to us are circuits with embedded *cryptographic keys* which make the output of the evaluated circuit *indistinguishable* from a distribution which could have been chosen *independently* of the input to the circuit, as, e.g., in IND-CPA-secure encryption where an encryption of a message m is indistinguishable from an encryption of $0^{|m|}$. To be useful for garbling security, we need the indistinguishable distribution to be generated by a circuit of the same size. Therefore, we demand that the function distribution consists of a fixed circuit C for which only the *randomness* is sampled, i.e., $C := C(\cdot\,; r)$ is a circuit with randomness r hardcoded into it.

$\underline{\mathsf{Sam}(1^\lambda)}$

$r \leftarrow\!\!\$\ \{0,1\}^{p(\lambda)}$
$C_r \leftarrow C_\lambda(.,r)$
$(\mathrm{lkg}_{C_r}, \mathcal{O}) \leftarrow \mathsf{Sam}_{\mathrm{leak}}(r)$
return $C_r, \mathrm{lkg}_{C_r}, \mathcal{O}$

$\underline{\mathsf{Out}^b_{\mathsf{Sam},\mathcal{D}}(1^\lambda)}$

$C_r, \mathrm{lkg}_{C_r}, \mathcal{O} \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
$(x_0, \mathrm{st}) \leftarrow\!\!\$\ \mathcal{D}^{\mathcal{O}}(1^\lambda, \mathrm{lkg}_{C_r})$
$x_1 \leftarrow 0^{|x_0|}$
$b^* \leftarrow\!\!\$\ \mathcal{D}^{\mathcal{O}_{C_r(x_b)}}(C_r(x_b), \mathrm{st})$
return b^*

Fig. 6: Output indistinguishability

Definition 8 (Output Indistinguishable Sampler). *We define the class* $\mathcal{C}^{out}$ *of* output indistinguishable *samplers as the set of PPT* Sam *such that*

Fixed circuit *there exists a circuit* $C = (C_\lambda)_\lambda$*, polynomial* p *and PPT* Sam_{leak} *such that* Sam *can be written as in Fig. 6 (left), and*

Output indistinguishability *for all PPT distinguishers* $\mathcal{D}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{Out}}_{\mathsf{Sam},\mathcal{D}}(1^\lambda) := \left|\Pr\left[1 = \mathsf{Out}^0_{\mathsf{Sam},\mathcal{D}}(1^\lambda)\right] - \Pr\left[1 = \mathsf{Out}^1_{\mathsf{Sam},\mathcal{D}}(1^\lambda)\right]\right| = \mathsf{negl}(\lambda),$$

where Fig. 6 defines $\mathsf{Out}^0_{\mathsf{Sam},\mathcal{D}}(1^\lambda)$ *and* $\mathsf{Out}^1_{\mathsf{Sam},\mathcal{D}}(1^\lambda)$*.*

Remark. Note that $\mathcal{D}$ does not need to receive any leakage about C since the circuit is fixed. We use the notation lkg_{C_r} just to be consistent with definition of DSIM.

5 Distributed Symmetric Encryption (DSE)

This section shows that DSIM security with respect to admissible samplers implies useful security properties on the example of distributed encryption that

[6] hence, in most cases where DSIM is a meaningful notion, Filter should just be identity.

was introduced in Section 2. For simplicity of exposition, we focus on the case of two servers and a single message. See Section 2 for further introduction to our example and a discussion on extending the example into a length-expanding case while maintaining the same online complexity.

Remember that we are in a setting with two servers who hold secret shares k_1, k_2 of a symmetric key $k = k_1 \oplus k_2$, and who wish to perform distributed encryptions under k. One way to implement such a protocol is by combining a garbling scheme with an oblivious transfer protocol to obtain a two-party computation protocol for evaluating $\mathsf{enc}(k_1 \oplus k_2, x)$ for message $x \in \{0,1\}^\lambda$.

Below, we describe the protocol between two stateful servers. We refer to the protocol below as *distributed symmetric encryption protocol (DSE)*, using a symmetric encryption scheme se and a garbling scheme Gb. We keep state implicit for conciseness and abstract away the OT protocol. DSE uses a trusted party in the setup phase who generates and secret-shares a symmetric key k.

Phase I	Phase II	
Server 1	Server 2	Server 2
Input: $k_1, 1^\lambda$	Input: $k_2, \text{msg}, \tilde{C}, d, 1^\lambda$	Input: $\tilde{x}, \tilde{C}, d, 1^\lambda$
$r_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$r_2 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$x' \leftarrow \mathsf{Gb.GEval}(\tilde{C}, \tilde{x}, d)$
$C \leftarrow \mathsf{se.enc}(k_1 \oplus \cdot, \cdot; r_1 \oplus \cdot)$	$x \leftarrow k_2 \| \text{msg} \| r_2$	**return** x'
$(\tilde{C}, K, d) \leftarrow\!\!\$\ \mathsf{Gb.GCircuit}(C)$	**return** x	
return $(\tilde{C}, d)$		

Security. The protocol shall provide security in a setting where the adversary obtains the key of one of the two servers. We here focus on corruption of the key of Server 2 and model security in this case. The security notion we consider is \$-AE security in the presence of the above distributed protocol under the same key. In this toy example, we consider only a single corrupted execution of the protocol.

Security is defined as indistinguishability of $\mathtt{Denc}^0_{\mathcal{A}}(1^\lambda)$ and $\mathtt{Denc}^1_{\mathcal{A}}(1^\lambda)$ in Fig. 7. The game starts by sampling key shares k_1 and k_2 and computing a garbled circuit $\tilde{C}$ and d, either honestly or via simulator $\mathcal{S}$. The adversary $\mathcal{A}$, acting as corrupted Server 2, is then given $\tilde{C}$, d and k_2 and outputs a message msg to be encrypted. The game proceeds by garbling this message together with encryption randomness r_2 and providing this $\tilde{x}$ to $\mathcal{A}$. At every step, $\mathcal{A}$ has access to encryption and decryption oracles, modeling honest executions run concurrently. Finally, $\mathcal{A}$ outputs a guess b'.

Definition 9 (Distributed Encryption Security (DENC)). *Let* se *be a symmetric encryption scheme where ciphertexts are twice as long as the plaintexts. Let* Gb *be a garbling scheme. Then, the DSE protocol using* se *and* Gb *is DENC-secure if for all PPT adversaries* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$ *such that*

$$\left|\Pr\left[1 = \mathtt{Denc}^0_{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 = \mathtt{Denc}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)\right]\right|$$

is negligible, where Fig. 7 defines $\mathtt{Denc}^0_{\mathcal{A}}(1^\lambda)$ *and* $\mathtt{Denc}^1_{\mathcal{A}}(1^\lambda)$.

$\mathtt{Denc}^0_{\mathcal{A}}(1^\lambda)$	$\mathtt{Denc}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)$	$\mathtt{Hybrid}_{\mathcal{A}}(1^\lambda)$
$k_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$k_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$k_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$k_2 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$k_2 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$k_2 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$k \leftarrow k_1 \oplus k_2$	$k \leftarrow k_1 \oplus k_2$	$k \leftarrow k_1 \oplus k_2$
$r_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$		
$C \leftarrow \mathsf{enc}(k_1 \oplus \cdot, \cdot; r_1 \oplus \cdot)$		
$(\tilde{C}, K, d) \leftarrow\!\!\$\ \mathsf{GCircuit}(C)$	$(\tilde{C}, d) \leftarrow\!\!\$\ \mathcal{S}(1^\lambda)$	$(\tilde{C}, d) \leftarrow\!\!\$\ \mathcal{S}^{\mathsf{ENC},\mathsf{DEC}}_{\text{hybrid}}(1^\lambda)$
$\text{msg} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC}}_{\mathsf{DEC}}(1^\lambda, k_2, \tilde{C}, d)$	$\text{msg} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC}}_{\mathsf{DEC}}(1^\lambda, k_2, \tilde{C}, d)$	$\text{msg} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC}}_{\mathsf{DEC}}(1^\lambda, k_2, \tilde{C}, d)$
$r_2 \leftarrow\!\!\$\ \{0,1\}^\lambda$		
$x \leftarrow k_2 \| \text{msg} \| r_2$		
$\tilde{x} \leftarrow\!\!\$\ \mathsf{GInput}(K, x)$	$\tilde{x} \leftarrow\!\!\$\ \mathcal{S}(1^\lambda)$	$\tilde{x} \leftarrow\!\!\$\ \mathcal{S}^{\mathsf{ENC},\mathsf{DEC}}_{\text{hybrid}}(1^\lambda)$
$S \leftarrow S \cup \{\tilde{C}(\tilde{x})\}$		$S \leftarrow S \cup \{\tilde{C}(\tilde{x})\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC}}_{\mathsf{DEC}}(1^\lambda, \tilde{x})$	$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC}}_{\mathsf{DEC}}(1^\lambda, \tilde{x})$	$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC}}_{\mathsf{DEC}}(1^\lambda, \tilde{x})$
return b'	**return** b'	**return** b'
$\mathsf{ENC}(m)$	$\mathsf{ENC}(m)$	$\mathsf{ENC}(m)$
assert $\|m\| = \lambda$	**assert** $\|m\| = \lambda$	**assert** $\|m\| = \lambda$
$c \leftarrow\!\!\$\ \mathsf{enc}(k, m)$	$c \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$	$c \leftarrow\!\!\$\ \mathsf{enc}(k, m)$
$S \leftarrow S \cup \{c\}$		$S \leftarrow S \cup \{c\}$
return c	**return** c	**return** c
$\mathsf{DEC}(c)$	$\mathsf{DEC}(c)$	$\mathsf{DEC}(c)$
assert $c \notin S$		**assert** $c \notin S$
$m \leftarrow \mathsf{dec}(k, c)$		$m \leftarrow \mathsf{dec}(k, c)$
return m	**return** $\perp$	**return** m

Fig. 7: Security games (left and middle) for DENC-security and hybrid game (right) for Theorem 3. The adversary $\mathcal{A}$ and simulators $\mathcal{S}$ and $\mathcal{S}_{\text{hybrid}}$ are *stateful*. We leave their state implicit for conciseness of notation.

Remark. The purpose of the ENC oracle is to model a weaker form of concurrent security. To closely model the real-life scenario of distributed encryption, it might seem useful to replace the ENC oracle also by a circuit garbling oracle. We remark that this would require studying concurrent composition of DSIM which is a non-trivial question. Concretely, when composing the same garbled circuit

$\underline{\mathsf{Sam}(1^\lambda)}$

$k_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$

$k_2 \leftarrow\!\!\$\ \{0,1\}^\lambda$

$k \leftarrow k_1 \oplus k_2$

$r_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$

$C \leftarrow \mathsf{enc}(k_1 \oplus \cdot, \cdot; r_1 \oplus \cdot)$

$\mathrm{lkg}_C \leftarrow k_2$

$\mathcal{O} \leftarrow \mathsf{ENC}, \mathsf{DEC}$

return $(C, \mathrm{lkg}_C, \mathcal{O})$

$\underline{\underline{\mathcal{O}_{C(x)}}}$

$\underline{\mathsf{ENC}(m)}$

assert $|m| = \lambda$

$c \leftarrow\!\!\$\ \mathsf{enc}(k, m)$

$S \leftarrow S \cup \{c\}$

return c

$\underline{\mathsf{DEC}(c)}$

assert $|m| = \lambda$

assert $c \notin S \cup \{C(x)\}$

$m \leftarrow\!\!\$\ \mathsf{dec}(k, c)$

return m

$\underline{\mathcal{C}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_C)}$

$k_2 \leftarrow \mathrm{lkg}_C$

$\mathrm{msg}, \mathrm{st} \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}(1^\lambda, k_2, \tilde{C}, d)$

$x \leftarrow \mathrm{msg}$

return (x, st)

$\underline{\mathcal{C}^{\mathcal{O}_{C(x)}}(\tilde{x}, \mathrm{st})}$

$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{C(x)}}(\tilde{x}, \mathrm{st})$

return b'

Fig. 8: Sampler Sam that emulates the sampling of $\mathtt{Denc}^0(1^\lambda)$ and a distinguisher $\mathcal{C}$ which just runs $\mathcal{A}$.

multiple times in parallel (with different, adversarially chosen inputs), we need to prove a hybrid argument which replaces real garbled circuits by simulated garbled circuits successively. When reducing one step in the hybrid argument to DSIM security, the other garbled circuits become leakage in the DSIM game. However, we need to prove that this additional leakage does not jeopardize security. Unfortunately, proving this property about the leakage requires another reduction to DSIM security. In order to avoid circular reasoning, new techniques are necessary to establish self-composability results for DSIM.

Theorem 3. *If* se *is an authenticated encryption scheme in the presence of linear related-key attacks (\$-RK-AE-secure) and* Gb *is DSim*$[\mathcal{C}^{Out}]$*-secure, then DSE is DENC-secure.*

Remark. We write DSim$[\mathcal{C}^{Out}]$ *to denote DSim*$_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{Out}]$ *where* Λ *is empty and* Filter *is identity. That is, the parameters* Λ *and* Filter *are not needed for this section.*

Proof. The proof of Theorem 3 proceeds via two high-level game-hops, see Fig. 7 (right) for the hybrid game between $\mathtt{DENC}^0_{\mathcal{A}}(1^\lambda)$ and $\mathtt{DENC}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)$. The first game hop from $\mathtt{DENC}^0_{\mathcal{A}}(1^\lambda)$ to $\mathtt{Hybrid}_{\mathcal{A}}(1^\lambda)$ reduces t to DSim$[\mathcal{C}^{\mathrm{Out}}]$-security, and the second game-hop from $\mathtt{Hybrid}_{\mathcal{A}}(1^\lambda)$ to $\mathtt{DENC}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)$ reduces to RK-AE security. We first provide the reduction for the 2nd game-hop, since it is easier than the first game-hop.

$\mathtt{Hybrid}_{\mathcal{A}}(1^\lambda)$ to $\mathtt{DENC}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)$: For any PPT simulator $\mathcal{S}_{\mathrm{hybrid}}$, we define a PPT simulator $\mathcal{S}$ as follows: The simulator $\mathcal{S}$ runs $\mathcal{S}_{\mathrm{hybrid}}$, but answers its ENC queries with random strings of length 2λ and its DEC queries with $\perp$. Now, to reduce the

indistinguishability of $\texttt{Hybrid}_{\mathcal{A}}(1^\lambda)$ and $\texttt{DENC}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)$ to \$-RK-AE, observe that k_1 is not used and thus, k_1 is perfectly random as required. Now, assume toward contradiction that there exists a pair of PPT algorithms $(\mathcal{A}, \mathcal{S}_{\text{hybrid}})$ such that the difference between $\Pr\left[1 = \texttt{Hybrid}_{\mathcal{A}}(1^\lambda)\right]$ and $\Pr\left[1 = \texttt{DENC}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)\right]$ is non-negligible, where $\mathcal{S}$ is derived from $\mathcal{S}_{\text{hybrid}}$ as previously described. Then, we can construct the following adversary $\mathcal{B}$ against \$-RK-AE: $\mathcal{B}$ emulates $\texttt{Hybrid}_{\mathcal{A}}(1^\lambda)$, except for all ENC and DEC queries which it forwards to its \$-RK-AE-game (with Δ being the all-zeroes string). $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. By construction, we have that

$$\Pr\left[1 = \mathcal{A}(1^\lambda)^{\mathsf{ENC},\mathsf{DEC}} \texttt{\$-RK-AE}^0(1^\lambda)\right] = \Pr\left[1 = \texttt{Hybrid}_{\mathcal{A}}(1^\lambda)\right].$$

Additionally, we claim that

$$\Pr\left[1 = \mathcal{A}(1^\lambda)^{\mathsf{ENC},\mathsf{DEC}} \texttt{\$-RK-AE}^1(1^\lambda)\right] = \Pr\left[1 = \texttt{DENC}^1_{\mathcal{A},\mathcal{S}}(1^\lambda)\right].$$

Namely, in both cases, the ENC queries of $\mathcal{A}$ are answered by random strings of length 2λ, and the DEC queries are answered by $\perp$, and the game is *stateless*, as there is no set S of previously obtained ciphertexts. Therefore, it does not matter whether queries to ENC and DEC are forwarded to the $\texttt{\$-RK-AE}^1$ game (as done by $\mathcal{B}$) or simulated as random answers and $\perp$ answers, respectively, without forwarding (as done by $\mathcal{S}$). Thus, we can conclude $\mathsf{Adv}^{\texttt{\$-RK-AE}}_{\mathsf{se},\mathcal{A}}(1^\lambda)$ is non-negligible and reach a contradiction.

$\texttt{DENC}^0_{\mathcal{A}}(1^\lambda)$ to $\texttt{Hybrid}_{\mathcal{A}}(1^\lambda)$: Assume towards contradiction that $\mathcal{A}$ is a PPT adversary which has non-negligible advantage in distinguishing between $\texttt{DENC}^0_{\mathcal{A}}(1^\lambda)$ and $\texttt{Hybrid}_{\mathcal{A},\mathcal{S}_{\text{hybrid}}}(1^\lambda)$, regardless of how we instantiate $\mathcal{S}_{\text{hybrid}}$. Now, we construct an adversary $(\mathsf{Sam}(1^\lambda), \mathcal{C})$ against DSim$[\mathcal{C}^{\text{Out}}]$-security. $(\mathsf{Sam}(1^\lambda), \mathcal{C})$ is shown in Fig. 8. We first prove that Sam is output indistinguishable.

Claim 1. *If* se *is an* \$*-RK-AE, then* $\mathsf{Sam} \in \mathcal{C}^{Out}$.

Written with explicit randomness $r := k_1||k_2||r_1$ of length $p(\lambda) := 3\lambda$, circuit $C_\lambda(.,r) := \mathsf{enc}(k_1 \oplus \cdot, \cdot; r_1 \oplus \cdot)$ and $\mathsf{Sam}_{\text{leak}}(r)$ which returns $\text{lkg}_C \leftarrow k_2$ and $\mathcal{O} \leftarrow \mathsf{ENC}, \mathsf{DEC}$, we can re-write Sam as in Fig. 9, as required to prove output indistinguishability. We now prove that for all PPT $\mathcal{D}$,

$\mathsf{Sam}(1^\lambda)$

$r \leftarrow\!\!\$\ \{0,1\}^{p(\lambda)}$
$C_r \leftarrow C_\lambda(.,r)$
$(\text{lkg}_{C_r}, \mathcal{O}) \leftarrow \mathsf{Sam}_{\text{leak}}(r)$
return $C_r, \text{lkg}_{C_r}, \mathcal{O}$

$\mathsf{Out}^b_{\mathsf{Sam},\mathcal{D}}(1^\lambda)$

$C_r, \text{lkg}_{C_r}, \mathcal{O} \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
$(x_0, \text{st}) \leftarrow\!\!\$\ \mathcal{D}^{\mathcal{O}}(1^\lambda, \text{lkg}_{C_r})$
$x_1 \leftarrow 0^{|x_0|}$
$b^* \leftarrow\!\!\$\ \mathcal{D}^{\mathcal{O}_{C_r(x_b)}}(C_r(x_b), \text{st})$
return b^*

Fig. 9: Rewritten version of Sam.

$$\left|\Pr\left[1 = \mathsf{Out}^0_{\mathsf{Sam},\mathcal{D}}(1^\lambda)\right] - \Pr\left[1 = \mathsf{Out}^1_{\mathsf{Sam},\mathcal{D}}(1^\lambda)\right]\right|$$

is negligible, via a sequence of game hops shown in Fig. 10.

We inline the code of Sam, move computations downwards when variables are not used before, and inline the code of the circuit C_r. We also write $S \leftarrow S \cup \{c^*\}$

$\mathsf{Out}^b_{\mathsf{Sam},\mathcal{D}}(1^\lambda)$	$\mathsf{Out}^b_{\mathsf{Sam},\mathcal{D}}(1^\lambda)$	$\mathsf{ENC}(m)$
$k_1 \leftarrow\!\$\ \{0,1\}^\lambda$	$k_1 \leftarrow\!\$\ \{0,1\}^\lambda$	**assert** $\lvert m\rvert = \lambda$
$k_2 \leftarrow\!\$\ \{0,1\}^\lambda$	$k_2 \leftarrow\!\$\ \{0,1\}^\lambda$	$c \leftarrow\!\$\ \mathsf{enc}(k, m)$
$k \leftarrow k_1 \oplus k_2$	$k \leftarrow\!\$\ \{0,1\}^\lambda$	$S \leftarrow S \cup \{c\}$
$r_1 \leftarrow\!\$\ \{0,1\}^\lambda$		**return** c
$C \leftarrow \mathsf{enc}(k_1 \oplus \cdot, \cdot; r_1 \oplus \cdot)$		
$(x_0, \mathrm{st}) \leftarrow\!\$\ \mathcal{D}^{\mathsf{ENC},\mathsf{DEC}}(1^\lambda, k_2)$	$(x_0, \mathrm{st}) \leftarrow\!\$\ \mathcal{D}^{\mathsf{ENC},\mathsf{DEC}}(1^\lambda, k_2)$	$\mathsf{DEC}(c)$
$x_1 \leftarrow 0^{\lvert x_0\rvert}$	$x_1 \leftarrow 0^{\lvert x_0\rvert}$	**assert** $\lvert m\rvert = \lambda$
	$\mathrm{msg}\lvert\lvert \Delta \rvert\rvert r' \leftarrow x_b$	**assert** $c \notin S$
$c^* \leftarrow C(x_b)$	$c^* \leftarrow\!\$\ \mathsf{enc}(k_1 \oplus \Delta, \mathrm{msg})$	$m \leftarrow\!\$\ \mathsf{dec}(k, c)$
$S \leftarrow S \cup \{c^*\}$	$S \leftarrow S \cup \{c^*\}$	**return** m
$b^* \leftarrow\!\$\ \mathcal{D}^{\mathsf{ENC},\mathsf{DEC}}(c, \mathrm{st})$	$b^* \leftarrow\!\$\ \mathcal{D}^{\mathsf{ENC},\mathsf{DEC}}(c, \mathrm{st})$	
return b^*	**return** b^*	

Fig. 10: Game hops to show Claim 1.

as an explicit state update instead of writing $c^* = C(x_b)$ as oracle subscript. The encryption process in the grey line uses fresh and uniform randomness since r_1 is uniformly random and so is r_1 xored with an adversarially chosen value.

We now reduce to \$-RK-AE security (Definition 1) in order to replace the ENC oracle by one that returns random strings of length 2λ, the string c^* by a random string of length 2λ and the DEC oracle by an oracle that always returns $\perp$. After the reduction to \$-RK-AE security, since c^* does not depend on b anymore, we have that $\Pr[b = b^*] = \frac{1}{2}$.

The reduction $\mathcal{B}$ to \$-RK-AE security can answer all ENC queries m by a query $(m, \Delta = 0^\lambda)$ to its own encryption oracle, and it can compute the ciphertext c by choosing $\Delta' := \Delta \oplus k_2$, since then,

$$k \oplus \Delta' = (k_1 \oplus k_2) \oplus (\Delta \oplus k_2) = k_1 \oplus \Delta,$$

which $\mathcal{B}$ expects. Analogously, the reduction proceeds with decryption queries. This concludes the proof of Claim 1 that $\mathsf{Sam} \in \mathcal{C}^{\mathrm{Out}}$.

Hence, by DSim$[\mathcal{C}^{\mathrm{Out}}]$-security, there exists a simulator $\mathcal{S}_{\mathrm{DSIM}}$ for $(\mathsf{Sam}, \mathcal{B})$ such that

$$\left|\Pr\left[1 = \mathtt{DSim}^0_{\mathsf{Sam},\mathcal{B},\mathsf{Gb},\Lambda}(1^\lambda)\right] - \Pr\left[1 = \mathtt{DSim}^1_{\mathsf{Sam},\mathcal{B},\mathcal{S}_{\mathrm{DSIM}},\Lambda}(1^\lambda)\right]\right|$$

is negligible. Given $\mathcal{S}_{\mathrm{DSIM}}$, the simulator $\mathcal{S}_{\mathrm{hybrid}}$ with oracle access to ENC and DEC runs

$$(\tilde{C}, d, \mathrm{st}_S) \leftarrow\!\$\ \mathcal{S}^{\mathsf{ENC},\mathsf{DEC}}_{\mathrm{DSIM}}(\mathrm{lkg}_C) \text{ and } \tilde{x} \leftarrow\!\$\ \mathcal{S}^{\mathsf{ENC},\mathsf{DEC}}_{\mathrm{DSIM}}([], \mathrm{st}_S)$$

and returns $(\tilde{C}, d, \tilde{x})$.

$$\begin{aligned}\text{Since } &\left|\Pr\left[1 = \mathtt{DSim}^0_{\mathsf{Sam},\mathcal{C},\mathsf{Gb}}(1^\lambda)\right] - \Pr\left[1 = \mathtt{DSim}^1_{\mathsf{Sam},\mathcal{C},\mathcal{S}_{\mathrm{DSIM}}}(1^\lambda)\right]\right| \\ =&\left|1 = \Pr\left[\mathtt{DENC}^0_{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 = \mathtt{Hybrid}_{\mathcal{A},\mathcal{S}_{\mathrm{hybrid}}(1^\lambda)}\right]\right|\end{aligned}$$

and the former is negligible and the latter non-negligible, we reached a contradiction.

□

6 Bootstrapping for output indistinguishable samplers

In this section, we bootstrap DSIM security for output indistinguishable samplers returning NC^0 circuits to output indistinguishable samplers returning arbitrary polynomial-size circuits. Let us denote the class of output indistinguishable samplers by $\mathcal{C}^{\mathrm{Out}}$ and output indistinguishable samplers which only return circuits in NC^0 by $\mathcal{C}^{\mathrm{Out},\mathrm{NC}^0} \subseteq \mathcal{C}^{\mathrm{Out}}$. Based on a $\mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out},\mathrm{NC}^0}]$-secure garbling scheme (for NC^0 circuits), we construct a $\mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out}}]$-secure garbling scheme (for arbitrary poly-size circuits): The new garbling scheme $\mathsf{Gb}_{\mathsf{comb}} = (\mathsf{GCircuit}_{\mathsf{comb}}, \mathsf{GInput}_{\mathsf{comb}}, \mathsf{GEval}_{\mathsf{comb}})$ which we construct (cf. Fig. 11) combines two garbling schemes, the *inner* SelSim-secure garbling scheme $\mathsf{Gb}_{\mathsf{in}} = (\mathsf{GCircuit}_{\mathsf{in}}, \mathsf{GInput}_{\mathsf{in}}, \mathsf{GEval}_{\mathsf{in}})$ for arbitrary polynomial-size circuits and an *outer* garbling scheme $\mathsf{Gb}_{\mathsf{outer}} = (\mathsf{GCircuit}_{\mathsf{outer}}, \mathsf{GInput}_{\mathsf{outer}}, \mathsf{GEval}_{\mathsf{outer}})$ for NC^0 circuits which is $\mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out},\mathrm{NC}^0}]$-secure, to obtain a *combined* garbling scheme $\mathsf{Gb}_{\mathsf{comb}}$ which is $\mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out}}]$-secure. $\mathsf{Gb}_{\mathsf{comb}}$ garbles a circuit $C(\cdot)$ and input x as depicted in Figure 11, where C_{in} is a circuit that takes as input x and produces as output the (selectively secure) garbling of C, i.e., $\tilde{C}_{\mathrm{in}}$, a garbling of x and the decoding information d_{in}. $\mathsf{Gb}_{\mathsf{in}}$ could be any SelSim-secure, projective[7] garbling scheme, e.g., $\mathsf{Gb}_{\mathsf{in}} = (\mathsf{GCircuit}_{\mathsf{in}}, \mathsf{GInput}_{\mathsf{in}}, \mathsf{GEval}_{\mathsf{in}}) := (\mathsf{GCircuit}_{\mathsf{yao}}, \mathsf{GInput}_{\mathsf{yao}}, \mathsf{GEval}_{\mathsf{yao}})$ is a valid choice (Yao's garbling is provably SelSim secure, assuming only the existence of \$-IND-CPA secure symmetric encryption scheme). We need to show that the circuit C_{in} is indeed in NC^0 and that $\mathsf{Gb}_{\mathsf{comb}}$ is $\mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out}}]$-secure.

Section 6.1 shows the (easy) statement that the circuit C_{in} is in NC^0. In a nutshell, this follows from $\mathsf{Gb}_{\mathsf{in}}$ being a projective garbling scheme and $\tilde{C}_{\mathrm{in}}$ and the decoding information d_{in} being just some constant bitstring in C_{in}.

Section 6.2 proves that if the sampler for circuit $C(x)$ is in $\mathcal{C}^{\mathrm{Out}}$, then the sampler for circuit $\tilde{C}(x; r)$ is in $\mathcal{C}^{\mathrm{Out},\mathrm{NC}^0}$. Section 6.3 then states and proves our main bootstrapping theorem.

6.1 C_{in} is low-depth

Recall that C_{in} is defined as $C_{\mathrm{in}}(\cdot) := [\tilde{C}_{\mathrm{in}}, \mathsf{GInput}_{\mathsf{in}}(\cdot\,, K_{\mathrm{in}}), d_{\mathrm{in}}]$ where $\tilde{C}_{\mathrm{in}}, d_{\mathrm{in}}$ and K_{in} are constant values (constant bitstrings). Hence, in order to show that

[7] A garbling scheme is projective, if for each input bit x_i, the input garbling is one out of two possible strings $K_0(i)$ and $K_1(i)$. For example, Yao's garbling scheme is projective.

$\mathsf{GCircuit}_{\mathsf{comb}}(C)$

$r_{\mathsf{in}} \leftarrow\!\!\$\ \{0,1\}^{12\lambda|C|}$

$\tilde{C}_{\mathrm{in}}, K_{\mathrm{in}}, d_{\mathrm{in}} \leftarrow \mathsf{GCircuit}_{\mathsf{in}}(C(\cdot); r_{\mathsf{in}})$

$C_{\mathrm{in}}(\cdot) \leftarrow [\tilde{C}_{\mathrm{in}}, \mathsf{GInput}_{\mathsf{in}}(\cdot\,, K_{\mathrm{in}}), d_{\mathrm{in}}]$

$\tilde{C}, K, d \leftarrow\!\!\$\ \mathsf{GCircuit}_{\mathsf{outer}}(C_{\mathrm{in}})$

return $\tilde{C}, K, d$

$\mathsf{GInput}_{\mathsf{comb}}(K, x)$

$\tilde{x} \leftarrow \mathsf{GInput}_{\mathsf{outer}}(K, x)$

return $\tilde{x}$

$\mathsf{GEval}_{\mathsf{comb}}(\tilde{C}, \tilde{x}, d)$

$\tilde{C}_{\mathrm{in}}, \tilde{x}_{\mathrm{in}}, d_{\mathrm{in}} \leftarrow \mathsf{GEval}_{\mathsf{outer}}(\tilde{C}, \tilde{x}, d)$

$y \leftarrow \mathsf{GEval}_{\mathsf{in}}(\tilde{C}_{\mathrm{in}}, \tilde{x}_{\mathrm{in}}, d_{\mathrm{in}})$

return y

Fig. 11: Garbling scheme $\mathsf{Gb}_{\mathsf{comb}}$. Length of r_{in} is chosen s.t. it is compatible with Yao's garbling scheme, however, the same results apply to any projective garbling scheme, just $|r_{\mathsf{in}}|$ might need to be adjusted.

the function C_{in} can be implemented as a constant depth circuit, it is enough to show the following lemma.

Lemma 1 (Low-depth Projective Input Garbling). *If $\mathsf{Gb}_{\mathsf{in}}$ is a projective garbling scheme, then $\mathsf{GInput}_{\mathsf{in}}(\cdot; K_{in}) : x \mapsto \mathsf{GInput}_{\mathsf{in}}(x; K_{in})$ can be described by a constant-depth circuit.*

Proof. Denote by x_i the ith bit of the input x. Now K_{in} consists of key pairs k_0^i, k_1^i for each bit of x. W.l.o.g., we can assume that K_{in} is a concatenation of all the key pairs in order. The input garbling of x, i.e. $\mathsf{GInput}_{\mathsf{in}}(x; K_{\mathrm{in}})$, outputs $k_{x_1}^1 || ... || k_{x_\lambda}^\lambda$. This can clearly be done in constant depth by a circuit that just checks each bit of x one by one (in parallel) and outputs the corresponding key k_{x_i} for each bit. □

6.2 Output Indistinguishable Sampling

We now prove that the above circuit transformation, when applied to a circuit sampler $\mathsf{Sam}_{\mathsf{comb}}(1^\lambda)$ in $\mathcal{C}^{\mathrm{Out}}$ yields a circuit sampler $\mathsf{Sam}_{\mathsf{outer}}(1^\lambda)$ in $\mathcal{C}^{\mathrm{Out},\mathrm{NC}^0}$. Since $\mathsf{Sam}_{\mathsf{comb}}$ is output indistinguishable, there exists a polynomial $p(\lambda)$ and a circuit $C = (C_\lambda)_{\lambda\in\mathbb{N}}$ such that $\mathsf{Sam}_{\mathsf{comb}}(1^\lambda)$ can be written as below (left). Then, we define the circuit sampler $\mathsf{Sam}_{\mathsf{outer}}(1^\lambda)$ (with randomness $r||r_{\mathrm{in}}$) as follows (right):

$\mathsf{Sam}_{\mathsf{comb}}(1^\lambda)$

$r \leftarrow\!\!\$\ \{0,1\}^{p(\lambda)}$

$C_r(.) \leftarrow C(\cdot, r)$

return $C_r, \mathrm{lkg}_{C_r}, \mathcal{O}$

$\mathsf{Sam}_{\mathsf{outer}}(1^\lambda)$

$r \leftarrow\!\!\$\ \{0,1\}^{p(\lambda)}$

$r_{\mathsf{in}} \leftarrow\!\!\$\ \{0,1\}^{12\lambda|C_r|}$

$\tilde{C}_{\mathrm{in}}, K_{\mathrm{in}}, d_{\mathrm{in}} \leftarrow \mathsf{GCircuit}_{\mathsf{in}}(C_r(\cdot); r_{\mathsf{in}})$

$C_{\mathrm{in}}(\cdot) \leftarrow [\tilde{C}_{\mathrm{in}}, \mathsf{GInput}_{\mathsf{in}}(\cdot\,, K_{\mathrm{in}}), d_{\mathrm{in}}]$

return $C_{\mathrm{in}}, \mathrm{lkg}_{C_r}, \mathcal{O}$

Lemma 2 (Output indististinguishability). *Let $\mathsf{Sam}_{\mathsf{comb}} \in \mathcal{C}^{Out}$. If Yao's garbling scheme is selectively secure, then $\mathsf{Sam}_{\mathsf{outer}}(1^\lambda) \in \mathcal{C}^{Out,NC^0}$.*

Proof. Firstly, by Lemma 1, $\mathsf{Sam}_{\mathsf{outer}}$ produces circuits in NC^0. Hence remains to show that $\mathsf{Sam}_{\mathsf{outer}}$ is output indistinguishable, i.e., we prove that for all PPT adversaries $\mathcal{D}_{\mathsf{outer}}$,

$$\left|\Pr\left[1 = \mathsf{Out}^0_{\mathsf{Sam}_{\mathsf{outer}},\mathcal{D}_{\mathsf{outer}}}(1^\lambda)\right] - \Pr\left[1 = \mathsf{Out}^1_{\mathsf{Sam}_{\mathsf{outer}},\mathcal{D}_{\mathsf{outer}}}(1^\lambda)\right]\right| = \mathsf{negl}(\lambda).$$

We prove indistinguishability of $\mathsf{Out}^0_{\mathcal{D}_{\mathsf{outer}}}(1^\lambda)$ and $\mathsf{Out}^1_{\mathcal{D}_{\mathsf{outer}}}(1^\lambda)$ via several hybrids. The games $\mathsf{Out}^b_{\mathcal{D}_{\mathsf{outer}}}(1^\lambda)$ and H^b are perfectly equivalent; we only inline the definition of circuit $\mathsf{Sam}_{\mathsf{outer}}$ and reorder lines whose order does not affect the functionality.

$\mathsf{Out}^b_{\mathsf{Sam}_{\mathsf{outer}},\mathcal{D}_{\mathsf{outer}}}(1^\lambda)$

$C_{\mathrm{in}}, \mathrm{lkg}_{C_r}, \mathcal{O} \leftarrow\!\$\ \mathsf{Sam}_{\mathsf{outer}}$

$(x_0, \mathrm{st}) \leftarrow\!\$\ {\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}}(1^\lambda, \mathrm{lkg}_{C_r})$

$x_1 \leftarrow 0^{|x_0|}$

$b^* \leftarrow\!\$\ {\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}_{C_r(x_b)}}(C_{\mathrm{in}}(x_b), \mathrm{st})$

return b^*

From $\mathsf{H}^b(1^\lambda)$ to $H^{b+2}_{\mathcal{S}}$, we reduce to the selective simulation-based security of $\mathsf{Gb}_{\mathsf{in}}$. The highlighted lines in H^b refer to the SelSim adversary (who first chooses the circuit, the input and its own state C_r, x_b, st, then receives a garbled circuit and garbled input and returns its guess). Note that the SelSim adversary knows r and can hence compute the algorithm $\mathcal{O}$ and pass it in st, so ${\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}}$ can be replaced by a code equivalent and efficient algorithm that does not make oracle queries to $\mathcal{O}$ (but simply runs the algorithm $\mathcal{O}$).

From $H^{0+2}_{\mathcal{S}}$ to $H^{1+2}_{\mathcal{S}}$, we reduce to the output indistinguishability of $\mathsf{Sam}_{\mathsf{comb}}$, note that the two first lines of $H^{b+2}_{\mathcal{S}}$ are code equivalent to $\mathsf{Sam}_{\mathsf{comb}}$ and the highlighted lines refer to the output indistinguishability adversary (note that the topology $\Phi(C_r)$ can be computed when you know C and C is constant). For this game hop we switch back to $\mathcal{O}$ being an oracle algorithm (as opposed to emulating ${\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}}$ by running the actual algorithm $\mathcal{O}$).

H^b

$r \leftarrow\!\$\ \{0,1\}^{p(\lambda)}$

$C_r, \mathrm{lkg}_{C_r} \leftarrow C(\cdot, r), \mathrm{lkg}_{C_r}$

$(x_0, \mathrm{st}) \leftarrow\!\$\ {\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}}(1^\lambda, \mathrm{lkg}_{C_r})$

$x_1 \leftarrow 0^{|x_0|}$

$r_{\mathsf{in}} \leftarrow\!\$\ \{0,1\}^{12\lambda|C_r|}$

$\tilde{C}_{\mathrm{in}}, K_{\mathrm{in}}, d_{\mathrm{in}} \leftarrow \mathsf{GCircuit}_{\mathsf{in}}(C_r(\cdot); r_{\mathsf{in}})$

$C_{\mathrm{in}}(\cdot) \leftarrow [\tilde{C}_{\mathrm{in}}, \mathsf{GInput}_{\mathsf{in}}(\cdot, K_{\mathrm{in}}), d_{\mathrm{in}}]$

$(\tilde{C}, \tilde{x}, d) \leftarrow C_{\mathrm{in}}(x_b)$

$b^* \leftarrow\!\$\ {\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}_{C_r(x_b)}}((\tilde{C}, \tilde{x}, d), \mathrm{st})$

return b^*

$H^{b+2}_{\mathcal{S}}$

$r \leftarrow\!\$\ \{0,1\}^{p(\lambda)}$

$C_r, \mathrm{lkg}_{C_r} \leftarrow C(\cdot, r), \mathrm{lkg}_{C_r}$

$(x_0, \mathrm{st}) \leftarrow\!\$\ {\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}}(1^\lambda, \mathrm{lkg}_{C_r})$

$x_1 \leftarrow 0^{|x_0|}$

$(\tilde{C}, \tilde{x}, d) \leftarrow \mathcal{S}(\Phi(C_r), C_r(x_b))$

$b^* \leftarrow\!\$\ {\mathcal{D}_{\mathsf{outer}}}^{\mathcal{O}_{C_r(x_b)}}((\tilde{C}, \tilde{x}, d), \mathrm{st})$

return b^*

□

6.3 Main theorem

Our main bootstrapping theorem now follows from the above lemmas.

Theorem 4 (Bootstrapping RIND from NC^0 to poly). *If the garbling scheme* $\mathsf{Gb}_{\mathsf{outer}} = (\mathsf{GCircuit}_{\mathsf{outer}}, \mathsf{GInput}_{\mathsf{outer}}, \mathsf{GEval}_{\mathsf{outer}})$ *is* $\mathit{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{Out,NC^0}]$*-secure and if* $\mathsf{Gb}_{\mathsf{in}} = (\mathsf{GCircuit}_{\mathsf{in}}, \mathsf{GInput}_{\mathsf{in}}, \mathsf{GEval}_{\mathsf{in}})$ *is a projective SelSim-secure garbling scheme, then the garbling scheme* $\mathsf{Gb}_{\mathsf{comb}} = (\mathsf{GCircuit}_{\mathsf{comb}}, \mathsf{GInput}_{\mathsf{comb}}, \mathsf{GEval}_{\mathsf{comb}})$ *achieves* $\mathit{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{Out}]$ *security.*

Proof. For all PPT $\mathsf{Sam}_{\mathsf{comb}} \in \mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out}}]$ and for all PPT $\mathcal{A}$, we want to construct a simulator $\mathcal{S}_{\mathsf{comb}}$ such that the following advantage

$$\left|\Pr\left[1 = \mathtt{DSim}^0_{\mathsf{Sam}_{\mathsf{comb}},\mathcal{A},\mathsf{Gb}_{\mathsf{comb}},\Lambda}(1^\lambda)\right] - \Pr\left[1 = \mathtt{DSim}^1_{\mathsf{Sam}_{\mathsf{comb}},\mathcal{A},\mathcal{S}_{\mathsf{comb}},\Lambda}(1^\lambda)\right]\right| \qquad (1)$$

is negligible.

$\mathtt{DSim}^0_{\mathsf{Sam}_{\mathsf{comb}},\mathcal{A},\mathsf{Gb}_{\mathsf{comb}},\Lambda}(1^\lambda)$

$C_r, \mathrm{lkg}_{C_r}, \mathcal{O} \leftarrow\$ \mathsf{Sam}_{\mathsf{comb}}(1^\lambda)$

$(\tilde{C}, K, d) \leftarrow\$ \mathsf{GCircuit}_{\mathsf{comb}}(C_r)$

$(x, \mathrm{st}_{\mathcal{A}}) \leftarrow\$ \mathcal{A}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_{C_r})$

$\tilde{x} \leftarrow\$ \mathsf{GInput}_{\mathsf{comb}}(K, x)$

$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}_{C_r(x)}}(\tilde{x}, \mathrm{st}_{\mathcal{A}})$

return b'

H^0

$r \leftarrow\$ \{0,1\}^{p(\lambda)}$

$C_r(.) \leftarrow C(\cdot, r)$

$r_{\mathsf{in}} \leftarrow\$ \{0,1\}^{12\lambda|C|}$

$\tilde{C}_{\mathsf{in}}, K_{\mathsf{in}}, d_{\mathsf{in}} \leftarrow \mathsf{GCircuit}_{\mathsf{in}}(C_r(\cdot); r_{\mathsf{in}})$

$C_{\mathsf{in}}(\cdot) \leftarrow [\tilde{C}_{\mathsf{in}}, \mathsf{GInput}_{\mathsf{in}}(\cdot, K_{\mathsf{in}}), d_{\mathsf{in}}]$

$\tilde{C}, K, d \leftarrow\$ \mathsf{GCircuit}_{\mathsf{outer}}(C_{\mathsf{in}})$

$(x, \mathrm{st}_{\mathcal{A}}) \leftarrow\$ \mathcal{A}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_{C_r})$

$\tilde{x} \leftarrow\$ \mathsf{GInput}_{\mathsf{outer}}(K, x)$

$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}_{\tilde{C}_r(\tilde{x})}}(\tilde{x}, \mathrm{st}_{\mathcal{A}})$

return b'

$\mathtt{DSim}^1_{\mathsf{Sam}_{\mathsf{comb}},\mathcal{A},\mathcal{S}_{\mathsf{comb}},\Lambda}(1^\lambda)$

$C_r, \mathrm{lkg}_{C_r}, \mathcal{O} \leftarrow\$ \mathsf{Sam}_{\mathsf{comb}}(1^\lambda)$

$(\tilde{C}, d, \mathrm{st}_S) \leftarrow\$ \mathcal{S}^{\mathcal{O}}(\mathsf{Filter}(\mathrm{lkg}_{C_r}))$

$(x, \mathrm{st}_{\mathcal{A}}) \leftarrow\$ \mathcal{A}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_{C_r})$

$\tilde{x} \leftarrow\$ \mathcal{S}^{\mathcal{O}}(\Lambda(\mathrm{st}_{\mathcal{A}}), \mathrm{st}_S)$

$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}_{\tilde{C}_r(\tilde{x})}}(\tilde{x}, \mathrm{st}_{\mathcal{A}})$

return b'

H^1

$r \leftarrow\$ \{0,1\}^{p(\lambda)}$

$C_r(.) \leftarrow C(\cdot, r)$

$r_{\mathsf{in}} \leftarrow\$ \{0,1\}^{12\lambda|C|}$

$\tilde{C}_{\mathsf{in}}, K_{\mathsf{in}}, d_{\mathsf{in}} \leftarrow \mathsf{GCircuit}_{\mathsf{in}}(C_r(\cdot); r_{\mathsf{in}})$

$C_{\mathsf{in}}(\cdot) \leftarrow [\tilde{C}_{\mathsf{in}}, \mathsf{GInput}_{\mathsf{in}}(\cdot, K_{\mathsf{in}}), d_{\mathsf{in}}]$

$\tilde{C}, d, \mathrm{st}_S \leftarrow\$ \mathcal{S}_{\mathsf{outer}}{}^{\mathcal{O}}(\mathsf{Filter}(\mathrm{lkg}_{C_r}))$

$(x, \mathrm{st}_{\mathcal{A}}) \leftarrow\$ \mathcal{A}^{\mathcal{O}}(\tilde{C}, d, \mathrm{lkg}_{C_r})$

$\tilde{x} \leftarrow\$ \mathcal{S}_{\mathsf{outer}}{}^{\mathcal{O}}(\Lambda(\mathrm{st}_{\mathcal{A}}), \mathrm{st}_S)$

$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}_{\tilde{C}_r(\tilde{x})}}(\tilde{x}, \mathrm{st}_{\mathcal{A}})$

return b'

The experiments $\mathtt{DSim}^0_{\mathsf{Sam}_{\mathsf{comb}},\mathcal{A},\mathsf{Gb}_{\mathsf{comb}},\Lambda}(1^\lambda)$ and the hybrid game H^0 are code equivalent by inlining the code of $\mathsf{Sam}_{\mathsf{comb}}$, $\mathsf{GCircuit}_{\mathsf{comb}}$ and $\mathsf{GInput}_{\mathsf{comb}}$. Now, observe that the lines highlighted in grey in H^0 describe $\mathsf{Sam}_{\mathsf{outer}}(1^\lambda)$ and thus, H^0 is also equivalent to $\mathtt{DSim}^0_{\mathsf{Sam}_{\mathsf{outer}},\mathcal{A},\mathsf{Gb}_{\mathsf{outer}},\Lambda}(1^\lambda)$, as required for the reduction.

Analogously, the hybrid game H^1 is equivalent to $\mathtt{DSim}^1_{\mathsf{Sam}_{\mathsf{outer}},\mathcal{A},\mathsf{Gb}_{\mathsf{outer}},\Lambda}(1^\lambda)$ with $\mathsf{Sam}_{\mathsf{outer}} \in \mathcal{C}^{\mathrm{Out},\mathrm{NC}^0}$ (Lemma 2) and thus, by $\mathrm{DSim}_{\Lambda,\mathsf{Filter}}[\mathcal{C}^{\mathrm{Out},\mathrm{NC}^0}]$-security of $\mathsf{Gb}_{\mathsf{outer}}$ (assumption), there is a simulator $\mathcal{S}_{\mathrm{outer}}$ s.t. H^0 is indistinguishable from H^1. If we define simulator $\mathcal{S}$ according to grey lines in H^1, we notice that H^1 is also code equivalent to $\mathtt{DSim}^1_{\mathsf{Sam}_{\mathsf{comb}},\mathcal{A},\mathsf{Gb}_{\mathsf{comb}},\Lambda}(1^\lambda)$, which concludes the proof. □

7 Large Yao-Incompressibility Entropy Implies Large Online Complexity

Conceptually, the AIKW lower bounds uses that $f(x)$ has real entropy (at most) $|x|$, but if f is a PRG, then $f(x)$ has (HILL) *pseudo-entropy* $|f(x)|$, since $f(x)$ is indistinguishable from a uniformly random string of length $|f(x)|$. AIKW conclude that the input garbling size must be proportional to $|f(x)|$ rather than to $|x|$. Using an analogous argument, HW directly formulate their lower bound for online complexity of maliciously secure 2PC in terms of pseudo-entropy, but rely on (Yao) incompressibility instead which is measured by the shortest efficient encoding of $f(x)$. High HILL pseudoentropy of $f(x)$ implies high incompressibility entropy of $f(x)$, since real entropy distribution $\mathcal{D}$ cannot be compressed, but the converse is not necessarily true [HLR07,HMS23], so the use of incompressibility, conceptually, yields a more general result. HW imply lower bounds also on SIM-secure garbling. In this section, we use the HW incompressibility entropy approach to bound the online complexity of a SIM-secure garbling scheme directly to obtain a self-contained proof, cf. Section 1.1.

Remark. Yao incompressibility has been defined in different variations, e.g., whether the definition considers *expected* ([TVZ05]) or *worst-case* (HW, [BSW03]) compression length and whether it uses *perfect* correctness ([TVZ05]) or *imperfect* correctness (HW, [BSW03]). We now state the Yao incompressibilitiy definitions of [TVZ05] and HW, the latter of which simplifies the definition by [BSW03] as is enough in this context.

Definition 10 (Yao Incompressibility Entropy by TVZ [TVZ05]). *Let Φ be a leakage function and $(C,x) \leftarrow\!\$\ \mathcal{D}(1^\lambda)$ be an efficiently sampleable distribution. The distribution $\mathcal{D}$ is k_λ-*incompressible*, if for every (non-uniform) polynomial-size circuit family pair of a compression algorithm $\mathsf{Cmpr}_\lambda(\cdot,\cdot)$ and a decompression function $\mathsf{Decmpr}_\lambda(\cdot,\cdot)$ s.t. for all x: $\mathsf{Decmpr}\big(\mathsf{Cmpr}(C(x),\Phi(C)),\Phi(C)\big) = C(x)$ it holds that*

$$\mathbb{E}_{(C,x)\leftarrow\!\$\mathcal{D}(1^\lambda)}[|\mathsf{Cmpr}(C(x),\Phi(C))|] \geq k_\lambda,$$

where $|.|$ is the output length of Cmpr.

Definition 11 (Yao Incompressibility Entropy by HW [HW15]). *Let Φ be a leakage function and $(C, x) \leftarrow\!\!\$\ \mathcal{D}(1^\lambda)$ be an efficiently sampleable distribution. The distribution $\mathcal{D}$ is k_λ-*incompressible*, if for every (non-uniform) polynomial-size circuit family pair of a compression algorithm $\mathsf{Cmpr}_\lambda(\cdot,\cdot)$ and a decompression function $\mathsf{Decmpr}_\lambda(\cdot,\cdot)$ it holds that if output length $l(\lambda)$ of Cmpr is $< k_\lambda$ then there exists a negligible function ϵ s.t.*

$$\Pr_{(C,x)\leftarrow\!\!\$\mathcal{D}}\left[\mathsf{Decmpr}\big(\mathsf{Cmpr}(C(x),\Phi(C)),\Phi(C)\big) = C(x)\right] \leq 1/2 + \epsilon(\lambda)$$

For completeness, we prove our theorems both for Yao incompressibility entropy as defined by HW and by Trevisan, Vadhan and Zuckermann (TVZ [TVZ05]) and refer to Karanko [Kar23] for a more comprehensive overview over different notions of incompressibility entropy and pseudo-entropy and clarifying of their relations—especially, high *expected* incompressibility entropy is implied by high pseudo *Shannon* entropy, while high *worst-case* incompressibility entropy is only implied by high pseudo *min* entropy.

Outline. For concreteness, we now revisit the AIKW lower bound which considers $C := \mathsf{PRG}$. Correctness of the garbling scheme implies that $\tilde{C}(\tilde{x}) = \mathsf{PRG}(x)$. Therefore, the simulator who first creates a garbled circuit $\tilde{C}$ and then gets $y = \mathsf{PRG}(x)$ also needs to create $\tilde{x}$ such that $\tilde{C}(\tilde{x}) = y$. Now, the check that $\tilde{C}(\tilde{x})$ evaluates to y can be performed only knowing y, and since the simulator gets only y, we can run the simulator on a *random* y instead of a PRG output—by PRG security, the simulator should be as successful in creating a simulated input $\tilde{x}$ such that $\tilde{C}(\tilde{x}) = y$. However, if $|\tilde{x}| < |y|$ the simulator has an impossible task, because y has more entropy than $\tilde{x}$. Therefore, $|\tilde{x}|$ should be proportional to the *computational entropy* of $C(x)$. We now prove that both high TVZ and HW incompressibility entropy of $C(x)$ imply high online complexity of a SIM-secure scheme Gb which garbles C and x.

Theorem 5 (TVZ Incompressibility ⇒ High Online Complexity). *Let Gb be a SIM-secure garbling scheme with leakage function Φ, and let $(C, x) \leftarrow\!\!\$\ \mathcal{D}(1^\lambda)$ be an efficiently samplable distribution with incompressibility-entropy $\geq k_\lambda$. If the online complexity $|\tilde{x}|$ is uniquely determined by $\Phi(C)$, then the expected online complexity is*

$$\mathbb{E}_{(C,x)\leftarrow\!\!\$\mathcal{D}(1^\lambda),\tilde{x}\leftarrow\!\!\$\mathsf{Gb.GInput}}[|\tilde{x}|] \geq k_\lambda - 2.$$

Proof. Assume towards contradiction that

$$\mathbb{E}_{(C,x)\leftarrow\!\!\$\mathcal{D}(1^\lambda),\tilde{x}\leftarrow\!\!\$\mathsf{Gb.GInput}}[|\tilde{x}|] < k_\lambda - 2, \tag{2}$$

Since the garbling scheme Gb is SIM-secure, there exists a PPT simulator $\mathcal{S}$, s.t.

$$\Pr\left[\begin{array}{r|l} \mathsf{Gb.GEval}(\tilde{C},\tilde{x},d) = C(x) & (C,x) \leftarrow\!\!\$\ \mathcal{D}(1^\lambda) \\ |\tilde{x}| = |\mathsf{Gb.GInput}(x)| & \tilde{C},d,\mathrm{st} \leftarrow\!\!\$\ \mathcal{S}(1^\lambda,\Phi(C)) \\ & \tilde{x} \leftarrow\!\!\$\ \mathcal{S}(1^\lambda,C(x),\mathrm{st}) \end{array}\right] \geq 1-\mu(\lambda),$$

where μ is negligible. The garbled input that the simulator returns must (almost always) have the correct online complexity, since otherwise we can use the length of the garbled input to distinguish real garbling from simulated garbling. We now use the simulator $\mathcal{S}$ to build a pair of efficient algorithms compressor Cmpr and decompressor Decmpr. Let $\mathcal{S}_\lambda^{\mathsf{det}}$ be a *deterministic* version of the simulator, i.e., $\mathcal{S}_\lambda^{\mathsf{det}}$ is equal to the simulator $\mathcal{S}(1^\lambda, \cdot; r_\lambda)$, where r_λ is the internal randomness that maximizes the above probability. We here use that [TVZ05] allow Cmpr and decompressor Decmpr to be a non-uniform circuit family. Then, we define compressor Cmpr and decompressor Decmpr as follows, writing lkg_Φ for a supposed output of $\Phi(C)$.

$\mathsf{Cmpr}(y, \mathsf{lkg}_\Phi)$

$\tilde{C}, d, \mathrm{st} \leftarrow \mathcal{S}_\lambda^{\mathsf{det}}(\mathsf{lkg}_\Phi)$ // 1st stage of simulator

$\tilde{x} \leftarrow \mathcal{S}_\lambda^{\mathsf{det}}(y, \mathrm{st})$ // 2nd stage of simulator

if $\mathsf{GEval}(\tilde{C}, \tilde{x}, d) = y$ AND $|\tilde{x}| = |\mathsf{Gb.GInput}(x)|$

  return $0||\tilde{x}$

return $1||y$

return $\tilde{x}_{1,\ldots,k-2}$

$\mathsf{Decmpr}(b||z, \mathsf{lkg}_\Phi)$

if $b = 1$

  return z

$\tilde{C}, d, \mathrm{st} \leftarrow \mathcal{S}_\lambda^{\mathsf{det}}(\mathsf{lkg}_\Phi)$

  // 1st stage of simulator

return $\mathsf{GEval}(\tilde{C}, z, d)$

The if-clause in the compressor is false only with negligible probability, because by averaging argument, we can show that there is simulator's randomness r_λ that achieves at least success probability $1 - 2\mu(\lambda)$. Hence,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[|\mathsf{Cmpr}(C(x), \Phi(C))|] &= \sum_{C,x \in \mathcal{D}, \text{if true}} \Pr_{\mathcal{D}}[C,x] \underbrace{|\mathsf{Cmpr}(C(x), \Phi(C))|}_{=|\mathsf{Gb.GInput}(x)|+1} \\
&\quad + \underbrace{\underbrace{\sum_{C,x \in \mathcal{D}, \text{if false}} \Pr_{\mathcal{D}}[C,x]}_{\mathsf{negl}(\lambda)} \underbrace{|\mathsf{Cmpr}(C(x), \Phi(C))|}_{\text{polynomial (WLOG)}}}_{\mathsf{negl}(\lambda)} \\
&\leq \mathbb{E}_{\mathcal{D}}[|\mathsf{Gb.GInput}(x)| + 1] + \mathsf{negl}(\lambda) \\
&= \underbrace{\mathbb{E}_{\mathcal{D}}[|\mathsf{Gb.GInput}(x)|]}_{\leq k-2 \text{ by } (2)} + 1 + \mathsf{negl}(\lambda) < k - 1 + \mathsf{negl}(\lambda)
\end{aligned}$$

which is a contradiction with incompressibility.

□

Remark. Note that as opposed to Theorem 5, in Theorem 6, the online complexity is k instead of $k-2$, but, in turn, we only obtain this lower bound on the online complexity for *some* (C, x) pair rather than *in expectation*.

Theorem 6 (HW Incompressibility ⇒ High Online Complexity). *Let* Gb *be a SIM-secure garbling scheme with leakage function* Φ*, and let* $(C, x) \leftarrow\!\!\$\ \mathcal{D}(1^\lambda)$ *be an efficiently samplable distribution with HW incompressibility-entropy*

$\geq k_\lambda$. *Then, there exists (at least) one pair in the support of* $\mathcal{D}$ *whose online complexity is* $|\tilde{x}| \geq k_\lambda$. *Again, we assume that the online complexity* $|\tilde{x}|$ *is uniquely determined by* $\Phi(C)$.

Proof. The part in gray here is copied in verbatim from proof of Theorem 5. Assume towards contradiction that for all (x, C) pairs in $\mathcal{D}$

$$|\tilde{x}| < k_\lambda, \tag{3}$$

Since the garbling scheme Gb is SIM-secure, there exists a PPT simulator $\mathcal{S}$, s.t.

$$\Pr\left[\begin{array}{r|l} \mathsf{Gb.GEval}(\tilde{C}, \tilde{x}, d) = C(x) & (C, x) \leftarrow\!\$\, \mathcal{D}(1^\lambda) \\ |\tilde{x}| = |\mathsf{Gb.GInput}(x)| & \tilde{C}, d, \mathrm{st} \leftarrow\!\$\, \mathcal{S}(1^\lambda, \Phi(C)) \\ & \tilde{x} \leftarrow\!\$\, \mathcal{S}(1^\lambda, C(x), \mathrm{st}) \end{array}\right] \geq 1 - \mu(\lambda),$$

where μ is negligible. The garbled input that the simulator returns must (almost always) have the correct online complexity, since otherwise we can use the length of the garbled input to distinguish real garbling from simulated garbling. We now use the simulator $\mathcal{S}$ to build a pair of efficient algorithms compressor Cmpr and decompressor Decmpr. Let $\mathcal{S}^{\mathsf{det}}_\lambda$ be a *deterministic* version of the simulator, i.e., $\mathcal{S}^{\mathsf{det}}_\lambda$ is equal to the simulator $\mathcal{S}(1^\lambda, \cdot; r_\lambda)$, where r_λ is the internal randomness that maximizes the above probability. We here use that HW allow Cmpr and decompressor Decmpr to be a non-uniform circuit family. Then, we define compressor Cmpr and decompressor Decmpr as follows, writing lkg_Φ for a supposed output of $\Phi(C)$.

$\mathsf{Cmpr}(y, \mathsf{lkg})$	$\mathsf{Decmpr}(z, \mathsf{lkg})$
$\tilde{C}, d, \mathrm{st} \leftarrow \mathcal{S}^{\mathsf{det}}_\lambda(\mathsf{lkg})$	$\tilde{C}, d, \mathrm{st} \leftarrow \mathcal{S}^{\mathsf{det}}_\lambda(\mathsf{lkg})$
$\tilde{x} \leftarrow \mathcal{S}^{\mathsf{det}}_\lambda(y, \mathrm{st})$	**return** $\mathsf{GEval}(\tilde{C}, z, d)$
return $\tilde{x}_{1,\ldots,k-1}$	

Now

$$\begin{aligned} &\Pr_{C,x \leftarrow\!\$ \mathcal{D}}\left[\mathsf{Decmpr}\big(\mathsf{Cmpr}(C(x), \Phi(C)), \Phi(C)\big) = C(x)\right] \\ &= \underbrace{\Pr\left[\mathcal{S}^{\mathsf{det}}_\lambda \text{ correct}\right]}_{\geq 1-2\mu \text{ by avg. arg.}} \underbrace{\Pr\left[|\tilde{x}| \leq k-1 \;\middle|\; \mathcal{S}^{\mathsf{det}}_\lambda \text{ correct}\right]}_{=1 \text{ by } (3)} \\ &\geq 1 - \mathsf{negl}(\lambda) \end{aligned}$$

which is a contradiction with HW incompressibility. □

$\mathsf{GCircuit}_{\mathsf{yao}}(C)$

for $g \in \text{Gates}(C)$
  $K_g(0) \leftarrow\!\!\$\ \{0,1\}^\lambda$
  $K_g(1) \leftarrow\!\!\$\ \{0,1\}^\lambda$
for $g \in \text{Gates}(C)$:
  $K_{\text{left}} \leftarrow K_{\mathsf{LeftPred}}(g, C)$
  $K_{\text{right}} \leftarrow K_{\mathsf{RightPred}}(g, C)$
  $K_{\text{output}} \leftarrow K_g$
  for $(b_{\text{left}}, b_{\text{right}}) \in \{0,1\}^2$:
    $d \leftarrow (op(g))(b_{\text{left}}, b_{\text{right}})$
    $c_{\text{in}} \leftarrow\!\!\$\ \mathsf{enc}(K_{\text{left}}(b_{\text{left}}), K_{\text{out}}(d))$
    $c_{\text{out}} \leftarrow\!\!\$\ \mathsf{enc}(K_{\text{right}}(b_{\text{right}}), c_{\text{in}})$
    $\tilde{g} \leftarrow \tilde{g} \cup \{c_{\text{out}}\}$
  $\tilde{C}[g] \leftarrow \tilde{g}$
$K \leftarrow \{(K_g, g), g \in \text{InputGates}(C)\}$
$K_{\text{out}} \leftarrow \{(\mathsf{sort}(K_g(0), K_g(1)), g),$
  $g \in \text{OutputGates}(C)\}$
for $g \in \text{OutputGates}(C)$:
  if $K_{\text{out}}(g) = (K_g(0), K_g(1))$
  then $\text{out}_g \leftarrow 0$
  else $\text{out}_g \leftarrow 1$
$d \leftarrow \{(\text{out}_g, g) : g \in \text{OutputGates}(C)\}$
return $((\tilde{C}, K_{\text{out}}), K, d)$

$\mathsf{GInput}_{\mathsf{yao}}(K, x)$

for $g \in \text{InputGates}(C)$:
  $d_g \leftarrow g$-th pos. of x
  $\tilde{x}[g] \leftarrow K_g(d_g)$
return $\tilde{x}$

$\mathsf{GEval}_{\mathsf{yao}}(\tilde{C}', \tilde{x}, d)$

$(\tilde{C}, K_{\text{out}}) \leftarrow \tilde{C}'$
for $g \in \text{InputGates}(C) : k_g \leftarrow \tilde{x}[g]$
for $g \in \text{Gates}(C) \setminus \text{InputGates}(C)$:
  $k_{\text{left}} \leftarrow k_{\mathsf{LeftPred}}(g, \Phi(C))$
  $k_{\text{right}} \leftarrow k_{\mathsf{RightPred}}(g, \Phi(C))$
  for $c \in \tilde{C}[g]$:
    $m_{\text{out}} \leftarrow \mathsf{dec}(k_{\text{right}}, c)$
    $m_{\text{in}} \leftarrow \mathsf{dec}(k_{\text{lef}}, m_{\text{out}})$
    if $m_{\text{in}} \neq \perp$ **then** $k_g \leftarrow m_{\text{in}}$
for $g \in \text{OutputGates}(C)$:
  $\text{map}_g \leftarrow d(g)$
  if $k_g = K_{\text{out}}(g)(\text{map}_g)$
    then $y[g] = 0$
  if $k_g = K_{\text{out}}(g)(1 - \text{map}_g)$
    then $y[g] = 1$
return y

Fig. 12: Yao's garbling scheme $\mathsf{Gb}_{\mathsf{yao}}$ with split output map.

8 Garbling NC^1 Circuits with Online Complexity $\mathcal{O}(|x|)$

In this section we construct a garbling scheme with online complexity $2n\lambda$ and prove that it is adaptively indistinguishable for NC^1 circuits. Our construction can thus be seen as JSW [JSW17] instantiated just with Yao's garbling scheme and without the somewhere equivocal encryption layer. In a nutshell, the JSW construction garbles a circuit C by first constructing a new circuit C' consisting of two copies of C connected through selector gates that each forward the output of one of the copies of C. The circuit C' is then garbled using Yao's garbling scheme and the resulting garbled circuit is encrypted with a somewhere equivocal encryption scheme [HJO+16]. The somewhere equivocal encryption scheme contributes to the online complexity as decryption keys need to be transmitted in the online phase. We show through a new security analysis that the final encryption step can actually be omitted, thus reducing the online complexity

to $2\lambda|x|$. We now present the construction and then discuss how our proof (cf. Appendix B) differs from JSW.

8.1 Yao's garbling scheme

Let $\mathsf{se} = (\mathsf{enc}, \mathsf{dec})$ be a symmetric encryption scheme with key space $\{0,1\}^\lambda$. Yao's garbling scheme $\mathsf{Gb}_{\mathsf{yao}}$ [Yao82a,Yao86] consists of algorithms $\mathsf{GCircuit}_{\mathsf{yao}}$, $\mathsf{GInput}_{\mathsf{yao}}$ and $\mathsf{GEval}_{\mathsf{yao}}$ that are shown in Fig. 12. For each wire, two uniformly random keys are sampled and assigned values 0 and 1, respectively. To garble a gate g, its left and right predecessor and the corresponding wire keys are determined, and four ciphertexts are computed according to gate operation $op(g)$.

Our description differs slightly from the literature in the split between garbled circuit and output decoding information: Instead of the more traditional view that treats the whole output map (consisting of wire keys and their mapping to bits) as part of the output decoding information, we separate the wire keys K_{out} (sorted lexicographically to hide their association with bits) from the actual map d (pointing now to entries in the key list). The wire keys then become part of the garbled circuit. This is in line with what JSW called *output-key security* in their construction: Yao's garbling scheme is actually adaptively simulatable even when the output wire keys (without the mapping to bits) are sent with the garbled gates in the offline phase. We refer to this version as *weak online Yao*. Interestingly, this is reminiscent of the point-and-permute technique [Rog91] applied only to the output wires, and reduces the online complexity to $n\lambda + m$ (assuming a suitable encoding of gate indices). Note that this change does not affect the combined information contained in the garbled circuit and output decoding information, we simply redistribute it.

$\underline{(\mathsf{Sel}_0 \circ C||C)\ (x)}$
$y_0 \leftarrow C(x)$
$y_1 \leftarrow C(x)$
$y \leftarrow y_0$
return y

Fig. 13: Circuit C'.

8.2 Our construction

For $b \in \{0,1\}$, a *selector gate* Sel_b takes as input two bits u_0 and u_1 and outputs u_b. We define our garbling scheme $\mathsf{Gb} = (\mathsf{GCircuit}, \mathsf{GInput}, \mathsf{GEval})$ as

- $\mathsf{GCircuit}(C) := \mathsf{GCircuit}_{\mathsf{yao}}(C')$, where $C' := \mathsf{Sel}_0 \circ C||C$ (Fig. 13) is a circuit consisting of two copies of C connected by m selector gates Sel_0.
- $\mathsf{GInput}(K, x) := \mathsf{GInput}_{\mathsf{yao}}(K, x||x)$.
- $\mathsf{GEval}(\tilde{C}, \tilde{x}, d) := \mathsf{GEval}_{\mathsf{yao}}(\tilde{C}, \tilde{x}, d)$

The input size of C' is twice that of C, and hence the input encoding size is twice that of Yao's garbling scheme.

Comparison to JSW Construction. JSW additionally encrypt the garbled circuit with a layer of somewhere equivocal encryption (SEE) where the simulator can revoke at pebbling complexity many ciphertexts. The overall construction has

an online complexity independent of the output size of the circuit. However, the online complexity of JSW is affected by the SEE decryption key, which is sent together with the garbled input. The decryption key removes the layer of somewhere equivocal encryption and then the garbled circuit can be evaluated on the garbled input. By the formula in [HJO+16] (Table 1), the size of the decryption key is $t \cdot s \cdot \lambda \cdot \log n$, where for JSW, t is the number of gates to be revoked and s is the size of a garbled gate, which is effectively linear in λ. Since the number of gates to revoked is either the *width* w or the *depth* d of the circuit, the size of the decryption key is either $\mathcal{O}(d\lambda^2)$ or $\mathcal{O}(w\lambda^2)$.

8.3 Security

Theorem 7. *Assuming IND-CPA security of symmetric encryption scheme* se*, the garbling scheme* Gb *is IND-secure (Def. 5) for* NC^1 *circuits and has an online complexity of* $2n\lambda$*.*

Our proof of Theorem 7 follows JSW and HJOSW [HJO+16], except for the game-hop moving from left selectors to right selectors. Here, we show that $C_0(x_0) = C_1(x_1)$ implies that selector gates do not need to be garbled in an input-dependent way. Namely, recall that $\mathsf{Sel}_b(u_0, u_1)$ returns u_b, but $C_0(x_0) = C_1(x_1)$ implies that $u_0 = u_1$ and so the output of $\mathsf{Sel}_b(u_0, u_1)$ is *independent* of bit b. Moreover, $u_0 = u_1$ implies that both input bits to Sel_b and the output bit of Sel_b are all equal, so regardless of b, the output 0-key is encrypted under the input left and right 0-keys and the output 1-key is encrypted under the input left and right 1-keys. The "mixed" ciphertexts (encryptions under the left 1-key and the right 0-key as well as encryptions under the left 0-key and the right 1-key) will never be decrypted, since $C_0(x_0) = C_1(x_1)$ implies that the input bit to the selector gate from $C_0(x_0)$ is equal to the input bit to the selector gate from $C_1(x_1)$, hence their content does not matter. We provide the proof of Theorem 7 in Appendix B.

Acknowledgments

We thank Christoph Egger, Pierre Meyer, the cryptography group at ENS Paris and the anonymous reviewers of Asiacrypt 2023 for the interesting discussion, and for pointing us towards the work of Hubáček and Wichs [HW15].

This work was supported by the Research Council of Finland, Blockchain Technology Laboratory at the University of Edinburgh and Input Output Global.

References

ADL+22. Shashank Agrawal, Wei Dai, Atul Luykx, Pratyay Mukherjee, and Peter Rindal. Paradise: Efficient threshold authenticated encryption in fully malicious model. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology – INDOCRYPT 2022*, pages 26–51, Cham, 2022. Springer International Publishing.

Agr19. Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 191–225. Springer, Heidelberg, May 2019.

AIK04. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Annual Symposium on Foundations of Computer Science*, pages 166–175. IEEE Computer Society Press, October 2004.

AIKW13. Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 166–184. Springer, Heidelberg, August 2013.

AMMR18. Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. DiSE: Distributed symmetric-key encryption. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1993–2010. ACM Press, October 2018.

App14. Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, Heidelberg, 2014.

AS16. Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 125–153. Springer, Heidelberg, January 2016.

BBO07. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, Heidelberg, August 2007.

BDF+18. Chris Brzuska, Antoine Delignat-Lavaud, Cédric Fournet, Konrad Kohbrok, and Markulf Kohlweiss. State separation for code-based game-playing proofs. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 222–249. Springer, Heidelberg, December 2018.

BHK13. Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415. Springer, Heidelberg, August 2013.

BHR12a. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 134–153. Springer, Heidelberg, December 2012.

BHR12b. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012: 19th Conference on Computer and Communications Security*, pages 784–796. ACM Press, October 2012.

BK03. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham,

editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, Heidelberg, May 2003.

BK08. Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

BLMR13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, Heidelberg, August 2013.

BO23. Chris Brzuska and Sabine Oechsner. A state-separating proof for Yao's garbling scheme. In *36th IEEE Computer Security Foundations Symposium - CSF 2023*, pages 137–152. IEEE, 2023.

Bod88. Hans L Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Automata, Languages and Programming: 15th International Colloquium*, pages 105–118. Springer, 1988.

Bod98. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.

BSW03. Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003*, volume 2764 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2003.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

CLTV15. Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497. Springer, Heidelberg, March 2015.

DNRS99. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 523–534. IEEE Computer Society Press, October 1999.

Fre90. Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 1*, AAAI'90, pages 4–9. AAAI Press, 1990.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, October 2013.

GM09. Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *Journal of Combinatorial Theory, Series B*, 99(1):218–228, 2009.

Gol93. Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, March 1993.

Gol11. Oded Goldreich. Candidate one-way functions based on expander graphs. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 76–87. Springer, 2011.

GS18. Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 535–565. Springer, Heidelberg, April / May 2018.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

HJO+16. Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178. Springer, Heidelberg, August 2016.

HLR07. Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 169–186. Springer, Heidelberg, May 2007.

HMS23. Iftach Haitner, Noam Mazor, and Jad Silbak. Incompressiblity and next-block pseudoentropy. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023*, volume 251 of *LIPIcs*, pages 66:1–66:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

HW15. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172. Association for Computing Machinery, January 2015.

JKKR17. Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 158–189. Springer, Heidelberg, August 2017.

JO20. Zahra Jafargholi and Sabine Oechsner. Adaptive security of practical garbling schemes. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology - INDOCRYPT 2020: 21st International Conference in Cryptology in India*, volume 12578 of *Lecture Notes in Computer Science*, pages 741–762. Springer, Heidelberg, December 2020.

JS14. Maurice J. Jansen and Jayalal Sarma. Balancing bounded treewidth circuits. *Theory Comput. Syst.*, 54(2):318–336, 2014.

JSW17. Zahra Jafargholi, Alessandra Scafuro, and Daniel Wichs. Adaptively indistinguishable garbled circuits. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 40–71. Springer, Heidelberg, November 2017.

JW16. Zahra Jafargholi and Daniel Wichs. Adaptive security of Yao's garbled circuits. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 433–458. Springer, Heidelberg, October / November 2016.

Kar23. Pihla Karanko. Different flavours of HILL pseudoentropy and Yao incompressibility entropy. Cryptology ePrint Archive, Paper 2023/1867, 2023. https://eprint.iacr.org/2023/1867.

Khu21. Dakshita Khurana. Non-interactive distributional indistinguishability (NIDI) and non-malleable commitments. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 186–215. Springer, Heidelberg, October 2021.

KKP21. Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. On treewidth, separators and Yao's garbling. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 486–517. Springer, 2021.

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

Muk20. Pratyay Mukherjee. Adaptively secure threshold symmetric-key encryption. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology - INDOCRYPT 2020: 21st International Conference in Cryptology in India*, volume 12578 of *Lecture Notes in Computer Science*, pages 465–487. Springer, Heidelberg, December 2020.

Rog91. Phillip Rogaway. *The round complexity of secure protocols*. PhD thesis, MIT, 1991.

RS86. Neil Robertson and P.D Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

TVZ05. Luca Trevisan, Salil Vadhan, and David Zuckerman. Compression of samplable sources. *Comput. Complex.*, 14(3):186–227, December 2005.

Yao82a. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society Press, November 1982.

Yao82b. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, November 1982.

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

A Invertibility of low treewidth functions

Recall that Kamath, Klein and Pietrzak (KKP [KKP21]) show indistinguishability-based security of Yao's garbling scheme for garbling circuits of constant treewidth. Existing results show that many NP-complete problems become efficiently solvable when restricted to instances with bounded treewidth [Bod88,Fre90,BK08] which implies that circuits of low treewidth are efficiently invertible. On the one hand, invertibility of low-treewidth circuits means that the KPP result is significantly stronger than orginally stated and shows that low-treewidth circuits can be garbled by Yao's garbling scheme with *simulation-based* security. On the other hand, invertibility of low-treewidth circuits also means that the KPP result is not useful for garbling cryptographic functions and especially not for garbling PRGs and circumventing the AIKW impossibility result.

For self-containedness, we now provide an explicit algorithm which inverts any circuit with constant treewidth in polynomial time, and any circuit with polylogarithmic treewidth in quasipolynomial time.

A.1 The inversion algorithm

The inversion algorithm $\mathsf{Inverter}_{\mathsf{Sep}}$ we describe below is an adaptation of Goldreich's inverter for circuits with low *expansion* [Gol11]. Expansion and treewidth of a graph both measure how well-connected a graph is, see Section A.4 for their formal relation. Goldreich's strategy is a divide-and-conquer approach: Split the graph of the circuit into two roughly equally sized halves, guess the values at the *merging points*, and then invert both halves individually, using the strategy recursively.

The algorithm $\mathsf{Inverter}_{\mathsf{Sep}}$ uses the same strategy and relies on separators as the merging points. A separator is a subset of vertices that, when removed, separates the graph into at least two components of roughly equal size, and we take advantage of the existence of separators whose size is at most the treewidth [RS86,Bod98]. The recursive inversion subroutine $\mathsf{Rec\text{-}Inverter}_{\mathsf{Sep}}$ of $\mathsf{Inverter}_{\mathsf{Sep}}$ takes as input a *set* Y of sinks together with their values and tries to invert these. This is useful because the output nodes of the main circuit become sinks of the sub-circuit and thus outputs of the sub-circuit which need to be inverted.

Both algorithms are parametrized with a partial function Sep which, given a circuit, outputs a smallest balanced separator for it. We assume that Sep works on C and all of its sub-components defined recursively by the separators provided by Sep. Sep can either be considered non-uniform information, but can also be computed without too much effort in some cases. Namely, for a circuit of size $|C|$ and treewidth tw, computing Sep takes time at most $|C|^{tw}$ which is polynomial for constant treewidth and quasipolynomial for logarithmic treewidth.

A.2 Definitions and notation

For a circuit C, we use the notation C both for the circuit as well as the underlying directed acyclic graph (DAG). We denote the vertex set and directed edge

set of C by $\mathcal{V}^C$ and $\mathcal{E}^G$, respectively. For any $A, B \subseteq \mathcal{V}^C$, the edges that have sources in A and are incident on B are denoted by $\mathcal{E}^C(A, B)$. For a subset of the nodes $S \subseteq \mathcal{V}^C$, we call a map $\mathsf{a} : S \to \{0,1\}$ an assignment.

Definition 12 (Tree-decomposition [RS86,Bod98]). *A tree decomposition of a directed graph $G = (\mathcal{V}, \mathcal{E})$ is a tree, T , with nodes $\mathcal{X}_1, ..., \mathcal{X}_p$, where each $\mathcal{X}_i \subseteq \mathcal{V}$, satisfies the following properties:*

1. *Each graph vertex is contained in at least one tree node (i.e., $\cup_{i\in[1,p]}\mathcal{X}_i = \mathcal{V}$).*
2. *For every edge $(v, w) \in \mathcal{E}$, there exists a node $\mathcal{X}_i$ that contains both v and w.*
3. *The tree nodes containing a vertex v form a connected subtree of T.*

The width of a tree decomposition is the size of its largest node $\mathcal{X}_i$ minus one. Its treewidth tw is the minimum width amongst all tree decompositions of G.

Definition 13 (Separators [RS86]). *For a graph $G = (\mathcal{V}, \mathcal{E})$, a set $S \subseteq \mathcal{V}$ is said to be a (balanced) separator if the graph restricted to $\mathcal{V} \setminus S$ has at least two components, and each of these components has size at most $\frac{2}{3}|\mathcal{V}|$.*

Theorem 8 ([RS86,Bod98]). *A graph G with treewidth tw has a balanced separator of size at most tw.*

A.3 Inverting circuits with bounded treewidth

Theorem 9. *Let C be a circuit of treewidth tw. Then, there exists a non-uniform algorithm $\mathsf{Inverter}_{\mathsf{Sep}}$ with advice Sep that runs in time $\mathcal{O}\big(2^{3tw\cdot\log|C|}\big)$ and inverts C with probability 1, i.e.,*

$$\Pr_{x\leftarrow\$\{0,1\}^n, y\leftarrow C(x)}\big[\mathsf{Inverter}_{\mathsf{Sep}}(y, C) \in C^{-1}(x)\big] = 1$$

where n is the input length of C. In particular, if C is polynomial in the security parameter, then $\mathsf{Inverter}_{\mathsf{Sep}}$ runs in polynomial time for constant treewidth and in quasipolynomial time for polylogarithmic treewidth.

The inversion algorithm $\mathsf{Inverter}_{\mathsf{Sep}}$ for advice Sep is shown in Fig. 14.

Proof. We start by introducing some notation. Given a set of separators S for a circuit C and an assignment $\mathsf{a} : S \to \{0,1\}$, we denote by $\mathrm{Partition}(C, S, \mathsf{a})$ the set of connected components CC emerging from splitting C at the separators. We consider the separators neighboring a connected component CC to be part of it. To make the splitting meaningful, we need to make the following modifications to C before splitting:

A separator can only be placed on a gate with a *single* input. If a separator were to be placed on a *two-input* node, then the two input wires and the single output wire[8] is replaced by an identity node, and instead of one separator, three separators are placed on each of the three identity gates.

[8] High fan-out can be placed below the new identity node.

$\mathsf{Inverter}_{\mathsf{Sep}}(C, y)$

bound $\leftarrow \log |C|$
$Y \leftarrow \{(i, y_i) | i \in \mathsf{sinks}(C)\}$
$X \leftarrow \emptyset$
$X' \leftarrow \mathsf{Rec\text{-}Inverter}(C, X, Y)$
$x' \leftarrow \mathsf{encodeAsg}(X')$
return x'

$\mathsf{bruteforce}(C, X, Y)$

// iterate over all assignments to X
for $\mathsf{a} : X \to \{0,1\}$:
  if $C(\mathsf{a}(X)) = Y$ **then**
    $X' \leftarrow \{(x, \mathsf{a}(x)) : x \in X\}$
    return X'
$X' \leftarrow \{(x, \perp) : x \in X\}$
return X'

$\mathsf{Rec\text{-}Inverter}_{\mathsf{Sep}}(C, X, Y)$

if $|C| \leq$ bound **then** :
  $X' \leftarrow \mathsf{bruteforce}(C, X, Y)$
  return X'
else :
  $S \leftarrow \mathsf{Sep}(C)$
  // iterate over all assignments to S
  for $\mathsf{a} : S \to \{0,1\}$:
    $X' \leftarrow \emptyset$
    for $(CC, X_{CC}, Y_{CC}) \in \mathrm{Partition}(C, S, \mathsf{a})$:
      $X'' \leftarrow\!\!\$\ \mathsf{Rec\text{-}Inverter}_{\mathsf{Sep}}(CC, X_{CC}, Y_{CC})$
      $X' \leftarrow X' \cup X''$
    if $X \setminus S = \{x : (x,0) \in X' \vee (x,1) \in X'\}$
      then $X' \leftarrow X' \cup \{(x, \mathsf{a}(x)) : x \in S \cap X\}$
      return X'
  $X' \leftarrow \{(x, \perp) : x \in X'\}$
  return X'

Fig. 14: Inversion algorithm for low treewidth circuits.

Now, a separator node is either a *constant value* source or a sink of the connected component CC (or disconnected from it in which case it is not included). If it is a sink, then it provides an additional output value which needs to be respected in the process of inversion.

Since the set of input values becomes smaller and the set of output nodes changes in their process, we actually consider the triple

$$(CC, X_{CC}, Y_{CC}) \in \mathrm{Partition}(C, S, \mathsf{a})$$

to be specified by $\mathrm{Partition}(C, S, \mathsf{a})$, i.e., the set X_{CC} of input nodes as well as the set Y_{CC} of pairs (g, y_g), where g is the index of the output node and y_g is its value.

If X is a set of input nodes and $\mathsf{a} : X \to \{0,1\}$ and Y is a set of pairs (g, y_g), then we use the notation

$$C(\mathsf{a}(X)) = Y$$

to express that circuit C when evaluated on inputs X carrying values according to a, the output gates g all carry the values y_g as specified by Y.

Finally, note that when using $|C|$, we refer to the number of *non-constant* gates in C.

Inverter runtime. We first turn to the runtime of inverting a circuit C when $|C| \leq$ bound $= \log |C|$ (base case). In particular, the input size is at a subset of $|C|$ and hence is upper bounded by $|C|$. Note that bruteforce runs in time

exponential in the input size, since it enumerates over all possible assignments to the input. Hence, inversion at the base case runs in time $\mathcal{O}\big(2^{\log|C|}\big) = \mathcal{O}(|C|)$.

For the recursion, observe that by definition of a balanced separator (Definition 13), each connected component is at most of size $\frac{2}{3}$ of the size of the previously largest connected component. Thus, the depth of the recursion is at most logarithmic in $|C|$. Each recursion involves 2^{3tw} possible branches. Hence, the overall runtime is $(2^{3tw})^{\log|C|}$, which is is polynomial in the security parameter if tw is constant and quasipolynomial if tw is logarithmic.

Inverter correctness. We now prove that if an inverse exists, then the inverter indeed finds an inverse.

Base Case. $\mathsf{bruteforce}(C, X, Y)$ tries out all possible inputs. Therefore, if a pre-image exists, then $\mathsf{bruteforce}$ finds it.

Recursion. Given input (C, X, Y) which has (at least) solution X^* and assuming that all recursion subroutines find pre-images if they exists are sound, we need to show that $\mathsf{RecInverter}(C, X, Y)$ finds a pre-image of Y under C. To show this, it suffices to argue that for *some* assignment $\mathsf{a} : S \to \{0,1\}$, all subroutines return a pre-image. Since we know that solution X^* exists, we can run C on solution X^* and see which values the separators S take to obtain assignment $\mathsf{a} : S \to \{0,1\}$. Since X^* is a solution for Y under C and is consistent with a, the corresponding restrictions of X^* are solutions for the subroutines. By the assumption that the subroutines find a pre-image if it exists, they all return a pre-image. As the connected components form a partition, each input is contained in exactly one partition or is a separator. Thus, indeed, all of X is covered by the union of the separators and the connected components and $\mathsf{RecInverter}(C, X, Y)$ returns a complete pre-image.

□

A.4 Connection to circuit expansion

Treewidth and expansion both capture *how connected* a graph is, and Grohe and Marx [GM09] show that treewidth and expansion are indeed closely related, as the treewidth of a graph is at least as high as its (vertex) expansion. We recall (parameterized) vertex expansion and Proposition 1 of [GM09]:

Definition 14 (Vertex Expansion [GM09]). *Let G be a graph. For a set X, let $S(X)$ denote the set of vertices in $V(G) \setminus X$ that are adjacent to X. For every $\alpha \in [0,1]$, the* vertex expansion *of G with parameter α is defined as*

$$\min_{\substack{X \subseteq V(G) \\ 0 < |X| \leq \alpha \cdot |V(G)|}} \frac{|S(X)|}{|X|}.$$

Theorem 10 (Grohe-Marx [GM09]). *Let $n \geq 1$ and $0 \leq \alpha \leq 1$. Then for every n-vertex graph G we have*

$$tw(G) \geq \lfloor vx_\alpha(G) \cdot (\alpha/2) \cdot n \rfloor.$$

Note that when $\alpha = \frac{1}{2}$, we recover the usual (un-parameterized) notion of vertex expansion. We then obtain from Theorem 10 that

$$tw(G) \geq \left\lfloor \frac{n}{4} \cdot vx(G) \right\rfloor .$$

Hence, as soon as $n > 3$ the treewidth of any n-vertex graph is at least that of its vertex expansion.

B Proof of Theorem 7

In order to prove adaptive indistinguishability of our construction, we define a number of hybrid games that differ in which circuits the circuit C' consists of. Each hybrid game $\mathtt{Hyb}_{\mathcal{A}}^{b_L,b_R,\mathrm{sel}}$ in Fig. 15 garbles circuit $C' = \mathsf{Sel}_{\mathrm{sel}} \circ C_{b_L} || C_{b_R}$. It is easy to see that $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$ is equivalent to game $\mathtt{Ind}_{\mathcal{A},\mathsf{Gb}}^{0}$, and $\mathtt{Hyb}_{\mathcal{A}}^{1,1,0}$ is equivalent to game $\mathtt{Ind}_{\mathcal{A},\mathsf{Gb}}^{1}$. We then establish two statements about the relation between hybrids that will allow us to modify the circuit C' that is garbled by switching the internal circuit and the selector gates.

$\mathtt{Hyb}_{\mathcal{A}}^{b_L,b_R,\mathrm{sel}}(1^\lambda)$

$(C_0, C_1, \mathrm{st}) \leftarrow\$ \mathcal{A}(1^\lambda)$
$n \leftarrow \mathsf{InputLength}(C_0)$
$K \leftarrow\$ \mathsf{Ksample}(1^n)$
$C' \leftarrow \mathsf{Sel}_{\mathrm{sel}} \circ C_{b_L} || C_{b_R}$
$(\tilde{C}, d) \leftarrow\$ \mathsf{GCircuit}_{\mathsf{yao}}(C')$
$(x_0, x_1, \mathrm{st}') \leftarrow\$ \mathcal{A}(\tilde{C}, d, \mathrm{st})$
$x \leftarrow x_{b_L} || x_{b_R}$
$\tilde{x} \leftarrow\$ \mathsf{GInput}_{\mathsf{yao}}(K, x)$
$b' \leftarrow\$ \mathcal{A}(\tilde{x}, \mathrm{st'})$
return b'

Fig. 15: Hybrid $\mathtt{Hyb}_{\mathcal{A}}^{b_L,b_R,\mathrm{sel}}$.

Circuit switch. The following lemma allows us to switch the circuit that is not selected by a circuit of equivalent topology as long as both circuit/input pairs lead to the same output:

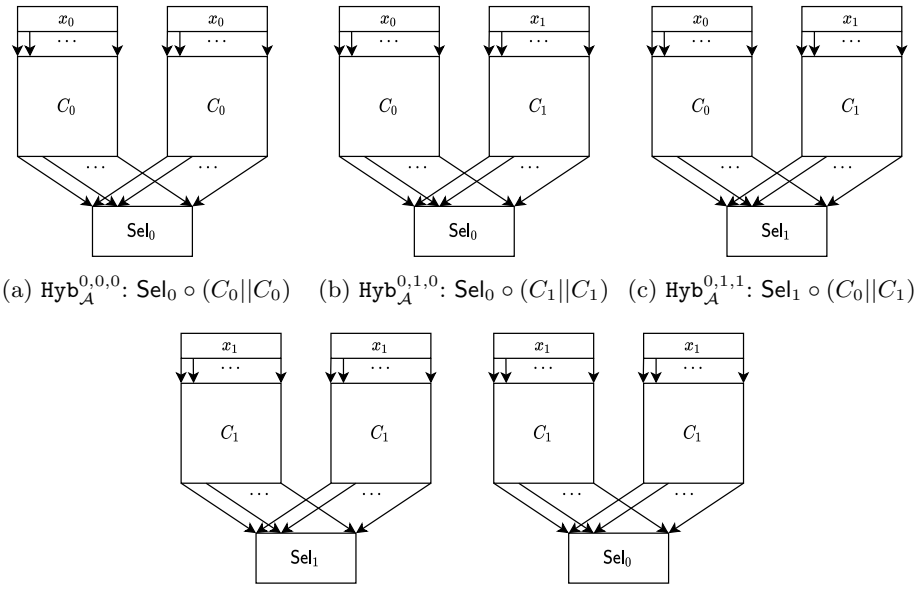
Lemma 3 (Circuit switch). *Assuming the IND-CPA security of the underlying encryption scheme, for any PPT adversary $\mathcal{A}$ which queries circuits C_0, C_1 with equal input length and inputs x_0 and x_1 such that $C_0(x_0) = C_1(x_1)$ and $\Phi(C_0) = \Phi(C_1)$,*

$$\left| \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,0,0}\right] - \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,0}\right] \right| = \mathsf{negl}(\lambda) \tag{4}$$

and

$$\left| \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,1}\right] - \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{1,1,1}\right] \right| = \mathsf{negl}(\lambda). \tag{5}$$

Lemma 3 will be proved in Appendix C through a reduction to the adaptive simulatability of Yao's garbling scheme: Intuitively, simulatability guarantees garbling circuit C_0 on input x_0 is indistinguishable from a simulation given topology $\Phi(C_0)$ and output $C_0(x_0)$, which in turn is indistinguishable from garbling C_1 on input x_1 as both circuits have the same topology and output.



(a) $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$: $\mathsf{Sel}_0 \circ (C_0||C_0)$ (b) $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$: $\mathsf{Sel}_0 \circ (C_1||C_1)$ (c) $\mathtt{Hyb}_{\mathcal{A}}^{0,1,1}$: $\mathsf{Sel}_1 \circ (C_0||C_1)$

(d) $\mathtt{Hyb}_{\mathcal{A}}^{1,1,1}$: $\mathsf{Sel}_1 \circ (C_1||C_1)$ (e) $\mathtt{Hyb}_{\mathcal{A}}^{1,1,0}$: $\mathsf{Sel}_0 \circ (C_0||C_0)$

Fig. 16: Overview of game hops in proof of Theorem 7.

Selector switch. The second lemma allows to switch the selector gates:

Lemma 4 (Selector switch). *Assuming the IND-CPA security of the underlying encryption scheme, we have that for any PPT adversary $\mathcal{A}$ which queries NC^1 circuits C_0, C_1 with equal input length and inputs x_0 and x_1 such that $C_0(x_0) = C_1(x_1)$ and $\Phi(C_0) = \Phi(C_1)$,*

$$\left|\Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,0}\right] - \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,1}\right]\right| = \mathsf{negl}(\lambda) \tag{6}$$

and

$$\left|\Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{1,1,1}\right] - \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{1,1,0}\right]\right| = \mathsf{negl}(\lambda). \tag{7}$$

Proof of Theorem 7. The proof proceeds as a sequence of game hops, where indistinguishability of games for each hop was shown in Lemma 3 and 4, respectively. Fig. 16 shows the circuit to be garbled in each game. Observing that $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$ is equivalent to game $\mathtt{Ind}_{\mathcal{A},\mathsf{Gb}}^{0}$, and $\mathtt{Hyb}_{\mathcal{A}}^{1,1,0}$ is equivalent to game $\mathtt{Ind}_{\mathcal{A},\mathsf{Gb}}^{1}$ then concludes our proof.

B.1 Pebbling Strategies

To prove Lemma 10, we now describe a pebbling strategy PebbleSelectors which moves from the left to the right configuration for a selector circuit using G moves and L-R moves. We analyze the complexity of the PebbleSelectors afterwards.

Pebbling Strategy. For every gate i in circuit C, we denote by $\mathsf{LeftPred}(C, i)$ and $\mathsf{RightPred}(C, i)$ the two predecessor gates.

- $\mathsf{PebbleSelectors}(C)$:
  1. For each selector gate i in C:
     - Run $\mathsf{RecPutGrey}(C, \mathsf{LeftPred}(C, i))$ and $\mathsf{RecPutGrey}(C, \mathsf{RightPred}(C, i))$
     - Replace the L pebble on gate i with a R pebble.
     - Run $\mathsf{RecRemoveGrey}(C, \mathsf{LeftPred}(C, i))$ and $\mathsf{RecRemoveGrey}(C, \mathsf{RightPred}(C, i))$

- $\mathsf{RecPutGrey}(C, i)$:
  1. If gate i an input gate, put a grey pebble on i and **return**.
  2. Run $\mathsf{RecPutGrey}(C, \mathsf{LeftPred}(C, i))$, $\mathsf{RecPutGrey}(C, \mathsf{RightPred}(C, i))$.
  3. Put a grey pebble on gate i.
  - Run $\mathsf{RecRemoveGrey}(C, \mathsf{LeftPred}(C, i))$ and $\mathsf{RecRemoveGrey}(C, \mathsf{RightPred}(C, i))$

- $\mathsf{RecRemoveGrey}(C, i)$: This is the same as $\mathsf{RecPutGrey}$, except instead of putting a grey pebble on gate i, in steps 1 and 3 we remove it.

To prove Lemma 10, we establish the following sublemma.

Lemma 5 (Single Output Gate Pebbling Complexity). *Let C be a selector circuit of depth d, with selector gates at the very bottom (on the outputs) and m outputs. If we start with an arbitrary configurations of left and right pebbles on the selector gates and white pebbles everywhere on C, then for every output gate i, there is a (G,LR)-pebbling strategy using at most $2d-1$ grey pebbles and t steps such that in the end, the i-th output (selector) gate is pebbled right instead of left (or vice versa), all other selector gates are in the same configuration as in the beginning, and all other gates are pebbled white.*

Here, $t = 4^{d-1}$.

Notice that our pebbling strategy is simply the JW pebbling strategy with the addition of (nearly identical) rules of pebbling selector gates, and hence the analysis is essentially the same, and is included here for completeness.

More specifically, we need to analyze both the runtime and space complexity (number of grey pebbles) of $\mathsf{RecPutGrey}$.

Runtime. We want to find a closed expression for $t : \mathbb{N} \to \mathbb{N}$, which is the number of steps made to place a pebble on layer d (and nowhere else). Note that the number of steps needed to remove a pebble and the number of steps to place a pebble are identical. Thus, we know that $t(1) = 1$ and $t(d) = 4t(d-1)$. Thus, we obtain $t(2) = 4$, $t(3) = 4 \cdot 4$ and $t(d) = 4^{d-1}$.

Pebbling complexity. Similarly, we want to find a closed expression for $s : \mathbb{N} \to \mathbb{N}$ which is the number of grey simultaneous pebbles needed to place a pebble on layer d. To count the number of grey pebbles needed, first observe that, in order to place a pebble on layer d, the algorithm first places a pebble on layer $d-1$ and then another pebble on layer $d-1$ and then a pebble on layer d. Then, to remove the first of the two pebbles from layer $d-1$, one needs $s(d-1)$ pebbles, while there are two more pebbles lying on layer d and lying on the co-parent. Thus, we have that $s(d) = s(d-1) + 2$. Moreover, $s(1) = 1$. Hence, we obtain $s(d) = 1 + 2 \cdot (d-1) = 2d - 1$. □

C Proof of Circuit Switch Lemma

We first establish the adaptive simulatability of weak online Yao (cf. Section 8.1) before returning to the proof of Lemma 3.

Lemma 6. *Assuming the IND-CPA security of the underlying encryption scheme, weak online Yao is adaptively SIM-secure for NC^1 circuits.*

Proof sketch. The statement follows directly from a modification to the adaptive simulatability proof due to Jafargholi and Wichs [JW16]: Observe that when the output map is computed in the online phase, the map actually only assigns bit values to existing output wire keys that were sampled in the offline phase. The output wire keys are moreover not used as encryption keys anywhere else in the circuit. Since the encryption scheme used to garble gates is further assumed to be IND-CPA secure, revealing the output wire keys (used as messages in the IND-CPA game) does not affect security. Wire keys are moreover chosen completely independently of the rest of the garbling procedure and hence we can use the messages provided by an IND-CPA adversary. □

Lemma 3. We are going to prove the first statement, the second statement follows from an analogous argument. Let $\mathcal{S}_{\mathsf{yao}}$ be the simulator for $\mathsf{Gb}_{\mathsf{yao}}$ obtained from Lemma 6 when garbling C_0. Games $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$ and $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$ differ only in the circuit whose output is not selected. Indistinguishability of the two games is shown in two steps. We define an intermediate game $\mathsf{Hyb}_{\mathcal{S}_{\mathsf{yao}}}$ that *simulates* the garbling of C_0 on input x_0 using $\mathcal{S}_{\mathsf{yao}}$, given only the topology of C_0 and output $y = C_0(x_0) = C_1(x_1)$. This game is shown to be indistinguishable from both $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$ and $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$ via reductions to adaptive simulatability of garbling C_0 and C_1 respectively. Since $C_0(x_0) = C_1(x_1)$ and both circuits have the same topology, and hence both circuits share the same simulator, game $\mathsf{Hyb}_{\mathcal{S}_{\mathsf{yao}}}$ can be shown to be indistinguishable from both $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$ and $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$.

For the game hop from $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$ to $\mathsf{Hyb}_{\mathcal{S}_{\mathsf{yao}}}$, we construct adaptive simulatability distinguisher $\mathcal{B}$ from adaptive indistinguishability distinguisher $\mathcal{A}$ as shown in Fig. 17. The reduction obtains two circuits C_0, C_1 from $\mathcal{A}$ and forwards C_0 to the challenger. Upon receiving garbled circuit $\tilde{C}_0$ and output wire keys L_{out}, $\mathcal{B}$ computes a random output map K_{out} using the keys L_{out}. With the help of K_{out}, the reduction then garbles $\mathsf{Sel}_0 \circ C_0$ that can be combined with $\tilde{C}_0$ to

Reduction $\mathcal{B}(1^\lambda)$:

$(C_0, C_1) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$

Send C_0 to the challenger and receive $(\tilde{C_0}, L_{\text{out}})$

for $g \in \text{OutputGates}(C_0)$:

$y_g \leftarrow\!\$\ \{0,1\}$

$(k_L, k_R) \leftarrow L_{\text{out}}(g)$

$K_{\text{out}}(y_g) \leftarrow k_L$

$K_{\text{out}}(1-y_g) \leftarrow k_R$

Garble $\mathsf{Sel}_0 \circ C_0$ according to $\mathsf{Gb}_{\mathsf{yao}}$ and obtain $(\tilde{C}, K, d^*)$, using K_{out} as $K_{\mathsf{RightPred}}$ for selector gates

$\tilde{C^*} \leftarrow \tilde{C} \cup \tilde{C_0}$

$(x_0, x_1) \leftarrow\!\$\ \mathcal{A}(\tilde{C^*}, d^*)$

Send x_0 to the challenger and receive $(\tilde{x^*}, d)$

$b' \leftarrow\!\$\ \mathcal{A}(\tilde{x^*})$

return b'

Fig. 17: Reduction $\mathcal{B}$.

obtain d^* as well as either the garbled circuit $\tilde{C^*} = \mathsf{Sel}_0 \circ C_0 || C_0$ or a similar garbled circuit $\tilde{C^*}$ where C_0 is simulated. Finally, $\mathcal{B}$ queries $\mathcal{A}$ with input $\tilde{C^*}$ and d^* to obtain inputs x_0, x_1, sends x_0 to the challenger, forwards the response to $\mathcal{A}$, and returns $\mathcal{A}$'s output bit b'.

Observe that the output map K_{out} computed by $\mathcal{B}$ does not necessarily match the one of the challenger. This does however not influence the distributions of ciphertext in a garbled selector gate since $\mathsf{Sel}_0(b_L, b_R) = \mathsf{Sel}_0(b_L, 1-b_R)$. Hence $\tilde{C^*}$ is either $\mathsf{Sel}_0 \circ C_0 || C_0$ garbled according to Yao, or a simulated garbled circuit. In the first case, $\mathcal{B}$ is equivalent to $\mathtt{Hyb}_{\mathcal{A}}^{0,0,0}$. In the second case, $\mathcal{B}$ is equivalent to $\mathsf{Hyb}_{\mathcal{S}_{\mathsf{yao}}}$. The game hop from $\mathsf{Hyb}_{\mathcal{S}_{\mathsf{yao}}}$ to $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$ follows via a similar reduction that sends C_1 and x_1 to the challenger instead. □

D Selector Switch

We use the language of pebbling games to express our result. We proceed as follows: To prove the indistinguishability of $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$ and $\mathtt{Hyb}_{\mathcal{A}}^{0,1,1}$, we define different *pebbles* which we can place on gates and which describe how different gates are garbled. Given such a *pebbling configuration*, we describe a hybrid game which garbles the circuit according to the pebbling configuration. We then define *legal moves* between pebbling configurations (also knowns as pebbling *rules*), and we show that

(1) each *legal move* reduces to IND-CPA security, and that
(2) we do not make too many moves.

Additionally, we also prove that we do not need more than a certain number of so-called grey pebbles, but let's return to this later.

D.1 Pebbling Colors and Rules

We now first introduce our circuit class, then our pebbling colors, and then our pebbling rules.

Definition 15 (Selector Circuit). *We call a circuit a* selector circuit *if it consists of two (potentially identical) circuits C_0 and C_1, composed in parallel and a layer of selector gates at the bottom determining whether the output of the left or the right circuit is returned.*

Colors Selector nodes either have a *left* or a *right* pebble on them. Other nodes have either a *white* or a *grey* pebble on them, leading to four colors: *left* (L), *right* (R), *white* (W) and *grey* (G).

Operations associated with colors. A gate with a *white* pebbled is garbled according to the operation specified by this gate. For a gate with a *grey* pebble on it, all ciphertexts contain the *active* output key, i.e., the operation computed by a grey bit depends on which key is active thus also called *input-dependent* garbling. For a selector gate with left pebble, we have as operation $\mathsf{Sel}_0(x_{\text{left}}, x_{\text{right}}) = x_{\text{left}}$, and for a selector gate with a right pebble on it, we have as operation $\mathsf{Sel}_1(x_{\text{left}}, x_{\text{right}}) = x_{\text{right}}$. Note that selector garblings are not input-dependent and that $\mathsf{Sel}_0(0,0) = \mathsf{Sel}_1(0,0) = 0$ and $\mathsf{Sel}_0(1,1) = \mathsf{Sel}_1(1,1) = 1$.

Configurations. In Section 8.3, different hybrid games used different variants of the circuit $C' = \mathsf{Sel}_{\text{sel}} \circ C_{b_L} || C_{b_R}$. In this section, we are interested in $C' = \mathsf{Sel}_{\text{sel}} \circ C_0 || C_1$ for $\text{sel} \in \{0,1\}$ and $C' = \mathsf{Sel}_{\text{sel}} \circ C_1 || C_0$ for $\text{sel} \in \{0,1\}$. Since we want to swap a left selector by a right selector (or vice versa), in this section, the left or right *pebble* on the selector gate determines whether it is a left or right selector. The hybrids $\mathtt{Hyb}_{\mathcal{A}}^{0,1,0}$ and $\mathtt{Hyb}_{\mathcal{A}}^{0,1,1}$ all garble the entire circuit as in Yao's garbling scheme and use either left or right selectors. We give names to the corresponding pebbling configurations.

Definition 16 (Left/Right Configuration). *If C' is a selector circuit, we call a pebbling configuration config_{left} of C' the* left configuration *if all selector gates are pebbled left and all other gates are pebbled white. We call a pebbling configuration config_{right} of C' the* right configuration *if all selector gates are pebbled right and all other gates are pebbled white.*

Pebbling Rules. Our pebbling games only allow the use of the following two pebbling rules, also illustrated in Fig. 18.

grey (G) If both parents of a node N are grey, then a white pebble on N can be replaced by a grey pebble and a grey pebble on N can be replaced by a white pebble.

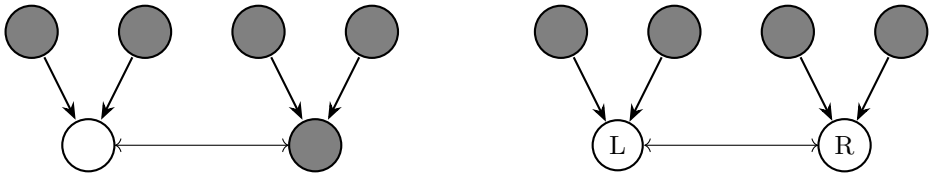


Fig. 18: Pebbling rules for input dependent gates (left) and left to right selector gate pebbling rules (right).

left-right (LR) If both parents of a selector node N are grey, then a left pebble on N can be replaced by a right pebble and a right pebble on N can be replaced by a left pebble.

Definition 17 (G-LR pebbling strategy). *We call a pebbling strategy G-LR, if it only uses the G rule and the LR rule.*

Garbling subroutine. We now define a new garbling subroutine $\mathsf{2Garble}_{\mathsf{config},\mathsf{bit}}$ in Fig. 19. It is parameterized by a configuration config that indicates how to garble each gate, and partial function bit that contains the current guessed values of the output of a gate. $\mathsf{Gb}_{\mathsf{config},\mathsf{bit}}$ takes as input two circuits C_0 and C_1, constructs the circuit $\mathsf{Sel}_{\mathsf{config}} \circ C_0||C_1$ and then garbles $\mathsf{Sel}_{\mathsf{config}} \circ C_0||C_1$ as determined by the colors of the pebbling configuration. Note that

$$\begin{aligned}
\mathsf{GCircuit}(\mathsf{Sel}_0 \circ C_0||C_1) &= \mathsf{2Garble}_{\mathsf{config}_{\mathrm{left}},\perp}(C_0, C_1) \\
\mathsf{GCircuit}(\mathsf{Sel}_1 \circ C_0||C_1) &= \mathsf{2Garble}_{\mathsf{config}_{\mathrm{right}},\perp}(C_0, C_1) \\
\mathsf{GCircuit}(\mathsf{Sel}_1 \circ C_1||C_1) &= \mathsf{2Garble}_{\mathsf{config}_{\mathrm{right}},\perp}(C_1, C_1) \\
\mathsf{GCircuit}(\mathsf{Sel}_0 \circ C_1||C_1) &= \mathsf{2Garble}_{\mathsf{config}_{\mathrm{left}},\perp}(C_1, C_1) \\
\mathsf{GInput}(K, x_0||x_1) &= \mathsf{2Input}(K, x_0, x_1) \\
\mathsf{GEval}(\tilde{C}, \tilde{x}, d) &= \mathsf{2Eval}(\tilde{C}, \tilde{x}, d)
\end{aligned} \tag{8}$$

D.2 Main lemmas for the proof of Lemma 4

For the proof of Lemma 4, we prove Equation 6, and the proof of Equation 7 is analogous. The proof of Equation 6 consists of four lemmas.

The first lemma introduces the guessing of ℓ many bits into the game and shows that we loose (exactly) a factor of $2^{-\ell}$ in security in this step. The next two lemmas bound the advantage of an adversary between two hybrids that differ exactly by one pebbling rule step. And the fourth lemma bounds the number of these steps by a polynomial. We now first state the lemmas, then show that Equation 6 follows from them, and then provide the proof of the lemmas in Appendix D.3, Appendix D.4, Appendix D.5 and Appendix B.1.

$\mathsf{2Garble}_{\mathsf{config},\mathsf{bit}}(C_0, C_1)$

for $g \in \mathrm{Gates}(\mathsf{Sel} \circ C_0||C_1)$:
  if $\mathsf{config}(g) = (\mathrm{reduction}, op_0, op_1)$ **then**
    $p^L \leftarrow \mathsf{LeftPred}(g, \mathsf{Sel} \circ C_0||C_1)$
    $p^R \leftarrow \mathsf{RightPred}(g, \mathsf{Sel} \circ C_0||C_1)$
for $g \in \mathrm{Gates}(\mathsf{Sel} \circ C_0||C_1) \backslash \{p^l, p^r\}$:
    $K_g(0) \leftarrow\!\!\$\ \{0,1\}^\lambda$, $K_g(1) \leftarrow\!\!\$\ \{0,1\}^\lambda$
$b^L \leftarrow \mathsf{bit}(p^L); b^R \leftarrow \mathsf{bit}^R(p^R)$
$K_{p^L}(1-b_L) \leftarrow \perp; K_{p^L}(1) \leftarrow \mathsf{GETA}_L(b_L)$
$K_{p^R}(1-b_R) \leftarrow \perp; K_{p^R}(1) \leftarrow \mathsf{GETA}_R(b_R)$
for $g \in \mathrm{Gates}(\mathsf{Sel} \circ C_0||C_1)$:
  $K_{\mathrm{left}} \leftarrow K_{\mathsf{LeftPred}}(g, \mathsf{Sel} \circ C_0||C_1)$
  $K_{\mathrm{right}} \leftarrow K_{\mathsf{RightPred}}(g, \mathsf{Sel} \circ C_0||C_1)$
  $K_{\mathrm{output}} \leftarrow K_g$
  if $\mathsf{config}(\tilde{g}) = W$:
    $\tilde{g} \leftarrow\!\!\$\ \mathsf{garble}^{\mathrm{white}}(K_{\mathrm{left}}, K_{\mathrm{right}}, K_{\mathrm{out}}, op(g))$
  if $\mathsf{config}(\tilde{g}) = L$:
    $\tilde{g} \leftarrow\!\!\$\ \mathsf{garble}^{\mathrm{white}}(K_{\mathrm{left}}, K_{\mathrm{right}}, K_{\mathrm{out}}, \mathsf{Sel}_0)$
  if $\mathsf{config}(\tilde{g}) = R$:
    $\tilde{g} \leftarrow\!\!\$\ \mathsf{garble}^{\mathrm{white}}(K_{\mathrm{left}}, K_{\mathrm{right}}, K_{\mathrm{out}}, \mathsf{Sel}_1)$
  if $\mathsf{config}(\tilde{g}) = G$:
    $d \leftarrow \mathrm{bit}(g)$
    $\tilde{g} \leftarrow\!\!\$\ \mathsf{garble}^{\mathrm{grey}}(K_{\mathrm{left}}, K_{\mathrm{right}}, K_{\mathrm{out}}, d)$
  if $\mathsf{config}(\tilde{g}) = (\mathrm{reduction}, op_0, op_1)$:
    $\tilde{g} \leftarrow\!\!\$\ \mathsf{garble}^{\mathrm{reduction}}(K_{\mathrm{out}}, op_0, op_1)$
  $\tilde{C}[g] \leftarrow \tilde{g}$
$K \leftarrow \{(K_g, g), g \in \mathrm{InputGates}(\mathsf{Sel} \circ C_0||C_0)\}$
for $g \in \mathrm{OutputGates}(C)$:
  if $K_{\mathrm{out}}(g) = (K_g(0), K_g(1))$
   then $\mathrm{out}_g \leftarrow 0$
  else $\mathrm{out}_g \leftarrow 1$
$d \leftarrow \{(\mathrm{out}_g, g) : g \in \mathrm{OutputGates}(C)\}$
return $((\tilde{C}, K_{\mathrm{out}}), K, d)$

$\mathsf{2Input}(K, x_0, x_1)$

for $g \in \mathrm{InputGates}(\mathsf{Sel}_0 \circ C_0||C_0)$:
  $d_g \leftarrow$ Value of $x_0||x_1$ at position g
  $\tilde{x}[g] \leftarrow K_g[d_g]$
return $\tilde{x}$

$\mathsf{garble}^{\mathrm{white}}(K_{\mathrm{left}}, K_{\mathrm{right}}, K_{\mathrm{out}}, op)$

$\tilde{g} \leftarrow \emptyset$
for $(b_{\mathrm{left}}, b_{\mathrm{right}}) \in \{0,1\}^2$:
  $d \leftarrow op(b_{\mathrm{left}}, b_{\mathrm{right}})$
  if $\mathsf{LeftPred}(g, \mathsf{Sel} \circ C_0||C_1) = p^{L/R}$ **then**
    $c_{\mathrm{in}} \leftarrow \mathsf{ENC}_{L/R}(b_{\mathrm{left}}, K_{\mathrm{out}}(d), K_{\mathrm{out}}(d))$
  else $c_{\mathrm{in}} \leftarrow\!\!\$\ \mathsf{enc}(K_{\mathrm{left}}(b_{\mathrm{left}}), K_{\mathrm{out}}(d))$
  if $\mathsf{RightPred}(g, \mathsf{Sel} \circ C_0||C_1) = p^{L/R}$ **then**
    $c_{\mathrm{out}} \leftarrow \mathsf{ENC}_{L/R}(b_{\mathrm{right}}, c_{\mathrm{in}}, c_{\mathrm{in}})$
  else $c_{\mathrm{out}} \leftarrow\!\!\$\ \mathsf{enc}(K_{\mathrm{right}}(b_{\mathrm{right}}), c_{\mathrm{in}})$
  $\tilde{g} \leftarrow \tilde{g} \cup \{c_{\mathrm{out}}\}$
return $\tilde{g}$

$\mathsf{garble}^{\mathrm{grey}}(K_{\mathrm{left}}, K_{\mathrm{right}}, K_{\mathrm{out}}, d)$

$\tilde{g} \leftarrow \emptyset$
for $(b_{\mathrm{left}}, b_{\mathrm{right}}) \in \{0,1\}^2$:
  if $\mathsf{LeftPred}(g, \mathsf{Sel} \circ C_0||C_1) = p^{L/R}$ **then**
    $c_{\mathrm{in}} \leftarrow \mathsf{ENC}_{L/R}(b_{\mathrm{left}}, K_{\mathrm{out}}(d), K_{\mathrm{out}}(d))$
  else $c_{\mathrm{in}} \leftarrow\!\!\$\ \mathsf{enc}(K_{\mathrm{left}}(b_{\mathrm{left}}), K_{\mathrm{out}}(d))$
  if $\mathsf{RightPred}(g, \mathsf{Sel} \circ C_0||C_1) = p^{L/R}$ **then**
    $c_{\mathrm{out}} \leftarrow \mathsf{ENC}_{L/R}(b_{\mathrm{right}}, c_{\mathrm{in}}, c_{\mathrm{in}})$
  else $c_{\mathrm{out}} \leftarrow\!\!\$\ \mathsf{enc}(K_{\mathrm{right}}(b_{\mathrm{right}}), c_{\mathrm{in}})$
  $\tilde{g} \leftarrow \tilde{g} \cup \{c_{\mathrm{out}}\}$
return $\tilde{g}$

$\mathsf{garble}^{\mathrm{reduction}}(K_{\mathrm{out}}, d)$

$\tilde{g} \leftarrow \emptyset$
for $(b_{\mathrm{left}}, b_{\mathrm{right}}) \in \{0,1\}^2$:
$d_0 \leftarrow op_0(b_{\mathrm{left}}, b_{\mathrm{right}}), d_1 \leftarrow op_1(b_{\mathrm{left}}, b_{\mathrm{right}})$
  $c^0_{\mathrm{in}} \leftarrow \mathsf{ENC}_L(b_{\mathrm{left}}, K_{\mathrm{out}}(d_0), K_{\mathrm{out}}(d_0))$
  $c^1_{\mathrm{in}} \leftarrow \mathsf{ENC}_L(b_{\mathrm{left}}, K_{\mathrm{out}}(d_0), K_{\mathrm{out}}(d_1))$
  $c_{\mathrm{out}} \leftarrow \mathsf{ENC}_R(b_{\mathrm{right}}, c^0_{\mathrm{in}}, c^1_{\mathrm{in}})$
return $\tilde{g}$

$\mathsf{2Eval}(\tilde{C}, \tilde{x}, d)$

Same as $\mathsf{GEval}_{\mathsf{yao}}$
See Fig. 12.

Fig. 19: 2Garble subroutine

$\mathtt{PebHyb}_{\mathcal{A}}^{b_L,b_R,\mathsf{config},\ell}(1^\lambda)$

$(C_0, C_1, \mathrm{st}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
$n \leftarrow \mathsf{InputLength}(C_0)$
$K \leftarrow\!\$\ \mathsf{Ksample}(1^n)$
$C' \leftarrow \mathsf{Sel} \circ C_{b_L} || C_{b_R}$
for every g s.t. $\mathsf{config}(g) = G$:
  $\mathsf{bit}(g) \leftarrow\!\$\ \{0,1\}$
$r \leftarrow \ell - |\mathrm{Dom}(\mathsf{bit})|$
  $s \leftarrow\!\$\ \{0,1\}^r$

$(\tilde{C}, d) \leftarrow\!\$\ \mathsf{2Garble}_{\mathsf{config},\mathsf{bit}}(C_{b_L}, C_{b_R})$
$(x_0, x_1, \mathrm{st}') \leftarrow\!\$\ \mathcal{A}(\tilde{C}, d, \mathrm{st})$
$x \leftarrow x_{b_L} || x_{b_R}$
for every g s.t. $\mathsf{config}(g) = G$:
  if value of g in $C'(x) \neq \mathsf{bit}(g)$
   then return 0
  if $s \neq 0^r$ **then**
    return 0
$\tilde{x} \leftarrow \mathsf{2Input}(K, x_{b_L} || x_{b_R})$
$b' \leftarrow\!\$\ \mathcal{A}(\tilde{x}, \mathrm{st'})$
return b'

$\mathcal{R}_{\mathsf{config}_{\mathrm{new}},\ell}^{\mathsf{config}_{\mathrm{old}},\mathcal{A}}(1^\lambda)$

$(C_0, C_1, \mathrm{st}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
$n \leftarrow \mathsf{InputLength}(C_0)$
$K \leftarrow\!\$\ \mathsf{Ksample}(1^n)$
$C' \leftarrow \mathsf{Sel} \circ C_{b_L} || C_{b_R}$
for every g s.t. $\mathsf{config}_{\mathrm{old}}(g) = G$:
  $\mathsf{bit}(g) \leftarrow\!\$\ \{0,1\}$
$r \leftarrow \ell - |\mathrm{Dom}(\mathsf{bit})|$
  $s \leftarrow\!\$\ \{0,1\}^r$
for every g :
  if $\mathsf{config}_{\mathrm{old}}(g) = \mathsf{config}_{\mathrm{new}}(g)$ **then** :
    $\mathsf{config}(g) \leftarrow \mathsf{config}_{\mathrm{old}}(g)$
  else $op_0 \leftarrow \mathsf{op}(g, \mathsf{config}_{\mathrm{old}}(g))$
    $op_1 \leftarrow \mathsf{op}(g, \mathsf{config}_{\mathrm{new}}(g))$
$(\tilde{C}, d) \leftarrow\!\$\ \mathsf{2Garble}_{\mathsf{config},\mathsf{bit}}(C_{b_L}, C_{b_R})$
$(x_0, x_1, \mathrm{st}') \leftarrow\!\$\ \mathcal{A}(\tilde{C}, d, \mathrm{st})$
$x \leftarrow x_{b_L} || x_{b_R}$
for every g s.t. $\mathsf{config}_{\mathrm{old}}(g) = G$:
  if value of g in $C'(x) \neq \mathsf{bit}(g)$
   then return 0
  if $s \neq 0^r$ **then**
    return 0
$\tilde{x} \leftarrow \mathsf{2Input}(K, x_{b_L} || x_{b_R})$
$b' \leftarrow\!\$\ \mathcal{A}(\tilde{x}, \mathrm{st'})$
return b'

Fig. 20: Hybrid pebbling games and reduction

Lemma 7 (Guessing). *Assuming the IND-CPA security of the underlying encryption scheme, we have that for any PPT adversary $\mathcal{A}$ which queries circuits C_0, C_1 with equal input length and inputs x_0 and x_1 such that $C_0(x_0) = C_1(x_1)$, and for all $\ell \in \mathbb{N}$,*

$$\begin{aligned}&\left|\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}_{left},\ell}\right] - \Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}_{right},\ell}\right]\right| \\ &\quad = 2^{-\ell}\left|\Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,0}\right] - \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,1}\right]\right| \end{aligned} \tag{9}$$

Lemma 8 (G Pebbling move). *Let* config^{old} *and* config^{new} *be two pebbling configurations which are one G pebbling move away from each other, and assume w.l.o.g., that a grey pebble is* removed *in this step. Then, for all PPT adversaries $\mathcal{A}$,*

$$\begin{aligned}&\left|\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{old},\ell}\right] - \Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{new},\ell}\right]\right| \\ \leq &\left|\Pr\left[1 = \mathcal{R}_{\mathsf{config}_{new},\ell}^{\mathsf{config}_{old},\mathcal{A}}, \mathtt{IND\text{-}CPA}_{L,R}^{0}\right] - \Pr\left[1 = \mathcal{R}_{\mathsf{config}_{new},\ell}^{\mathsf{config}_{old},\mathcal{A}}, \mathtt{IND\text{-}CPA}_{L,R}^{1}\right]\right|\end{aligned}$$

where Fig. 20 defines $\mathcal{R}_{\mathsf{config}_{new},\ell}^{\mathsf{config}_{old},\mathcal{A}}$.

Lemma 9 (Bounding a L-R Pebbling move). *Let* config^{old} *and* config^{new} *be two pebbling configurations which are one L-R pebbling move away from each other, and assume w.l.o.g., that a left pebble is replaced by a right pebble in this step. Then, for all PPT adversaries $\mathcal{A}$,*

$$\begin{aligned}&\left|\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{old},\ell}\right] - \Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{new},\ell}\right]\right| \\ \leq &\left|\Pr\left[1 = \mathcal{R}_{\mathsf{config}_{new},\ell}^{\mathsf{config}_{old},\mathcal{A}}, \mathtt{IND\text{-}CPA}_{L,R}^{0}\right] - \Pr\left[1 = \mathcal{R}_{\mathsf{config}_{new},\ell}^{\mathsf{config}_{old},\mathcal{A}}, \mathtt{IND\text{-}CPA}_{L,R}^{1}\right]\right|\end{aligned}$$

where Fig. 20 defines $\mathcal{R}_{\mathsf{config}_{new},\ell}^{\mathsf{config}_{old},\mathcal{A}}$.

Lemma 10 (Pebbling Complexity). *Let C be a selector circuit of depth d, with selector gates at the very bottom (on the outputs) and m outputs. The algorithm* $\mathsf{PebbleSelectors}(C)$ *is a (G,LR) pebbling strategy and requires $2d-1$ grey pebbles and $m \cdot t$ steps to move from the left to the right configuration. Here, $t = m \cdot 4^d$*

To see that Equation 6 follows, observe that Lemma 10 sais that there is a pebbling strategy which needs at most $2d-1$ grey pebbles, thus, we can choose $\ell = 2d-1$. Since the depth d is logarithmic in the security parameter, the factor $2^{-\ell}$ which we lose in Lemma 7 is polynomial in the security parameter.

Moreover, Lemma 8 and Lemma 9 ensure that the distinguishing advantage between two adjacent hybrids is negligible, and Lemma 10 ensures that we need only $m \cdot 4^{d-1}$ hybrid games. $m \cdot 4^{d-1}$ is also a polynomial in the security parameter, since d is logarithmic in the security parameter and since the number of inputs m is determined by a circuit written by a PPT adversary. Thus, indeed, if the underlying symmetric encryption scheme is IND-CPA secure, Equation 6 holds. □

D.3 Proof of Lemma 7 (Guessing)

Due to Equation 8, we have that for all $\ell \in \mathbb{N}$,

$$\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}_{\mathrm{left}},\ell} \;\middle|\; s = 0^{|\ell|}\right] = \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,0}\right]$$

and

$$\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}_{\mathrm{left}},\ell} \;\middle|\; s \neq 0^{|\ell|}\right] = 1.$$

Moreover, $\Pr\left[s = 0^{\ell}\right] = 2^{-\ell}$ and thus,

$$\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}_{\mathrm{left}},\ell}\right] = 1 \cdot (1 - 2^{-\ell}) + 2^{-\ell} \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,0}\right] \tag{10}$$

Analogously,

$$\Pr\left[1 = \mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}_{\mathrm{right}},\ell}\right] = 1 \cdot (1 - 2^{-\ell}) + 2^{-\ell} \Pr\left[1 = \mathtt{Hyb}_{\mathcal{A}}^{0,1,1}\right] \tag{11}$$

Putting equation 10 and 11 together directly yields Equation 9. □

D.4 Proof of Lemma 8 (G move)

The reduction $\mathcal{R}_{\mathsf{config}_{\mathrm{new}},\ell}^{\mathsf{config}_{\mathrm{old}},\mathcal{A}}$ (cf. Fig. 20) proceeds exactly as $\mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{\mathrm{old}},\ell}$ except for the gate g where $\mathsf{config}^{\mathrm{old}}$ and $\mathsf{config}^{\mathrm{new}}$ differ. Here, the operation induced by $\mathsf{config}^{\mathrm{old}}$ always encrypts the active key (denoted op_0), and the operation induced by $\mathsf{config}^{\mathrm{new}}$ (denoted op_1) encrypts whichever key should be encrypted according to the circuit $C' = \mathsf{Sel} \circ C_0 || C_1$. Here, the reduction uses its left-or-right oracles for both parents of g.

In order to simulate the rest of the garbling properly, the reduction also queries the encryption oracles (twice with the same message) whenever the keys of the parents of g are used for encrypting a key. Since the parents of g are grey-pebbled, we know that they only encrypt the active key. The active key is accessible to the reduction because it can retrieve it via the GETA query from the IND-CPA game and does so in the very beginning.

To see the soundness of the reduction, we can inline the code of the IND-CPA game into the reduction and observe that indeed, the emulation is perfect.

To see this from the code of the reduction, first observe that the code marked in pink (cf. Fig. 20) is not used by $\mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{\mathrm{old/new}},\ell}$. The reduction computes a new configuration config based on $\mathsf{config}^{\mathrm{old}}$ and $\mathsf{config}^{\mathrm{new}}$ which marks g as a special reduction gate. For the encryptions of all gates which are not g, the reduction perfectly emulates $\mathtt{PebHyb}_{\mathcal{A}}^{0,1,\mathsf{config}^{\mathrm{old/new}},\ell}$. For g itself, if the $\mathtt{IND\text{-}CPA}_{L/R}^{b}$ game has bit 0, the encryption always encrypts according to op_0, i.e., $\mathsf{config}^{\mathrm{old}}$. If the bit is 1, then the encryption always encrypts according to op_1, except for

the active keys. For active keys, however, op_0 and op_1 are identical, they both encrypt the active key. Thus, we have that

$$\mathcal{R}^{\mathsf{config}_{\mathrm{old}},\mathcal{A}}_{\mathsf{config}_{\mathrm{new}},\ell} \to \texttt{IND-CPA}^0_{L/R} = \mathsf{PebHyb}^{0,1,\mathsf{config}^{\mathrm{old}},\ell}_{\mathcal{A}}$$
$$\mathcal{R}^{\mathsf{config}_{\mathrm{old}},\mathcal{A}}_{\mathsf{config}_{\mathrm{new}},\ell} \to \texttt{IND-CPA}^1_{L/R} = \mathsf{PebHyb}^{0,1,\mathsf{config}^{\mathrm{new}},\ell}_{\mathcal{A}}$$

and Lemma 8 follows. □

D.5 Proof of Lemma 9 (L-R move)

The reduction and essentially all arguments in this proof are analogous to the proof of Lemma 8 (G move). The core difference is the argument for why, in this game hop, changing from a left to a right pebble on a selector gate does not change which output key is encrypted under the two active input keys when garbling the selector gate.

Since $C_0(x_0) = C_1(x_1)$, we have two cases. In case 0, both the 0-keys of the parent nodes of the selector gate are active and in this case $\mathsf{Sel}_0(0,0) = \mathsf{Sel}_1(0,0) = 0$ as required. In case 1, both the 1-keys of the parent nodes of the selector gate are active and in this case, we also have $\mathsf{Sel}_0(1,1) = \mathsf{Sel}_1(1,1) = 1$ as needed. □