Adaptive Security of Yao's Garbled Circuits

Zahra Jafargholi * Daniel Wichs*

Abstract

A garbling scheme is used to garble a circuit C and an input x in a way that reveals the output $C(x)$ but hides everything else. Yao's construction from the 80's is known to achieve *selective security*, where the adversary chooses the circuit C and the input x in one shot. It has remained as an open problem whether the construction also achieves adaptive security, where the adversary can choose the input x after seeing the garbled version of the circuit C.

A recent work of Hemenway et al. (CRYPTO '16) modifies Yao's construction and shows that the resulting scheme is adaptively secure. This is done by encrypting the garbled circuit from Yao's construction with a special type of "somewhere equivocal encryption" and giving the key together with the garbled input. The efficiency of the scheme and the security loss of the reduction is captured by a certain pebbling game over the circuit.

In this work we prove that Yao's construction itself is already adaptively secure, where the security loss can be captured by the same pebbling game. For example, we show that for circuits of depth d, the security loss of our reduction is $2^{O(d)}$, meaning that Yao's construction is adaptively secure for NC1 circuits without requiring complexity leveraging.

Our technique is inspired by the "nested hybrids" of Fuchsbauer et al. (Asiacrypt '14, CRYPTO '15) and relies on a careful sequence of hybrids where each hybrid involves some limited guessing about the adversary's adaptive choices. Although it doesn't match the parameters achieved by Hemenway et al. in their full generality, the main advantage of our work is to prove the security of Yao's construction as is, without any additional encryption layer.

*Research supported by NSF grants CNS-1347350, CNS-1314722, CNS-1413964. This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

1 Introduction

Garbled circuits, introduced by Yao in (oral presentations of) [Yao82, Yao86], can be used to garble a circuit C and an input x in a way that reveals $C(x)$ but hides everything else. Yao's construction is based on one-way functions and achieves a number of desirable properties with countless applications. One of the features of this construction is that a circuit C can be garbled *off-line* in time proportional to $|C|$ which is presumably large, but an input x can later be garbled very efficiently *on-line* in time only proportional to $|x|$ which is presumably much smaller. We consider the *on-line complexity* (i.e., time to garble the input x) as the main measure of efficiency.

Selective vs. Adaptive Security. Unfortunately, Yao's construction is only known to satisfy selective security where the adversary must choose the circuit C and the input x to be garbled in one shot. It has remained an open problem whether Yao's construction also achieves the stronger notion of adaptive security where the adversary can choose the input x after seeing the garbled circuit. Adaptive security is especially important in the off-line/on-line setting where the adversary often sees the garbled circuit first and may be able to influence the choice of the input x.

Prior Work on Adaptive Security. The work of Bellare, Hoang and Rogaway [BHR12a] raised the question of whether Yao's construction or indeed any construction of garbled circuits achieves adaptive security. They showed a simple adaptively secure construction where the on-line complexity is proportional to the *circuit size*, but left it as an open problem to do better.

The work of Applebaum et al. [AIKW13] shows that the on-line complexity in the adaptive setting must at least exceed the *output size* of the circuit. This is in contrast to the selective setting, where Yao's garbling scheme achieves on-line complexity that depends only on the input size and not the output size. However, there is a small variant of Yao's scheme (by giving the mapping of output labels to output bits with the garbled input) which is natural in the adaptive setting and which raises the on-line complexity to also depend on the output size. We refer to this variant as Yao's construction when we consider the adaptive setting and it has remained as an open problem if this variant is adaptively secure.

Another approach to proving adaptive security of Yao's construction is to use *complexity leveraging* where we guess the adversary's choice of x a-priori. A direct approach results in a security loss of 2^n where n is the input size to the circuit. In particular, if we insist on polynomial security loss then this approach can only handle circuits with a logarithmic input size.

We mention that there are also other approaches that depart from Yao's construction and/or rely on significantly heavier assumptions than one-way functions. For example [BHR12a] show how to get an optimal solution (in fact one that bypasses the lower-bound of [AIKW13]) in the random oracle model. The work of [BHK13] shows that this solution also works in the standard-model based on non-standard hash-function assumption referred to as UCE. Boneh et al. [BGG+14] implicitly provides an adaptive garbling scheme with low on-line complexity that scales with the depth of the circuit under LWE, while the work of Ananth and Sahai [AS15] shows how to get an essentially optimal schemes assuming indistinguishability obfuscation.

Work of Hemenway et al. (CRYPTO '16). The most relevant prior work is a recent result of Hemenway et al. [HJO+15]. This work modifies Yao's construction by encrypting the garbled circuit with a special type of "somewhere equivocal encryption" and giving the key together with the garbled input. The encryption scheme has an "equivocation parameter" which determines its key size and therefore affects the on-line complexity of the garbling. They show that the resulting scheme is adaptively secure where the equivocation parameter needed and the security loss of the reduction are captured by a certain pebbling game over the circuit. In particular, if a circuit with input size n and output size m can be pebbled with t pebbles in γ steps then the resulting scheme can be instantiated so as to achieve on-line complexity $O(n + m + t)$ and security loss γ. Furthermore they show that any circuit of size q, width w, and depth d can be pebbled with $t = O(w)$ pebbles in $\gamma = O(q)$ steps or alternatively with $t = O(d)$ pebbles in $\gamma = q \cdot 2^{O(d)}$ steps. In particular, this means that (without complexity leveraging):

- For any circuit of width w, the on-line complexity can be made $O(w)$.
- For NC1 circuits, the on-line complexity can be just $O(n + m)$.

Our Results. In this work we revisit the question of whether Yao's construction itself (without modification) is adaptively secure. We give a new reduction which connects the security of Yao's construction with the same pebbling game as studied by Hemenway et al. [HJO+15]. In particular, we show that for circuits that can be pebbled with t pebbles in γ steps, Yao's construction is adaptively secure with a security loss of $\gamma 2^{O(t)}$. For example, since circuits of size q and depth d can be pebbled in $\gamma = q2^{O(d)}$ steps with $t = O(d)$ pebbles we get a security loss of $q2^{O(d)}$. This means that Yao's construction is already adaptively secure for NC1 circuits, without the use of complexity leveraging.[1]

Next we describe our techniques and compare to those of [HJO+15]. On a very high level, the work of [HJO+15] proves security via a sequence of hybrids, where in each hybrid some small number of garbled gates of the Yao garbled circuit are "equivocal" and only needed to be specified by the reduction in the on-line phase after the input x is known. In this work we replace the role of "equivocation" with the careful use of "guessing". Instead of simply guessing the entire input x, our reduction consists of a sequence of hybrids where in each hybrid we guess some small number of the wire values in the circuit and abort if the guess is incorrect. We then show how to patch together hybrids that contain different guessed wires (and even a different number of guessed wires) to get a security proof. This approach is reminiscent of the "nested hybrids" technique employed by Fuchsbauer et al. [FKPR14, FJP15] and we believe our abstraction of this technique via pebbling will be useful in other contexts.

1.1 Our Techniques

1.1.1 Yao's Scheme and The Challenge of Adaptive Security ([HJO+15])

To describe our technical contribution, we must first describe Yao's construction and the difficulty one faces when trying to prove adaptive security. The following discussion is taken essentially verbatim from [HJO+15], following the ideas of Lindell and Pinkas [LP09] who gave the first detailed proof of security for Yao's garbled circuits in the selective security setting.

Yao's Scheme. For each wire w in the circuit, we pick two keys k_w^0, k_w^1 for a symmetric-key encryption scheme. For each gate in the circuit computing a function $g : \{0,1\}^2 \to \{0,1\}$ and having input wires a, b and output wire c we create a *garbled gate* consisting of 4 randomly ordered ciphertexts created as:

$$\begin{aligned} c_{0,0} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^{g(0,0)})) \qquad & c_{1,0} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^{g(1,0)})), \\ c_{0,1} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^{g(0,1)})) \qquad & c_{1,1} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^1}(k_c^{g(1,1)})) \end{aligned} \tag{1}$$

where $(\mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme. The garbled circuit $\widetilde{C}$ consists of all of the gabled gates, along with an *output mapping* $\{k_w^0 \to 0, k_w^1 \to 1\}$ which gives the correspondence between the keys and the bits they represent for each output wire w . To garble an n-bit value $x = x_1 x_2 \cdots x_n$, the garbled input $\widetilde{x}$ consists of the keys $k_{w_i}^{x_i}$ for the n input wires w_i.

To evaluate the garbled circuit on the garbled input, it's possible to decrypt (exactly) one ciphertext in each garbled gate and get the key $k_w^{v(w)}$ corresponding to the bit $v(w)$ going over the wire w during the computation $C(x)$. Once the keys for the output wires are computed, it's possible to recover the actual output bits by looking them up in the output mapping.

Selective Security Simulator. To prove the selective security of Yao's scheme, we need to define a simulator that gets the output $y = y_1 y_2 \cdots y_m = C(x)$ and must produce $\widetilde{C}$, $\widetilde{x}$. The simulator picks random keys k_w^0, k_w^1 for each wire w just like the real scheme, but it creates the garbled gates as follows:

$$\begin{aligned} c_{0,0} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^0)) \qquad & c_{1,0} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^0)), \\ c_{0,1} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^0)) \qquad & c_{1,1} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^1}(k_c^0)) \end{aligned} \tag{2}$$

where all four ciphertext encrypt the same key k_c^0. It creates the output mapping $\{k_w^0 \to y_w, k_w^1 \to 1 - y_w\}$ by "programming it" so that the key k_w^0 corresponds to the correct output bit y_w for each output wire w.

[1] Unfortunately, we cannot get a meaningful analogue of the width based result of [HJO+15] since the security loss would be 2^w which exceeds the trivial security loss of 2^n obtained by simply guessing the input.

This defines the simulated garbled circuit $\widetilde{C}$. To create the simulated garbled input $\widetilde{x}$ the simulator simply gives out the keys k_w^0 for each input wire w. Note that, when evaluating the simulated garbled circuit on the simulated garbled input, the adversary only sees the keys k_w^0 for every wire w.

Selective Security Hybrids. To prove indistinguishability between the real world and the simulation, there is a series of carefully defined hybrid games that switch the distribution of one garbled gate at a time. Unfortunately, we cannot directly switch a gate from the real distribution (1) to the simulated one (2) and therefore must introduce an intermediate distribution (3) as below:

$$\begin{aligned} c_{0,0} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^{v(c)})) \qquad & c_{1,0} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^{v(c)})), \\ c_{0,1} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^{v(c)})) & c_{1,1} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^1}(k_c^{v(c)})) \end{aligned} \tag{3}$$

where $v(c)$ is the correct *value* of the bit going over the wire c during the computation of $C(x)$.

Let us give names to the three modes for creating garbled gates that we defined above: (1) is called RealGate mode, (2) is called SimGate mode, and (3) is called InputDepSimGate mode, since the way that it is defined depends adaptively on the choice of the input x.

We can switch a gate from RealGate to InputDepSimGate mode if the *predecessor* gates are in InputDepSimGate mode (or we are in the input level). This follows by CPA security of encryption. In particular, we are *not* changing the value contained in ciphertext $c_{v(a),v(b)}$ encrypted under the keys $k_a^{v(a)}, k_b^{v(b)}$ that the adversary obtains during evaluation, but we *can* change the values contained in all of the other ciphertexts since the keys $k^{1-v(a)}, k^{1-v(b)}$ do not appear anywhere inside the predecessor garbled gates as long as they are already in InputDepSimGate mode.

We can also switch a gate from InputDepSimGate to SimGate mode if the *successor* gates are in InputDepSimGate or SimGate mode (or we are at the output level). This is actually an information theoretic step; since the keys k_c^0, k_c^1 are used completely symmetrically in the successor gates there is no difference between always encrypting $k_c^{v(c)}$ as in InputDepSimGate mode or encrypting k_c^0 as in SimGate. This allows us to first switch every gate from RealGate to InputDepSimGate mode and then from InputDepSimGate to SimGate, proving the selective security of Yao's construction.

Challenges in Achieving adaptive security. There are two issues in using the above strategy in the adaptive setting: an immediate but easy to fix problem and a more subtle but difficult to overcome problem.

The first immediate issue is that the selective simulator needs to know the output $y = C(x)$ to create the garbled circuit $\widetilde{C}$ and in particular to program the output mapping $\{k_w^0 \to y_w, k_w^1 \to 1 - y_w\}$ for the output wires w. However, the adaptive simulator does not get the output y until *after* it creates the garbled circuit $\widetilde{C}$. Therefore, we cannot (even syntactically) use the selective security simulator in the adaptive setting. This issue turns out to be easy to fix by modifying the construction to send the output-mapping as part of the garbled input $\widetilde{x}$ in the on-line phase, rather than as part of the garbled circuit $\widetilde{C}$ in the off-line phase. This modification raises on-line complexity to also being linear in the output size of the circuit, which we know to be necessary by the lower bound of [AIKW13]. We refer to this modification as Yao's garbled circuit construction in the adaptive setting. With this modification, the adaptive simulator can program the output mapping after it learns the output $y = C(x)$ in the on-line phase and therefore we get a syntactically meaningful simulation strategy in the adaptive setting.

The second problem is where the true difficulty lies. Although we have a syntactically meaningful simulation strategy, the previous proof of indistinguishability of the real world and the simulation completely breaks down in the adaptive setting. In particular InputDepSimGate mode as specified in equation (3) is syntactically undefined in the adaptive setting. Recall that in this mode the garbled gate is created in a way that depends on the input x, but in the adaptive setting the input x is chosen adaptively after the garbled circuit is created! Therefore, *although we have a syntactically meaningful simulation strategy for the adaptive setting, we do not have any syntactically meaningful sequence of intermediate hybrids to prove indistinguishability between the real world and the simulated world.*

1.1.2 Our Solution

As described above, in the selective setting there is a proof of security via a sequence of hybrids that changes the distribution of gates from RealGate mode to InputDepSimGate mode to SimGate mode. Unfortunately, InputDepSimGate mode does not make sense (even syntactically) in the adaptive setting since it relies on knowing the value on the outgoing wire of that gate, which isn't defined until the input x is given.

To overcome this problem, the work of [HJO+15] encrypted the entire garbled circuit with a somewhere equivocal encryption scheme which allowed the simulator to put dummy values in place of all of the gates in InputDepSimGate mode and only later after the input x was known replace the dummy values with correctly distributed garbled gates by equivocating the encryption.

Our Idea: Guess and Hope for the Best. Our idea to overcome this problem is very different. We define hybrid games in the adaptive setting where we guess the value $v(c)$ on the outgoing wire c of every gate in InputDepSimGate mode a-priori and abort if the adversary's adaptive choice of the input x doesn't match our guesses. Note that although the goal is to have the garbled gates in InputDepSimGate mode depend on the input x, we choose them independently of x and only abort later if we chose incorrectly. This defines syntactically meaningful hybrid games, but unfortunately the set of guessed wires and even the number of guessed wires is different in each hybrid making it impossible to compare them directly. However, we show that by carefully adding and removing guesses in different parts of the proof and then only comparing hybrids with an equivalent set of guesses, we can patch together this sequence of a-priori incomparable hybrids and give an indistinguishability reduction. Overall, we can take any valid sequence of γ hybrid games that would give an indistinguishability proof in the selective setting and translate it into a proof of security in the adaptive setting with a security loss of $\gamma 2^{O(t)}$ where t is the maximum number of gates in InputDepSimGate mode in any hybrid. This idea of "carefully" guessing different components in different hybrids is reminiscent of the *nested hybrids* technique of Fuchsbauer et al. [FKPR14, FJP15].

In comparison to [HJO+15], we rely on "guessing" instead of "equivocating". Whereas [HJO+15] had to modify Yao's scheme and pay for gates in InputDepSimGate mode by increasing the "equivocation parameter" which resulted in larger key size for the somewhere equivocal encryption, we get to keep the scheme unmodified but pay for gates in InputDepSimGate mode in the security loss of our reduction.

Sequences of Hybrids and Pebbling. With the above framework, the goal of proving adaptive security reduces to the goal of giving a sequence of hybrids in the selective setting where the number of gates in InputDepSimGate mode in any hybrid is as small as possible. This is the same challenge as faced in the work of [HJO+15] and we can rely on the same idea.

Recall that we need to start with the real world where all gates are in RealGate mode and end with the simulated world where all gates are in SimGate mode. As discussed in the overview of the selective security proof of Yao's garbled circuits, we are allowed to change a gate from RealGate to InputDepSimGate if all of its predecessors are in InputDepSimGate (or it's an input gate) and we are allowed to change InputDepSimGate to SimGate if all of the successors are in InputDepSimGate or SimGate modes (or it's an output gate). A naive sequence of hybrids, corresponding to the proof of selective security of Lindell and Pinkas [LP09], would first change all the gates from RealGate mode to InputDepSimGate mode one level at a time starting from the input level, and then change them all to SimGate mode by again changing one level at a time starting from the input level. However, this requires that there is a hybrid step where all of the gates are in InputDepSimGate mode, while our goal is to minimize the number of such gates. It turns out that one can do much better.

The work of [HJO+15] abstracts the above problem as a pebbling game. We associate RealGate mode with not having a pebble, InputDepSimGate mode with having a *black pebble* and SimGate mode with having a *gray pebble*. The rules of the game go as follows:

- We can place or remove a black pebble on a gate as long as both predecessors of that gate have black pebbles on them (or the gate is an input gate).
- We can replace a black pebble with a gray pebble on a gate as long as all successors of that gate have black or gray pebbles on them (or the gate is an output gate).

The goal of the game is to end up with a gray pebble on every gate while using as few black pebbles as possible at any point in time. It was shown that any circuit of size q, width w and depth d can be pebbled

in two different ways: either with $t = O(w)$ black pebbles in $\gamma = O(q)$ steps or with $t = O(d)$ black pebbles in $\gamma = q \cdot 2^{O(d)}$ steps.

Our Parameters. Using the second pebbling strategy based on depth, we get a security proof of Yao's garbled circuits in the adaptive setting with a security loss of $q2^{O(d)}$ where q is the circuit size and d is the circuit depth. In particular, for NC1 circuits we get a security reduction showing the adaptive security of Yao's garbled circuits without complexity leveraging.

2 Preliminaries

General Notation. For a positive integer n, we define $[n] := \{1, \ldots, n\}$. We use the notation $x \leftarrow X$ for the process of sampling a value x according to the distribution X. We use U_n for uniform distribution over n-bit strings. A function $\mu(\cdot)$ is negligible in x if $\mu(x) \leq 1/p(x)$ for any polynomial function p and all sufficiently large x. We use $\mathsf{poly}(x)$ to denote the set of all polynomial functions $p(x)$. For an interactive game $\textsc{Game}$ with an adversary $\mathcal{A}$, we use $\textsc{Game}_{\mathcal{A}}$ to denote the outcome of the game played with $\mathcal{A}$.

Definition 1. *Two distributions X and Y are (T, ε)-indistinguishable, denote $\mathbf{D}_T[X, Y] = \varepsilon$ if for any probabilistic algorithm $\mathcal{A}$, running in time T,*

$$|\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]| \leq \varepsilon.$$

For two games $\textsc{Game}$ and $\textsc{Game}'$ we say they are $(T(\lambda), \varepsilon(\lambda))$-indistinguishable, $\mathbf{D}_{T(\lambda)}[\textsc{Game}, \textsc{Game}'] = \varepsilon(\lambda)$, if for any adversary $\mathcal{A}$ running in time $T(\lambda)$,

$$\left|\Pr[\textsc{Game}_{\mathcal{A}} = 1] - \Pr\left[\textsc{Game}'_{\mathcal{A}} = 1\right]\right| \leq \varepsilon(\lambda).$$

Let games $\textsc{Game}(\lambda)$ and $\textsc{Game}'(\lambda)$ be games parameterized by the security parameter λ. If for any polynomial function $T(\lambda)$, there exists a negligible function $\varepsilon(\lambda)$, such that for all λ, $\mathbf{D}_{T(\lambda)}\left[\textsc{Game}(\lambda), \textsc{Game}'(\lambda)\right] \leq \varepsilon(\lambda)$, we say the two games are computationally indistinguishable and denote this by $\textsc{Game}(\lambda) \overset{\text{comp}}{\approx} \textsc{Game}'(\lambda)$.

Circuit Notation. A boolean circuit C consists of gates $\mathsf{gate}_1, \ldots, \mathsf{gate}_q$ and wires $w_1, w_2, \ldots, w_p$. A gate is defined by the tuple $\mathsf{gate}_i = (g, w_a, w_b, w_c)$, where $g : \{0,1\}^2 \to \{0,1\}$ is the function computed by the gate, w_a, w_b are the incoming wires, and w_c is the outgoing wire. Although each gate has a unique outgoing wire w_c, this wire can be used as an incoming wire to several different gates and therefore this models a circuit with fan-in 2 and unbounded fan-out. We also allow $w_a = w_b$, for gates with fan-in 1. We denote the number of gates with q, input wires with m and output wires with m. The total number of wires is $p = n + q$ (since each wire can either be input wire or an outgoing wire of some gate). For convenience, we denote the n input wires by $\mathsf{in}_1, \ldots, \mathsf{in}_n$ and the m output wires by $\mathsf{out}_1, \ldots, \mathsf{out}_m$. We also use reserve a, b and c as labels for input wires to a gate and output wire of the same gate (instead of w_a, w_b, and w_c). For $x \in \{0,1\}^n$ we write $C(x)$ to denote the output of evaluating the circuit C on input x.

We say C is leveled, if each gate has an associated level and any gate at level l has incoming wires only from gates at level $l-1$ and outgoing wires only to gates at level $l+1$. We let the *depth* d denote the number of levels and the *width* w denote the maximum number of gates in any level.

A circuit C is fully specified by a list of gate tuples $\mathsf{gate}_i = (g, a, b, c)$. We use $\Phi_{\mathsf{topo}}(C)$ to refer to the topology of a circuit– which indicates how gates are connected, without specifying the function implement by each gate. In other words, $\Phi_{\mathsf{topo}}(C)$ is the list of *sanitized gate tuples* $\widehat{\mathsf{gate}}_i = (\perp, a, b, c)$ where the function g that the gate implements is removed from the tuple.

3 Garbling Scheme and Adaptive Security ([HJO+15])

The bulk of this section defining what garbled circuits are and presenting Yao's construction is taken verbatim from [HJO+15].

3.1 Garbling Scheme

We now give a formal definition of a garbling scheme. There are many variants of such definitions in the literature, we use the definition given in [HJO+15] and refer the reader to [BHR12b] for a comprehensive treatment.

Definition 2. *A Garbling Scheme is a tuple of PPT algorithms* $\mathsf{GC} = (\mathsf{GCircuit}, \mathsf{GInput}, \mathsf{Eval})$ *such that:*

- $(\widetilde{C}, k) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, C)$*: takes as input a security parameter* λ*, a circuit* $C : \{0,1\}^n \to \{0,1\}^m$*, and outputs the* garbled circuit $\widetilde{C}$*, and* key k*.*
- $\tilde{x} \leftarrow \mathsf{GInput}(k, x)$*: takes as input,s* $x \in \{0,1\}^n$*, and key* k *and outputs* $\tilde{x}$*.*
- $y = \mathsf{Eval}(\widetilde{C}, \tilde{x})$*: given a garbled circuit* $\widetilde{C}$ *and a garbled input* $\tilde{x}$ *output* $y \in \{0,1\}^m$*.*

Correctness *There is a negligible function* ε *such that for any* $\lambda \in \mathbb{N}$*, any circuit* C *and input* x *it holds that* $\Pr[C(x) = \mathsf{Eval}(\widetilde{C}, \tilde{x})] = 1 - \varepsilon(\lambda)$*, where* $(\widetilde{C}, k) \leftarrow \mathsf{GCircuit}(1^\lambda, C)$*,* $\tilde{x} \leftarrow \mathsf{GInput}(k, x)$*.*

Adaptive Security.

- GC *is* $(T(\lambda), \varepsilon(\lambda))$*-adaptively secure garbling scheme, if there exists a probabilistic polynomial time simulator* $\mathsf{Sim} = (\mathsf{SimC}, \mathsf{SimIn})$ *such that, for any probabilistic adversary* $\mathcal{A}$*, running in time* $T(\lambda)$*,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathsf{GC},\mathsf{Sim}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathsf{GC},\mathsf{Sim}}(1^\lambda, 1) = 1]\right| \leq \varepsilon(\lambda).$$

  In other words, $\mathbf{D}_{T(\lambda)}\left[\mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 0), \mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 1)\right] = \varepsilon(\lambda)$*.*

- GC *is adaptively secure if* $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 0) \overset{\text{comp}}{\approx} \mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 1)$

where the experiment $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathsf{GC},\mathsf{Sim}}(1^\lambda, b)$ *is defined as follows:*

1. *The adversary* $\mathcal{A}$ *specifies* C *and gets* $\widetilde{C}$ *where* $\widetilde{C}$ *is created as follows:*
   - *if* $b = 0$*:* $(\widetilde{C}, k) \leftarrow \mathsf{GCircuit}(1^\lambda, C)$*,*
   - *if* $b = 1$*:* $(\widetilde{C}, \mathsf{state}) \leftarrow \mathsf{SimC}(1^\lambda, \Phi_{\mathsf{topo}}(C))$*, where* $\Phi_{\mathsf{topo}}(C)$ *reveals the topology of* C*.*
2. *The adversary* $\mathcal{A}$ *specifies* x *and gets* $\tilde{x}$ *created as follows:*
   - *if* $b = 0$*,* $\tilde{x} \leftarrow \mathsf{GInput}(k, x)$*,*
   - *if* $b = 1$*,* $\tilde{x} \leftarrow \mathsf{SimIn}(C(x), \mathsf{state})$*.*
3. *Finally, the adversary outputs a bit* b'*, which is the output of the experiment.*

On-line Complexity. The time it takes to garble an input x, (i.e., time complexity of $\mathsf{GInput}(\cdot,\cdot)$) is the *on-line complexity* of the scheme. Clearly the on-line complexity of the scheme gives a bound on the size of the garbled input $\widetilde{x}$. Ideally, the on-line complexity should be much smaller than the circuit size $|C|$.

Projective Scheme. We say a garbling scheme is *projective* if each bit of the garbled input $\widetilde{x}$ only depends on one bit of the actual input x. In other words, each bit of the input, is garbled independently of other bits of the input. Projective schemes are essential for two-party computation where the garbled input is transmitted using an oblivious transfer (OT) protocol. Our constructions will be projective.

Hiding Topology. A garbling scheme that satisfies the above security definition may reveal the topology of the circuit C. However, there is a way to transform any such garbling scheme into one that hides everything, including the topology of the circuit, without a significant asymptotic efficiency loss. More precisely, we rely on the fact that there is a function $\mathsf{HideTopo}(\cdot)$that takes a circuit C as input and outputs a functionally equivalent circuit C', such that for any two circuits C_1, C_2 of equal size, if $C_1' = \mathsf{HideTopo}(C_1)$ and $C_2' = \mathsf{HideTopo}(C_2)$, then $\Phi_{\mathsf{topo}}(C_1') = \Phi_{\mathsf{topo}}(C_2')$. An easy way to construct such function $\mathsf{HideTopo}$ is by setting C' to be a universal circuit, with a hard-coded description of the actual circuit C. Therefore, to get a topology-hiding garbling scheme, we can simply use a topology-revealing scheme but instead of garbling the circuit C directly, we garble the circuit $\mathsf{HideTopo}(C)$.

3.2 Yao's Garbling Scheme

In this section we describe Yao's garbling scheme and in the next section we give the simulation strategy.

Construction. Let C be a leveled boolean circuit with fan-in 2 and unbounded fan-out, with inputs size n, output size m, depth d. Let q denote the number of gates in C. Recall that wires are uniquely identified with labels and a circuit C is specified by a list of gate tuples $\mathsf{gate} = (g, a, b, c)$, where g computes the gate and a, b are the input wire labels and c is the output wire label. The topology of the circuit $\Phi_{\mathsf{topo}}(C)$ consists of the sanitized gate tuples $\widehat{\mathsf{gate}} = (\perp, a, b, c)$. For simplicity, we implicitly assume that $\Phi_{\mathsf{topo}}(C)$ is public and known to the circuit evaluator without explicitly including it as part of the garbled circuit $\widetilde{C}$. To simplify the description of our construction, we first describe the procedure for garbling a single gate, that we denote by $\mathsf{GarbleGate}$.

Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure symmetric-key encryption scheme satisfying the special correctness property defined in Appendix A. $\mathsf{GarbleGate}$ is defined as follows.

- $\widetilde{g} \leftarrow \mathsf{GarbleGate}(g, \{k_a^\sigma, k_b^\sigma, k_c^\sigma\}_{\sigma\in\{0,1\}})$: This function computes 4 ciphertexts $\mathtt{ct}_{\sigma_0,\sigma_1} \ : \ \sigma_0, \sigma_1 \in \{0,1\}$ as defined below and outputs them in a random order as $\widetilde{g} = [\mathtt{ct}_1, \mathtt{ct}_2, \mathtt{ct}_3, \mathtt{ct}_4]$.

$$\mathtt{ct}_{0,0} \leftarrow \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^{g(0,0)})), \quad \mathtt{ct}_{0,1} \leftarrow \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^{g(0,1)}))$$
$$\mathtt{ct}_{1,0} \leftarrow \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^{g(1,0)})), \quad \mathtt{ct}_{1,1} \leftarrow \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^{g(1,1)}))$$

3.3 Adaptive Simulator

The adaptive security simulator for our garbling scheme is essentially the same as the selective security simulator for Yao's scheme (as in [LP09]), with the only difference that the output table is sent in the on-line phase, and is computed adaptively to map to the correct output.

More specifically, the adaptive simulator $(\mathsf{SimC}, \mathsf{SimIn})$ works as follows. In the off-line phase, SimC computes the garbled gates using procedure $\mathsf{GarbleSimGate}$, that generates 4 ciphertexts that encrypt the same output key. More precisely,

- $\mathsf{GarbleSimGate}(\{k_a^\sigma, k_b^\sigma\}_{\sigma\in\{0,1\}}, k_c')$ takes both keys for input wires w_a, w_b and a single key for the output wire w_c, that we denote by k_c'. It then output $\widetilde{g}_c = [\mathtt{ct}_1, \mathtt{ct}_2, \mathtt{ct}_3, \mathtt{ct}_4]$ where the ciphertexts, arranged in random order, are computed as follows.

$$\mathtt{ct}_{0,0} \leftarrow \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c')) \quad \mathtt{ct}_{1,0} \leftarrow \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c'))$$
$$\mathtt{ct}_{0,1} \leftarrow \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c')) \quad \mathtt{ct}_{1,1} \leftarrow \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c'))$$

The simulator invokes $\mathsf{GarbleSimGate}$ on input $k_c' = k_c^0$.

In the on-line phase, SimIn, on input $y = C(x)$ adaptively computes the output tables so that the evaluator obtains the correct output. This is easily achieved by associating each bit of the output, y_j, to the only key encrypted in the output gate g_{out_j}, which is $k^0_{\mathsf{out}_j}$. For the input keys, SimIn just sends keys $k^0_{\mathsf{in}_i}$ for each $i \in [n]$. The detailed definition of $(\mathsf{SimC}, \mathsf{SimIn})$ is provided in Fig. 2.

$\mathsf{GCircuit}(1^\lambda, C)$

- (Wires) $k_i^\sigma \leftarrow \mathsf{KeyGen}(1^\lambda)$ for $i \in [p]$, $\sigma \in \{0,1\}$.
  (Input wires) $K = (k^0_{\mathsf{in}_i}, k^1_{\mathsf{in}_i})_{i\in[n]}$.
- (Gates) For each $\mathsf{gate}_i = (g, a, b, c)$ in C:
  $\widetilde{g}_i \leftarrow \mathsf{GarbleGate}(g, \{k_a^\sigma, k_b^\sigma, k_c^\sigma\}_{\sigma\in\{0,1\}})$.
- (Output tables) For each output $\ell \in [m]$:
  $\widetilde{d}_\ell := [(k^0_{\mathsf{out}_\ell} \to 0), (k^1_{\mathsf{out}_\ell} \to 1)]$.
- (Garbled Circuit) $\widetilde{C} := (\widetilde{g}_1, \ldots, \widetilde{g}_q)$.

Output $\widetilde{C}$, $k = (K, (\widetilde{d}_\ell)_{\ell\in[m]})$.

$\mathsf{GInput}(x, k)$

- (Select input keys) $K^x = (k^{x_1}_{\mathsf{in}_1}, \ldots, k^{x_n}_{\mathsf{in}_n})$.

Output $\tilde{x} = (K^x, (\widetilde{d}_\ell)_{\ell\in[m]})$.

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

- Parse $\widetilde{x} = (K, (\widetilde{d}_\ell)_{\ell\in[m]})$.
- Evaluate Circuit.
  Parse $K = (k_{\mathsf{in}_1}, \ldots, k_{\mathsf{in}_n})$.
  For each level $j = 1, \ldots, d$,
  For each $\widehat{\mathsf{gate}}_i = (\bot, a, b, c)$ at level j:
  Let $\widetilde{g}_i = [\mathtt{ct}_1, \mathtt{ct}_2, \mathtt{ct}_3, \mathtt{ct}_4]$ For $\delta \in [4]$ let $k'_c \leftarrow \mathsf{Dec}_{k_a}(\mathsf{Dec}_{k_b}(\mathtt{ct}_\delta))$ If $k'_c \neq \bot$ then set $k_c := k'_c$.
- Decrypt output.
  For $\ell \in [m]$:
  Parse $\widetilde{d}_\ell = [(k^0_{\mathsf{out}_\ell} \to 0), (k^1_{\mathsf{out}_\ell} \to 1)]$.
  Set $y_\ell = b$ iff $k_{\mathsf{out}_\ell} = k^b_{\mathsf{out}_\ell}$.

Output $y_1, \ldots, y_m$.

Figure 1: Yao's Garbling Scheme.

4 Hybrid Games

Our goal is to show the indistinguishability of the real world and the simulation in the adaptive setting. We do so by first introducing a template that allows us to define various hybrid games and then showing how to patch such games together to get a full security proof.

4.1 Template for Defining Hybrid Games

Garbling Mode / Guessed Wires. A gate's garbling mode indicates the way it is computed and can be one of the following RealGate, SimGate, InputDepSimGate which corresponds to the distributions outlined in Figure 3. A *circuit configuration* is consists of two sets. A set the garbling modes for each gate in the circuit (i.e. mode_i, $i \in [q]$) and as set of *guessed wires* $I \subseteq [p]$. We use the pair ($(\mathsf{mode}_i)_{i\in[q]}, I$) to denote a circuit configuration. A circuit configuration is *valid* if the outgoing wire of every gate in InputDepSimGate mode, is contained in the set of guessed wires I.

The Hybrid Game $\mathsf{Hyb}((\mathsf{mode}_i)_{i\in[q]}, I)$. Every valid circuit configuration defines a hybrid game as specified formally in Figure 4 and described informally below. The hybrid game consists of a *guessing* step and a *garbling* step. The garbling step has two procedures: one for creating the garbled circuit $\widetilde{C}$ and one for creating the garbled input $\widetilde{x}$. The initial guessing step, is necessary in order to create gates in InputDepSimGate mode. For any such gate it is essential to know what is the bit on its output wire, (referred to as $v(c)$ in Figure 3) once the circuit is computed. However the input is not known at the time of circuit garbling. Therefore we guess it! In some hybrid games we also need to guess values on other wires in the circuit. We define a set (called Guess), that stores all these guessed values for the marked wires. Hyb creates the garbled circuit by picking random keys $k^\sigma_{w_i}$ for each wire w_i. For each gate i, $\mathsf{mode}_i \in \{\mathsf{RealGate}, \mathsf{SimGate}, \mathsf{InputDepSimGate}\}$, it creates a garbled gate $\widetilde{g}_i$ according to the corresponding distribution as described in Figure 3, and using $\mathsf{Guess}(c)$ instead of the unknown $v(c)$. Once Hyb has the input, it checks whether all the guesses were made correctly. If not, the game is over with a fixed and dedicated output (say 0). However if they are correct, it follows the rules below to create the garbled input and map the output wires to $\{0, 1\}$.

Simulator

$\underline{\mathsf{SimC}(1^\lambda, \Phi_{\mathsf{topo}}(C))}$

- (Wires) $k^\sigma_{w_i} \leftarrow \mathsf{KeyGen}(1^\lambda)$ for $i \in [p]$, $\sigma \in \{0,1\}$.
- (Garbled gates) For each gate $\widetilde{\mathsf{gate}}_i = (\bot, a, b, c)$ in $\Phi_{\mathsf{topo}}(C)$:
  $\widetilde{g}_i \leftarrow \mathsf{GarbleSimGate}\left(\{k^\sigma_a, k^\sigma_b\}_{\sigma\in\{0,1\}}, k^0_c\right)$.

Output $\widetilde{C}, \mathsf{state} = \left(\{k^\sigma_{w_i}\}\right)$.

$\underline{\mathsf{SimIn}(y, \mathsf{state})}$

- (Output table) $\widetilde{sd}_\ell \leftarrow \left[\left(k^{y_\ell}_{\mathsf{out}_\ell} \to 0\right), \left(k^{1-y_\ell}_{\mathsf{out}_\ell} \to 1\right)\right]_{\ell\in[m]}$. // ensures $k^0_{\mathsf{out}_\ell} \to y_\ell$

Output $\widetilde{x} = ((k^0_{\mathsf{in}_i})_{i\in[n]}, (\widetilde{sd}_\ell)_{\ell\in[m]})$.

Figure 2: Simulator for Adaptive Security.

RealGate	SimGate	InputDepSimGate
$\mathtt{ct}_{0,0} \leftarrow \mathsf{Enc}_{k^0_a}(\mathsf{Enc}_{k^0_b}(k^{g(0,0)}_c))$	$\mathtt{ct}_{0,0} \leftarrow \mathsf{Enc}_{k^0_a}(\mathsf{Enc}_{k^0_b}(k^0_c))$	$\mathtt{ct}_{0,0} \leftarrow \mathsf{Enc}_{k^0_a}(\mathsf{Enc}_{k^0_b}(k^{v(c)}_c))$
$\mathtt{ct}_{0,1} \leftarrow \mathsf{Enc}_{k^0_a}(\mathsf{Enc}_{k^1_b}(k^{g(0,1)}_c))$	$\mathtt{ct}_{0,1} \leftarrow \mathsf{Enc}_{k^0_a}(\mathsf{Enc}_{k^1_b}(k^0_c))$	$\mathtt{ct}_{0,1} \leftarrow \mathsf{Enc}_{k^0_a}(\mathsf{Enc}_{k^1_b}(k^{v(c)}_c))$
$\mathtt{ct}_{1,0} \leftarrow \mathsf{Enc}_{k^1_a}(\mathsf{Enc}_{k^0_b}(k^{g(1,0)}_c))$	$\mathtt{ct}_{1,0} \leftarrow \mathsf{Enc}_{k^1_a}(\mathsf{Enc}_{k^0_b}(k^0_c))$	$\mathtt{ct}_{1,0} \leftarrow \mathsf{Enc}_{k^1_a}(\mathsf{Enc}_{k^0_b}(k^{v(c)}_c))$
$\mathtt{ct}_{1,1} \leftarrow \mathsf{Enc}_{k^1_a}(\mathsf{Enc}_{k^1_b}(k^{g(1,1)}_c))$	$\mathtt{ct}_{1,1} \leftarrow \mathsf{Enc}_{k^1_a}(\mathsf{Enc}_{k^1_b}(k^0_c))$	$\mathtt{ct}_{1,1} \leftarrow \mathsf{Enc}_{k^1_a}(\mathsf{Enc}_{k^1_b}(k^{v(c)}_c))$

Figure 3: Garbling Gate modes: RealGate (left), SimGate (center), InputDepSimGate (right). The value $v(c)$ depends on the input x and corresponds to the bit going over the wire c in the computation $C(x)$.

- If all of the gates having in_i as an input wire are in SimGate mode, then $K[i] := k^0_{\mathsf{in}_i}$ else $K[i] := k^{x_i}_{\mathsf{in}_i}$.
- If the unique gate having out_ℓ as an output wire is in SimGate mode, then we give the output map the simulated values $\widetilde{d}_\ell := [(k^{y_\ell}_{\mathsf{out}_\ell} \to 0), (k^{1-y_\ell}_{\mathsf{out}_\ell} \to 1)]$ else the real ones $\widetilde{d}_\ell := [(k^0_{\mathsf{out}_\ell} \to 0), (k^1_{\mathsf{out}_\ell} \to 1)]$.

Real game and Simulated Game. By the definition of adaptively secure garbled circuits (Definition 2), the real game $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathsf{GC},\mathsf{Sim}}(1^\lambda, 0)$ is equivalent to $\mathsf{Hyb}^\lambda_{\mathcal{A}}((\mathsf{mode}_i = \mathsf{RealGate})_{i\in[q]}, \emptyset)$ and the simulated game $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathsf{GC},\mathsf{Sim}}(1^\lambda, 1)$ is equivalent to $\mathsf{Hyb}^\lambda_{\mathcal{A}}((\mathsf{mode}_i = \mathsf{SimGate})_{i\in[q]}, \emptyset)$. Therefore, the main aim is to show that these hybrids are indistinguishable.

4.2 Rules for Indistinguishable Hybrids

We provide rules that allow us to move from one configuration to another and prove that the corresponding hybrid games are indistinguishable. We define two rules that allow us to do this.

Definition 3 (Neighboring Hybrids)**.** *We say two valid hybrids or configurations* $((\mathsf{mode}_i)_{i\in[q]}, I), ((\mathsf{mode}'_i)_{i\in[q]}, I)$ *are "neighboring", if the set of guessed wires* I *is the same in both of them and the garbling modes of all gates except one are the same; i.e. there exists some* $j \in [q]$ *such that for all* $i \neq j$ *we have* $\mathsf{mode}_i = \mathsf{mode}'_i$*. We call* gate_j *the target gate of the two hybrids or configurations.*

Definition 4 (Predecessor/Successor/Sibling Gates)**.** *[HJO+15] Given a circuit* C *and a gate* $j \in [q]$ *of the form* $\mathsf{gate}_j = (g, w_a, w_b, w_c)$ *with incoming wires* w_a, w_b *and outgoing wire* w_c*:*

- *We define the* predecessors *of* j*, denoted by* $\mathsf{Pred}(j)$*, to be the set of gates whose outgoing wires are either* w_a *or* w_b*. If* w_a, w_b *are input wires then* $\mathsf{Pred}(j) = \emptyset$*, else* $|\mathsf{Pred}(j)| = 2$*.*
- *We define the* successors *of* j*, denoted by* $\mathsf{Succ}(j)$ *to be the set of gates that contain* w_c *as an incoming wire. If* w_c *is an output wires then* $\mathsf{Succ}(j) = \emptyset$*.*

Game $\mathsf{Hyb}^{\lambda}_{\mathcal{A}}((\mathsf{mode}_i)_{i\in[q]}, I)$

1. (Guesses) For all $w_i \in I$,
   - Let $\mathsf{Guess}(w_i) \leftarrow \{0,1\}$
2. **Receive C from $\mathcal{A}$**
   Garble circuit C:
3. (Wires) $k_i^{\sigma} \leftarrow \mathsf{KeyGen}(1^{\lambda})$ for $i \in [p]$, $\sigma \in \{0,1\}$.
4. (Gates) For each $\mathsf{gate}_i = (g, a, b, c)$ in C.
   - If $\mathsf{mode}_i = \mathsf{RealGate}$:
     run $\widetilde{g}_i \leftarrow \mathsf{GarbleGate}(g, \{k_a^{\sigma}, k_b^{\sigma}, k_c^{\sigma}\}_{\sigma\in\{0,1\}})$.
   - If $\mathsf{mode}_i = \mathsf{InputDepSimGate}$:
     run $\widetilde{g}_i \leftarrow \mathsf{GarbleSimGate}((k_a^{\sigma}, k_b^{\sigma})_{\sigma\in\{0,1\}}, k_c^{\mathsf{Guess}(c)})$.
   - If $\mathsf{mode}_i = \mathsf{SimGate}$:
     run $\widetilde{g}_i \leftarrow \mathsf{GarbleSimGate}(\{k_a^{\sigma}, k_b^{\sigma}\}_{\sigma\in\{0,1\}}, k_c^{0})$.
5. **Send $\widetilde{C}$ to $\mathcal{A}$ and get back x**
   Garble Input x:
6. (Check the guesses) For each $\forall\ i \in I$,
   - Let $v(w_i)$ be the bit on the wire w_i during the computation $C(x)$.
   - if $v(w_i) \neq \mathsf{Guess}(w_i)$ **Output 0** and abort the game.
7. (Output tables) Let $y = C(x)$. For $\ell = 1, \ldots, m$:
   Let i be the index of the gate with output wire $\mathtt{out}_{\ell}$.
   - If $\mathsf{mode}_i \neq \mathsf{SimGate}$, set $\widetilde{d}_{\ell} := [(k^0_{\mathsf{out}_{\ell}} \to 0), (k^1_{\mathsf{out}_{\ell}} \to 1)]$,
   - else, set $\widetilde{d}_{\ell} := [(k^{y_{\ell}}_{\mathsf{out}_{\ell}} \to 0), (k^{1-y_{\ell}}_{\mathsf{out}_{\ell}} \to 1)]$.
8. (Select input keys) For $\ell = 1, \ldots, n$:
   - If all gates i having in_{ℓ} as an input wire satisfy $\mathsf{mode}_i = \mathsf{SimGate}$, then set $K[\ell] := k^0_{\mathsf{in}_{\ell}}$,
   - else set $K[\ell] := k^{x_{\ell}}_{\mathsf{in}_{\ell}}$.
9. **Send $\widetilde{x} := (K, \{\widetilde{d}_{\ell}\}_{\ell\in[m]})$ to $\mathcal{A}$ and receive $\mathcal{A}$'s output**
10. **Output $\mathcal{A}$'s output**

Figure 4: The Hybrid Game.

- *We define the* siblings *of j, denoted by* $\mathsf{Siblings}(j)$ *to be the set of gates that contain either w_a or w_b as an incoming wire.*

We define $\mathsf{TimeGC}(x)$ to be the time it takes to garble a circuit of size x using Yao's garbling scheme. For convenience, we let $\mathsf{mode} \stackrel{\text{def}}{=} (\mathsf{mode}_i)_{i\in[q]}$ and omit writing the security parameter λ in the superscript of the hybrid games, since it is the same for all the games discussed here. For the same reason we use, ε and T instead of $\varepsilon(\lambda)$ and $T(\lambda)$.

4.2.1 Indistinguishability Rule 1: $\mathsf{RealGate} \leftrightarrow \mathsf{InputDepSimGate}$

This rule allows us to change the garbling mode of a gate from $\mathsf{RealGate}$ to $\mathsf{InputDepSimGate}$. It says that one can move from a circuit configuration (mode, I) to neighboring circuit configuration (mode', I) where the mode of the target gate changes from $\mathsf{RealGate}$ in mode to $\mathsf{InputDepSimGate}$ in mode' (and vice versa).

Lemma 1. *Let* $\mathsf{Hyb}(\mathsf{mode}, I)$ *and* $\mathsf{Hyb}(\mathsf{mode}', I)$ *be two neighboring hybrids, with target* gate_j *such that* $\mathsf{mode}_j = \mathsf{RealGate}$ *and* $\mathsf{mode}'_j = \mathsf{InputDepSimGate}$*. In addition, for all* $i \in \mathsf{Pred}(j)$*:* $\mathsf{mode}_i = \mathsf{InputDepSimGate}$*. Then* $\mathsf{Hyb}(\mathsf{mode}, I)$ *and* $\mathsf{Hyb}(\mathsf{mode}', I)$ *are* $(T(\lambda), \varepsilon(\lambda))$*-indistinguishable as long as* $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is an encryption scheme* $(T'(\lambda), \varepsilon(\lambda))$*-secure under* CPA double encryption *as per Definition 6 and* $T'(\lambda) = T(\lambda) + \mathsf{TimeGC}(|C|)$.

Proof. Let (mode, I) and (mode', I) be as in the statement of the lemma, two valid circuit configurations. Towards a contradiction, assume that there exists a adversary $\mathcal{A}$ who runs in time T and distinguishes $H^0 := \mathsf{Hyb}(\mathsf{mode}, I)$ and $H^1 := \mathsf{Hyb}(\mathsf{mode}', I)$. i.e.,

$$\left|\Pr\left[H^0_{\mathcal{A}} = 1\right] - \Pr\left[H^1_{\mathcal{A}} = 1\right]\right| > \varepsilon.$$

We construct an adversary $\mathcal{B}$, running in time T' that breaks the double CPA-security of the encryption scheme $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ which is used to garble gates. More specifically, we show that $\mathcal{B}$ wins the chosen double encryption security game (Def. 6) which is implied by CPA security. The formal description of adversary $\mathcal{B}$ is provided in Fig. 5.

Informally, $\mathcal{B}$ –on input (mode, I) and target gate j– aims to use her CPA-oracle access in $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$ to generate distribution H^b. The only difference between H^0 and H^1 is in the way gate $\widetilde{g}_j$ is computed. On a high level, the reduction $\mathcal{B}$ will compute all garbled gates $\widetilde{g}_i$ for $i \neq j$, according to experiment $\mathsf{Hyb}(\mathsf{mode}, I)$, and will compute the garbled gate $\widetilde{g}_j$ using the ciphertexts obtained as a challenge in the experiment $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$.

In more detail, let $\mathsf{gate}_j = (g^*, a^*, b^*, c^*)$ be the target gate. Recall, the predecessors of gate_j (with output wires a^* and b^*) are in $\mathsf{InputDepSimGate}$ mode. Therefore garbling of each gate in $\mathsf{Pred}(j)$, includes encryptions of one wire label only. We call these wires (which are fixed by the bit values guessed in step 1, $\alpha, \beta \in \{0,1\}$) $k^\alpha_{a^*}$ and $k^\beta_{b^*}$. Consequently the wire label decrypted during the evaluation of gate_j is also the same wire label in both games, $k^{g(\alpha,\beta)}_{c^*}$. The difference is $\mathsf{mode}_j = \mathsf{RealGate}$ in $\mathsf{Hyb}(\mathsf{mode}, I)$, meaning, there is another wire label, which was used to garble gate_j and its ciphertext is one of the four ciphertexts $\mathtt{ct}_s$, $s \in \{0,1\}^2$. But in $\mathsf{Hyb}(\mathsf{mode}', I)$, garbling mode of gate_j is $\mathsf{InputDepSimGate}$ and the only wire label used is $k^{g(\alpha,\beta)}_{c^*}$. To create the same garbled gate distributions using the challenger of the $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$, the reduction $\mathcal{B}$ –who knows all wire keys *except* for $k^{1-\alpha}_{a^*}, k^{1-\beta}_{b^*}$– will create $\mathtt{ct}_{\alpha,\beta}$ as an encryption of $k^{g(\alpha,\beta)}_{c^*}$ on its own, but the remaining three ciphertexts $\mathtt{ct}_{\alpha',\beta'}$ will come from the experiment $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$ as either encryptions of different values $k^{g(\alpha',\beta')}_{c^*}$ (real) or of the same value $k^{g(\alpha,\beta)}_{c^*}$[2].

The one subtlety is that the reduction needs to create encryptions under the keys $k^{1-\alpha}_{a^*}, k^{1-\beta}_{b^*}$ to create garbled gates $\widetilde{g}_i$ for gates i that are siblings of gate j. It can do that by using the encryption oracles which are given to it as part of the experiment $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$. The formal description of the reduction $\mathcal{B}$ is provided

[2] If $a^* = b^*$, (gate_j has fan-in 1), then $\mathcal{B}$ uses the challenger of the CPA encryption instead of the double-encryption scheme. The reduction considers the CPA challenger's key as $k^{1-\alpha}_{b^*}$, and using appropriate queries garbles gate_j.

Adversary $\mathcal{B}$ (Reduction)
Input mode, I and j.

1. (Guesses) For all $w_i \in I$,
   - Let $\mathsf{Guess}(w_i) \leftarrow U_1$
2. **Receive C from $\mathcal{A}$**
   Garble circuit C:
3. (Wires) $k_i^\sigma \leftarrow \mathsf{KeyGen}(1^\lambda)$ for $i \in [p]$, $\sigma \in \{0,1\}$. *except* for the two keys $k_{a^*}^{1-\alpha}, k_{b^*}^{1-\beta}$.
4. (Gates) For each $\mathsf{gate}_i = (g, a, b, c)$ in C except gate_j.
   - If $\mathsf{mode}_i = \mathsf{RealGate}$:
     run $\widetilde{g}_i \leftarrow \mathsf{GarbleGate}(g, \{k_a^\sigma, k_b^\sigma, k_c^\sigma\}_{\sigma\in\{0,1\}})$.
   - If $\mathsf{mode}_i = \mathsf{InputDepSimGate}$:
     run $\widetilde{g}_i \leftarrow \mathsf{GarbleSimGate}((k_a^\sigma, k_b^\sigma)_{\sigma\in\{0,1\}}, k_c^{\mathsf{Guess}(c)})$
   - If $\mathsf{mode}_i = \mathsf{SimGate}$:
     run $\widetilde{g}_i \leftarrow \mathsf{GarbleSimGate}(\{k_a^\sigma, k_b^\sigma\}_{\sigma\in\{0,1\}}, k_c^0)$.

   4·A) Let $\alpha := \mathsf{Guess}(a^*)$, $\beta := \mathsf{Guess}(b^*)$

   4·B) Let $x_0 = k_{c^*}^{g^*(1-\alpha,\beta)}, y_0 = k_{c^*}^{g^*(\alpha,1-\beta)}, z_0 = k_{c^*}^{g^*(1-\alpha,1-\beta)}$ and $x_1 = y_1 = z_1 = k_{c^*}^{g^*(\alpha,\beta)}$.

   4·C) Give $k_{a^*}^\alpha, k_{b^*}^\beta$ and $(x_0, y_0, z_0), (x_1, y_1, z_1)$ to the challenger of $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$. The challenger of $\mathsf{Exp}^{\mathsf{double}}(1^\lambda, b)$ chooses two keys which we implicitly define as $k_{a^*}^{1-\alpha}, k_{a^*}^{1-\beta}$. It gives B the ciphertexts $\mathtt{ct}_x, \mathtt{ct}_y, \mathtt{ct}_z$ and oracle access to $\mathsf{Enc}_{k_{a^*}^{1-\alpha}}(\cdot)$ and $\mathsf{Enc}_{k_{b^*}^{1-\beta}}(\cdot)$.

   4·D) For the gate j :
   - Compute $\mathtt{ct}_{\alpha,\beta} \leftarrow \mathsf{Enc}_{k_{a^*}^\alpha}(\mathsf{Enc}_{k_{b^*}^\beta}(k_{c^*}^{g(\alpha,\beta)}))$.
   - Set $\mathtt{ct}_{1-\alpha,\beta} := \mathtt{ct}_x$, $\mathtt{ct}_{\alpha,1-\beta} := \mathtt{ct}_y$, $\mathtt{ct}_{1-\alpha,1-\beta} := \mathtt{ct}_z$.
   - Let $\widetilde{g}_j$ be a random ordering of $[\mathtt{ct}_{0,0}, \mathtt{ct}_{0,1}, \mathtt{ct}_{1,0}, \mathtt{ct}_{1,1}]$

5. **Send $\widetilde{C}$ to $\mathcal{A}$. Obtain x from $\mathcal{A}$.**

   Garble Input x:
6. (Check the guesses) For each $\forall\ i \in I$,
   - Let $v(w_i)$ be the bit on the wire w_i during the computation $C(x)$.
   - if $v(w_i) \neq \mathsf{Guess}(w_i)$ **Output 0** and abort.
7. (Output tables) Let $y = C(x)$. For $\ell = 1, \ldots, m$:
   Let i be the index of the gate with output wire $\mathtt{out}_\ell$.
   - If $\mathsf{mode}_i \neq \mathsf{SimGate}$, set $\widetilde{d}_\ell := [(k_{\mathsf{out}_\ell}^0 \to 0), (k_{\mathsf{out}_\ell}^1 \to 1)]$,
   - else, set $\widetilde{d}_\ell := [(k_{\mathsf{out}_\ell}^{y_\ell} \to 0), (k_{\mathsf{out}_\ell}^{1-y_\ell} \to 1)]$.
8. (Select input keys) For $\ell = 1, \ldots, n$:
   - If all gates i having in_ℓ as an input wire satisfy $\mathsf{mode}_i = \mathsf{SimGate}$, then set $K[\ell] := k_{\mathsf{in}_\ell}^0$,
   - else set $K[\ell] := k_{\mathsf{in}_\ell}^{x_\ell}$.
9. **Set $\widetilde{x} := (K, \{\widetilde{d}_\ell\}_{\ell\in[m]})$. Send $\widetilde{x}$ to $\mathcal{A}$ and output whatever $\mathcal{A}$ outputs.**

Figure 5: Proof of security for rule 1: the reduction B uses an adversary $\mathcal{A}$ that distinguishes the hybrids to play the chosen double encryption security game (Def. 6) denoted by $\mathsf{Exp}^{\mathsf{double}}$.

in Fig. 5. Finally notice that $\mathcal{B}$'s running time is, the time it takes to create the garble circuit plus the time it takes to run $\mathcal{A}$, so $T' = T + \mathsf{TimeGC}(|C|)$.

$$\begin{aligned}&\left|\Pr[H^0_{\mathcal{A}} = 1] - \Pr[H^1_{\mathcal{A}} = 1]\right| \\ \leq &\left|\Pr[\mathsf{Exp}^{\mathsf{double}}_{\mathcal{B}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Exp}^{\mathsf{double}}_{\mathcal{B}}(1^\lambda, 1) = 1]\right| \leq \varepsilon.\end{aligned}$$

which proves the Lemma. □

4.2.2 Indistinguishability Rule 2. $\mathsf{InputDepSimGate} \leftrightarrow \mathsf{SimGate}$

This rule allows us to change the mode of a gate j from $\mathsf{InputDepSimGate}$ to $\mathsf{SimGate}$ under the condition that all successor gates $i \in \mathsf{Succ}(j)$ satisfy that $\mathsf{mode}_i \in \{\mathsf{InputDepSimGate}, \mathsf{SimGate}\}$.

Lemma 2. *Let* $\mathsf{Hyb}(\mathsf{mode}, I)$ *and* $\mathsf{Hyb}(\mathsf{mode}', I)$ *be two neighboring hybrids, with target* gate_j *such that* $\mathsf{mode}_j = \mathsf{InputDepSimGate}$ *in* mode *and* $\mathsf{mode}_j = \mathsf{SimGate}$ *in* mode'. *In addition, for all* $i \in \mathsf{Succ}(j)$ *we have* $\mathsf{mode}_i \in \{\mathsf{SimGate}, \mathsf{InputDepSimGate}\}$. *Then for any* $\mathcal{A}$, $\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}, I)$ *and* $\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}', I')$ *are identically distributed.*

Proof. Fix any adversary $\mathcal{A}$. Define $H_0 := \mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}, I)$ and $H_1 := \mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}', I)$. The difference between the hybrids is in how the garbled gate $\widetilde{g}_j$ is created:

- In H_0, we set $\widetilde{g}_j \leftarrow \mathsf{GarbleSimGate}((k^\sigma_{a^*}, k^\sigma_{b^*})_{\sigma\in\{0,1\}}, k^{\mathsf{Guess}(c^*)}_{c^*})$.
- In H_1, we set $\widetilde{g}_j \leftarrow \mathsf{GarbleSimGate}((k^\sigma_{a^*}, k^\sigma_{b^*})_{\sigma\in\{0,1\}}, k^0_{c^*})$.

If j is not an output gate, and all successor gates $i \in \mathsf{Succ}(j)$ are in $\{\mathsf{SimGate}, \mathsf{InputDepSimGate}\}$ modes then the keys $k^0_{c^*}$ and $k^1_{c^*}$ are treated symmetrically everywhere in the game other than in $\widetilde{g}_j$. Therefore, by symmetry, there is no difference between using $k^0_{c^*}$ and $k^{\mathsf{Guess}(c^*)}_{c^*}$ in $\widetilde{g}_j$

If j is an output gate then the keys $k^0_{c^*}$ and $k^1_{c^*}$ are only used in $\widetilde{g}_j$ and in the output map $\widetilde{d}_j$. Therefore, by symmetry, there is no difference between using $k^{y_j}_{c^*}$ in $\widetilde{g}_j$ and setting $\widetilde{d}_j := [(k^0_{\mathsf{out}_j} \to 0), (k^1_{\mathsf{out}_j} \to 1)]$ (in H_0) versus using $k^0_{c^*}$ in $\widetilde{g}_j$ and setting $\widetilde{d}_j := [(k^{y_j}_{\mathsf{out}_j} \to 0), (k^{1-y_j}_{\mathsf{out}_j} \to 1)]$ (in H_1).

One last difference between the hybrids occurs if some wire in_i becomes only connected to gates that are in $\mathsf{SimGate}$ in H_1. In this case, when we create the garbled input $\widetilde{x}$, then in H_0 we give $K[i] := k^{x_i}_{\mathsf{in}_i}$ but in H_1 we give $K[i] := k^0_{\mathsf{in}_i}$. Since the keys $k^0_{\mathsf{in}_i}, k^1_{\mathsf{in}_i}$ are treated symmetrically everywhere in the game (both in H_0 and H_1) other than in $K[i]$, there is no difference between setting $K[i] := k^0_{\mathsf{in}_i}$ versus $K[i] := k^{x_i}_{\mathsf{in}_i}$.

□

4.2.3 Scaling Indistinguishability

We now show that by adding guesses we can make the hybrids more indistinguishable, or equivalently, removing guesses makes the hybrids more distinguishable. This lemma is crucial for comparing hybrids with different guesses by scaling the number of guesses up or down to make the comparison possible.

Lemma 3. *If* $\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}, I), \mathsf{Hyb}(\mathsf{mode}', I)\right] = \varepsilon$ *and* J *is a set of wires, disjoint from* I *then*

$$\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}, I \cup J), \mathsf{Hyb}(\mathsf{mode}', I \cup J)\right] = 2^{-|J|} \cdot \varepsilon.$$

Proof. For any probabilistic T bounded adversary $\mathcal{A}$, we have

$$\Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}, I \cup J) = 1\right] = 2^{-|J|} \Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}, I) = 1)\right].$$

Because with probability $2^{-|J|}$, (the probability of guessing the extra $|J|$ wires correctly) $\mathcal{A}$ playing the game $\mathsf{Hyb}(\mathsf{mode}, I \cup J)$ has the exact same interactions as in game $\mathsf{Hyb}(\mathsf{mode}, I)$ and therefore the same exact outputs. The same holds for $\Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}', I \cup J) = 1\right]$ therefore,

$$\begin{aligned}&\left[\Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}, I \cup J) = 1\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}', I \cup J) = 1\right]\right] \\ = \; &2^{-|J|}\left|\Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}, I) = 1\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{A}}(\mathsf{mode}', I) = 1\right]\right| \leq 2^{-|J|} \cdot \varepsilon\end{aligned}$$

□

5 Pebbling and Sequences of Hybrid Games

In the last section we defined hybrid games parameterized by a configuration (mode, I). We also gave 2 rules, which describe ways that allow us to move from one configuration to another in indistinguishable steps. Now our goal is to use the given rules so as to define a *sequence of indistinguishable hybrid games* that takes us from the *real game* $\mathsf{Hyb}((\mathsf{mode}_i = \mathsf{RealGate})_{i\in[q]}, I = \emptyset)$ to the *simulation* $\mathsf{Hyb}((\mathsf{mode}_i = \mathsf{SimGate})_{i\in[q]}, I = \emptyset)$.

Pebbling Game. We capture the problem of finding a sequences of hybrid games using a certain type of *pebbling game* on the graph of circuit C.

- *Graph of circuit C* is obtained by assigning a node to each gate, and a directed edge from node i to node j for each wire going out of gate i and into gate j. To make this consistent, we think of each input wire (in) as outgoing wire of an empty (dummy) gate, going into a gate in level 1 of the circuit. Since we are always considering a pebbling on the graph of a circuit, we use words gate/node and wire/edge interchangeably.
- *Pebbles.* Each gate can either have *no pebble*, a *black pebble*, or *a gray pebble* on it (this will correspond to $\mathsf{RealGate}, \mathsf{InputDepSimGate}$ and $\mathsf{SimGate}$ modes respectively). Initially, the circuit starts out with no pebbles on any gate. The game consist of the following possible moves:

  Pebbling Rule A. We can place or remove a black pebble on a gate as long as both predecessors of that gate have black pebbles on them (or it is an input gate).

  Pebbling Rule B. We can replace a black pebble with a gray pebble on a gate as long as all successors of that gate have black or gray pebbles on them (or the gate is an output gate).
- A *pebbling* of a circuit C starts with no pebbles on the graph and is a sequence of γ moves that follow rules A and B and that end up with a gray pebble on every gate. We say that a pebbling uses t black pebbles if this is the maximal number of black pebbles on the circuit at any point in time during the game.
- A *pebble configuration* specifies for each gate, whether it contains no pebble, a gray pebble, or a black pebble.

From Pebbling to Sequence of Hybrids. A pebbling in γ moves has a sequence of $\gamma + 1$ pebble configurations starting with no pebbles and ending with a gray pebble on each gate. Each pebble configuration follows from the preceding one by a move that satisfies pebbling rules A or B. Next we create a sequence of hybrids by defining one hybrid from each pebbling configuration.

- For every gate $i \in [q]$, we set $\mathsf{mode}_i = \mathsf{RealGate}$ if gate i has no pebble, $\mathsf{mode}_i = \mathsf{InputDepSimGate}$ if gate i has a black pebble, and $\mathsf{mode}_i = \mathsf{SimGate}$ if gate i has a gray pebble.
- We set I to be the set of the output wires of the gates with black pebbles.

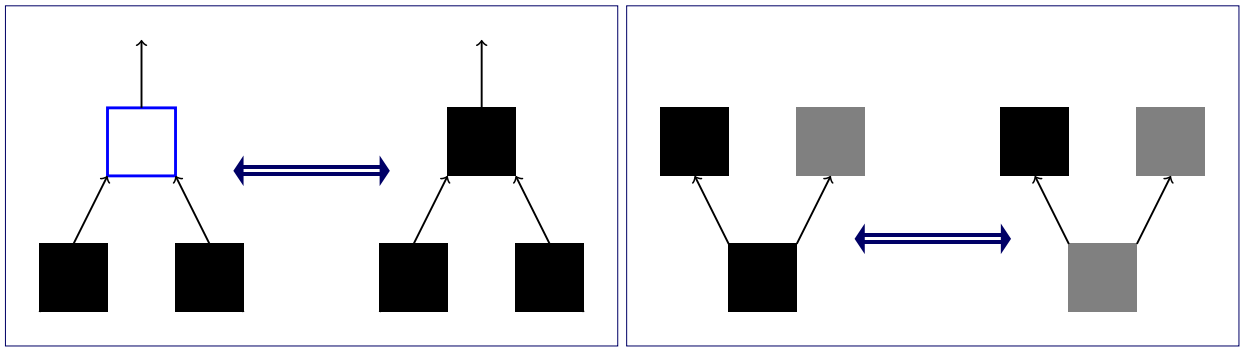

Figure 6: Pebbling Rules

Therefore a pebbling defines a sequence of hybrids $H_\alpha = \mathsf{Hyb}(\mathsf{mode}^\alpha, I^\alpha)$ for $\alpha = 0, \ldots, \gamma$ where $H_0 = \mathsf{Hyb}((\mathsf{mode}_i^0 = \mathsf{RealGate})_{i\in[q]}, \emptyset)$ is the real game and $H_\gamma = \mathsf{Hyb}((\mathsf{mode}_i^\gamma = \mathsf{SimGate})_{i\in[q]}, \emptyset)$ is the simulated game, and each H_α is induced by the pebbling configuration after α moves. In our next theorem and the following corollary, we prove that the sequence of hybrids obtained from a pebbling, as explained above, shows indistinguishability of the real and simulated games.

Theorem 1. *Assume that there is a pebbling of circuit C in γ moves, using t black pebbles. Also assume that the encryption scheme $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is $(T + \mathsf{TimeGC}(|C|), \varepsilon)$-secure under* CPA double encryption.

Then, the sequence of hybrids obtained from such pebbling as described above has the following property. For any $\alpha \in \{0, 1, \cdots, \gamma\}$, $H_\alpha = \mathsf{Hyb}(\mathsf{mode}^\alpha, I^\alpha)$

$$\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^\alpha), H_\alpha\right] \le \sum_{i=1}^{\alpha} 2^{r_i - |I^\alpha|} \cdot \varepsilon \le \alpha 2^{t-|I^\alpha|}\varepsilon$$

where $r_\alpha = \max\left(\left|I^{\alpha-1}\right|, |I^\alpha|\right) \le t$, for $\alpha \in [\gamma]$.

Proof. We show the claim holds for mode_0 and any configurations; $(\mathsf{mode}^\alpha, I^\alpha)$, $\alpha \in \{0, 1, \cdots \gamma\}$ by induction on the number of pebbling steps taken so far (i.e., α). For convenience, let $s_\alpha = |I^\alpha|$ and remember $r_\alpha = \max(s_{\alpha-1}, s_\alpha)$.

Base case. Let $\alpha = 0$, $\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^0), H_0\right] = \mathbf{D}_T[H_0, H_0] = 0$.

Inductive step. Assume the claim holds for α, we show it holds for $\alpha + 1$.

- If the $\alpha+1$st move in the pebbling game is to add a black pebble: $s_{\alpha+1} = s_\alpha + 1$ and $r_{\alpha+1} = s_{\alpha+1}$

$$\begin{aligned}
&\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha+1}), \mathsf{Hyb}(\mathsf{mode}^{\alpha+1}, I^{\alpha+1})\right] \\
\le\ &\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha+1}), \mathsf{Hyb}(\mathsf{mode}^{\alpha}, I^{\alpha+1})\right] \\
&\qquad + \mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^\alpha, I^{\alpha+1}), \mathsf{Hyb}(\mathsf{mode}^{\alpha+1}, I^{\alpha+1})\right] && (4)\\
\le\ &2^{-1} \cdot \mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha}), \mathsf{Hyb}(\mathsf{mode}^{\alpha}, I^{\alpha})\right] + \varepsilon && (5)\\
\le\ &2^{-1}\sum_{i=1}^{\alpha} 2^{r_i - s_\alpha} \cdot \varepsilon + \varepsilon \le \sum_{i=1}^{\alpha} 2^{r_i - s_\alpha - 1} \cdot \varepsilon + \varepsilon && (6)\\
\le\ &\sum_{i=1}^{\alpha} 2^{r_i - s_{\alpha+1}} \cdot \varepsilon + 2^{r_{\alpha+1} - s_{\alpha+1}} \cdot \varepsilon \le \sum_{i=1}^{\alpha+1} 2^{r_i - s_{\alpha+1}} \cdot \varepsilon
\end{aligned}$$

  Line 4 follows from the previous line by the Triangle Inequality. By Lemma 1 or 2, $\mathbf{D}_T[\mathsf{Hyb}(\mathsf{mode}^\alpha, I^{\alpha+1}), \mathsf{Hyb}(\mathsf{mode}^{\alpha+1}, I^{\alpha+1})] \le \varepsilon$. By Lemma 3 we have that $\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha+1}), \mathsf{Hyb}(\mathsf{mode}^{\alpha}, I^{\alpha+1})\right] \le \mathbf{D}_T[\mathsf{Hyb}(\mathsf{mode}^0, I^\alpha), \mathsf{Hyb}(\mathsf{mode}^\alpha, I^\alpha)]/2$. Combining the two we get Line 5. We use the induction hypothesis to arrive at Line 6. The last line follows by noticing $s_{\alpha+1} = s_\alpha + 1$ and $r_{\alpha+1} = s_{\alpha+1}$.

- If the $\alpha+1$st move in the pebbling game is to remove a black pebble: $s_{\alpha+1} = s_\alpha - 1$ and $r_{\alpha+1} = s_\alpha$

$$\begin{aligned}
&\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha+1}), \mathsf{Hyb}(\mathsf{mode}^{\alpha+1}, I^{\alpha+1})\right] \\
\le\ &2\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha}), \mathsf{Hyb}(\mathsf{mode}^{\alpha+1}, I^{\alpha})\right] && (7)\\
\le\ &2\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^0, I^{\alpha}), \mathsf{Hyb}(\mathsf{mode}^{\alpha}, I^{\alpha})\right] \\
&\qquad + 2\mathbf{D}_T\left[\mathsf{Hyb}(\mathsf{mode}^\alpha, I^{\alpha}), \mathsf{Hyb}(\mathsf{mode}^{\alpha+1}, I^{\alpha})\right] && (8)\\
\le\ &2\left(\sum_{i=1}^{\alpha} 2^{r_i - s_\alpha} \cdot \varepsilon + \varepsilon\right) \le \sum_{i=1}^{\alpha} 2^{r_i - s_\alpha + 1} \cdot \varepsilon + 2\varepsilon && (9)\\
\le\ &\sum_{i=1}^{\alpha} 2^{r_i - s_{\alpha+1}} \cdot \varepsilon + 2^{r_{\alpha+1} - s_{\alpha+1}} \cdot \varepsilon \le \sum_{i=1}^{\alpha+1} 2^{r_i - s_{\alpha+1}} \cdot \varepsilon
\end{aligned}$$

  Similar to the last case, Line 7 follows from the previous line By Lemma 3. Line 8 follows from the Triangle inequality. By Lemma 1 or 2 and the induction hypothesis we arrive at Line 9. The last line follows by noticing $s_{\alpha+1} = s_\alpha - 1$ and $r_{\alpha+1} = s_\alpha$.

Rule	wires in I^α, $I^{\alpha+1}$	$\mathsf{mode}_j^\alpha \to \mathsf{mode}_j^{\alpha+1}$	Hybrids	Lemma
A	$\mathtt{WPred} \subseteq I^\alpha$, $\mathtt{WPred} \cup \{c^*\} \subseteq I^{\alpha+1}$	RealGate $\to$ InputDepSimGate	$(\mathsf{mode}^\alpha, I^{\alpha+1})$ $(\mathsf{mode}^{\alpha+1}, I^{\alpha+1})$	1
	$\mathtt{WPred} \cup \{c^*\} \subseteq I^\alpha$, $\mathtt{WPred} \subseteq I^{\alpha+1}$	InputDepSimGate $\to$ RealGate	$(\mathsf{mode}^\alpha, I^\alpha)$ $(\mathsf{mode}^{\alpha+1}, I^\alpha)$	
B	$\mathtt{WPred} \cup \{c^*\} \subseteq I^\alpha$, $\mathtt{WPred} \subseteq I^{\alpha+1}$	InputDepSimGate $\to$ SimGate	$(\mathsf{mode}^\alpha, I^\alpha)$ $(\mathsf{mode}^{\alpha+1}, I^\alpha)$	2
	$\mathtt{WPred} \subseteq I^\alpha$, $\mathtt{WPred} \cup \{c^*\} \subseteq I^{\alpha+1}$	SimGate $\to$ InputDepSimGate	$(\mathsf{mode}^\alpha, I^{\alpha+1})$ $(\mathsf{mode}^{\alpha+1}, I^{\alpha+1})$	

Figure 7: From Pebbling Rules to Indistinguishable Hybrids. $\mathtt{WPred} :=$ {output wires of $\mathsf{Pred}(j)$}, $\mathtt{WSucc} :=$ {output wires of $\mathsf{Succ}(j)$}.

The reason we can apply lemmas 1 or 2, is that the pebbling game rules (A and B) guarantee that the garbling modes of each two hybrids in our sequence have the necessary properties for applying lemmas 1 or 2. In addition we created the set I such that it includes all the necessary wires to keep the configuration valid. For more details, see Figure 7, where we change gate_j's mode at step $\alpha+1$, following rule A or B. □

Corollary 1. *Assume that* $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is an encryption scheme which is* $(T(\lambda), \varepsilon(\lambda))$*-secure under* CPA double encryption. *If there is a pebbling of circuit* C *in* γ *moves, using* t *black pebbles then* $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 0)$ *and* $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 1)$ *are* $(T'(\lambda), \varepsilon'(\lambda))$*-indistinguishable where*

- $\varepsilon'(\lambda) \leq \sum_{i=1}^{\gamma} 2^{r_i} \cdot \varepsilon(\lambda) \leq \gamma \cdot 2^t \cdot \varepsilon(\lambda)$
- $T'(\lambda) = T(\lambda) - \mathsf{TimeGC}(|C|)$.

where $r_i = \max(s_{i-1}, s_i)$ *and* s_i *is the number of black pebbles used at the* i*th pebbling step.*

Proof. By definition $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 0) = \mathsf{Hyb}^\lambda(\mathsf{mode}^0, I^0)$ and $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathsf{GC},\mathsf{Sim}}(1^\lambda, 1) = \mathsf{Hyb}^\lambda(\mathsf{mode}^\gamma, I^\gamma)$ where $I^0 = I^\gamma = \emptyset$. By Theorem 1 with $\alpha = \gamma$, we have $\mathbf{D}_{T(\lambda)}\left[\mathsf{Hyb}^\lambda(\mathsf{mode}^0, \emptyset), \mathsf{Hyb}^\lambda(\mathsf{mode}^\gamma, \emptyset)\right] \leq \sum_{i=1}^{\gamma} 2^{r_i} \cdot \varepsilon(\lambda)$ which proves the Corollary. □

Corollary 2. *If there is a pebbling of circuit* C *in* γ *moves, using* t *black pebbles then* GC *is adaptively secure with online complexity*

1. $(m+n)\lambda$, *when* Γ *is secure under* CPA double encryption *and* $2^t\gamma = \mathsf{poly}(\lambda)$.
2. $(m+n)\mathsf{poly}(\lambda + \log\gamma + t)$, *when* Γ *is sub-exponentially secure under* CPA double encryption *and* $\log(\gamma) + t = \mathsf{poly}(\lambda)$.

Proof. The online complexity of the garbling scheme consist of $(m+n)$ secret keys of the scheme Γ.

For case (1) we only need standard security of Γ to survive a polynomial security loss of $2^t\gamma = \mathsf{poly}(\lambda)$. Therefore, we can set the security parameter of Γ to λ, which gives a key size of λ.

For case (2) we need to survive a security loss of $2^t\gamma = 2^{\mathsf{poly}(\lambda)}$. If the encryption scheme Γ is sub-exponentially secure it means that when instantiated with security parameter λ' it has security $\varepsilon(\lambda') \leq 2^{-(\lambda')^\nu}$ for some constant ν and all large enough λ'. Therefore we need to set $\lambda' = (\lambda + \log(\gamma) + t)^{1/\nu}$ to ensure that $2^t\gamma\varepsilon(\lambda')$ is negligible, which results in a key size of $\lambda' = \mathsf{poly}(\lambda + \log(\gamma) + t)$. □

5.1 Pebbling Strategies

We now rely on a result of [HJO+15] to instantiate Corollary 2. In particular, it shows that for any circuit with q gates and depth d there is a pebbling strategy which makes at most $\gamma = q \cdot 2^{2d}$ moves and uses $t = 2d$ black pebbles. See Appendix B for the description of the strategy. By instantiating Corollary 2 with the above strategy, we obtain the following corollary.

Corollary 3. *Assuming the existence of (standard) one-way functions, Yao's garbling schemes is adaptively secure with on-line complexity* $(n+m)\lambda$ *for all circuits of depth* $d = O(\log \lambda)$.

Assuming the existence of sub-exponentially secure one-way functions Yao's garbling schemes is adaptively secure with on-line complexity $(n+m)\mathsf{poly}(\lambda, d)$*, for arbitrary circuits of depth* $d = \mathsf{poly}(\lambda)$.

6 Conclusions

We show that Yao's garbled circuit construction is already adaptively secure, without the need for any modification, at least when it comes to NC1 circuits. More generally, we give a reduction where the security loss is related (exponentially) to the pebble complexity of the circuit, which can often be much smaller than the input size, and therefore beats the naive reduction that guesses the entire input. It remains as an open problem to improve the reduction further or to give some negative results showing that it cannot be done.

References

[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, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 166–184. Springer, Heidelberg, August 2013.

[AS15] Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. Cryptology ePrint Archive, Report 2015/776, 2015. http://eprint.iacr.org/.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, 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, *ASIACRYPT 2012*, volume 7658 of *LNCS*, 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 12*, pages 784–796. ACM Press, October 2012.

[FJP15] Georg Fuchsbauer, Zahra Jafargholi, and Krzysztof Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 601–620. Springer, Heidelberg, August 2015.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014.

[HJO+15] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. *IACR Cryptology ePrint Archive*, 2015:1250, 2015.

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

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Symmetric-Key Encryption with Special Correctness [LP09]

In our construction of the garbling scheme, we use a symmetric-key encryption scheme $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ which satisfies the standard definition of CPA security and an additional *special correctness* property below (this is a simplified and sufficient variant of the property described in from [LP09]). We need this property to ensure the correctness of our garbled circuit construction.

Definition 5 (Special Correctness)**.** *A CPA-secure symmetric-key encryption* $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies special correctness if there is some negligible function* ε *such that for any message* m *we have:*

$$\Pr[\mathsf{Dec}_{k_2}(\mathsf{Enc}_{k_1}(m)) \neq \perp \ : \ k_1, k_2 \leftarrow \mathsf{KeyGen}(1^\lambda)] \leq \varepsilon(\lambda).$$

Construction. Let $F = \{f_k\}$ be a family of pseudorandom functions where $f_k : \{0,1\}^\lambda \to \{0,1\}^{\lambda+s}$, for $k \in \{0,1\}^\lambda$ and s is a parameter denoting the message length. Define $\mathsf{Enc}_k(m) = (r, f_k(r) \oplus m0^\lambda)$ where $m \in \{0,1\}^s$, $r \xleftarrow{\$} \{0,1\}^\lambda$ and $m0^\lambda$ denotes the concatenation of m with a string of 0s of length λ. Define $\mathsf{Dec}_k(c)$ which parses $c = (r, z)$, computes $w = z \oplus f_k(r)$ and if the last λ bits of w are 0's it outputs the first s bits of w, else it outputs $\perp$.

It's easy to see that this scheme is CPA secure and that it satisfies the special correctness property.

Double Encryption Encryption Security. For convenience, we define a notion of double encryption security, following [LP09]. This notion is implied by standard CPA security but is more convenient to use in our security proof of garbled circuit security.

Definition 6 (Double-encryption security)**.** *An encryption scheme* $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$

- *is* $(T(\lambda), \varepsilon(\lambda))$*-secure under* chosen double encryption *if*

$$\mathbf{D}_{T(\lambda)}\left[\mathsf{Exp}^{\mathsf{double}}(1^\lambda, 0), \mathsf{Exp}^{\mathsf{double}}(1^\lambda, 1)\right] = \varepsilon(\lambda).$$

- *is secure under* chosen double encryption *if*

$$\mathsf{Exp}^{\mathsf{double}}(1^\lambda, 0) \overset{\text{comp}}{\approx} \mathsf{Exp}^{\mathsf{double}}(1^\lambda, 1).$$

- *is sub-exponentially secure if*

$$\exists\ \nu > 0, \forall T(\lambda) \in \mathsf{poly}(\lambda),\ \mathbf{D}_{T(\lambda)}\left[\mathsf{Exp}^{\mathsf{double}}(1^\lambda, 1), \mathsf{Exp}^{\mathsf{double}}(1^\lambda, 0)\right] \leq \varepsilon(\lambda) = 1/2^{\lambda^\nu}$$

where the experiment $\mathsf{Exp}^{\mathsf{double}}_{\mathcal{A}}$ *is defined as follows. s* **Experiment** $\mathsf{Exp}^{\mathsf{double}}_{\mathcal{A}}(1^\lambda, b)$

1. *The adversary* $\mathcal{A}$ *on input* 1^λ *outputs two keys* k_a *and* k_b *of length* λ *and two triples of messages* (x_0, y_0, z_0) *and* (x_1, y_1, z_1) *where all messages are of the same length.*
2. *Two keys* $k'_a, k'_b \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ *are chosen.*
3. $\mathcal{A}^{\mathsf{Enc}_{k'_a}(\cdot), \mathsf{Enc}_{k'_b}(\cdot)}$ *is given the challenge ciphertexts* $c_x \leftarrow \mathsf{Enc}_{k_a}(\mathsf{Enc}_{k'_b}(x_b))$, $c_y \leftarrow \mathsf{Enc}_{k'_a}(\mathsf{Enc}_{k_b}(y_b))$, $c_z \leftarrow \mathsf{Enc}_{k'_a}(\mathsf{Enc}_{k'_b}(z_b))$ *as well as* **oracle access** *to* $\mathsf{Enc}_{k'_a}(\cdot)$ *and* $\mathsf{Enc}_{k'_b}(\cdot)$.
4. $\mathcal{A}$ *outputs* b' *which is the output of the experiment.*

The following lemma is essentially immediate - see [LP09] for a formal proof.

Lemma 4. *If* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is CPA-secure then it is secure under chosen double encryption with the same security parameter.*

B Pebbling Strategy [HJO+15]

This is a recursive strategy defined as follows.

- $\mathsf{Pebble}(C)$:
  For each gate i in C starting with the gates at the top level moving to the bottom level:
  1. $\mathsf{RecPutBlack}(C, i)$
  2. Replace the black pebble on gate i with a gray pebble.
- $\mathsf{RecPutBlack}(C, i)$: // Let $\mathsf{LeftPred}(C, i)$ and $\mathsf{RightPred}(C, i)$ be the two predecessors of gate i in C.
  1. If gate i is an input gate, put a black pebble on i and **return**.
  2. Run $\mathsf{RecPutBlack}(C, \mathsf{LeftPred}(C, i))$, $\mathsf{RecPutBlack}(C, \mathsf{RightPred}(C, i))$
  3. Put a black pebble on gate i.
  4. Run $\mathsf{RecRemoveBlack}(C, \mathsf{LeftPred}(C, i))$ and $\mathsf{RecRemoveBlack}(C, \mathsf{RightPred}(C, i))$,
- $\mathsf{RecRemoveBlack}(C, i)$: This is the same as $\mathsf{RecPutBlack}$, except that instead of putting a black pebble on gate i, in steps 1 and 3, we remove it.

To analyze the correctness of this strategy, we note the following invariants: if the circuit C is in a configuration where it does not contain any pebbles at any level below that of gate i, then (1) the procedure $\mathsf{RecPutBlack}(C, i)$ results in a configuration where a single black pebble is added to gate i, but nothing else changes, (2) the procedure $\mathsf{RecRemoveBlack}(C, i)$ results in a configuration where a single black pebble is removed from gate i, but nothing else changes. Using these two invariants the correctness of of the entire strategy follows.

To calculate the number of black pebbles used and the number of moves that the above strategy takes to pebble C, we use the following simple recursive equations. Let $\#\mathsf{PebPut}(d)$ and $\#\mathsf{PebRem}(d)$ be the number of black pebbles on gate i and below it used to execute $\mathsf{RecPutBlack}$ and $\mathsf{RecRemoveBlack}$ on a gate at level d, respectively. We have,

$$\#\mathsf{PebPut}(1) = 1, \quad \#\mathsf{PebPut}(d) \leq \max(\#\mathsf{PebPut}(d-1), \#\mathsf{PebRem}(d-1)) + 2$$
$$\#\mathsf{PebRem}(1) = 1, \quad \#\mathsf{PebRem}(d) \leq \max(\#\mathsf{PebPut}(d-1), \#\mathsf{PebRem}(d-1)) + 2$$

Therefore the strategy requires at most $2d$ black pebbles to pebble the circuit.

To calculate the number of moves it takes run $\mathsf{Pebble}(C)$, we use the following recursive equations. Let $\#\mathsf{Moves}(d)$ be the number of moves it takes to put a black pebble on, or remove a black pebble from, a gate at level d. Then

$$\#\mathsf{Moves}(1) = 1, \quad \#\mathsf{Moves}(d) = 4(\#\mathsf{Moves}(d-1)) + 1$$

Hence, each call of $\mathsf{RecPutBlack}$ takes at most 4^d moves, and the total number of moves to pebble the circuit is at most $q4^d$. In summary, the above gives us a strategy to pebble any circuit with at most $\gamma = q4^d$ moves and $t = 2d$ black pebbles.