On Garbled Circuits and Constant Round Secure Function Evaluation*

Stephen R. Tate Ke Xu

Department of Computer Science
University of North Texas
Denton, TX 76203

Abstract

In this paper, we examine a form of garbled (or encrypted) circuit introduced by Beaver, Micali, and Rogaway as part of their design of a constant-round secure function evaluation (SFE) protocol [5]. We show that a subtle flaw in their construction allows even a simple passive adversary (also known as an "honest-but-curious adversary") to discover private data when evaluating such a garbled circuit. In particular, information leaks from the garbled circuit at places where multiple gates share a common input wire, and is extracted by exploiting dependencies between the gate labels of the multiple gates that share that input wire. In addition to showing how this flaw manifests itself and how it can be exploited, we pinpoint the errors in the corresponding security proof [19]. Finally, we introduce a new type of gate called a "splitter" which corrects the security flaw by removing all instances of shared input wires, and using this we can correct the problems in the proof as well, giving a secure garbled circuit. This corrected circuit can be substituted for the original construction in applications such as the constant round SFE protocol of Beaver, Micali, and Rogaway and secure mobile agent protocols such as described by Algesheimer *et al.* [1]

Keywords: Secure Function Evaluation, Garbled Circuits, Encrypted Circuits

1 Introduction

A central problem in modern cryptography is the design of secure distributed protocols for multiple parties to jointly evaluate a given function. This problem, which is generally referred to as secure function evaluation (SFE), consists of n parties, $n \geq 2$, each of whom possesses a private input x_i that it wants to keep secret. The n parties want to collaboratively evaluate a function $f(x_1, \ldots, x_n)$. A SFE protocol enables them to do so both correctly and securely without the help of any third party, and in the end each party would obtain essentially the same information as it would if all the parties had sent their inputs secretly to a trusted party who evaluates the function f on these inputs and returns the desired output directly to the parties. That is, each party cannot learn more about the other parties' private inputs than what is revealed by the output of the function.

The pursuit of a general solution to secure function evaluation which is suitable for any function was pioneered by Yao [21] for the two-party case, and by Goldreich, Micali, and Wigderson [14, 15] for the multiparty case. Subsequent work has provided protocols for various communication models as well as against different adversarial behavior. Some representative work includes [6, 12, 18, 5,

*This research is supported in part by NSF award 0208640.

19, 8, 10, 11]. Meanwhile, satisfactory definition of security for secure multiparty computation has been developed due to [16, 2, 3, 9].

1.1 Garbled Circuits

The idea of "garbled circuits," first described by Yao [21], is central to protocols for general secure function evaluation. Garbled circuits are encrypted versions of standard bounded fan-in boolean circuits in which the normal boolean values are replaced by encrypted signals, and the gates provide enough information to obliviously evaluate the circuit gate-by-gate to produce a garbled output (more details are given below).

Unfortunately, while Yao introduced the general idea, his paper contains no details on how such circuits are constructed. To fill this gap, subsequent authors have given details on how garbled circuits can be constructed. Goldreich, Micali, and Wigderson [14] and Beaver [4] both describe what they describe as Yao's construction, so we will refer to this particular circuit construction as the Yao circuit construction. Naor, Pinkas, and Sumner [17] give a slightly different construction, and we will refer to this construction as the NPS circuit construction. Finally, Beaver, Micali, and Rogaway [5] give yet a different construction, which we refer to as the BMR construction. The problems we discuss in this paper are specific to the BMR construction, and not to the idea of garbled circuits in general or to the other constructions. However, this particular construction is the only one which is specifically designed for constant round multiparty SFE, so correcting the flaw in this particular construction corrects security problems in the constant round multiparty SFE problem. We comment more fully on the applicability of the Yao and NPS constructions in a following section, "Our Results, in Context."

1.1.1 Two party SFE

The garbled circuit idea as introduced by Yao [21], as well as the NPS construction, is designed specifically for the two-party setting. In this setting there are two parties, A and B, holding private values x and y, respectively. The goal is for B to learn the output of a function $f(x, y)$ while learning nothing about A's input x other than what follows from the computed value $f(x, y)$. In order to accomplish this, we first represent the function f with a bounded fan-in boolean circuit, and then form the garbled version of the circuit which then can be evaluated by B (after B learns the garbled versions of x and y). For simplicity we assume the fan-in of every gate in the circuit is at most 2.

In a garbled circuit, each wire is associated with a pair of random binary strings called *wire signals*, corresponding to the 0 and 1 bit values in the boolean circuit, which are known as the *semantics* of the signals since this reflects the meaning of the signal in the "cleartext" circuit. The correspondence between the wire signals and their semantics is random and is kept secret. Each two-input gate in the garbled circuit consists of four *gate labels* presented at random order, corresponding to the four possible input values. Holding the left and right input wire signals, to evaluate a gate, one first locates the correct gate label, then applies some computation on the gate label and the input signals to obtain the output wire signal. The gate labels are constructed in a way that respects the functionality of the gate. That is, the semantics of the two input signals and the output signal obtained from the above computation complies to the gate functionality. Given a garbled circuit and the wire signals for a particular instance of input, one can obliviously evaluate the garbled circuit without learning anything about the input, output, or the internal states. Based on this, Yao's two-party protocol consists of three stages. First, player A creates a garbled circuit for computing the desired function. In the second stage, A sends the garbled circuit as well as the wire signals for its input to player B, and also reveals the semantics of the signals for the

output wires. Then A and B engages in a 1-out-of-2 oblivious transfer protocol so that B receives the wire signals for its input but nothing else, and A learns nothing about which signals B has received. In the final stage, B independently evaluates the garbled circuit with both parties' input signals, obtains the output signals and recovers their semantics. As stated, this is secure against passive adversaries, but this can be made secure against active adversaries as well with appropriate zero-knowledge proofs showing that the players are behaving properly.

1.1.2 Multiparty SFE

While the scenario just described works well in the two-party setting, it does not scale up well to n parties. In fact, while the Goldreich, Micali, and Wigderson paper [14] was cited above for its description of the two-party circuit construction, the primary contribution of their paper is a multi-party SFE protocol that uses the two-party construction as one piece of the larger protocol. Their approach, which we refer to as the GMW multi-party protocol, is based on n-way secret sharing.

First, each of the n players shares its private input among all the players. As a result, each player holds a piece of the shared n inputs. Next, all the players collaboratively evaluate the circuit for computing the desired function in a gate-by-gate manner, from the input gates to the output gates[1]. With each player holding its share of the values entering a gate, the idea is to have the player obtain its share of the output value of that gate. This can be done by a series of privacy-preserving two-party computations, which are in turn performed using Yao's two-party method. Finally, the shares of the output are combined to get the final result. Since evaluating each gate potentially requires interaction among all the players, the GMW protocol requires unbounded rounds of communication, as do other protocols which follow the same paradigm, such as [6, 12, 18].

Beaver, Micali, and Rogaway [5] show that multi-party SFE in the case of honest majority can actually be done in a constant number of communication rounds, and essentially uses the earlier multiparty protocols to construct a garbled circuit that computes the desired function and the garbled forms of the parties' inputs. After this multiparty protocol execution, the garbled circuit and inputs are known by each player, and then each player evaluates the circuit without further interaction. We will refer to this protocol as the BMR protocol. The form of the garbled circuit constructed by the BMR protocol is carefully designed so that most of the construction is done locally by each individual player, and the whole circuit is easily assembled from the individual pieces. As a result, the construction requires only constant rounds of communication. To guard against malicious adversaries, the players have to commit their inputs, the random bits used by each player have to be jointly generated and committed, and each player has to prove that its computation is done correctly. All these can be done within constant rounds of communication, so the overall round complexity is constant.

1.2 The Security Flaw of the BMR Construction

As mentioned above, the constant round complexity of the BMR protocol comes from the carefully designed garbled circuit whose construction requires only a constant number of communication rounds. Recall that the 0 and 1 values for each wire are represented in the garbled circuit by random, independently chosen binary strings called wire signals, and a way is given so that for each gate the correct output signal can be computed from the input signals using supplemental information called "gate labels". A subtle point, which results in the flaws in the BMR circuit

[1]Note that the original GMW paper converts the circuit to a straight-line program, and then securely evaluates this, but the technique can be understood without this additional step.

construction, is that while the wire signals are independent, the gate labels are *not*. In particular, all gates which share a particular wire have some degree of correlation between them, whether it is a correlation between the gate which produces a signal as output and a gate which uses the signal as input, or between multiple gates which share a common input wire. The first of these correlations causes a technical problem in the security proof of the BMR protocol as presented in Rogaway's thesis [19], which is easily corrected and done later in this paper. The second of these correlations is more serious, and is the basis of a technique which can compromise the privacy of players' private inputs in many circuits. As such, this requires a correction to the circuit construction, and our approach involves making easy modifications to the circuit wherever a gate has fan-out greater than one. Note that any interesting circuit will have such gates, because if all wires (including input wires) are restricted to being used only once, then we are restricted to computing only read-once boolean formulas — a serious limitation to what can be evaluated.

The BMR garbled circuit construction can be used back in Yao's original two-party protocol, which gives the design more general use than just for the multi-party SFE protocol. Yao's protocol has its own applications, one of the most recent being secure mobile agent computation [7, 1, 20]. To protect mobile agents against malicious hosts, the critical part of the agent code can be implemented as a garbled circuit. In the protocol proposed by Algesheimer, Cachin, Camenisch, and Karjoth [1], the originator of the mobile agent creates a garbled circuit for each host, and uses a trusted third party to implement the oblivious transfer for a host to receive its input signals. It's obvious that the garbled circuit design of the BMR protocol can be used here, and in fact, these papers refer to [19] as a detailed description of the garbled circuit technique (as does some other literature [13]). The only difference is that here the garbled circuits are created by one party, the originator, instead of multiple parties. Because the security flaw has nothing to do with the multiparty setting but is inherent to the circuit design, all protocols which adopt this design would suffer from it. Therefore, we believe correcting this flaw has important and wide-ranging consequences.

1.3 Our Results, in Context

We correct the flaw in the BMR circuit construction by introducing a new type of gate, which we call a *splitter*. This gate has a single input and two outputs, with both outputs being independently coded versions of the input value. Trees of splitters can be built to accommodate any desired fan-out. Obviously this is completely uninteresting in a standard (non-garbled) boolean circuit, but allows us to create a circuit in which each output drives only a single input. Because of this property, the gate labels have more independence than in the BMR construction, and in particular the second type of inter-gate correlation mentioned above completely disappears. In addition to describing this solution, we'll also prove that the new design meets the standard security goals.

How do our results relate to other garbled circuit constructions and results? Both of the other two garbled circuit constructions mentioned above (the Yao and NPS constructions) do not seem to contain the same weakness as the BMR construction, and hence can't be exploited in the same way. However, both of these constructions are designed specifically for the two-party setting, so don't apply directly to the multi-party case. It is not clear that the Yao construction can even be adapted to directly work in the multi-party case — in particular, note that the original GMW protocol [14] was specifically addressing the multi-party SFE problem using Yao's circuit construction, and used an involved construction built up from multiple two-party computations.

On the other hand, the NPS circuit construction has more similarities with the BMR construction, and we believe that the techniques of Beaver, Micali, and Rogaway could be applied to this construction to make it work within a protocol for multiparty SFE. Furthermore, due to a feature of the NPR construction (specifically, the use of pseudorandom functions whose parameters change

for each gate in the circuit) this approach may provide an alternative fix for the problems with the BMR construction that we identify in this paper — in fact, our own first attempt at correcting the BMR construction was precisely along these lines, but we changed to the use of splitters due to the proof of security. In particular, by using splitters we are able to only slightly modify the security proof for the original BMR construction [19], fixing the subtle bugs while keeping the bulk of the proof that was correct. Unfortunately, the NPR approach would require a significantly different approach to the security proof, as the gate-by-gate substitution used in the proof no longer works due to the problem of making a gate constructed from a pseudo-random sample with an unknown seed properly consistent with other pseudo-random gates that share the same input signal (this is the consistency problem that we address more fully in Section 3.1). And since we are not aware of any formal security proof for the NPR protocol, such a proof would have to be built from scratch.

Summarizing the preceding discussion, the advantages to our approach of correcting the BMR protocol are that we directly deal with a solution to the multi-party SFE problem rather than having to adapt a two-party solution, and we are able to give a solid security proof. Furthermore, clearly identifying the subtle problem of correlations between gate labels provides valuable insight for people designing such constructions in the future, and the general lessons learned from this are important.

The rest of the paper is organized as follows. Section 2 presents the BMR garbled circuit construction in detail and demonstrates the security flaw with an example. In Section 3 we summarize Rogaway's proof of the security and identify two places where the proof breaks down (corresponding to the two types of inter-gate correlations mentioned earlier). Section 4 presents our correction and proves the security of the new design using the basic outline from Rogaway's proof method. Section 5 gives conclusions.

2 The BMR Construction

The BMR protocol [5, 19] assumes existence of private channels among the players as well as a broadcast channel. Its security is based on the basic assumption that a one-way function exists, and hence that a secure pseudorandom number generator exists. The protocol is claimed to be computationally secure against an adaptive, active adversary which corrupts strictly less than half of the parties. In order to limit its round complexity to a constant, the BMR protocol is composed of two phases.

- In Phase I, the n players collaboratively construct a garbled circuit and the garbled inputs corresponding to their private inputs. The joint computation can be done through one of the earlier information-theoretically secure multiparty protocols [6, 12, 18]. The constructed garbled circuit and inputs exist as shared information among the players in such a way that no coalition of t or fewer players (where $t < n/2$) is able to recover the shared information from their pieces.

- In Phase II, the garbled circuit and the garbled inputs are publicly revealed by the players and each player gets a copy of both. Also revealed are the semantics of the signals for the output wires. Semantics of the signals for other wires are kept secret from all players. After this, each player evaluates the garbled circuit with the garbled inputs and determines the semantics of the output, with no further communication required.[2]

[2]This definition allows only a single shared output to be computed, but it can be modified to produce private outputs for each player if that is needed — see the original SFE papers for details.

Since the security flaw is in the design of the garbled circuit, not how it is constructed, we only describe the form of the garbled circuit and the garbled inputs. More precisely, what we describe is what each player gets in Phase II before it starts the non-interactive evaluation. For details related to the joint construction and the constant round complexity, the readers are referred to the original papers.

2.1 The Garbled Circuit and The Garbled Inputs

We summarize the definitions of the garbled circuit and the garbled inputs from the original paper [5]. Suppose there are n parties, and let k be a security parameter, where we require that n be bounded by a polynomial in k (for example, we could require that $n \leq k^{10}$). For simplicity of presentation, suppose the desired function f takes n ℓ-bit inputs and outputs a single ℓ-bit value, that is $f : (\Sigma^\ell)^n \longrightarrow \Sigma^\ell$, where $\Sigma = \{0, 1\}$. Note that in parts of their presentation Beaver, Micali, and Rogaway consider only boolean functions, so the output is a single bit. As they note, ℓ-bit output is an obvious generalization, and in fact they state their final result for an even more general situation in which the output is from $(\Sigma^\ell)^n$ (so is an n-tuple of ℓ-bit values). The results in this paper apply just as well in the more restricted setting of boolean functions, as the flaws we identify can be exploited on sub-circuits, even if the full circuit has a single-bit output. For example, the specific function described in Section 2.2 to demonstrate the security flaw could be changed from computing the larger of two numbers to computing the parity of the bits in the larger of two numbers (a boolean function). Then the flaw is exploited easily on the subcircuit that corresponds to the circuit shown in Figure 1.

Let C be a boolean circuit that computes f using Γ gates, where each gate has fan-in at most 2. There are W wires, and we typically use small Greek letters to represent wire numbers $0, \ldots, W-1$. Wires $0, \ldots, n\ell-1$ are the inputs, while wires $W-\ell, \ldots, W-1$ are the outputs. In a garbled version of C, each wire has associated with it a pair of *signals*, which are nk-bit strings. The signals are numbered in the same order as the wires, so if α is a wire then the two signals associated with this wire are $s_{2\alpha}$ and $s_{2\alpha+1}$. We often define a signal base value, for example $a = 2\alpha$, so the notation is simpler, giving the signal pair (s_a, s_{a+1}). Each signal has an associated *semantics* that corresponds to the signal's "plaintext" value, with one signal of a pair having semantics 0 and the other having semantics 1. For a wire α, a semantics variable $\lambda_\alpha \in \{0, 1\}$ is chosen randomly and indicates that signal s_a has semantics λ_α while signal s_{a+1} has semantics $\overline{\lambda_\alpha}$. The *garbled inputs* are the signals of the input wires which correspond to the actual input bits, so for example if $b_\omega \in \{0, 1\}$ is the plaintext input bit for wire $\omega \leq n\ell - 1$, then the input signal $\sigma_\omega = s_{2\omega+(b_\omega \oplus \lambda_\omega)}$ is the garbled input for this bit. When the garbled circuit is evaluated using the garbled inputs, each wire takes on the value of exactly one of its signals, which is denoted σ_α for wire α. Note that since all semantics variables are randomly chosen and kept secret (except those of the output wires), knowledge of which wire signal becomes σ_α gives no information about the actual plaintext value of the bit.

Let $\mathcal{G}$ be a pseudorandom generator that generates a $(k + 2nk)$-bit string from a k-bit seed. Define F, G, and H to be the first k, next nk, and last nk bits of the $\mathcal{G}$'s output, respectively. Therefore, $F(s) = \mathcal{G}(s)[1 : k]$, $G(s) = \mathcal{G}(s)[k + 1 : k + nk]$, $H(s) = \mathcal{G}(s)[k + nk + 1 : k + 2nk]$. In the following definitions, $\circ$ denotes the concatenation of two strings, and $\oplus$ denotes XOR.

The signals associated with a wire ω can be broken into n pieces of k bits each, so $s_{2\omega} = s^1_{2\omega} \circ \cdots \circ s^n_{2\omega}$ and $s_{2\omega+1} = s^1_{2\omega+1} \circ \cdots \circ s^n_{2\omega+1}$, where $|s^i_j| = k$ for $1 \leq i \leq n$ and $0 \leq j \leq 2W - 1$ (so superscript ranges over the n players, while the subscript ranges over the wire indices). These pieces are then used to compute the gate labels as follows. Let $f^i_j = F(s^i_j)$, $g^i_j = G(s^i_j)$, $h^i_j = H(s^i_j)$, for $1 \leq i \leq n$ and $0 \leq j \leq 2W - 1$. Suppose gate g computes the function $\otimes$. If the left and right incoming wires and the outgoing wire of gate g are α, β, and γ respectively, then writing $a = 2\alpha$,

$b = 2\beta$, and $c = 2\gamma$, the gate labels for g are

$$\begin{aligned}
A_g &= g_a^1 \oplus \cdots \oplus g_a^n \oplus g_b^1 \oplus \cdots \oplus g_b^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \lambda_\alpha \otimes \lambda_\beta = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases} \\
B_g &= h_a^1 \oplus \cdots \oplus h_a^n \oplus g_{b+1}^1 \oplus \cdots \oplus g_{b+1}^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \lambda_\alpha \otimes \overline{\lambda_\beta} = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases} \\
C_g &= g_{a+1}^1 \oplus \cdots \oplus g_{a+1}^n \oplus h_b^1 \oplus \cdots \oplus h_b^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \overline{\lambda_\alpha} \otimes \lambda_\beta = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases} \\
D_g &= h_{a+1}^1 \oplus \cdots \oplus h_{a+1}^n \oplus h_{b+1}^1 \oplus \cdots \oplus h_{b+1}^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \overline{\lambda_\alpha} \otimes \overline{\lambda_\beta} = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases}
\end{aligned} \tag{1}$$

In order to use the garbled circuit, each player learns all the gate labels and the garbled inputs, and also the *wire labels* f_j^i, for $1 \le i \le n$ and $0 \le j \le 2W - 1$. The semantics λ_ω for the circuit outputs are revealed, but all other semantics variables (for the inputs and intermediate values) remain secret. Note that the wire labels do not leak any information about their corresponding signals s_j^i. Holding signal $\sigma_\omega^1 \circ \cdots \circ \sigma_\omega^n$ for a wire ω, a player can determine whether this signal is $s_{2\omega}^1 \circ \cdots \circ s_{2\omega}^n$ if $F(\sigma_\omega^1) = f_{2\omega}^1$ (in which case we say the signal has index 0), or $s_{2\omega+1}^1 \circ \cdots \circ s_{2\omega+1}^n$ (in which case we say the signal has index 1). This information allows the player to locate the right gate label for evaluating a gate. For a gate g, if the player holds $s_{a+p}^1 \circ \cdots \circ s_{a+p}^n$ for the left incoming wire α, and $s_{b+q}^1 \circ \cdots \circ s_{b+q}^n$ for the right incoming wire β, where $a = 2\alpha$, $b = 2\beta$, and $p, q \in \{0, 1\}$ are the signal indices, and suppose the outgoing wire is γ, then the outgoing signal σ_γ is computed as

$$\sigma_\gamma = \begin{cases} g_{a+p}^1 \oplus \cdots \oplus g_{a+p}^n \oplus g_{b+q}^1 \oplus \cdots \oplus g_{b+q}^n \oplus A_g & \text{if } p = 0 \text{ and } q = 0 \\ h_{a+p}^1 \oplus \cdots \oplus h_{a+p}^n \oplus g_{b+q}^1 \oplus \cdots \oplus g_{b+q}^n \oplus B_g & \text{if } p = 0 \text{ and } q = 1 \\ g_{a+p}^1 \oplus \cdots \oplus g_{a+p}^n \oplus h_{b+q}^1 \oplus \cdots \oplus h_{b+q}^n \oplus C_g & \text{if } p = 1 \text{ and } q = 0 \\ h_{a+p}^1 \oplus \cdots \oplus h_{a+p}^n \oplus h_{b+q}^1 \oplus \cdots \oplus h_{b+q}^n \oplus D_g & \text{if } p = 1 \text{ and } q = 1 \end{cases} \tag{2}$$

It is easy to verify that equation (2) computes the correct outgoing signal according to the gate function $\otimes$, and this is done without revealing any information about the semantics of the incoming and outgoing signals. Notice that each gate label is composed of n substrings which are combined using a simple XOR, so the gate labels can be easily constructed through a multiparty protocol. When applying this design to a case where the garbled circuit is created by a single party (such as Yao's two-party protocol), the signals don't have to be divided into substrings. A single string will work. Except this, the design can be directly used in this special case without modification.

As another alternative, notice the signal index can be simplified by adding an additional bit to the signal as the index. As a result, determining which signal one is holding becomes simply inspecting the last bit of the signal. This technique was used in Rogaway's thesis [19], and has the benefit of simplifying the garbled circuit without sacrificing security.

2.2 Exploiting Inter-gate Dependencies

Before pointing out the security flaw in a general sense, let's look at an example. Suppose two players A and B each hold ℓ-bit numbers, x_1 and x_2 respectively, and want to find out the larger number of the two (this example can obviously be extended to an arbitrary number of players). After joint computation, the two players should learn the larger number, but no information about the smaller number should be revealed. Hence, if player A has the larger number, she should essentially gain no new information after the computation. We show that using the BMR protocol,

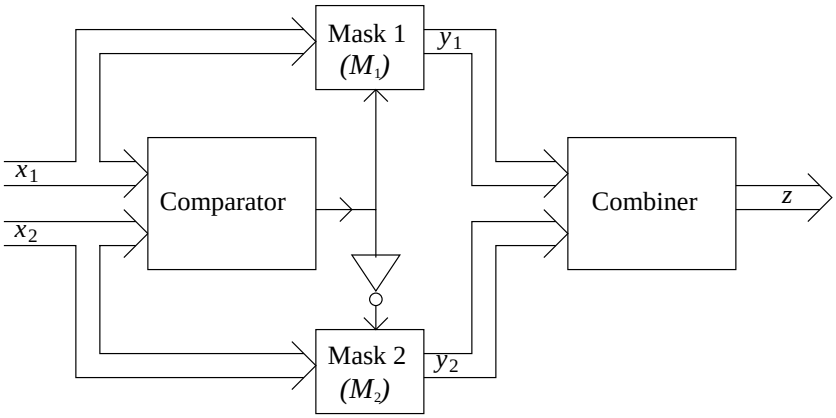

Figure 1: A circuit that computes $z = \max(x_1, x_2)$.

even an honest-but-curious A is able to learn B's number x_2 with high probability, as long as x_2 has a nontrivial number of zero bits.

Figure 1 shows a circuit that computes $z = \max(x_1, x_2)$. Each rectangle represents a subcircuit. If $x_1 \geq x_2$, the comparator outputs 1, otherwise 0. This bit serves as a selector to the mask circuits M_1 and M_2, so that if the bit is 1, then $y_1 = x_1$ and $y_2 = 0$; otherwise, $y_1 = 0$ and $y_2 = x_2$. The "combiner" subcircuit is just a bitwise OR of y_1 and y_2. Following the BMR protocol, the two players A and B first collaboratively construct a garbled version of the circuit and the garbled inputs corresponding to x_1 and x_2. Mask subcircuit M_2 simply ANDs each bit from x_2 with the selector bit (the complemented output of the comparator), and the garbled version of this subcircuit is shown in Figure 2. Suppose the input wires of M_2 are $\alpha_1, \ldots, \alpha_\ell$ for x_2 and β for the selector bit, the output wires are $\gamma_1, \ldots, \gamma_\ell$ for y_2, and the gate labels are $(A_{g_1}, B_{g_1}, C_{g_1}, D_{g_1}), \ldots, (A_{g_\ell}, B_{g_\ell}, C_{g_\ell}, D_{g_\ell})$. In Phase II of the protocol, both A and B receive a copy of the garbled circuit and inputs, and evaluate the circuit on their own.

According to equation (1), the labels A_{g_i}, B_{g_i}, C_{g_i}, D_{g_i}, for $1 \leq i \leq \ell$, are

$$\begin{aligned}
A_{g_i} &= g^1_{a_i} \oplus g^2_{a_i} \oplus g^1_b \oplus g^2_b \oplus \begin{cases} s^1_{c_i} \circ s^2_{c_i} & \text{if } \lambda_{\alpha_i} \wedge \lambda_\beta = \lambda_{\gamma_i} \\ s^1_{c_i+1} \circ s^2_{c_i+1} & \text{otherwise} \end{cases} \\
B_{g_i} &= h^1_{a_i} \oplus h^2_{a_i} \oplus g^1_{b+1} \oplus g^2_{b+1} \oplus \begin{cases} s^1_{c_i} \circ s^2_{c_i} & \text{if } \lambda_{\alpha_i} \wedge \overline{\lambda_\beta} = \lambda_{\gamma_i} \\ s^1_{c_i+1} \circ s^2_{c_i+1} & \text{otherwise} \end{cases} \\
C_{g_i} &= g^1_{a_i+1} \oplus g^2_{a_i+1} \oplus h^1_b \oplus h^2_b \oplus \begin{cases} s^1_{c_i} \circ s^2_{c_i} & \text{if } \overline{\lambda_{\alpha_i}} \wedge \lambda_\beta = \lambda_{\gamma_i} \\ s^1_{c_i+1} \circ s^2_{c_i+1} & \text{otherwise} \end{cases} \\
D_{g_i} &= h^1_{a_i+1} \oplus h^2_{a_i+1} \oplus h^1_{b+1} \oplus h^2_{b+1} \oplus \begin{cases} s^1_{c_i} \circ s^2_{c_i} & \text{if } \overline{\lambda_{\alpha_i}} \wedge \overline{\lambda_\beta} = \lambda_{\gamma_i} \\ s^1_{c_i+1} \circ s^2_{c_i+1} & \text{otherwise} \end{cases}
\end{aligned} \tag{3}$$

where $a_i = 2\alpha_i$, $b = 2\beta$, and $c_i = 2\gamma_i$. Therefore, the contribution of wire β to all output signals (through the gate labels) comes from a limited set of four possible values consisting of $g^1_b \oplus g^2_b$, $g^1_{b+1} \oplus g^2_{b+1}$, $h^1_b \oplus h^2_b$, and $h^1_{b+1} \oplus h^2_{b+1}$. This can be exploited by an adversary as follows.

Let b_β denote the plaintext bit on wire β. Assume $x_1 \geq x_2$, so that $b_\beta = 0$ which in turn makes all the output wires of M_2 0 and so masks out x_2. However, this does not prevent us from learning x_2. Essentially we are going to deduce what the output of M_2 would be if b_β were 1. Let p_i denote the index of signal σ_{α_i} (recall that σ_{α_i} is the signal held by wire α_i, so $\sigma_{\alpha_i} = s_{2\alpha_i+p_i}$). Because the signals on input wires $\alpha_1, \ldots, \alpha_\ell$ don't change when β changes, indices p_i won't change;

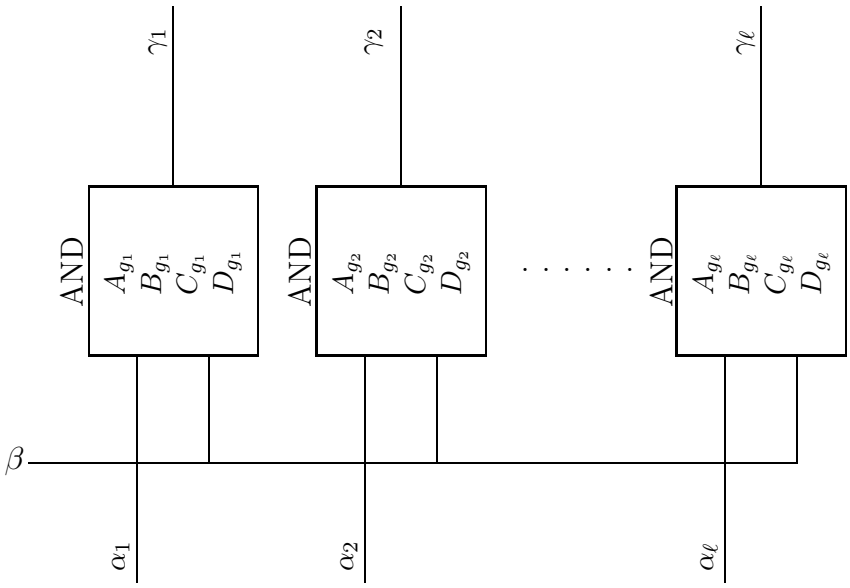

Figure 2: The garbled subcircuit $\mathcal{M}_2$. Wires α_k get the input bits from x_2, and output wires γ_k provide the bits of y_2.

furthermore, while we don't know what the signal on wire β will be when it is complemented, we *do* know its index. Therefore, we can locate the labels that would be used for evaluating the gates if b_β were 1. In addition, since σ_{α_i} is known we can compute the contribution of σ_{α_i} to the output value of each gate after β is changed. Next we make a (likely wrong) guess that all bits of x_2 are 0. If this were true, evaluating the gates with $b_\beta = 1$ would give the same result as when $b_\beta = 0$, so we in fact know the output signals because we computed them when the circuit was evaluated with the correct, legally obtained inputs. Therefore, of the four basic parts of the gate label equation when $b_\beta = 1$ we know three of them: The contribution of σ_{α_i}, the proper gate label, and the output signal σ_{γ_i}. Now we can solve equation (2) with the following results. Without loss of generality, suppose that $\lambda_\beta = 1$, so the index q for wire β is 1 when $b_\beta = 0$, and $q = 0$ when $b_\beta = 1$. Then for gate g_i,

- If p_i, the index for wire α_i, is 0, then A_{g_i} is the correct gate label to use when $b_\beta = 1$, as $p_i = q = 0$. Compute $\mu_i = \sigma_{\gamma_i} \oplus g^1_{2\alpha_i} \oplus g^2_{2\alpha_i} \oplus A_{g_i}$. If our guess was right, which means the ith bit of x_2 is actually 0, then σ_{γ_i} is the correct output signal, and so according to equation (3) $A_{g_i} = g^1_{2\alpha_i} \oplus g^2_{2\alpha_i} \oplus g^1_{2\beta} \oplus g^2_{2\beta} \oplus \sigma_{\gamma_i}$. Thus $\mu_i = g^1_{2\beta} \oplus g^2_{2\beta}$. Otherwise (if our guess was not correct), then following the equations we see that $\mu_i = s_{2\gamma_i} \oplus s_{2\gamma_i+1} \oplus g^1_{2\beta} \oplus g^2_{2\beta}$, and since the wire labels are chosen randomly and independently, μ_i is a random value in this case.
- If p_i is 1, compute $\mu_i = \sigma_{\gamma_i} \oplus g^1_{2\alpha_i+1} \oplus g^2_{2\alpha_i+1} \oplus C_{g_i}$. If our guess is right, then $\mu_i = h^1_{2\beta} \oplus h^2_{2\beta}$. Otherwise, as before it is just another random string.

Therefore, all the resulting μ_i's can be divided into 3 groups. In each of the first 2 groups, the μ_i's are equal to each other, since they are equal to either $g^1_{2\beta} \oplus g^2_{2\beta}$ or $h^1_{2\beta} \oplus h^2_{2\beta}$. These μ_i's correspond to correct guesses of the output bit, or in other words the 0 bits of x_2. Note that we have no way of knowing these values ahead of time, since we don't know the proper signal for the complement of β, but we *can* recognize subsets with equal μ_i values. The third group will be just some random strings, corresponding to the 1 bits, and the probability that these values are equal to the values

produced by the first two sets, or any other μ_i value, is very small. Thus by identifying repeated values produced by this computation we have determined x_2 with high probability. We summarize this attack in the following theorem.

Theorem 2.1 *Let M be a mask circuit as in the previous discussion that correctly masks out an ℓ-bit value x that has z zero bits, for $z \geq 3$. If the garbled wire signals have nk bits, then the attack described above correctly recovers x with probability at least $1 - \frac{z}{2^{z-2}}$, for sufficiently large k.*

The proof of this theorem is straightforward and included in the full paper. Note that while we attack specific masking circuits, similar techniques could apply in many other situations where multiple gates share a common input.

3 The Flaw in the Proof

Rogaway [19] presented a proof of security for the BMR protocol. Clearly, because the attack described in the last section exists, the proof is not valid, and in this section we identify the flaws in the proof. Rogaway's proof is extremely intricate, but the vast majority of the proof deals with oracle constructions that address how active adversaries interact with the secure function evaluation protocol. Fortunately for us, the flaws are in a more basic portion of the proof, so we simplify things greatly here by looking at a related problem which turns out to be one piece in building up the entire proof. While our simplifications sweep many details under the rug, the ideas presented here translate directly into the full proof for secure function evaluation.

We can construct a "fake" garbled circuit given just knowledge of the output of f as follows: Random signals are assigned to all wires, and the semantics are defined such that the semantics of the output signals match the desired output of the function f, and all non-output signals have unknown semantics. Each signal is made to have a random index in the pair of signals for that wire. For any gate, the gate labels are constructed by finding the one label that corresponds to the indices of its input signals[3] and correctly setting that label to produce the previously selected output signal. The other three labels are set to random values. The correct gate labels are called the *on-path* labels, and other than making a consistent path through the circuit they have no real meaning (notice that the definition doesn't even depend on the functionality of the gate or the semantics of the signals). Noticing that these fake circuits can be constructed from just knowledge of the output and the structure of the circuit, it is clear that this fake circuit reveals no information about private inputs or intermediate values of the computation (since none exist!). The goal is to show that an ensemble of distributions (indexed by k, n, and C) of these fake circuits and fake inputs is computationally indistinguishable from an ensemble of real garbled circuits and real garbled inputs. If this is true, then no more information can be obtained from a real garbled circuit than the fake one, which is to say that no information can be obtained other than what follows from the output value.

Denote the distributions of fake garbled circuits and the real garbled circuits as $\tilde{\mathcal{C}}$ and $\mathcal{C}$ respectively. To prove their indistinguishability, Rogaway makes a sequence of hybrid garbled circuit distributions as

$$\tilde{\mathcal{C}} = \mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \cdots \Rightarrow \mathcal{C}_{\Gamma-1} \Rightarrow \mathcal{C}_\Gamma = \mathcal{C}.$$

For $\mathcal{C}_i$, real gate labels (for all 4 labels) are generated as in $\mathcal{C}$ for gates numbered $1, \ldots, i$, and the remaining gates are fake (only the one on-path label, which is required for proper computation of

[3]Since the indices are random, the gate label selected will also be random.

the output, is correctly computed from pseudorandom strings, while the remaining labels are truly random strings). It is obvious that at the endpoints of this sequence, $\mathcal{C}_0 = \tilde{\mathcal{C}}$ and $\mathcal{C}_\Gamma = \mathcal{C}$.

We wish to show that $\tilde{\mathcal{C}}$ and $\mathcal{C}$ are computationally indistinguishable, so assume for the sake of contradiction that this is not true and that a polynomial time distinguisher for these two distributions exists. It immediately follows that somewhere in the above sequence there exists a non-negligible "jump" in the distinguishing probabilities. Let g_k denote the jumping point, meaning that distributions $\mathcal{C}_{g_k-1}$ and $\mathcal{C}_{g_k}$ are distinguishable. The strategy is to make use of this distinguisher to construct another polynomial-time distinguisher which is able to distinguish a truly random string from a pseudorandom string. This contradicts the assumption that the pseudorandom generator $\mathcal{G}$ is secure.

Definition 3.1 *We define the two probability distributions:*

- U_{2nk+k} *is the uniform distribution on length* $2nk+k$ *binary strings;*
- $G(U_k)$ *is the distribution of length* $2nk+k$ *binary strings that are produced by pseudorandom generator* $\mathcal{G}$ *on uniformly selected seeds of length* k*.*

Consider two $(2nk+k)$-bit strings, X and Y, which either both come from U_{2nk+k} or both come from $G(U_k)$. Our goal is to create a distinguisher for these distributions. Define four substrings $X_g = X[k+1 : k+nk]$, $X_h = X[k+nk+1 : k+2nk]$, $Y_g = Y[k+1 : k+nk]$, and $Y_h = Y[k+nk+1 : k+2nk]$ — note that if X and Y are pseudorandom these correspond to precisely the pseudorandom substrings produced by the $G(s)$ and $H(s)$ functions, which were defined earlier.

We use these strings to define a new circuit distribution $\mathcal{C}^{X,Y}$ which involves constructing as in $\mathcal{C}_{g_k}$ but substituting the substrings into the off-path signal contributions to gate g_k. It is not clear how Rogaway had intended to do this in his proof because there are two possibilities: either only the labels in gate g_k are changed (a theory supported by Rogaway's statement that labels in $\mathcal{C}^{X,Y}$ are "computed differently ... but just for the case of g_k"), or all gates which share these off-path signals change (this alternative is supported by the actual formulas given for computing the gate labels). While there is some ambiguity on what is intended, we take the latter option for several reasons, the most important being that the first interpretation creates additional problems in the proof, since the real gates don't have the necessary inter-gate dependencies unless substitutions are made in all the gates sharing input wires.

Referring back to equation (1), if the on-path signal indices for the two input wires α and β are p and q, then we call g^j_{a+p}, h^j_{a+p}, g^j_{b+q} and h^j_{b+q} "player j's on-path contributions to gate g_k," while $g^j_{a+(1-p)}$, $h^j_{a+(1-p)}$, $g^j_{b+(1-q)}$, and $h^j_{b+(1-q)}$ are "player j's off-path contributions to gate g_k." In order to create a circuit from distribution $\mathcal{C}^{X,Y}$, we create the circuit just like $\mathcal{C}_{g_k}$ except that we randomly select a player j and then substitute X_g, X_h, Y_g, and Y_h in place of the off-path contributions to g_k. Note that the random selection of player j is important in the original proof due to the model of the adversary used there, but is not really needed here (we could always use player 1, for instance). However, we keep the random selection of j anyway in order to simplify translation back to the larger context of the original proof.

As an illustration, suppose that for gate g_k, the on-path indices are $p = q = 0$, so A_{g_k} is the on-path gate label used in constructing the output signal. Further suppose that the randomly chosen player is $j = 1$. Then the gate labels for gate g_k in $\mathcal{C}^{X,Y}$ are calculated as follows:

$$
\begin{aligned}
A_g &= g_a^1 \oplus \cdots \oplus g_a^n \oplus g_b^1 \oplus \cdots \oplus g_b^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \lambda_\alpha \otimes \lambda_\beta = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases} \\
B_g &= h_a^1 \oplus \cdots \oplus h_a^n \oplus Y_g \oplus \cdots \oplus g_{b+1}^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \lambda_\alpha \otimes \overline{\lambda_\beta} = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases} \\
C_g &= X_g \oplus \cdots \oplus g_{a+1}^n \oplus h_b^1 \oplus \cdots \oplus h_b^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \overline{\lambda_\alpha} \otimes \lambda_\beta = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases} \\
D_g &= X_h \oplus \cdots \oplus h_{a+1}^n \oplus Y_h \oplus \cdots \oplus h_{b+1}^n \oplus \begin{cases} s_c^1 \circ \cdots \circ s_c^n & \text{if } \overline{\lambda_\alpha} \otimes \overline{\lambda_\beta} = \lambda_\gamma \\ s_{c+1}^1 \circ \cdots \circ s_{c+1}^n & \text{otherwise} \end{cases}
\end{aligned}
$$

Note that this substitution leaves all the on-path labels the same as before, and consistent throughout the circuit. If we ignore issues of inter-gate dependencies, and if X and Y are $G(U_k)$-distributed, then the labels in gate g_k are made from pseudorandom strings just as they would be in a real garbled gate. Furthermore (again ignoring inter-gate dependencies), if X and Y are U_{2nk+k}-distributed then each of the off-path gate labels is XORed with a truly random string from X or Y, so the off-path gate labels should be random just as they would be in a fake gate. This leads to the following two claims in Rogaway's proof [19], with notation adjustments to match our presentation:

Claim 11: If X and Y are $G(U_k)$-distributed, then $\mathcal{C}^{X,Y} = \mathcal{C}_{g_k}$.
Claim 12: If X and Y are U_{2nk+k}-distributed, then $\mathcal{C}^{X,Y} = \mathcal{C}_{g_k-1}$.

Clearly, since we have a distinguisher for $\mathcal{C}_{g_k}$ and $\mathcal{C}_{g_k-1}$, these claims would imply that we can use this distinguisher to create a polynomial time distinguisher for $G(U_k)$ and U_{2nk+k}, which would complete the proof by contradiction, thus proving the security of this construction. The claims seem likely enough, but unfortunately, as presented, both claims are false due to two different types of inter-gate dependencies that aren't taken into account. We correct Claim 11 below, but postpone the correction of Claim 12 until the next section where we fix the problems in the circuit construction.

3.1 Correcting Claim 11

The problem with both claims is that gate labels are not independent of other gate labels in the circuit: the gate labels of gate g are correlated with the labels in the gates whose output provide the inputs to g (this is the failing of Claim 11), and are correlated with other gates that share the same input wire (this is the failing of Claim 12). In particular, for $\mathcal{C}^{X,Y} = \mathcal{C}_{g_k}$ it is not sufficient that the labels on gate g_k be based on a pseudorandom number generator, but that pseudorandom number generator *must* use as a seed the values provided by the gate whose output feeds it, even for the off-path labels (since the first g_k gates must form a subcircuit of a "real" garbled circuit). In other words, to properly substitute the substrings of X and Y, we would need to modify the predecessor gates to output appropriate seeds to generate X and Y. Since we don't know the seeds for X and Y this is clearly not possible.

Fortunately, this is not a hard problem to correct, and can in fact be corrected without changing the circuit construction. First, let's put a more specific ordering on the gates: The gates of the circuit are labeled in the order of a valid topological sorting, so the gates providing input to a gate g always have lower gate numbers than g. We define a new "walk" from $\tilde{\mathcal{C}}$ to $\mathcal{C}$ as

$$\tilde{\mathcal{C}} = \overline{\mathcal{C}}_0 \Rightarrow \overline{\mathcal{C}}_1 \Rightarrow \cdots \Rightarrow \overline{\mathcal{C}}_{\Gamma-1} \Rightarrow \overline{\mathcal{C}}_\Gamma = \mathcal{C},$$

where $\overline{\mathcal{C}}_i$ consists of fake gate labels in gates $1, \ldots, \Gamma - i$ and real gate labels in the remaining gates. The difference with the earlier walk is that we now change fake gates to real gates starting from

the outputs and working back to the inputs, rather than vice versa. Define i_k to be a position in this sequence where $\overline{\mathcal{C}}_{i_k-1}$ is distinguishable from $\overline{\mathcal{C}}_{i_k}$, and let $g_k = \Gamma - i_k + 1$ so that g_k is the gate whose change from fake to real enables the distinguisher to detect the difference. $\overline{\mathcal{C}}^{X,Y}$ is defined similarly to $\mathcal{C}^{X,Y}$, but with respect to the $\overline{\mathcal{C}}_i$ distributions.

Lemma 3.1 *If X and Y are $G(U_k)$-distributed, then $\overline{\mathcal{C}}^{X,Y} = \overline{\mathcal{C}}_{i_k}$.*

PROOF SKETCH: The subcircuit of $\overline{\mathcal{C}}^{X,Y}$ made up of gates $g_k, \ldots, \Gamma$ now makes up a subcircuit of a real garbled circuit — the only unusual gate is gate g_k, but since only the pseudorandom strings themselves (and not the seeds) affect later gate labels, the later gates are easily made consistent. Predecessor gates have fake labels so no correlation with predecessor gates is needed or desired for the off-path gate labels. Since even in $\overline{\mathcal{C}}_{i_k}$ the seeds producing the pseudorandom strings affecting the off-path gate labels of gate g_k are not used for anything, our ignorance of what seeds generated X and Y is unimportant. ∎

To summarize, we have removed the problem of correlations between labels in gate g_k and in the gates providing the input values to g_k by converting fake gates to real gates in a different order, thereby removing the correlations in the hybrid circuits. Unfortunately, no such fix is possible for Claim 12, as correlations between gates sharing the same input wire simply can't be removed from the hybrid circuit. Because of this, we modify the main circuit construction in the next section so that these correlations no longer exist.

4 Using Splitters to Correct the Problem in the Construction

The correction we need to make to the circuit construction is straightforward: we add a "splitter" gate which has a single input and two outputs. In the garbled version, the two outputs will have independent pairs of wire signals, effectively decorrelating the gates which use these shared input values. We build up trees of splitters to accommodate arbitrary fanout.

A splitter simply copies the value from the input to each output. In the garbled version, there are four labels just as in the standard logic gates. Let the input wire be α, the two output wires be γ and δ, and let $a = 2\alpha$, $c = 2\gamma$, and $d = 2\delta$. Then the output signal pairs for the two output wires are (s_c, s_{c+1}) and (s_d, s_{d+1}), and the gate labels are defined as follows:

$$A_g = g_a^1 \oplus \cdots \oplus g_a^n \oplus \begin{cases} s_c & \text{if } \lambda_\alpha = \lambda_\gamma, \\ s_{c+1} & \text{otherwise;} \end{cases}$$

$$B_g = g_{a+1}^1 \oplus \cdots \oplus g_{a+1}^n \oplus \begin{cases} s_c & \text{if } \overline{\lambda_\alpha} = \lambda_\gamma, \\ s_{c+1} & \text{otherwise;} \end{cases}$$

$$C_g = h_a^1 \oplus \cdots \oplus h_a^n \oplus \begin{cases} s_d & \text{if } \lambda_\alpha = \lambda_\delta, \\ s_{d+1} & \text{otherwise;} \end{cases}$$

$$D_g = h_{a+1}^1 \oplus \cdots \oplus h_{a+1}^n \oplus \begin{cases} s_d & \text{if } \overline{\lambda_\alpha} = \lambda_\delta, \\ s_{d+1} & \text{otherwise.} \end{cases}$$

Then in the evaluation phase, the output signals are calculated as follows:

$$\sigma_\gamma = \begin{cases} g_a^1 \oplus \cdots \oplus g_a^n \oplus A_g & \text{if } p = 0; \\ g_{a+1}^1 \oplus \cdots \oplus g_{a+1}^n \oplus B_g & \text{if } p = 1. \end{cases}$$

and

$$\sigma_\delta = \begin{cases} h_a^1 \oplus \cdots \oplus h_a^n \oplus C_g & \text{if } p = 0; \\ h_{a+1}^1 \oplus \cdots \oplus h_{a+1}^n \oplus D_g & \text{if } p = 1. \end{cases}$$

It is not difficult to see that this properly reflects the semantics of what we are trying to do, and results in two independent codings of the input value as signals on the outputs.

4.1 Proving The Security of The New Design

First, we will be specific about the flaw in Claim 12 from Rogaway's proof. Assume that gate g_k shares an input wire with many other gates, and further assume that several of these other gates are on the "real side" of the circuit (so have already been converted from fake gates to real ones). When we switch to constructing a circuit from $\overline{\mathcal{C}}^{X,Y}$ by substituting substrings from X and Y into the off-path contributions to gate g_k, we must also make substitutions into all the other "real" gates that share an input wire (if we did not do so, then there would be serious problems with Claim 11, as pointed out in the previous section). However, this means that even if X and Y are U_{2nk+k}-distributed random strings, they are *not* independent of other gate labels in the circuit, and so this distribution of circuits is in fact different from $\overline{\mathcal{C}}_{i_k-1}$. While not immediately obvious, it becomes clear with a minimal amount of work that this is precisely the set of dependencies that we exploited in compromising the BMR construction in Section 2.2.

In the new construction, using splitters to avoid any direct sharing of input wires, we must extend the construction for $\overline{\mathcal{C}}^{X,Y}$ in order to accommodate this new type of gate. Unlike standard logic gates, a splitter will actually have *two* on-path gate labels, corresponding to the two outputs that must be computed. To construct a circuit from distribution $\overline{\mathcal{C}}^{X,Y}$ when gate g_k is a splitter, we simply substitute into the two off-path labels in the obvious way, using only X_g and X_h. Note that string Y is not used at all in this case, but that is not important — X and Y are just two samples from the same distribution, and there is no requirement that the distinguisher use all strings available as input. We now have corrected Claim 12, which we restate here as a lemma for the corrected version.

Lemma 4.1 *If X and Y are U_{2nk+k}-distributed, then $\overline{\mathcal{C}}^{X,Y} = \overline{\mathcal{C}}_{i_k-1}$.*

PROOF SKETCH: As in our discussion of the original "Claim 12", if X and Y are U_{2nk+k}-distributed then each off-path label in g_k is XORed with a random value, so the off-path label is itself truly random. The difference with the earlier discussion is that now the inputs of any gate are not shared with any other gate, so the question of whether to substitute in only gate g_k or in all gates that share an input wire is irrelevant. The substitution is not used anywhere other than in gate g_k, so the off-path labels are not only truly random, but they are independent of all other gate labels in the garbled circuit. Therefore, distributions $\overline{\mathcal{C}}^{X,Y}$ and $\overline{\mathcal{C}}_{i_k-1}$ are identical. ∎

We now combine Lemma 3.1 and Lemma 4.1 in the obvious way to get the following theorem.

Theorem 4.1 *The circuit construction from BMR, modified with the use of splitter gates as described in this paper so that the fanout of all gates is at most one, is secure against a passive, honest-but-curious adversary.*

Placed into the context of Rogaway's larger proof of security against active, adaptive adversaries, we get a corrected version of the constant round secure function evaluation protocol. We restate the security theorem from the Beaver, Micali, Rogaway paper [5] here for completeness.

Theorem 4.2 *Assume the standard model, $t < n/2$, $f : (\Sigma^\ell)^n \to (\Sigma^\ell)^n$ a function realized by a circuit C. Assume that a secure pseudorandom generator exists. Then there is a protocol $\mathcal{P}$ that strongly t-securely computes f in* `const` *rounds, where* `const` *is an absolute constant. Furthermore, a natural encoding of protocol $\mathcal{P}$ can be found by a fixed algorithm in time polynomial in $|C|$. Local computation in $\mathcal{P}$ is* `poly`$(|C|, k)$ *time bounded, where* `poly` *is a fixed polynomial.*

5 Conclusion

We have shown that the constant round SFE protocol of Beaver, Micali, and Rogaway is not secure against even a passive adversary due to a flaw in its garbled circuit design. Our correction fixes the flaw while still maintaining the constant round complexity. Our modified circuit construction can also be used on its own in many other applications that make use of the garbled circuit technique.

Acknowledgment

The authors are grateful to Oded Goldreich for his comments, which improved the presentation of these results.

References

[1] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth, *Cryptographic security for mobile code*, in Proc. IEEE Symposium on Security and Privacy, May 2001, pp. 2–11.

[2] D. Beaver, *Foundations of secure interactive computing*, in Advances in Cryptology — Crypto'91 Proceedings, J. Feigenbaum, ed., vol. 576 of Lecture Notes in Computer Science, Springer Verlag, 1991, pp. 377–391.

[3] ——, *Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority*, Journal of Cryptology, 4 (1991), pp. 75–122.

[4] ——, *Correlated pseudorandomness and the complexity of private computation*, in 28th Annual ACM Symposium on the Theory of Computing, 1996, pp. 479–488.

[5] D. Beaver, S. Micali, and P. Rogaway, *The round complexity of secure protocols*, in Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC), 1990, pp. 503–513.

[6] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, in Proc. 20th Annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 1–10.

[7] C. Cachin, J. Camenisch, J. Kilian, and J. Müller, *One-round secure computation and secure autonomous mobile agents*, in Proc. 27th International Colloquium on Automata, Languages and Programming (ICALP), U. Montanari, J. P. Rolim, and E. Welzl, eds., vol. 1853 of Lecture Notes in Computer Science, 2000, pp. 512–523.

[8] R. Canetti, *Studies in Secure Multiparty Computation and Applications*, PhD thesis, Weizmann Institute of Science, June 1995.

[9] ——, *Security and composition of multi-party cryptographic protocols*, Journal of Cryptology, 13 (2000), pp. 143–202.

[10] R. Canetti, U. Feige, O. Goldreich, and M. Naor, *Adaptively secure multiparty computation*, in Proc. 28th Annual ACM Symposium on Theory of Computing (STOC), 1996, pp. 639–648.

[11] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, *Universally composable two-party and multi-party secure computation*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 494–503.

[12] D. Chaum, C. Crépeau, and I. Damgård, *Multiparty unconditionally secure protocols*, in Proc. 20th Annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 11–19.

[13] O. Goldreich, *Draft of a chapter on general protocols (first posted version)*. Extracts from a working draft for Volume 2 of Foundations of Cryptography, October 2002. Available at `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`.

[14] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game*, in Proc. 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, pp. 218–229.

[15] ——, *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*, Journal of the ACM, 38 (1991), pp. 691–729. Prelimilary version appeared in Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science, pp. 174-187, 1986.

[16] S. Micali and P. Rogaway, *Secure computation*, in Advances in Cryptology — Crypto'91 Proceedings, J. Feigenbaum, ed., vol. 576 of Lecture Notes in Computer Science, Springer Verlag, 1991, pp. 392–404.

[17] M. Naor, B. Pinkas, and R. Sumner, *Privacy preserving auctions and mechanism design*, in 1st ACM Conf. on Electronic Commerce, 1999, pp. 129–139.

[18] T. Rabin and M. Ben-Or, *Verifiable secret sharing and multiparty protocols with honest majority*, in Proc. 21st Annual ACM Symposium on Theory of Computing (STOC), 1989, pp. 73–85.

[19] P. Rogaway, *The Round Complexity of Secure Protocols*, PhD thesis, Laboratory for Computer Science, MIT, April 1991.

[20] S. R. Tate and K. Xu, *Mobile agent security through multi-agent cryptographic protocols*, in The 4th International Conference on Internet Computing (IC 2003), 2003.

[21] A. C. Yao, *How to generate and exchange secrets*, in Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS), 1986, pp. 162–167.