Garbling 3-input AND gates

Tomer Ashur[1], Carmit Hazay[2], and Rahul Satish[3]

[1] Cryptomeria, Leuven, Belgium tomer@cryptomeria.tech
[2] Bar-Ilan University, Ramat Gan, Israel carmit.hazay@biu.ac.il
[3] Bar-Ilan University, Ramat Gan, Israel rahul.satish@live.biu.ac.il

Abstract. This paper describes a failed extension to the 3-halves technique. The idea is clever and novel, however, unfortunately, the laws of linear algebra prevent it from working.

Keywords: Garbled Circuits · Free-XOR · Gate-by-Gate Garbling

1 Introduction

Garbled Circuits(GC) have been one of the major paradigms to achieve secure multiparty computation right from its introduction in the 80s by Yao in his seminal paper [Yao86]. Ever since, it has been an active area of study. Garbled circuits today are built using efficient symmetric key primitives, making them extremely practical in situations where the computation or liveness of parties is a bottleneck. However, this comes at the cost of communicating more bits. Hence, there has been a long line of work that is focused on reducing the concrete communication size of garbled circuits [BMR90,NPS99,KS08,PSSW09, KMR14,GLNP15,ZRE15,AAC+21]. The best concrete communication complexity today is achieved by the three-halves garbling scheme [RR21]. The three-halves scheme requires communicating $\frac{3}{2}\kappa$ bits for every AND gate, and no communication is needed for XOR gates.

In [KS08], Kolesnikov and Schneider introduced the free-XOR technique which removed communication for all XOR gates in a circuit. In [ZRE15], Zahur et al. show a lower bound of 2κ bits to garble an AND gate in a model they define as linear garbling. They also show how the linear garbling model captures most of the practical garbling techniques known at that time. Following this, various works [KKS16,BMR16,WmM17] showed how to construct garbling schemes for an AND gate by using techniques not defined in the linear garbling model but could not extend this technique to improve communication for garbling general circuits. Rosulek and Roy, in [RR21], showed how one could garble any general circuit, using $\frac{3}{2}\kappa$ bits per AND gate while maintaining the free-XOR optimization, using the three-halves garbling scheme. They achieved this using two techniques, **Slicing** and **Dicing**. They sliced the input labels into halves and then used the control bit randomization technique, introduced in [KKS16].

However, most of these techniques concern circuits representing a given functionality with just fan-in two gates. State of the art today to garble a higher fan-in gate uses a circuit composed of a sequence of fan-in two gates to generate the higher fan-in gate. For example, to garble a n-input AND gate, $(n-1)$ 2-input gates are used, and hence with a scheme like three-halves, one incurs a communication cost of $\frac{3(n-1)}{2}\kappa$ bits. While it is a viable solution, a vital question would be whether this is the most efficient way to garble such gates.

2 Linear Garbling as a system of equations

Zahur et al. in [ZRE15] introduced the linear garbling model. The model encompassed most of the practical garbling constructions known at that time. They also proved a lower bound on the concrete communication complexity of a garbled gate for a 2-input AND gate. We now discuss how some existing schemes in the linear garbling model can be seen as a system of linear equations, as shown in [RR21]. We discuss garbling AND gates only as XOR is free.

Notations: For this section, we consider 2-input AND gates with input wires a, b and output wire c. Each wire x has two wire labels, a 0-label X_0, and 1-label X_1. To comply with the free-XOR framework [KS08], we also have a global Δ such that $X_1 = X_0 \oplus \Delta$ for all wire labels. Therefore we can always consider a pair of wire labels to be denoted $\{X_0, X_0 \oplus \Delta\}$. We denote by G_{ij} the ciphertext for the row $2i+j$ in the permuted table. Therefore, the garbled gate is represented as $G = \{G_{00}, G_{01}, G_{10}, G_{11}\}$. We also assume oracle access of the garbler and evaluator to a hash function H satisfying some security assumption.

Yao's Garbling Scheme: We now describe Yao's Garbling Scheme, used along with the point and permute technique. The garbler encrypts the labels using a key and generates four ciphertexts $G_{00}, G_{01}, G_{10}, G_{11}$, corresponding to each row of the permuted truth table. A key is derived by hashing the input labels corresponding to that particular row. This gives the following four equations, assuming permute bits $p_a = 1, p_b = 0$ (the second row encrypts $C \oplus \Delta$):

$$\begin{aligned} G_{00} &= H(A_0, B_0) \oplus C \\ G_{01} &= H(A_0, B_1) \oplus (C \oplus \Delta) \\ G_{10} &= H(A_1, B_0) \oplus C \\ G_{11} &= H(A_1, B_1) \oplus C \end{aligned}$$

GRR3 Row Reduction: Randomly sampling A, B, C and a fixed global offset Δ fixes the garbled gate G above for the sampled values. Hence its size is also fixed to 4κ bits. However, as observed by Naor et al. in [NPS99], we need a uniformly sampled C. Since the output of H is already pseudorandom, we can set $C = H(A_0, B_0)$, which would imply $G_{00} = 0$, hence decreasing the size of the garbled gate to 3κ. The set of equations then reduces to:

$$\begin{aligned} 0 &= H(A_0, B_0) \oplus C \\ G_{01} &= H(A_0, B_1) \oplus (C \oplus \Delta) \\ G_{10} &= H(A_1, B_0) \oplus C \\ G_{11} &= H(A_1, B_1) \oplus C \end{aligned}$$

Half Gates: Zahur et al. in [ZRE15] suggested something that can be seen to be significantly different in hindsight but still retains a structure similar to previous schemes. They introduce the possibility of having a 2κ sized garbled gate by making one of the ciphertexts depend on the other two ciphertexts in the garbled gate and the active input label A that the evaluator holds. They achieve this by replacing the term $H(A, B)$ in the previous schemes with $H(A) \oplus H(B)$, which results in the following equations:

$$\begin{aligned} 0 &= H(A_0) \oplus H(B_0) \oplus C \\ G_{01} &= H(A_0) \oplus H(B_1) \oplus (C \oplus \Delta) \oplus A_0 \\ G_{10} &= H(A_1) \oplus H(B_0) \oplus C \\ G_{01} \oplus G_{10} &= H(A_1) \oplus H(B_1) \oplus C \oplus (A_0 \oplus \Delta) \end{aligned}$$

Three Halves: At this point, one could see that by carefully restructuring the half-gates equation, everything the garbler can't control can be shifted to one side of the equation. The garbler cannot control the value of Δ as it is fixed for the entire circuit, the hash outputs, which are pseudorandom by assumption, and the input label A, which we now move to the left of the equation. On the right are the elements the garbler has to compute: the output label and the garbled gate. We now denote the garbled gate by $G = \{G_1, G_2\}$

$$\begin{aligned} C &= H(A_0) \oplus H(B_0) \\ C \oplus G_1 &= H(A_0) \oplus H(B_1) \oplus \Delta \oplus A_0 \\ C \oplus G_2 &= H(A_1) \oplus H(B_0) \\ C \oplus G_1 \oplus G_2 &= H(A_1) \oplus H(B_1) \oplus (A_0 \oplus \Delta) \end{aligned}$$

This can also be done with the equations of Yao's Garbling scheme or the GRR3 row reduction technique. A traditional way to represent this system of equations is $\mathbf{A}x = b$, with its solution $x = \{C, G_1, G_2\}$. For half-gates, the following system holds:

$$\begin{bmatrix} 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \end{bmatrix} \begin{bmatrix} C \\ G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0 \\ 1\,0\,0\,1 \\ 0\,1\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix} \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix} \oplus \begin{bmatrix} 0\,0\,0 \\ 1\,0\,0 \\ 0\,0\,0 \\ 1\,0\,1 \end{bmatrix} \begin{bmatrix} A_0 \\ B_0 \\ \Delta \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \Delta$$

A more concise representation of the above equation is:

$$V \begin{bmatrix} C \\ G_1 \\ G_2 \end{bmatrix} = M \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(B_0) \\ H(B_1) \end{bmatrix} \oplus R \begin{bmatrix} A_0 \\ B_0 \\ \Delta \end{bmatrix} \oplus T\Delta$$

Here T represents the output column of the permuted truth table. Applying the left inverse of the matrix V to both sides gives the solution to this system, and the garbler can then send $G = \{G_1, G_2\}$ to the evaluator. Rosulek et al. in [RR21] showed that the above observation reduces constructing linear garbling schemes to finding solutions to such a system of equations. They showed how one could do better than half-gates using a novel technique called **slicing**. Indeed one can slice the output labels into two halves of size $\frac{\kappa}{2}$ bits each and then garble each slice separately. Applying half-gates on each slice would generate a garbled gate of size $2 \times \frac{\kappa}{2}$ bits for each output slice. Hence, the scheme will still require 2κ bits of communication. However, if the two garbled gates could share a ciphertext, then the size of the garbled gate would be $3\frac{\kappa}{2}$ bits.

They then show how one could garble each slice such that they share a ciphertext. For a label X, let's denote the left slice by X_L and it's right slice by X_R. To derive the output label C_L, we use hash of the form $H(A) \oplus H(A \oplus B)$, and for C_R, we use hash of the form $H(B) \oplus H(A \oplus B)$, where A, B are the active labels corresponding to a particular row being garbled. We now have two equations:

$$\begin{bmatrix} 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,1 \\ 1\,1\,0 \end{bmatrix} \begin{bmatrix} C_L \\ G_1 \\ G_3 \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0 \\ 1\,0\,0\,1 \\ 0\,1\,0\,1 \\ 0\,1\,1\,0 \end{bmatrix} \begin{bmatrix} H(A_0) \\ H(A_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \end{bmatrix} \oplus R_1 \begin{bmatrix} A_L \\ A_R \\ B_L \\ B_R \\ \Delta_L \\ \Delta_R \end{bmatrix} \oplus T\Delta_L \tag{1}$$

$$\begin{bmatrix} 1\,0\,0 \\ 1\,1\,1 \\ 1\,0\,1 \\ 1\,1\,0 \end{bmatrix} \begin{bmatrix} C_R \\ G_2 \\ G_4 \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,0\,1 \\ 0\,1\,1\,0 \end{bmatrix} \begin{bmatrix} H(B_0) \\ H(B_1) \\ H(A_0 \oplus B_0) \\ H(A_0 \oplus B_1) \end{bmatrix} \oplus R_2 \begin{bmatrix} A_L \\ A_R \\ B_L \\ B_R \\ \Delta_L \\ \Delta_R \end{bmatrix} \oplus T\Delta_R \tag{2}$$

Before finding an R that works, if we solve the equation just in terms of the hashes, output label, and garbled gate, we observe that the solution is of the following form:

$$\begin{aligned} G_1 &= H(A_0) \oplus H(A_1) \\ G_2 &= H(B_0) \oplus H(B_1) \\ G_3 &= H(A_0 \oplus B_0) \oplus H(A_0 \oplus B_1) \\ G_4 &= H(A_0 \oplus B_0) \oplus H(A_0 \oplus B_1) \end{aligned}$$

Barring any addition of input labels to this equation, we see that $G_3 = G_4$ is the common G_i for the two sliced output labels. It is incorrect, as we still have not accounted for T, which is essentially the truth table. T enforces the necessity of having an R, and in this case, unlike in half-gates, there is no R_1, R_2 that satisfies this set of equations for all values of T. That means that revealing R gives away the permute bits. Three-halves solves this problem through the idea of **dicing**. These input label combinations can be encrypted using a constant-size ciphertext for each row. The evaluator can only decrypt those combinations needed for the row it is evaluating. This is called dicing. In three halves, they search for R_1, R_2 from all possible matrices satisfying some given conditions such that it satisfies this equation for a given T. They then show how to use the garbling equation to encrypt them using 5 bits, hence setting the cost of the garbled gate to $3\frac{\kappa}{2} + 5$ bits.

3 Extending linear garbling to 3-input AND gates

Most garbling schemes cater to 2-input gates. To garble higher fan-in gates, we need to construct a higher fan-in input gate using 2-input gates. For example, in the 3-input case, a 3-input AND gate is constructed using two 2-input AND gates. We will need $2 \cdot \frac{3\kappa}{2} = 3\kappa$ bits to garble a 3-input AND gate when we construct higher fan-in gadgets using the current approach. Therefore, it will be interesting to see if one could do better by just garbling a 3-input truth table by extending their techniques.

In three halves, an essential factor contributing towards the improvement is the availability of an additional hash, $H(A \oplus B)$. As it does not help as is, they then use slicing to reduce the overall size of the garbled gate. Similarly, we have additional hashes when looking at a three-input case with input labels A, B, and C. The number of available hashes available is exponential in the number of inputs. So we now do the following: We slice the labels into three slices. Let X be a label. We denote a sliced version of X as $X = X^1||X^2||X^3$. What we desire is to have an equation of the following form:

$$V \begin{bmatrix} \vec{D} \\ \vec{G} \end{bmatrix} = M\vec{H} \oplus R \begin{bmatrix} \vec{A} \\ \vec{B} \\ \vec{C} \\ \vec{\Delta} \end{bmatrix} \oplus T\vec{\Delta}$$

We can now, just like in three halves, use the additional hashes, along with $H(A), H(B)$, and $H(C)$ for the three different output slices, respectively. However, this direct extension doesn't work as we have repeating rows for different inputs in the above matrix. We need a hash combination that would change for any change in the input to the circuit. To address this, we choose an approach where we discard $H(X)$ values for wire labels X. We see that $H(X)$ only captures a change in the wire x, $H(X \oplus Y \oplus Z)$ captures a change in the input of one or three wires x, y, z. Neither of them captures a change in two inputs to the circuit as we are in the free-XOR setting. For example, for inputs $(0,0,0)$ and $(0,1,1)$, the first slice of the output labels will always use the hash $H(A_0)$ and the hash $H(A_0 \oplus B_0 \oplus C_0)$, as the free-XOR collapses the change in inputs of wires b and c, to the same hash value. A linear combination of the values of the type $H(X \oplus Y)$ captures an input change made simultaneously to two input wires. So, the garbler includes the following hash combinations for encrypting a slice of the output label D, also :

$$\begin{aligned} D^1 &\leftarrow H(A \oplus B) \oplus H(B \oplus C) \oplus H(A \oplus B \oplus C) \\ D^2 &\leftarrow H(A \oplus C) \oplus H(B \oplus C) \oplus H(A \oplus B \oplus C) \\ D^3 &\leftarrow H(A \oplus B) \oplus H(A \oplus C) \oplus H(A \oplus B \oplus C) \end{aligned}$$

Choosing the hash inputs and combinations for each row fixes matrix M and V, just like in the three-halves garbling scheme. Just like in three halves, we know that the matrices on both sides should have the same column space, and we also see that columns of M, V already span a space of dimension 7. Let the common column space be $\mathcal{G}$. Then,

$$\mathcal{G} = \text{colspace}(V) = \text{colspace}(M) \supseteq \text{colspace}(R \oplus [\mathbf{0}|t])$$

We now examine the constraints on choosing the control matrix R. Let matrix K be a basis for the co-kernel of M. As colspace$(M) \supseteq$ colspace$(R \oplus [\mathbf{0}|t])$, We need $KR = K[\mathbf{0}|t]$. Firstly, we must ensure that the control matrix only uses the combinations the evaluator can use from its available wire labels. We know that R acts on the vector $\begin{bmatrix}\vec{A_0} \ \vec{B_0} \ \vec{C_0} \ \vec{\Delta}\end{bmatrix}^T$. Therefore, the last four columns of R indicate the inclusion of Δ. This depends on whether the evaluator can access X_0 or $X_0 \oplus \Delta$. Thus, R decomposes into 3×3 sub-matrices in the following way:

$$R = \begin{bmatrix} R_{0,0,0,A} & R_{0,0,0,B} & R_{0,0,0,C} & 0 \\ R_{0,0,1,A} & R_{0,0,1,B} & R_{0,0,1,C} & R_{0,0,1,C} \\ R_{0,1,0,A} & R_{0,1,0,B} & R_{0,1,0,C} & R_{0,1,0,B} \\ R_{0,1,1,A} & R_{0,1,1,B} & R_{0,1,1,C} & R_{0,1,1,B} \oplus R_{0,1,1,C} \\ R_{1,0,0,A} & R_{1,0,0,B} & R_{1,0,0,C} & R_{1,0,0,A} \\ R_{1,0,1,A} & R_{1,0,1,B} & R_{1,0,1,C} & R_{1,0,1,A} \oplus R_{1,0,1,C} \\ R_{1,1,0,A} & R_{1,1,0,B} & R_{1,1,0,C} & R_{1,1,0,A} \oplus R_{1,1,0,B} \\ R_{1,1,1,A} & R_{1,1,1,B} & R_{1,1,1,C} & R_{1,1,1,A} \oplus R_{1,1,1,B} \oplus R_{1,1,1,C} \end{bmatrix}$$

We define the marginal view of an input combination for matrix $R(i,j,k)$ to be $[R_{i,j,k,A}, R_{i,j,k,B}, R_{i,j,k,C}]$, which would be sufficient for the evaluator to know which linear combination is to be applied when it holds the labels A_i, B_j and C_k.

We now study the structure of $K[\mathbf{0}|t]$. We notice that we get eight matrices when we compute $K[\mathbf{0}|t]$ for all t values. Let us call these matrices $T_1, \cdots, T_8$ respectively. They correspond to a distinct set of permutations for point and permute $p_1, \cdots, p_8$. This is a complete set of possible permutations using point and permute for the truth table.

Like in the three-halves scheme, finding an R that satisfies the above garbling equation for all t is impossible. So we extend the same re-randomize and encrypt technique as in three halves. There are some changes in how we structure R. Let us assume that there are fixed matrices $R_1, \cdots, R_8$ such that $KR_i = T_i$ for permutation p_i. Let's denote s_i as the activation variable for the permutation p_i as in $s_i = 1$ if p_i is used, and 0 otherwise. Then, we can write

$$R = \sum_{i=1}^{8} s_i R_i + R_\$ \tag{3}$$

such that $KR_\$ = 0$, and $R_\$$ coming from a distribution $\mathcal{R}$ such that the marginal views for every matrix $R_\$ \leftarrow \mathcal{R}$ is uniform for all input pairs (i, j, k). We see that this uniform randomness in $R_\$$ for the marginal views induces uniform randomness in the marginal views for the matrix R, as the matrices R_i are fixed.

Sampling $R_\$$: $R_\$$ should follow the exact structure as R as shown above. $R_\$$ is a 24×12 matrix. Let $C_1, \cdots, C_{12}$ be the 12 columns in $R_\$$. Then, because of the structure of R, we see that columns C_{10} becomes dependent on C_1, C_4, C_7. Similarly, C_{11} becomes dependent on C_2, C_5, C_8 and C_{12} becomes dependent on C_3, C_6, C_9 respectively. Moreover, we observe the dependency across all three sets of columns is the same. So it is enough to find a span for the matrix $[C_1, C_4, C_7, C_{10}]$ and reuse it for the other two column sets to construct the entire matrix $R_\$$.

Also, because we want $KR_\$ = 0$, the columns of $R_\$$ will be constructed by a span of the basis of the right kernel of K. We compute the basis of the right kernel of K, which is of dimension 7, as it is the same space as that of M, which is also of rank 7, as discussed before. We search over all possible 2^7 linear combinations of this basis for each column C_1, C_4, C_7 (note that this fixes C_{10}) such that they form a valid matrix $[C_1, C_4, C_7, C_{10}]$ following the structure of $R_\$$.

This gives us a matrix space M of dimension 2^9. To get the complete matrix $R_\$$, we randomly choose three matrices of 2^9 elements in this space. This gives us the following matrices

$$R_{\$,1} = [C_1, C_4, C_7, C_{10}]$$

$$R_{\$,2} = [C_2, C_5, C_8, C_{11}]$$

$$R_{\$,3} = [C_3, C_6, C_9, C_{12}]$$

Hence, with augmenting and rearranging some columns, we have the matrix

$$R_\$ = [C_1, \cdots, C_{12}].$$

4 The failure.

Unfortunately, the R_i values assumed in (3) do not exist.

References

AAC+21. Anasuya Acharya, Tomer Ashur, Efrat Cohen, Carmit Hazay, and Avishay Yanai. A new approach to garbled circuits. Cryptology ePrint Archive, Report 2021/739, 2021. https://ia.cr/2021/739.

BMR90. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *ACM*, pages 503–513, 1990.

BMR16. Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, page 565–577, 2016.

GLNP15. Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, page 567–578, 2015.

KKS16. Carmen Kempka, Ryo Kikuchi, and Koutarou Suzuki. How to circumvent the two-ciphertext lower bound for linear garbling schemes. In *ASIACRYPT*, pages 967–997, 2016.

KMR14. Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. Flexor: Flexible garbling for XOR gates that beats free-xor. In *CRYPTO*, pages 440–457, 2014.

KS08. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, pages 486–498, 2008.

NPS99. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *ACM-EC*, pages 129–139, 1999.

PSSW09. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *ASIACRYPT*, pages 250–267, 2009.

RR21. Mike Rosulek and Lawrence Roy. Three halves make a whole? beating the half-gates lower bound for garbled circuits. In *CRYPTO*, pages 94–124, 2021.

WmM17. Yongge Wang and Qutaibah m. Malluhi. Reducing garbled circuit size while preserving circuit gate privacy. Cryptology ePrint Archive, Paper 2017/041, 2017.

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

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT*, pages 220–250, 2015.