A Multiple Valued Logic Approach for the Synthesis of Garbled Circuits

Stelvio Cimato
Dipartimento di Informatica "Giovanni Degli Antoni"
Università degli Studi di Milano, Milano, Italy
stelvio.cimato@unimi.it

Valentina Ciriani
Dipartimento di Informatica "Giovanni Degli Antoni"
Università degli Studi di Milano, Milano, Italy
valentina.ciriani@unimi.it

Ernesto Damiani
Information Security Research Center
Khalifa University, Abu Dhabi, United Arab Emirates
ernesto.damiani@kustar.ac.ae

Maryam Ehsanpour
Dipartimento di Informatica "Giovanni Degli Antoni"
Università degli Studi di Milano, Milano, Italy
maryam.ehsanpour@unimi.it

***Abstract*—Secure Multi-party Computation (SMC) protocols enable two or more parties to compute collaboratively generic functions while keeping secret their inputs, sharing only the final result. To achieve this goal, a technique relying on the design of Garbled Circuits (GC) has been firstly proposed by Yao. Garbled circuits are Boolean circuits that can be evaluated using a distributed protocol for computing the result for each gate, till computing the output values. To improve the efficiency of this technique and exploit SMC protocols in practical applications, such as computation outsourcing in untrusted environments, a number of optimizations have been introduced. In this paper we analyze the deployment of Multiple Valued Logic techniques for the design of GC, discussing their impact on the overall computation and communication costs.**

I. Introduction

Secure computation protocols have been introduced to provide two or more interacting parties with the capability of computing a function f of their respective inputs x_1 and x_2, while keeping their inputs private. At the end of the computation, which involves no trusted third party, the participants share only the final result. In the literature [1], different paradigms for solving secure computation problem have been proposed, some relying on Homomorphic Encryption (HE), others on Linear Secret Sharing (LSS), usually deployed when more than two parties are involved, others again on Garbled Circuit (GC) design. After an initial period when research has focused on theoretical and feasibility results, interest in practical SMC systems is growing and different privacy-preserving frameworks are being developed [18], [17]. Significant improvements in efficiency have been achieved and a number of SMC-based solutions to a number of problems have been delivered, such as private auctions, tax-fraud detection, email filtering and others [1].

In this paper we focus on the design and synthesis of Garbled Circuits, whose evaluation is performed gate by gate using a protocol where the participants compute the final output without disclosing their inputs. For each gate in the Boolean circuit, an Oblivious Transfer (OT) protocol [14] is run between the two parties so that the resulting output value can be computed without knowing the input value of the other party. Since the execution of the OT protocol requires interaction between the collaborating parties, reducing the number of gates is important to reduce the overall communication cost for the evaluation of the GC. Kolesnikov et al. in [10] proposed an approach to increase the efficiency of the GC technique, enabling the evaluation of XOR gates essentially for free with respect to the communication cost. Other optimizations have been studied, reducing the size of the communication needed for the garbled tables, as well as implementing other techniques to reduce the GC size and the computation complexity [13].

In our work, we will explore alternative GC representations, focusing on *Multiple Valued Logic* ones. In the field of circuit design, *Multiple Valued Logic* (*MVL*) [3], [4], [6], [8], [11], [12], [15], [20] is a direct generalization of the standard Boolean logic, where the classical Boolean domain $B = \{0, 1\}$ is replaced by $P = \{0, 1, \ldots, |P| - 1\}$ with $|P| > 1$. Multiple valued networks are in general more compact in area than the corresponding Boolean circuits [5], [9], [15]. Moreover, for high level design it is natural to think of multiple valued variables, rather than Boolean ones. During the MVL design process, the problem is described with multiple valued variables, which are later transformed into Boolean ones. This phase, called *encoding*, is particularly critical and potentially onerous. Boolean encoding of MVL is a hard problem, especially for large circuits, since it is difficult to correlate encoding decisions with the final logic optimization of the circuit [5]. Therefore, generally, encoding is done at the beginning without any specific criterion, and then Boolean logic synthesis is applied to an arbitrary encoded circuit [5]. This is an evident disadvantage for the optimization of the final circuit, due to the Boolean nature of the standard circuits. Clearly, the realization of a MVL circuit would directly solve the problem. However, in circuit design, potential advantages of MVL are countered by the difficulty of MVL circuit implementation. Here, we do not exploit a hardware MVL circuit, but only its algebraic description. Therefore, we can completely bypass all technology issues that are the typical of this ctical problem in the Computer-Aided-Design (CAD) context.

To avoid (or limit) the encoding of the function to be securely computed, and in order to obtain a more compact

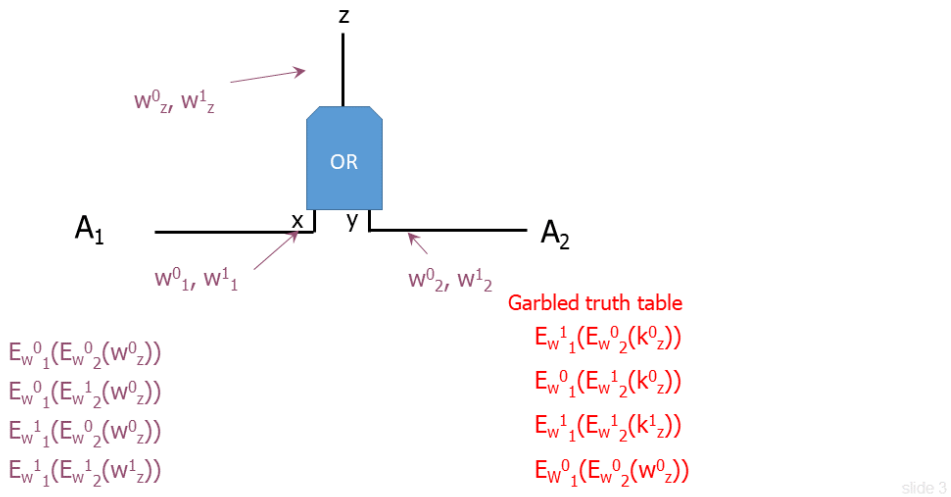

Fig. 1. Construction of a garbled trth table for an OR gate.

circuit description, in this paper we study the generalization of the classical Yao's protocol in the MVL context. We show also how in this context it is possible to extend some of the techniques to improve the evaluation of multiple valued gates, having no communication costs. The paper is organized as follows. In section II-A and II-B we give basic information on secure multi-party computation and multiple valued logic, respectively. In section III we present the generalization of Yao protocol, and in section IV the extension of the optimization technique for the evaluation of some multiple valued gates. In section V we draw final conclusions.

II. Preliminaries

A. Secure Multi-Party Computation

The protocol involves two players, A_1 and A_2 who evaluate together the garbled circuit produced by A_1. The garbling is obtained by producing a garbled table for each gate contained in the circuit. Considering a gate g with input wires W_i and W_j, Player A_1 randomly chooses two secret keys, w_1^0 and w_1^1 corresponding to the garbling of input values 0 and 1, and does the same for the other input wire, selecting the keys w_2^0 and w_2^1. Further, for each gate g, A_1 creates and sends to A_2 a garbled table T, allowing A_2 to recover the garbling of the corresponding g output, and nothing else. Then garbling of players' inputs are (obliviously) transferred to A_2, who obtain the garbled output simply by evaluating the garbled circuit gate by gate, using the tables T_i. Figure 1 depicts the construction of the garbled truth table for the evaluation of an OR gate between A_1 and A_2.

Notice that at during of the computation, A_2 doesn't know the input values of A_1 and the intermediate values of the computation, which are randomly chosen values. A number of tricks can also be implemented to reduce the number of encryptions needed to build and communicate the garbled truth table to A_2. Recently, there have been a lot of research studies on SMC that caused tremendous improvements in terms of performance of GC protocols, resulting in many efficient prototypes such as JustGarble [2], SCAPI (Secure Computation API) [7], and TinyGarble [19].

B. Multiple Valued Logic

In this section we briefly review some basic definitions for Multiple Valued Logic (*MVL*). Complete surveys on this topic can be found in [6], [12].

Let P_i be the finite subset of natural numbers $P_i = \{0, 1, \ldots, |P_i| - 1\}$ with $|P_i| > 1$. A multiple valued logic is a generalization of the classical Boolean logic, as described below.

Definition 1: A multi-valued variable x_i is a variables that takes on values from P_i.

Definition 2: A multi-valued function F is a function such that $F : \{P_1, P_2, \ldots, P_n\} \rightarrow P_F$.

In particular, when $P_1 = P_2 = \ldots = P_n = P_F = P$ we have that $F : P^n \rightarrow P$. Note that, in this case, there are $|P|^{|P|^n}$ possible different MVL functions.

MVL gates are a direct generalization of standard Boolean gates. Of course, the number of two-input gates grows exponentially with the dimension of P. Therefore, while the number of two-input Boolean gates is 16, the number of two-input MVL gates $g : P^2 \rightarrow P$ is $|P|^{|P|^2}$ (e.g., 19.683 for $|P| = 3$). In particular, the standard Boolean AND gate is the minimum gate (i.e., $x \cdot y = \min(x, y)$), OR gate can be generalized to

the maximum (i.e., $x + y = \max(x, y)$) or to the truncated sum gate (i.e., $x +_t y = \min(|P| - 1, sum(x, y))$), the XOR gate can be generalized to the mod sum (i.e., $x \oplus y = (x + y) \mod |P|$) or to the mod difference (i.e., $x \ominus y = (x - y) \mod |P|$), as shown in Figure 2. A *MVL circuit* is a circuit composed by MVL gates.

The objective of multiple-valued logic optimization is to reduce the size of the algebraic expression (e.g., a SOP form) representing a MVL function. As in the Boolean domain, this algebraic expression optimization directly implies the minimization of the corresponding MVL circuit. Minimization techniques have been studied since the late nineteen-sixties, and a large variety of two-level (e.g., Espresso-MV [16]) and multi-level (e.g., MVSIS [8]) optimization tools have been proposed.

III. Multiple Valued Yao's Protocol

In this section we generalize Yao's secure two-party computation protocol to multiple-valued logic. According to the method presented by Yao et al in [21], standard function F is encoded in a Boolean function F_B that can be represented by a Boolean circuit. Starting from their input values, the parties interact to compute the final result by exchanging some encrypted information in order to evaluate the output of each Boolean gate in the circuit. For this reason, the cost of the secure two-party computation protocol is generally proportional to the number of logic gates in the Boolean circuit. In our scenario, we consider a general multiple-valued function F that is represented by a multiple-valued circuit composed by multiple-valued gates.

The aim of two-party secure computation protocols is performing collaborative computation between two parties who do not want to disclose the input values they own [21]. In the general formulation, two parties A_1 and A_2 want to compute a function on their respective inputs, while maintaining the privacy of the inputs. The first protocol was presented in the seminal work by Yao et al. [21] and is usually referred to as the Garbled Circuit (GC) construction of Yao, where the function to be computed securely is represented as a simple Boolean circuit. In order to generalize Yao's construction, let us consider a circuit composed of a single multiple valued gate with two input wires, w_1 and w_2, and one output wire w_3. Let x_1 denote the input multiple-valued value known only to A_1, and x_2 the input multiple-valued value known only to A_2. The construction allows A_2 to evaluate the gate on the two inputs ignoring the value of x_1, while A_1 doesn't learn anything about x_2 (apart from what the parties can deduce from the resulting value and their current input).

Consider the set $P = \{0, 1, \ldots, p - 1\}$ with $p = |P| > 1$ and the multiple-valued gate $G : P^2 \to P$. To proceed with the computation, for each wire w_i A_1 generates p randomly selected different cryptographic keys, one for the input value 0 denoted by k_i^0, one for the input value 1, k_i^1, ..., one for the input value $p - 1$, k_i^{p-1}.

The keys are used as input to a selected encryption algorithm, denoted as $E_{k_1,k_2,\ldots,k_{p-1}}(m)$. Using those keys, A_1 can compute the garbled truth table for the function computed by the gate, whose entry each is obtained using a combination of the input keys corresponding to the possible input values, and contains the encryption of the corresponding output value of the gate. Note that a truth table for a gate $G : P^2 \to P$ has $|P|^2$ entries. For example, while a truth table in the Boolean domain has 4 Boolean entries, in the case of $|P| = 3$ the truth table has 9 multiple valued values (see for example, the truth tables in Figure 2).

Once A_1 has computed the garbled values for all the entries of the table, she can send a permutation of the truth table, together with the keys corresponding to her inputs.

Notice that knowledge of the keys, does not allow A_2 to learn anything on the input values. At this point, A_2 needs, from A_1, the keys corresponding to her own input values without disclosing them to A_1. For this purpose, the party can engage in an Oblivious Transfer protocol (OT), allowing A_2 to learn the keys corresponding to her inputs.

Generally speaking, GC construction relies on $1 - out - of - 2 -$ OT protocol between a sender and a receiver. The sender A_1 has two secret values v_0 and v_1, and the receiver has a secret bit i. At the end of the protocol A_2 learns v_i, but nothing about v_{1-i}, while P_1 learns nothing about the selection bit i. The OT protocol is a widely studied cryptographic primitive, with different variants and implementation, whose robustness has been considered under different security models. In the Yao extension, we consider a $1 - out - of - n$ OT protocol that can be defined as a natural generalization of a $1 - out - of - 2$ OT, where the sender has n values, and the receiver has an index i, corresponding to the value he owns and for which she wishes to receive the i-th key, without the sender learning i. At the end, the OT protocol allows A_2 to retrieve the right key corresponding to her input value for the gate, while A_1 doesn't learn anything about the selected input. Since now A_2 knows the two keys, she can decrypt the right entry in the table, and retrieve the key of the output value.

In general, for a circuit composed of multiple gates, A_1 should compute garbled values for all the input wires and use them for computing the truth table for each multiple-valued gate. Then she should send all the truth tables, and all her input values to A_2, who can invoke the OT protocol for each needed input value to the circuit. Once retrieved all the values, A_2 can compute the output keys for all the gates of the circuit.

At the end of the protocol, A_2 has generated the key k of the final output and sends it to A_1. The party A_1 can now send to A_2 the value, corresponding to k, of the output of the entire function. This value is the final result of the computation.

IV. Improved Evaluation for Multiple Valued Gates

In [10], Kolesnikov and Schneider presented an optimisation, which allows the evaluation of XOR gates to for free, avoiding any interaction between the two parties for such gates. In other words, there is no need to compute and send the garbed tables for the XOR gates. The optimisation requires that there is a global random value R known only to P_1, such that for all garbled wires w_i it holds that $k_i^1 = k_i^0 \oplus R$, i.e. the garbled value corresponding to 1 for a wire, is determined by XOR-ing the garbled 0 value with the random quantity R. In this way, computing the output value for a XOR gate amounts to compute the value resulting by the XOR of the two input

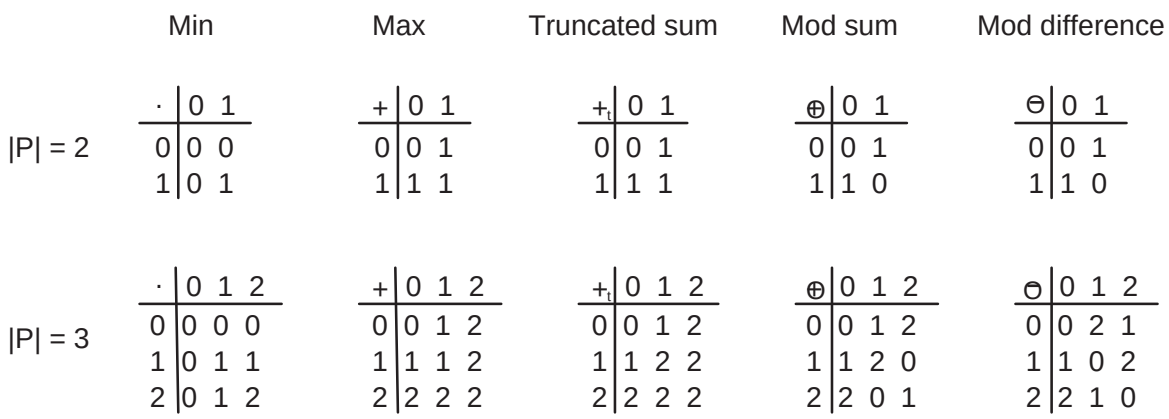

Fig. 2. Truth tables for two-input Boolean and MVL operators.

values. Security of this solution has been proved in different context in [10], [13] under different assumptions.

Here we show how these improved evaluation techniques can be readily extended to the case of MVL gates. Specifically, we show how the SUM gate G, in the case of three-valued logic can be can be evaluated without communication between the parties (the reasoning holds for gates in the case of P values).

Let G have two input wires W_a and W_b and output wire W_c. Garble the wire values as follows. Randomly choose w_a^0, $w_b^0, R_1, R_2 \in_R \{0,1,2\}^N$, with the following properties for R_1 and R_2:

$$R_1 \oplus R_2 = 0$$

,

$$R_1 \oplus R_1 = R_2$$

,

$$R_2 \oplus R_2 = R_1$$

Set $w_c^0 = w_a^0 \oplus w_b^0$, and $\forall i \in (a,b,c) : w_i^1 = w_i^0 \oplus R_1$ and $w_i^2 = w_i^0 \oplus R_2$

It is easy to see that the garbled gate output is simply obtained by summing the garbled gate inputs:

$$w_c^0 = w_a^0 \oplus w_b^0 = (w_a^0 \oplus R_1) \oplus (R_2 \oplus w_b^0) = w_a^1 \oplus w_b^2 = (w_a^0 \oplus R_2) \oplus (R_1 \oplus w_b^0) = w_a^2 \oplus w_b^1$$

$$w_c^1 = w_c^0 \oplus R_1 = w_a^0 \oplus (w_b^0 \oplus R_1) = w_a^0 \oplus w_b^1 = (w_a^0 \oplus R_1) \oplus \oplus w_b^0 = w_a^1 \oplus w_b^0 = (w_a^0 \oplus R_1 \oplus R_1) \oplus (R_2 \oplus w_b^0) = w_a^2 \oplus w_b^2$$

$$w_c^2 = w_c^0 \oplus R_2 = w_a^0 \oplus (w_b^0 \oplus R_2) = w_a^0 \oplus w_b^2 = (w_a^0 \oplus R_2) \oplus \oplus w_b^0 = w_a^2 \oplus w_b^0 = (w_a^0 \oplus R_2 \oplus R_2) \oplus (R_1 \oplus w_b^0) = w_a^1 \oplus w_b^1$$

We omit here the security proofs for this approach.

V. Conclusion

Secure multi-party computation protocols are increasingly deployed for solving practical problems, where privacy of inputs must be preserved and the result can be collaboratively computed. Maturity of the current frameworks is testified by the number of applications developed in different fields [1]. To improve efficiency of the Garbled Circuit construction, a number of optimization techniques have been proposed in literature, aiming at reducing the number of operations and the communication costs, that means reducing the circuit size and the gates for which interaction is required. In this paper we pursue the idea of using Multiple Valued Logic for the synthesis of garbled circuits, showing an extension of the classic Yao protocol for the evaluation of multiple valued gates. Ongoing work includes developing security proofs for our proposed protocols, as well as a full set of experimental prototypes to compare the size and the performance of the multiple valued circuits with respect the original circuits.

References

[1] D. W. Archer, D. Bogdanov, B. Pinkas, and P. Pullonen, "Maturity and performance of programmable secure computation," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 48–56, 2016.

[2] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, 2013, pp. 478–492. [Online]. Available: https://doi.org/10.1109/SP.2013.39

[3] A. Bernasconi and V. Ciriani, "Autosymmetric multiple-valued functions: Theory and spectral characterization," in *41st IEEE International Symposium on Multiple-Valued Logic, ISMVL 2011, Tuusula, Finland, May 23-25, 2011*, 2011, pp. 10–15.

[4] ——, "Autosymmetric and dimension reducible multiple-valued functions," *Multiple-Valued Logic and Soft Computing*, vol. 23, no. 3-4, pp. 265–292, 2014.

[5] R. K. Brayton and S. P. Khatri, "Multi-valued logic synthesis," in *Proceedings Twelfth International Conference on VLSI Design*, 1999, pp. 196–205.

[6] E. Dubrova, "Multiple-valued logic synthesis and optimization," in *Logic Synthesis and Verification*, S. Hassoun and T. Sasao, Eds., 2002, pp. 89–114.

[7] Y. Ejgenberg, M. Farbstein, M. Levy, and Y. Lindell, "Scapi: The secure computation application programming interface." *IACR Cryptology EPrint Archive*, vol. 2012, p. 629, 2012.

[8] M. Gao, J.-H. Jiang, Y. Jiang, Y. Li, S. Sinha, and R. Brayton, "MVSIS," in *Notes of the International Workshop on Logic Synthesis*, 2001.

[9] A. K. Jain, R. J. Bolton, and M. H. Abd-El-Barr, "Cmos multiple-valued logic design – part i: Circuit implementation," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 8, pp. 503–514, 1993.

[10] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, ser. Lecture Notes in Computer Science, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds., vol. 5126. Springer, 2008, pp. 486–498. [Online]. Available: https://doi.org/10.1007/978-3-540-70583-3_40

[11] D. M. Miller and R. Drechsler, "On the construction of multiple-valued decision diagrams," in *32nd IEEE International Symposium on Multiple-Valued Logic (ISMVL 2002), May 15-18, 2002, Boston, Massachusetts, USA*, 2002, pp. 245–253.

[12] D. M. Miller and M. A. Thornton, *Multiple Valued Logic: Concepts and Representations*. Morgan & Claypool Publishers, 2007.

[13] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Springer, 2009, pp. 250–267. [Online]. Available: https://doi.org/10.1007/978-3-642-10366-7_15

[14] M. O. Rabin, "How to exchange secrets with oblivious transfer." *IACR Cryptology ePrint Archive*, vol. 2005, p. 187, 2005.

[15] D. A. Rich, "A survey of multivalued memories," *IEEE Trans. Comput.*, vol. 35, no. 2, pp. 99–106, 1986.

[16] R. L. Rudell, "Multiple-valued logic minimization for pla synthesis," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M86/65, 1986.

[17] M. Sepehri, S. Cimato, and E. Damiani, "A multi-party protocol for privacy-preserving range queries," in *Secure Data Management - 10th VLDB Workshop, SDM 2013, Trento, Italy, August 30, 2013, Proceedings*, 2013, pp. 108–120. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-06811-4_15

[18] ——, "Privacy-preserving query processing by multi-party computation," *Computer Journal*, vol. 58, no. 10, pp. 2195–2212, 2015. [Online]. Available: http://dx.doi.org/10.1093/comjnl/bxu093

[19] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, "Tinygarble: Highly compressed and scalable sequential garbled circuits," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 411–428.

[20] B. Steinbach, M. A. Perkowski, and C. Lang, "Bi-decompositions of multi-valued functions for circuit design and data mining applications," in *29th IEEE International Symposium on Multiple-Valued Logic, ISMVL 1999, Freiburg im Breisgau, Germany, May 20-22, 1999, Proceedings*, 1999, pp. 50–58.

[21] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *FOCS*, 1986, pp. 162–167.