SAND2020-5499C

Garbled circuits for enabling privacy preserving safeguards

Mitchell Negus[1], Rachel Slaybaugh[1], and David Farley[2]

[1]University of California, Berkeley
[2]Sandia National Laboratories[*]

July 2020

Abstract

International safeguards rely on mutual cooperation between an international inspector and a State facility. However, only safeguards data agreed upon through the joint Comprehensive Safeguards Agreement (CSA) is necessarily made available to the IAEA. On one hand, the inspector desires as much information as is needed to be assured that material is not diverted, or that such a diversion is detected in a timely manner. On the other, the State facility aims to minimize operational disruptions and prevent the exposure of any information that it deems private or proprietary. These separate concerns can be at odds, but recent developments in the field of secure multi-party computation may offer a solution. Secure multi-party computation allows multiple parties to evaluate functions together, learning the answer to a given question without exposing the private inputs of any party. For safeguards, an inspector and a facility could corroborate evidence of compliance without the facility ever needing to share any data beyond its current commitments. Garbled circuits, a class of multi-party computation algorithms, can facilitate this interaction. As a demonstration, garbled circuits are applied to time series data sets and correctly identify irregular patterns without ever accessing the data directly. This result suggests that garbled circuits can also be used to detect anomalous safeguards events—or ideally their absence—without access to the original data streams.

1 Introduction

International nuclear safeguards are the primary key mechanism for preventing the proliferation of nuclear weapons and nuclear weapons technology. These safeguards—most often established by a Comprehensive Safeguards Agreement (CSA) concluded between the International Atomic Energy Agency (IAEA) and a member State—provide for the monitoring of all peaceful nuclear material in the possession of the member[1]. While the safeguards practices outlined in these CSAs share many of the same foundational principals, procedures, and technical implementations, each CSA is negotiated on a case-by-case basis. The agreements are crafted to balance the reasonable objectives of both parties. For the IAEA, this objective

[*]*Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

is to prevent the diversion of nuclear material to unauthorized, or worse, malicious entities [1]. States likely share this objective, but must also consider the security and privacy of their data. A prudent state may be reluctant to share any data that is not explicitly required under its CSA, even if it might be useful in identifying material diversions or could corroborate a narrative of smooth and compliant operation.

Modern data-analytics present a myriad of opportunities for expanding nuclear safeguards capabilities [2, 3, 4]. When applied to increasingly prevalent digital instrumentation and sensor networks, pattern recognition and anomaly detection algorithms could facilitate near-instantaneous alerts of abnormal circumstances, including the potential diversion of nuclear material. However, for data sources that are not already covered by a safeguards agreement, it would be within the rights of a State to refuse to provide collected information to inspectors. To improve the chances that a State is willing to share such supplemental information, inspectors could leverage advancements in the field of secure multiparty computation (MPC). MPC allows two or more parties to jointly perform a calculation using inputs from both parties, learn the common output, but never learn the input from any other party [5]. In this scenario, a State or facility could provide some safeguards relevant data stream, the inspector could provide criteria supporting normal operation, and together the two parties jointly determine if the facility was operating as expected. To demonstrate this technique, this work applies multiparty computation to time series data, identifying abnormal events while never exposing the time series to the evaluator.

Garbled circuits were chosen as the specific subclass of MPC used to evaluate the time series data. The underlying concepts were postulated by Andrew Yao in the mid-1980s to extend privacy-preserving computation to arbitrary functions [6, 7, 8]. Then, by framing a calculation as a boolean circuit of logic gates, any computable function can be evaluated securely. Generally, the process entails two parties, a circuit generator and a circuit evaluator, executing an interactive procedure.

2 The Garbled Circuit Protocol

The protocol begins with both parties agreeing on the function to compute, and then developing a virtual logic circuit (like the one shown in Figure 1) that properly evaluates the selected function. In the most basic case, this circuit will consist of some number of virtual boolean gates (NOT, AND, OR, NAND, NOR, XOR, XNOR) connected by virtual wires [8, 5]. The input wires to this circuit represent the inputs of the two parties to the function, with some subset of the wires corresponding to the input of one party and the remainder corresponding to the input of the other. Given the values on those combined inputs, the circuit solves the function with an answer represented on the circuit's output wires. To protect the privacy of each party, the circuit undergoes a process called "garbling".

The following description of the garbling process adopts the following definitions:

- k denotes the circuit security parameter, a number of bits that defining the computational size of the problem
- $\in_R$ represents a uniform random selection from a given set
- $\{0,1\}^n$ gives a string of bits (ones and zeros) of length n; if n is omitted, it is assumed to be 1

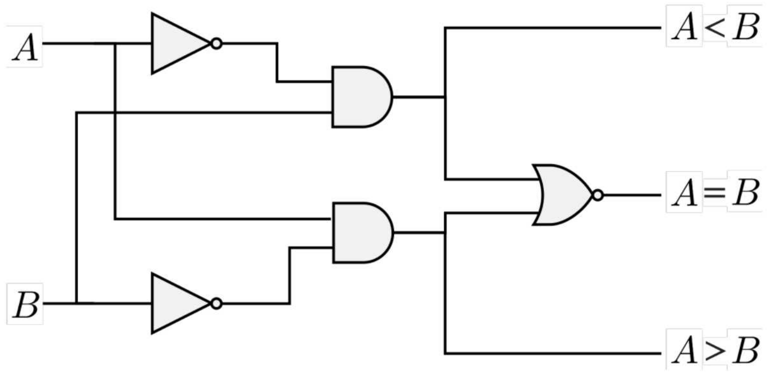

Figure 1: A simple logic circuit for comparing two bits, A and B. The value on each output wire on the right will be either 0 if the statement is false, or 1 if the statement is true.

(a)

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

(b)

x	y	z
ℓ_x^0 : 6fd0	ℓ_y^0 : c4d4	ℓ_z^0 : ed56
ℓ_x^0 : 6fd0	ℓ_y^1 : 32f1	ℓ_z^0 : ed56
ℓ_x^1 : 131b	ℓ_y^0 : c4d4	ℓ_z^0 : ed56
ℓ_x^1 : 131b	ℓ_y^1 : 32f1	ℓ_z^1 : bd67

Figure 2: (2a) The truth table for a logical AND gate. (2b) The truth table for an AND gate, with random labels assigned. Sample 16-bit hexadecimal labels are given as an example.

- $E_{\alpha,\beta}(x), D_{\alpha,\beta}(x)$ represent two-key symmetric encryption and decryption functions, such that $E_{\alpha,\beta}\left(D_{\alpha,\beta}\left(x\right)\right) = x$ and α, β are the two encryption keys

2.0.1 Circuit Generator

The circuit generator begins the garbling process by privately assigning random labels to each potential value on every wire in the circuit. For some wire x with potential values 0 and 1, these labels would be two security-parameter-length strings of random bits (mathematically these are $\ell_x^{b_x} \in_R \{0,1\}^k$, where $b_x \in \{0,1\}$ is the bit value on wire x). Starting with the input wires, the circuit generator then creates the truth table for each gate in the circuit. Figure 2a shows the truth table for an AND gate.

The circuit generator then replaces each wire value in the truth table with the corresponding random wire label. Figure 2b gives a circuit with replaced labels. To hide the mapping between wire values and wire labels, the circuit generator encrypts each output value in the truth table using the input values on the same row. For a gate with two wire inputs, this would be

$$e_{b_x,b_y} = E_{\ell_x^{b_x},\ell_y^{b_y}}\left(\ell_z^{b_z}\right),$$

resulting in four outputs e_{b_x,b_y}, one for each combination of $b_x, b_y \in \{0,1\}$. Finally, these encrypted tokens are shuffled randomly to remove all correlation with the initial truth table.

After garbling the circuit in this way, the circuit generator passes the encrypted tables of tokens to the circuit evaluator. Along with the shuffled tables, the generator also passes the evaluator the label corresponding to any circuit input wire constituting part of the generator's circuit input. Since wire labels are uniformly random bit strings, the evaluator cannot identify whether a label represents an input value of 0 or 1. The generator's private inputs remain secure.

2.0.2 Circuit Evaluator

The circuit evaluator begins by receiving the garbled circuit and random input labels from the generator. From the evaluator's perspective, these labels and tables are just meaningless random information. Since the generator sent only their own labels (not knowing the labels corresponding to values on the evaluator's input wires), the evaluator must request its labels from the generator. The evaluator cannot do this directly without exposing their own input value to the generator, and instead engages the generator in an oblvious transfer (OT) protocol. In this process, the evaluator learns exactly one wire label (e.g. $\ell_y^{b_y}$ for $b \in \{0,1\}$) while the generator never learns the evaluator's selection (either $b_y = 0$ or $b_y = 1$). Once the evaluator concludes the OT protocol and possesses the complete set of wire input labels, they may begin evaluating the circuit.

For each gate in topological order, the evaluator uses the input wires and table of encrypted tokens to determine the random label on the gate's output wire. Given any two wire labels ℓ_x and ℓ_y, the evaluator can succssfully decrypt one (and only one) entry in the shuffled table of tokens. For example, reversing the process for the example gate with two input wires, the evaluator computes

$$\ell_z^{b_z} = D_{\ell_x^{b_x}, \ell_y^{b_y}} \left(e_{b_x, b_y} \right).$$

Note that in this example, the evaluator does not actually know b_x or b_y unless wire x or y is one of the evaluator's input wires to the circuit. All decrypted wire labels still appear to be random strings of bits, with no correlation to the value that they represent. Since only one (shuffled) table entry is decrypted, the evaluator cannot discern anything about the original unshuffled table. the Every wire label that is decrypted can then be used to decrypt tables for subsequent gates. This process continues until the evaluator possesses the labels for each output wire of the circuit.

At this point, the two parties may come together to determine the output of the solved function. The generator applies the mapping between output labels and output values to the labels determined by the evaluator to reveal the ouput bits produced by the circuit.

3 Time Series Analysis

As mentioned in previous work [INMM 2019], this demonstration project ultimately intends to apply garbled circuits to safeguards-relevant data. This could include datasets such as those from the MINOS testbed. To prepare for a successful demonstration with safeguards data, this project first focused on building garbled circuits that could be applied to arbitrary time series data streams. This general algorithm can be extended to handle the intricacies of operating on safeguards-relevant data as needed.

Drawing on experience of project collaborators in performing time series anomaly detection[9], a procedure was developed to perform the discord discovery anomaly detection algorithm using

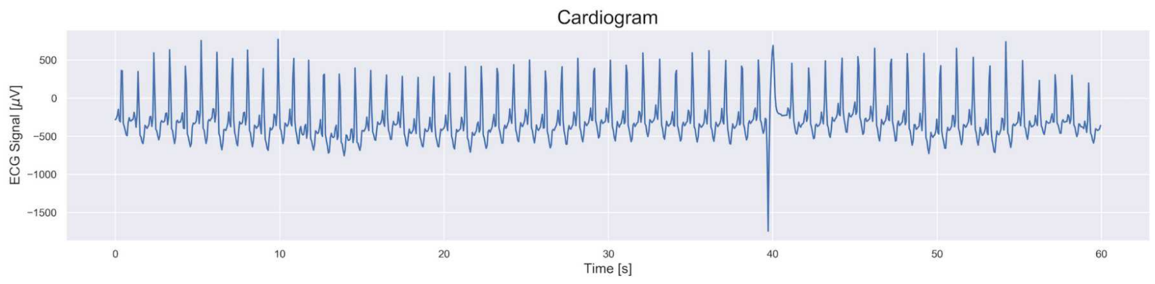

Figure 3: A 60 second time series sample of an electrocardiogram. A heartbeat anomaly is visually apparent near the 40 second mark.

garbled circuits.

The discord discovery algorithm identifies anomalous events by searching the time series for a subsequence (a subset of consecutive data points) that is the most different from any other subsequence. The subsequence that is most different is, by definition, the most anomalous subsequence. Uniqueness can be characterized using a distance metric, most often chosen to be the Euclidean distance between subsequences. A sample discord discovery procedure would select an appropriate subsequence length (chosen based on *a priori* knowledge or a best guess regarding the length of a potential anomaly), then loop through each time series subsequence. For each subsequence, the distance to every other subsequence is calculated, and the minimum calculated distance is stored. The most unique subsequence would then be given by the subsequence with the largest minimum distance to every other subsequence.

The discord discovery method offers a robust and consistent procedure, and is therefore straightforward to implement using garbled circuits. The algorithm has been shown to successfully identify anomalies in electrocardiogram (ECG) time series [9], and so ECG datasets were therefore the dataset of choice for testing a privacy-preserving implementation of the algorithm. ECG data was acquired from the publicly available Beth Israel Deaconess dataset [10, 11].

4 The *CypherCircuit* Package

With the final project objective being to demonstrate the viability of MPC as a nuclear safeguards tool, the garbled circuit implementation must be both highly efficient and clearly written. While several garbled circuit frameworks and implementations exist [12, 13, 14, 15, 16, 17], none met the criteria to the extent necessary. Many are state-of-the-art research codes and are intended for cryptographic experts to showcase optimization advancements, are still in-development, or are built on proprietary foundations. The former characteristic is unlikely to be well suited for convincing safeguards administrators that the technique is secure, inhibiting adoption of the method. At the same time, while the latter may be reasonable for use in production, it is less suitable for this demonstration. As a solution, the *CypherCircuit* Python package is being developed to implement garbled circuits for arbitrary functions—including time series discord discovery—with code transparency, accessibility, and intuitiveness as priorities.

The *CypherCircuit* package allows users to build circuits from elementary components—namely `Gate` objects connected by `Wire` objects. Utilizing the object-oriented nature of Python, all gate types (`Not`, `And`, `Or`, `Xor`, etc.) are provided as straightforward descendants of the base

`Gate` class. These components are added as members of a circuit, accessed altogether through a `CircuitBoard` unit. To simplify operation for a user, a standard set of more complex components (e.g. circuit adders, comparators, multipliers) are provided in one of the packages modules, while a set of pre-built circuits are provided in another.

Once the desired circuit has been constructed (e.g. by a user acting as the circuit generator), each `CircuitBoard` can be garbled, a process in which labels are assigned, truth tables are generated, and truth table rows are encrypted. The circuit generator can then encode the circuit using a binary input vector before passing the circuit components to a circuit evaluator. The evaluator executes the analogous method for decoding the circuit, solving for the output.

While a user is most likely to elect to exchange information between parties over a network (an option which is faster and requires less storage space), the framework also provides the option for all exchanged information to be written to files for the sake of transparency. These files can be inspected by a curious, if not skeptical, party to ensure that no sensitive information was revealed in the process.

4.1 ***CypherCircuit*** Application to Time Series Data

After the basic functionality was completed, the *CypherCircuit* package was tasked with detecting anomalous events in a patient's ECG. A time series interval of 60 seconds was selected from one patient in the complete BIDMC Congestive Heart Failure database, with 250 voltages recorded per second. Each consecutive set of 15 data points from the original time series was averaged together to produce a time series with 1,000 points, in order to reduce the overall size of the data set while preserving key features.

To complete the analysis, a circuit was built to calculate the square of the Euclidean distance between any two n-dimensional vectors. Each n-length subsequence of the time series could be considered an n-dimensional vector, with each subsequence point representing the vector's component in one dimension. Following the procedure for discord discovery, each subsequence pair was assigned an equivalent circuit. Evaluating each circuit yielded a collection of distance values which could then be compared. The minimum squared distance between each subsequence and all other subsequences was identified. Since the square root function is monotonic, the minimum squared distance is equivalent to the true minimum distance. The square root function is not necessary and may be omitted to reduce runtime.

As can be seen in Figure 4, the algorithm correctly identified the anomalous subsequence. The highlighted region indicates the anomaly identified by the algorithm. In total, each circuit consisted of 25,567 wires and 30,953 logic gates. With optimizations, 14,718 (48%) of these gates were "free" and did not need to be evaluated. The evaluation time for each circuit was approximately 8 minutes per subsequence, and when performed for the nearly 100,000 subsequence pairs in the time series, took a total time of about 2 days on a standard laptop computer.

5 Conclusion

This demonstration suggests that time series anomaly detection using garbled circuits is possible. To be practical however, significant gains in efficiency must be realized. A time series analysis taking several minutes to process seconds of data is unlikely to be adopted. However, while the *CypherCircuit* package already takes advantage of several known optimizations (e.g. FreeXOR optimization [18] and an intelligent evaluation scheme to avoid garbling gates

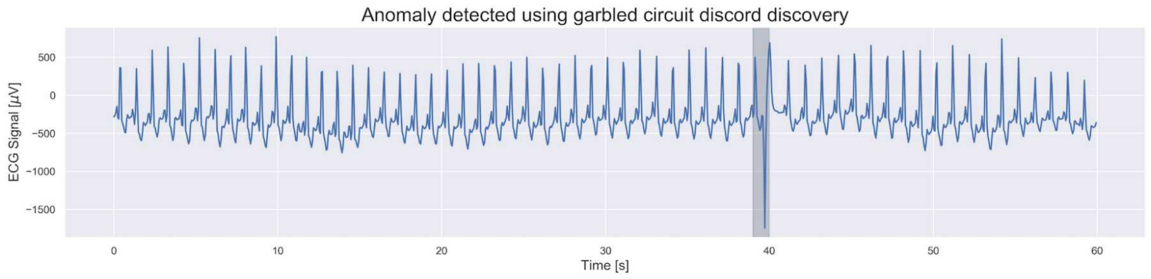

Figure 4: The result of the discord discovery method applied to the cardiogram, when operated using garbled circuits. The highlighted region is the algorithmically selected anomaly. The evaluator identified the time at which the anomaly occurred without ever receiving access to the ECG time series.

with publicly known output values) there are still significant enhancement opportunities available. Notably, the code could leverage algorithmic improvements such as row-reduction [19] or powerful arithmetic logic circuits [20], implement structural upgrades to enable faster code execution using Cython or other optimized Python libraries, or simply be modified to run utilizing parallel processing.

Though efficiency gains are necessary, this successful demonstration suggests that garbled circuits should be constructed to operate on nuclear safeguards specific time series data. While these data sets may include some two-dimensional time series data streams, like the cardiogram, time series consisting of radiation spectra represent a three-dimensional (energy, count, time) data stream. Data of this type will likely require a more sophisticated treatment.

More generally, the *CypherCircuit* package is designed to be flexible, and can be applied to any potential safeguards challenge where privacy-preserving computation may be useful. This scope is not limited to time series analysis. The garbled circuit technique could be used, for instance, to detect inconsistencies in two sets of independently collected records. Even outside of safeguards-specific applications, multiparty computation may be of use to the nuclear non-proliferation community. Research has already explored using physical zero-knowledge proofs to perform secure warhead verification, and similar endeavors could return to using digital cryptography for those applications. In short, MPC presents a valuable tool for the nuclear safeguards and non-proliferation community going forward. In a modern, complex world with increasing international tension, systems like MPC which obviate trust requirements between parties may actually become a preferred solution for building international cooperation.

References

[1] IAEA Safeguards Glossary, 2002.

[2] J. Kornell, et al. Informational sensing for nonproliferation. SAND2016-5573C, Sandia National Laboratories, 2016.

[3] Z. N. Gastelum and Shead T. M. Preliminary application of neural networksto support assessmentof ground-based imagery for international safeguards analysis. SAND2017-4857C, Sandia National Laboratories, 2017.

[4] K. J. Dayman, et al. Transformative Data Analytics Capabilities for Nuclear Forensics and Safeguards. In *Proceedings of the 60th Annual Meeting of the Institute of Nuclear Materials Management*, July 2019.

[5] D. Beaver, et al. The Round Complexity of Secure Protocols. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC 90, page 503513, New York, NY, USA, 1990. ACM.

[6] A. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.

[7] A. C. Yao. How to Generate and Exchange Secrets. In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE, October 1986.

[8] O. Goldreich, et al. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC 87, page 218229, New York, NY, USA, 1987. Association for Computing Machinery.

[9] E. Keogh, et al. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *The Fifth IEEE International Conference on Data Mining*, 2005.

[10] Baim D. S., et al. Survival of patients with severe congestive heart failure treated with oral milrinone. *J. American College of Cardiology*, pages 661–670, March 1986.

[11] A. Goldberger, et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, 2000.

[12] A. Ben-David, et al. Fairplaymp - a system for secure multi-party computation. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 257–266, 01 2008.

[13] D. Bogdanov, et al. Sharemind: A Framework for Fast Privacy-Preserving Computations. In S. Jajodia and J. Lopez, editors, *Computer Security - ESORICS 2008*, pages 192–206, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[14] E. M. Songhori, et al. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *2015 IEEE Symposium on Security and Privacy*, pages 411–428, 2015.

[15] D. Demmler, et al. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *ENCRYPTO*, 01 2015.

[16] E. M. Songhori, et al. Arm2gc: Succinct garbled processor for secure computation. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC 19, New York, NY, USA, 2019. ACM.

[17] Keller M. MP-SPDZ: A versatile framework for multi-party computation. Cryptology ePrint Archive, Report 2020/521, 2020. `https://eprint.iacr.org/2020/521`.

[18] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In L. Aceto, et al., editors, *Automata, Languages and Programming*, pages 486–498, Berlin, Heidelberg", 2008. Springer Berlin Heidelberg.

[19] M. Naor, et al. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC 99, pages 129–139, New York, NY, USA, 1999. Association for Computing Machinery.

[20] M. Ball, et al. Garbling gadgets for boolean and arithmetic circuits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS 16, pages 565–577, New York, NY, USA, 2016. Association for Computing Machinery.