A Survey of Optimizations on Garbled Circuits and Their Application to Private Set Intersection

AJ Wisniewski
austinw6@illinois.edu
University of Illinois at Urbana-Champaign
Champaign, IL, USA

ABSTRACT

Distributed computing has become a fundamental component of data analysis due to the upsurge of big data pipelines. While these technologies have facilitated new advancements that benefit humanity, they also provide an attack vector for compromising the privacy and security of the underlying data. Additionally, real-world applications increasingly employ multiple entities to evaluate desired analytical functions, either for efficiency or security reasons, or both. In the case of security, researchers have proposed various cryptographic schemes to ensure data privacy and protection throughout this process. Yao's garbled circuit protocol is one of the most robust solutions to meet these requirements and continues to be optimized to this day. This paper first describes Yao's original protocol and related fundamental cryptographic primitives, then discusses recent advancements made in optimizing garbled circuits, and finally presents a basic prototype implementation and application of garbled circuits to private set intersection.

1 INTRODUCTION

Secure computation is an active subfield of cryptography devoted to evaluating functions in a secure manner. One of the first secure computation problems was coined by Yao and is known as the "millionaires' problem" [32]. It describes a scenario where two millionaires want to determine who is richer without revealing their actual wealth to each other. In his paper, Yao also proposed a cryptographic solution to this and generalized it to the secure computation of any function [5]. In cases involving two or more parties, this process is known as *secure multi-party computation* (MPC). There are many real-world scenarios where MPC techniques can be applied, including financial systems [6], secure collaboration between governments and companies [20, 13], electronic elections [9], as well as many others.

Researchers have proposed various means for realizing MPC, including through generic methods and function specific methods [5]. Generic methods are used for cases in which the number of possible functions supported by the protocol is infinite. Yao's garbled circuit (GC) protocol remains one of the most important generic paradigms for MPC, especially in the case of *secure two-party computation* (2PC) [33, 5]. Since it was introduced in 1986, it has remained a prominent area of cryptographic research with new optimizations continuing to be made in terms of both communication and computational complexity.

The structure of this paper is as follows: Section 2 introduces fundamental topics pertaining to Boolean circuits, cryptography and two-party computation, Section 3 outlines the process for executing Yao's garbled circuit protocol, Section 4 summarizes recent optimizations to Yao's protocol, and Section 5 describes a basic prototype implementation and application of the protocol to private set intersection (PSI).

2 PRELIMINARIES

2.1 Secure Multi-Party Computation

Distributed computing considers the scenario where multiple distinct, yet connected, computing devices (or parties) wish to carry out a joint computation of some function [19]. For example, each party may contribute to a distributed database system, and the function to be computed may be a database update or analysis of some kind. The aim of *secure multi-party computation* (MPC) is to enable parties to carry out such distributed computing tasks in a secure manner. *Two-party computation* (2PC) is a special case of MPC with only two parties.

To formally claim and prove the security of an MPC protocol, there are certain general security requirements that must be satisfied [20]. The most central of these properties are:

- *Privacy:* None of the parties should learn anything more than their prescribed output or information that can be derived from the output itself. For example, in an auction setting where only the highest bid is revealed as the output, it is clearly possible to derive from the output that all other bids were lower.
- *Correctness:* The output that is delivered to each party in the protocol is guaranteed to be correct. Using the same auction example, only the party with the highest bid is guaranteed to win and no party, including the auctioneer, can alter this.
- *Independence of Inputs:* No party should be able to choose their input based on the inputs of other parties. For example, it may be possible to generate a higher bid without knowing the exact value of the one in question.
- *Guaranteed Output:* At the end of the protocol, honest parties should receive their outputs regardless of attempts by corrupted parties to prevent it.
- *Fairness:* Any party, corrupted or honest, can receive their output if all of the parties receive their outputs.

The above list does not constitute a definition of security, but rather a set of requirements that should hold for any secure MPC protocol [20].

2.2 Adversary Models

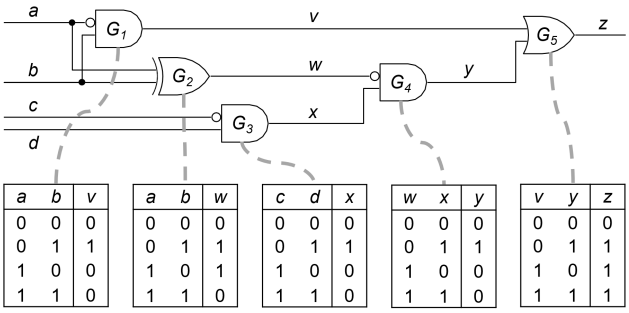
The standard approach to formally prove the security of cryptographic protocols is to consider adversaries with different capabilities [29]. The three most common groups of adversaries are:

- *Malicious (or Active) Adversaries:* These are the strongest type of adversaries and are allowed to deviate from the protocol arbitrarily. Two of their main aims are to learn the private inputs of other parties and to influence the outcome of the computation. Due to their unpredictable behavior, protection against such attacks is relatively expensive.
- *Covert Adversaries:* These are similar to malicious adversaries, but with the restriction that they must avoid being caught cheating. Due to this extra constraint, protocols secure against covert adversaries can be substantially more efficient than those against malicious ones.
- *Semi-Honest (or Passive) Adversaries:* These do not deviate from the protocol and only attempt to learn additional information based upon what an honest party would normally have access to. This is why they are also sometimes referred to as *honest-but-curious adversaries*. This model encompasses many practical settings, such as protection against insider attacks. As a result, most practical privacy-preserving applications rely on the semi-honest model for the underlying protocol and implementation.

2.3 Boolean Circuits

Boolean circuits are a classical representation of functions in engineering and computer science [29]. They are comprised of two main components: gates and wires. Figure 1 is an example of a Boolean circuit containing wires *a, b, c, d, v, w, x, y,* and *z* as well as gates $G_1, \cdots, G_5$. Wires *a, b, c,* and *d* are the *input wires* to the circuit, wires *v, w, x,* and *y* are *intermediate wires,* and *z* is the *output wire*. Each wire is exactly 1 bit that may encode one of two truth values: either `TRUE` or `FALSE`, also represented as 1 and 0 respectively. Each gate takes in two inputs on the left and outputs a truth value on the right based on an internal *truth table* shown at the bottom of Figure 1. Some common types of logic gates include AND ($\wedge$), OR ($\vee$), NOT ($\neg$), and XOR ($\oplus$).

The *size* of a Boolean circuit is represented by the number of gates it has [31]. Decreasing the number of gates in a circuit results in reducing the overall cost of an MPC protocol in terms of both computation and communication complexity [5]. Section 4 will discuss optimizations that rely upon restructuring circuits to improve efficiency.



a	b	v
0	0	0
0	1	1
1	0	0
1	1	0

a	b	w
0	0	0
0	1	1
1	0	1
1	1	0

c	d	x
0	0	0
0	1	1
1	0	0
1	1	0

w	x	y
0	0	0
0	1	1
1	0	0
1	1	0

v	y	z
0	0	0
0	1	1
1	0	1
1	1	1

Figure 1: An example Boolean circuit. Figure adapted from Mike Rosulek's presentation entitled *A Brief History of Practical Garbled Circuit Optimizations* [26].

2.4 Cryptography Fundamentals

2.4.1 Symmetric Encryption. A *symmetric encryption* scheme uses the same cryptographic key k for both encrypting plaintext messages and decrypting ciphertexts [10]. The notation $c \leftarrow Enc_k(m)$ means that a plaintext message m is encrypted with a key k, resulting in a ciphertext c [5]. Similarly, decryption can be denoted as either $m \leftarrow Dec_k(c)$ or $m \leftarrow Enc_k^{-1}(c)$ since it is the inverse of encryption. One of the most well-known symmetric encryption schemes is `AES256` [10]. There are other schemes that build on top of `AES256` to provide additional security features, including `Fernet` which is discussed in Subsection 5.2.

2.4.2 Cryptographic Hash Functions. A *cryptographic hash function* $H(m)$ maps an arbitrary size message m to a fixed size ℓ-bit string $c \leftarrow H(m)$ [25]. In cryptography literature, the term *hash function* generally refers to a cryptographic hash function. Hash functions have two common security properties [27]:

- *Collision resistance:* It should be computationally infeasible for an adversary to compute any collision $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$. In other words, an adversary should not be able to find two messages that hash to the same output.
- *Second-preimage resistance:* Given a message x, it should be computationally infeasible for an adversary to compute any collision $x' \neq x$ such that $H(x) = H(x')$.

Hash functions can be ideally modelled as a *random oracle* to assess their security [16]. A random oracle is a theoretical black-box that responds to every unique query with a true random number chosen from its output domain [5]. To achieve this, it records its responses to unique queries so that it can respond to repeated queries with the same random output. One well-known hash function is `SHA256`, which is used in the garbled circuit implementation in Subsection 5.2.

2.4.3 Pseudo-Random Functions. A *pseudo-random function* (PRF) is a function that can be used for generating pseudo-random output that is indistinguishable from true random output [5, 27]. In other words, a PRF can be modelled as a random oracle. It is denoted as $PRF_k(x)$ on an input x using a private key k. A PRF can be instantiated with a block cipher, such as `AES256`, or a cryptographic hash function, such as `SHA256`) [29].

2.4.4 Message Authentication Codes. A *message authentication code* (MAC) is a fixed-size tag that is used to provide authentication for a message [27]. It is denoted as $MAC_k(m)$ on an input message m and a private key k [29]. MAC tags provide protection for both the integrity and authenticity of a message since verifiers who possess the private key k can use them to detect any unwarranted changes to the message content [5].

2.4.5 Commitment Schemes. A *secure commitment scheme* allows a committer to publish a value, called the *commitment*, that binds them to a message without revealing it [15]. Later, the committer may *open* the commitment and reveal the committed message to a verifier, who can validate that it is consistent with the commitment. Commitment schemes have two main security properties [11]:

- *Binding:* The committer should not be able to alter the message after committing to it.

- *Hiding:* No information about the committed message should be revealed to other parties.

2.4.6 Secret Sharing. A *secret sharing scheme* distributes *shares*, or portions, of a secret value amongst a group of n parties [5]. To reconstruct the secret value, the parties need to combine a sufficient number of shares together since each individual share should reveal no information about the secret. *Shamir's secret sharing* is one well-known *threshold secret sharing scheme*, where only $k+1$ of the n shares are required to reconstruct the secret. It also ensures that no information about the secret value can be recovered with k or less shares.

To achieve this, each party's share is represented by a distinct point in a Galois field in the set $\{(x_1, y_1), \cdots, (x_n, y_n)\} \subseteq (\mathbb{Z}_p)^2$, where p is prime [27]. All of the points must be chosen such that they lie on the same degree-bound k polynomial f. The secret value is chosen to be $f(0)$, or the resulting polynomial evaluated at zero. Then by using Lagrange interpolation, any collection of $k+1$ shares can uniquely determine the polynomial f and recover the secret value. This is an essential component for the 4-to-2 garbled row reduction (GRR) scheme in Subsection 4.5.

2.4.7 Oblivious Transfer. An *oblivious transfer* (OT) scheme allows one party to select inputs from another party without revealing which were chosen. For 1-out-of-n OT, a sender has n input messages $\{M_1, \cdots, M_n\}$ and a receiver has a number $c \in \{1, \cdots, n\}$ corresponding to the index of the message they would like to learn [8]. At the end of the protocol, the receiver should learn the message M_c and nothing else, while the sender should learn nothing. This is a fundamental component of Yao's garbled circuit protocol discussed in Section 3.

3 YAO'S GARBLED CIRCUIT PROTOCOL

3.1 Protocol Overview

Yao's garbled circuit protocol enables secure two-party computation as outlined in Subsection 2.1. To achieve this, it requires modeling the desired function as a Boolean circuit and then masking the inputs and outputs of each gate so that the party evaluating the circuit cannot discern any information about the function inputs or intermediate wire values in the circuit [30].

Suppose two parties, Alice and Bob, are trying to compute a function f whose Boolean representation is given in Figure 1. To carry out the garbled circuit protocol, the parties must assume the role of either the *garbler* or the *evaluator*. In this example, assume Alice is the garbler and Bob is the evaluator. Alice's input includes bits a and c and Bob's input includes bits b and d. There are four main stages to the protocol [5]:

- *Garbling:* Alice (the garbler) generates a *garbling* of the circuit following the procedure specified in Subsection 3.3. This includes generating *wire labels* described in Subsection 3.2 that are used to garble input values. Figure 2 depicts a garbling of the circuit from Figure 1.
- *Input Transfer:* Alice sends the garbling of the circuit and her garbled input values to Bob (the evaluator). She then uses oblivious transfer to send Bob his garbled input values without her discovering his underlying input values. The oblivious transfer protocol is summarized in Subsection 2.4.7 and its relevance to garbled circuits is discussed in Subsection 3.4.
- *Evaluation:* Bob evaluates the garbling of the circuit using both garbled inputs. Details for how the garbled circuit is evaluated are included in Subsection 3.5.
- *Output Reveal:* Bob sends Alice the garbled output value to decipher and she returns the final output of the function $f(a, b, c, d)$ to Bob.

The above protocol uses a semi-honest model which is most common. Modifications to make the protocol more robust and protect against other types of adversaries are mentioned in Subsection 5.3.3.

3.2 Garbled Wires

The garbler is responsible for generating a pair of *wire labels* for every wire in the circuit. Each wire label is a uniformly random string corresponding to one of the raw truth values on the wire, namely 0 or 1. For example in Figure 2, A_0 is the wire label corresponding to 0 for the a wire and A_1 corresponds to 1 respectively. For simplicity, all wire labels in this paper will follow this mapping. However, in practice, this should be random for each wire label to mask the underlying truth value from the evaluator.

3.3 Garbled Gate Generation

Garbled gates maintain the semantics of a logic gate by mapping input wire labels to output wire labels, rather than with raw wire values. Inside of each garbled gate, the output wire labels are encrypted with input wire labels as keys using a symmetric encryption scheme, resulting in four ciphertexts per gate. In this way, both input wire labels are needed to decipher the corresponding output wire label. There are many ways to perform encryption using two keys, but one of the most intuitive formulas is $c \leftarrow Enc_{k_2}(Enc_{k_1}(m))$. More efficient schemes are mentioned in Subsection 5.3.1.

An example computation of garbled gates is illustrated in Figure 2. The collection of garbled gates at the bottom is typically referred to as a *garbled circuit* or a *garbling*. Although not depicted in the figure for simplicity, it is imperative that the rows of each garbled gate are randomly permuted. Otherwise, the evaluator could deduce the garbler's input based on which row successfully decrypts. This is because the gate types are shared between the two parties and their respective truth tables are well-known.

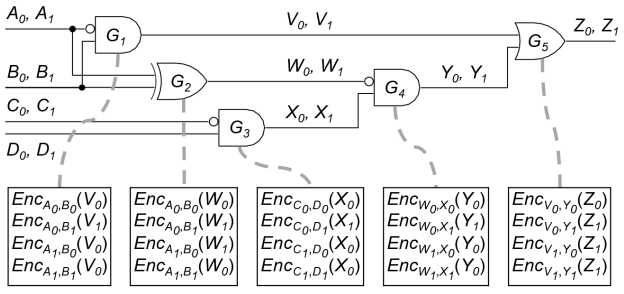


Figure 2: Garbling of the circuit in Figure 1. Figure adapted from Mike Rosulek's presentation [26].

3.4 Input Transfer

Since generation of the garbled gates is performed by the garbler, they possess the mapping between wire labels and raw wire values, hence why the evaluator must evaluate the circuit. However, the evaluator should not learn both wire labels in a particular pair, as they could then decrypt additional rows of the truth table and learn undisclosed information about the garbler's inputs. As a result, oblivious transfer is employed to allow the evaluator to learn only the wire labels corresponding to their own input values without revealing any information to the garbler. Additionally, the garbler sends all of the garbled gates and their own garbled inputs to the evaluator to perform the circuit evaluation.

3.5 Circuit Evaluation

Once the evaluator receives the garbled inputs and garbled circuit, they can perform the evaluation by decrypting the proper ciphertext gate-by-gate in *topological order*. In this scenario, a topological ordering means that if the output of a gate G_1 is input to another gate G_2, then G_1 must be evaluated before G_2 [5]. In Figure 2, this could be done in ascending order from G_1 to G_5.

In Yao's original proposal, the evaluator must try to decrypt all four ciphertexts in a garbled gate to discover which one is successful. However, Subsection 4.2 describes an alternate method to eliminate this heuristic.

3.6 Security

There are three main security parameters of a garbling scheme [4]:

- *Privacy:* The garbled circuit, garbled input, and decoding information reveal nothing besides the output of the function f modeled by the garbled circuit.
- *Obliviousness:* The garbled circuit and garbled input alone reveal no information other than underlying details about the circuit or input that the protocol does not intend to hide, such as topology or size.
- *Authenticity:* Given a valid garbled circuit and garbled input, it should be computationally infeasible for an adversary to construct a garbled output that is not authentic.

4 GARBLED CIRCUIT OPTIMIZATIONS

4.1 Parameters

There are three parameters related to Yao's protocol that researchers have focused on optimizing [17]:

- *Size:* The *size* of a garbled circuit is the number of ciphertexts in each garbled gate. Size is typically considered the primary parameter to optimize due to the communication complexity cost of garbled circuits. Often times reducing size at the expense of the other two parameters is preferable [17].
- *Computation Time:* Computational complexity is measured by the number of calls to the underlying protocol used for encryption and decryption, typically either PRFs or hash functions.
- *Hardness Assumption:* Due to their construction, certain schemes may require stricter security assumptions, resulting in a less robust protocol. Some schemes outlined in this paper rely upon the hardness assumption of hashes or PRFs, while others, such as Free XOR, require a stronger circularity assumption [7].

Table 1 shows a brief history of optimizations to garbled circuits, with an emphasis on those that reduce the size.

4.2 Point and Permute

As indicated in Subsection 3.5, Yao's original proposal required that the evaluator try to decrypt all four ciphertexts in a garbled gate to discover which one is successful. The aim of *point and permute* (P&P) is to inform the evaluator which one of the ciphertexts in a gate must be decrypted without revealing the truth value of any inputs or outputs [2].

To accomplish this, each wire label is first assigned a *color bit*, or *select bit*, with value either 0 or 1. The color bits must be different between both wire labels in a pair. Next, rather than randomly permuting the ordering of rows in the garbled gate, a canonical ordering for every gate in the circuit is chosen based on the color bits. For example, in Figure 3, the canonical ordering is ●●, ●●, ●●, ●●. Finally, during evaluation, the evaluator only needs to decrypt the row indexed by the color bits of the given wire labels. This method does not reduce the number of ciphertexts, but does reduce the computational cost of decryption to one.

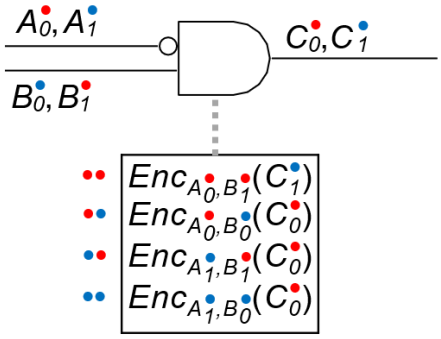


Figure 3: Point and permute on a NAND gate. Figure adapted from Mike Rosulek's presentation [26].

4.3 4-to-3 Garbled Row Reduction

Garbled row reduction (GRR) allows the elimination of one ciphertext through careful selection of wire labels [21]. Normally, the garbler chooses wire labels uniformly at random. However, it is only important that the labels be pseudo-random, meaning indistinguishable from those chosen uniformly at random [27]. Therefore, the garbler can choose an output wire label such that the first row in the garbled gate reduces to a constant provided that it remains pseudo-random.

For example, in Figure 3, the garbler can set $C_1^\bullet := Dec_{A_0^\bullet, B_1^\bullet}(0^n)$, where n is the size of the wire labels. This causes the first row of the garbled gate to become 0^n. The output wire label is still pseudo-random under the assumption that the encryption scheme is secure. As a result, the first row of the garbled gate need not be sent since if it were the proper ciphertext to decrypt, the evaluator would know in advance that the resulting output wire label is 0^n.

Scheme	GC Size (κ bits / gate) AND	XOR	Calls to H per gate: Garbler AND	Garbler XOR	Evaluator AND	Evaluator XOR	Hardness Assumption
Unoptimized Textbook Yao [32]	8	8	4	4	2.5	2.5	PRF
Point & Permute [2]	4	4	4	4	1	1	PRF
4 ⟶ 3 Row Reduction [21]	3	3	4	4	1	1	PRF
Free XOR [18]	3	0	4	0	1	0	CCR
4 ⟶ 2 Row Reduction [22]	2	2	4	4	1	1	PRF
FleXOR [17]	2	{0,1,2}	4	{0,2,4}	1	{0,1,2}	CCR
Half Gates [33]	2	0	4	0	2	0	CCR
Slice & Dice [28]	1.5	0	≤ 6	0	≤ 3	0	CCR

Table 1: Comparison of efficient garbling schemes. Gate size ignores small constant additive terms. Table adapted from Rosulek and Roy [28].

4.4 Free XOR

One of the greatest advancements in garbled circuit optimizations is free XOR that eliminates the need for any ciphertext transmission or cryptographic calculation for XOR gates [18, 5]. Rather than choosing pairs of wire labels independently and uniformly at random, the garbler chooses the first label randomly and offsets the second label by an constant Δ common to all wire labels. As long as Δ remains secret and unknown to the evaluator, then the resulting wire labels are still pseudo-random.

Note that if all wire label pairs are offset by the same constant Δ, then an XOR operation can be performed merely by computing the XOR of the input wire labels themselves. This can be verified by the truth table in Figure 4. Free XOR causes the transmission and computation cost of XOR gates to be completely free for both the garbler and the evaluator. Performance can be improved further by restructuring circuits to replace existing non-XOR gates with additional XOR gates that result in the same circuit behavior. Another benefit is that this scheme is compatible with 4-to-3 garbled row reduction. However, free XOR requires a stronger hardness assumption entitled circular correlation robustness (CCR) [7]. This is because the secret offset Δ is used in both the secret key and input of the chosen encryption scheme when garbling non-XOR gates.

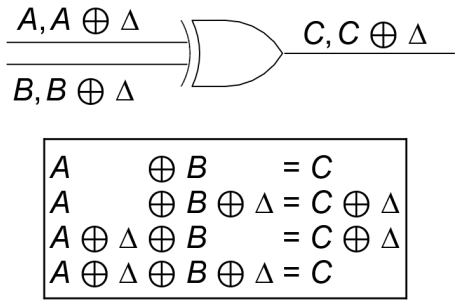
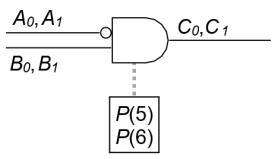


Figure 4: An XOR gate with the same offset Δ between wire label pairs. To transform this scheme into free XOR, set C := A ⊕ B and the truth table at the bottom verifies correctness. Figure adapted from Mike Rosulek's presentation [26].

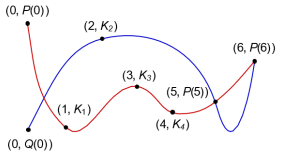
4.5 4-to-2 Garbled Row Reduction

This second form of garbled row reduction eliminates two ciphertexts by employing Shamir's secret sharing protocol outlined in Subsection 2.4.6 [22]. Four keys K_1, K_2, K_3, and K_4 are derived from the input wire labels to form secret shares. These keys are computed as $K_i := Dec_{A_j,B_k}(0^n)$, where $i \in \{1,2,3,4\}$ and $j,k \in \{0,1\}$. For simplicity, keys are indexed based on their row in the garbled gate, so K_1 corresponds to input wire labels A_0 and B_0, K_2 to A_0 and B_1, and so on. In Figure 5, based on the truth table for a NAND gate, the garbler groups K_1, K_3, and K_4 such that if the evaluator obtains one of these keys, they learn C_0. The garbler does something similar for K_2 and C_1. Note that the evaluator should only be able to derive one of these keys.

Using Lagrange interpolation, $(1, K_1)$, $(3, K_3)$, and $(4, K_4)$ form a unique degree-bound 2 polynomial P in the finite field GF(2^n). Note that any three points form a unique degree-bound 2 polynomial here, including the polynomial Q derived by interpolating $(2, K_2)$, $(5, P(5))$, and $(6, P(6))$. The key idea is that if the garbler sets $C_0 := P(0)$ and $C_1 := Q(0)$, then the proper output wire label can be found by interpolating through $P(5)$, $P(6)$, and the point corresponding to the key computed by the evaluator. In other words, once the evaluator interpolates and computes either P or Q, they can evaluate it at 0 and obtain the proper output wire label. Although this reduces the number of ciphertexts to two, it is not compatible with free XOR since the garbler cannot guarantee $C_0 \oplus C_1 = \Delta$.



(a) The garbled NAND gate with ciphertexts P(5) and P(6).



(b) Plots of the polynomials P and Q obtained from keys K_1, K_2, K_3, and K_4. Note that this plot is over the finite field GF(2^n).

Figure 5: 4-to-2 garbled row reduction on a NAND gate. Figure adapted from Mike Rosulek's presentation [26].

4.6 FleXOR

Kolesnikov et al. incorporated the underlying principles behind free XOR while maintaining compatibility with 4-to-2 garbled row reduction by relaxing certain constraints [17]. This was done using a gadget derived from a unary identity gate illustrated in Figure 6. This gadget merely transfers the offset of the input wire label from Δ_1 to Δ_2 in the output. Since this gate is compatible with garbled row reduction, it only costs one ciphertext.

Naively speaking, suppose that an XOR gate has pairs of wire labels all with different offsets Δ_1, Δ_2, and Δ_3. Transferring $\Delta_1 \rightarrow \Delta_3$ and $\Delta_2 \rightarrow \Delta_3$ costs two ciphertexts in total, so this scheme is no better than 4-to-2 garbled row reduction. However, if either Δ_1 or Δ_2, or both, were already equal to Δ_3, then the number of ciphertexts required would be one or zero.

This produces a combinatorial optimization problem where the garbler attempts to select wire offsets that minimize the total cost of XOR gates while also being subject to compatibility with 4-to-2 garbled row reduction for AND gates. Although the actual cost of an XOR gate varies from 2 to 0, Kolesnikov et al. found through applying certain heuristics that FleXOR was more efficient than free XOR in most cases [17].

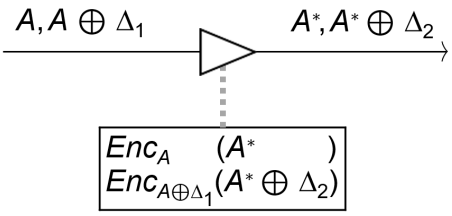


Figure 6: Unary identity gate transferring the wire label offset from Δ_1 to Δ_2. Note that this is compatible with garbled row reduction, so only one ciphertext need be sent. Figure adapted from Mike Rosulek's presentation [26].

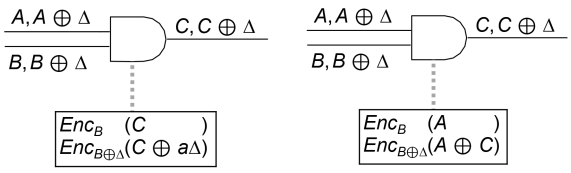
4.7 Half Gates

Half gates were the first technique compatible with free XOR that only require two ciphertexts per garbled AND gate [33]. The key idea is that only one ciphertext is needed per garbled AND gate if a party were to somehow know the truth value on one of the input wires. This can be broken down into two cases for each of the parties:

- *Garbler:* In Figure 7(a), the garbler is assumed to know in advance the truth value on the A input wire. If the underlying wire value a is 0, then both values of b result in c being 0, corresponding to the output wire label C. If a is 1, then the output wire label will either be C if b is 0 or $C \oplus \Delta$ if b is 1. These four cases are reduced to the truth table at the bottom of Figure 7(a). Since this is compatible with garbled row reduction, only one ciphertext need be sent in this case.
- *Evaluator:* Similarly in Figure 7(b), the evaluator is assumed to know the truth value on the B input wire. If b is 0, this results in c being 0, corresponding to wire label C. If b is 1, then it suffices to learn $A \oplus C$ to determine the proper output wire label. This is done by computing the XOR between $A \oplus C$ and the input wire label corresponding to a. For example, if the evaluator is given the input wire label A, then $A \oplus (A \oplus C) = C$ and is the proper output wire label since a is 0. Otherwise, if the evaluator is given $A \oplus \Delta$, then $(A \oplus \Delta) \oplus (A \oplus C) = C \oplus \Delta$ which is correct. Again, this is compatible with garbled row reduction, so only one ciphertext need be sent.

To combine these two AND half gates, Zahur et al. recognized that $a \wedge b = ((a \oplus r) \wedge b) \oplus (r \wedge b)$, where r is a random bit assigned by the garbler [33]. If the evaluator were allowed to learn the value $a \oplus r$, then the left-hand side of $(a \oplus r) \wedge b$ in the previous equation would be known to the evaluator. Similarly, since the garbler chose the bit r, they would know the left-hand side of $r \wedge b$.

To satisfy these requirements, r is chosen to be the color bit of the wire label A, or whichever corresponds to $a = 0$, and this association is known to the garbler. This allows the evaluator to learn $a \oplus r$ since it is the color bit of the wire label given to them, namely A or $A \oplus \Delta$. Therefore, by using the half gates in Figure 7, these two AND operations cost one ciphertext each, and the XOR operation is free by applying free XOR, making the overall cost two ciphertexts. This method remained the most efficient in terms of size for six years until Rosulek and Roy published their slice and dice scheme in 2021 [28].



(a) The garbler is assumed to know in advance the truth value on the A input wire.

(b) The evaluator is assumed to know in advance the truth value on the B input wire.

Figure 7: Two garbled AND half gates. Note that both are compatible with garbled row reduction, so only one ciphertext need be sent in each case. Figures adapted from Mike Rosulek's presentation [26].

4.8 Slice and Dice

Rosulek and Roy made a breakthrough discovery in 2021 by approaching garbled circuits with a linear algebraic perspective [28]. By including the additional hash $H(A_i \oplus B_j)$, where the evaluator has the input wire labels A_i and B_j, the resulting garbled gate ciphertext formulas can exploit further redundancies compared to half gates and reduce the number of ciphertexts required to 1.5.

In particular, they formulate a new technique called *slicing* that splits each wire label into two pieces: a left portion and a right portion. For a wire label C_{ij} in the case with input wire labels A_i and B_j, the half wire labels are calculated as $C_{ijL} \coloneqq H(A_i) \oplus H(A_i \oplus B_j)$ and $C_{ijR} \coloneqq H(B_j) \oplus H(A_i \oplus B_j)$. Note that the evaluator is not able to obtain any output wire label halves associated with cases other than the given A_i and B_j.

On the other hand, *dicing* is unique to the linear algebraic perspective and is essential to obfuscate plaintext inputs from the evaluator inside of the garbled gate ciphertext equations. Although this scheme reduces the size to 1.5 ciphertexts, it does increase the computational costs for garbling and evaluation.

5 APPLICATION TO PRIVATE SET INTERSECTION

5.1 Overview

Private set intersection (PSI) allows two parties to compute the intersection of their sets without revealing any information about items not in the intersection to the other party [24]. Circuit-based PSI schemes allow for the secure evaluation of arbitrary functions, expressed as either arithmetic or Boolean circuits. Compared to other PSI techniques, circuit-based methods have the advantage that the functionality of the protocol can easily be extended, without having to change the protocol or the security of the resulting protocol.

5.2 Prototype Implementation

For a practical application scenario of PSI, this work examines comparing two lists of emails hashed using `SHA256`. This simplifies the underlying circuit representation since the length of emails may differ between entries. The circuit used to compare the entries is based on a 256-bit comparator that is automatically generated based on the chosen hash size. For simplicity, an unoptimized textbook implementation of Yao's protocol is used. `Fernet` was chosen as the symmetric encryption scheme for Yao's protocol because of the multiple security mechanisms provided by its utilization. Namely, `Fernet` guarantees that encrypted messages cannot be manipulated or read without the key and offers higher security, trust, privacy, authenticity, authorization, data confidentiality, and integrity [1].

Due to the implementation simplicity and the communication bottleneck imposed by garbled circuits, comparing two lists containing 5 emails each takes 37s to compute on a standard desktop computer, averaging 1.48s per circuit computation. However. there exist many avenues for further optimization outlined in Subsection 5.3.

5.3 Future Optimizations

5.3.1 Protocol Design. The size of the garbled circuit is the bottleneck for most implementations, including the current one. An unoptimized textbook implementation of Yao's protocol is used for simplicity, but introducing one of the schemes outlined in Section 4 could significantly improve performance.

Furthermore, there exist other ways to perform symmetric encryption using two keys. The current prototype adopts the intuitive approach using $c \leftarrow Enc_{k_2}(Enc_{k_1}(m))$ and `Fernet` as the symmetric encryption scheme. However, Bellare et al. introduced an efficient scheme for garbling by fixing the key used in the block cipher [3]. This is done by instead encrypting a constant value and incorporating the input wire labels into the secret key. It was shown to provide a performance improvement of over 400x for garbling the `AES` circuit compared to previous methods.

5.3.2 Circuit Efficiency. Another source of inefficiency in the current prototype is related to the underlying comparison circuit. Since the emails may appear in any order, each entry in one list must be compared with all entries in the other list. This results in $O(n_1 n_2)$ comparisons, where n_1 and n_2 are the sizes of the two lists respectively. For simplicity, assume $n_1 = n_2 = n$, as placeholders can be introduced to make the list sizes equal, so the computational complexity becomes $O(n^2)$.

Huang et al. introduced the sort-compare-shuffle (SCS) circuit that first *sorts* both sets into a single sorted list, then *compares* all neighboring elements for equality, and finally *shuffles* the intersecting elements to hide any information that could be obtained from the resulting order [14, 24]. These three operations are all done obliviously such that both parties do not learn any information about elements outside of the intersection. Sorting can be achieved in $O(nlogn)$ and the other operations require less computation, resulting in an overall complexity of $O(nlogn)$.

More recently, Pinkas et al. proposed a new protocol for computing PSI variants [23]. In particular, each party first puts their input elements into bins according to a new Cuckoo hashing algorithm, and then the intersection is obtained by securely computing a Boolean comparison circuit over the bins. In terms of computational complexity, this scheme only requires $\omega(n)$ comparisons.

5.3.3 Security Assumptions. The current prototype relies upon the semi-honest adversary model. If increased security is desired for certain applications, *cut-and-choose* is a popular technique that plays a fundamental role in cryptographic protocol design, particularly for secure two-party computation in the malicious model [34]. To prevent the garbler from maliciously modifying the circuit and violating privacy and correctness, the garbler must create N copies of the garbled circuit and generate commitments to each of them. Then, the evaluator selects a subset of the circuits and sends them back to the garbler as a challenge. For the ones chosen, the garbler must reveal the committed labeling and prove that they decrypt correctly to ensure no unwarranted changes were made. Finally, if the revealed commitments are valid, then the evaluator can safely evaluate the remaining circuits and obtain the proper output. Researchers have shown that selecting half of the circuits to verify and evaluating the other half is optimal for ensuring a negligible probability of the evaluated circuits being malicious [34].

6 CONCLUSION

This paper detailed Yao's original garbled circuit protocol and outlined recent optimizations that have been made towards improving its communication and computational complexity. A basic prototype implementation of Yao's protocol applied to private set intersection was also introduced as an example library that can be expanded upon for improved efficiency or security. Several open problems pertaining to garbled circuits still remain, including further optimizing the size, computation cost, and hardness assumption compared to the slice and dice scheme by Rosulek and Roy [28]. Another area of research pertains to *privacy-free* garbled circuits in which only the authenticity property is required of the garbling scheme [12, 33].

ACKNOWLEDGMENTS

This work was performed in part for the National Science Foundation under the CyberCorps: Scholarship for Service (SFS) program. I would like to thank my mentor Andrew Miller for his guidance and support throughout this project as well as my principal investigator Masooda Bashir for affording me the opportunity to participate in the SFS program.

REFERENCES

[1] Guma Ali, Mussa Ally Dida, and Anael Elikana Sam. 2021. A secure and efficient multi-factor authentication algorithm for mobile money applications. *Future Internet*, 13, 12, 299. DOI: 10.3390/fi13120299.

[2] D. Beaver, S. Micali, and P. Rogaway. 1990. The round complexity of secure protocols. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing* (STOC '90). Association for Computing Machinery, Baltimore, Maryland, USA, 503–513. ISBN: 0897913612. DOI: 10.1145/100216.100287.

[3] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. 2013. Efficient garbling from a fixed-key blockcipher. Cryptology ePrint Archive, Report 2013/426. https://ia.cr/2013/426. (2013).

[4] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. 2012. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (CCS '12). Association for Computing Machinery, Raleigh, North Carolina, USA, 784–796. ISBN: 9781450316514. DOI: 10.1145/2382196.2382279.

[5] Osman Biçer. 2017. Efficiency optimizations on yao's garbled circuits and their practical applications. (2017). DOI: 10.48550/ARXIV.1703.03473.

[6] Dan Bogdanov, Riivo Talviste, and Jan Willemson. 2011. Deploying secure multi-party computation for financial data analysis. *IACR Cryptol. ePrint Arch.*, 662. http://eprint.iacr.org/2011/662.

[7] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. 2012. On the security of the "free-xor" technique. In *Theory of Cryptography* (Lecture Notes in Computer Science). Vol. 7194. Springer, 39–53. DOI: 10.1007/978-3-642-28914-9_3.

[8] Tung Chou and Claudio Orlandi. 2015. The simplest protocol for oblivious transfer. In *Proceedings of the 4th International Conference on Progress in Cryptology – LATINCRYPT 2015 - Volume 9230*. Springer-Verlag, Berlin, Heidelberg, 40–58. ISBN: 9783319221731. DOI: 10.1007/978-3-319-22174-8_3.

[9] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. 1997. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding* (Lecture Notes in Computer Science). Vol. 1233. Springer, 103–118. DOI: 10.1007/3-540-69053-0_9.

[10] Joan Daemen and Vincent Rijmen. 2002. *The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography)*. (1st ed.). Springer. ISBN: 3540425802.

[11] Ivan Damgård. 1998. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998* (Lecture Notes in Computer Science). Ivan Damgård, (Ed.) Vol. 1561. Springer, 63–86. DOI: 10.1007/3-540-48969-X_3.

[12] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. 2015. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II* (Lecture Notes in Computer Science). Elisabeth Oswald and Marc Fischlin, (Eds.) Vol. 9057. Springer, 191–219. DOI: 10.1007/978-3-662-46803-6_7.

[13] Brett Hemenway, Steve Lu, Rafail Ostrovsky, and William Welser IV. 2016. High-precision secure computation of satellite collision probabilities. Cryptology ePrint Archive, Report 2016/319. https://ia.cr/2016/319. (2016).

[14] Yan Huang, David Evans, and Jonathan Katz. 2012. Private set intersection: are garbled circuits better than custom protocols? In *NDSS*.

[15] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. 2010. Constant-size commitments to polynomials and their applications. In *ASIACRYPT* (Lecture Notes in Computer Science). Masayuki Abe, (Ed.) Vol. 6477. Springer, 177–194. ISBN: 978-3-642-17372-1. http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2010.html#KateZG10.

[16] Neal Koblitz and Alfred J. Menezes. 2015. The random oracle model: a twenty-year retrospective. *Des. Codes Cryptography*, 77, 2–3, (Dec. 2015), 587–610. DOI: 10.1007/s10623-015-0094-2.

[17] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. 2014. Flexor: flexible garbling for xor gates that beats free-xor. Cryptology ePrint Archive, Report 2014/460. https://ia.cr/2014/460. (2014).

[18] Vladimir Kolesnikov and Thomas Schneider. 2008. Improved garbled circuit: free xor gates and applications. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II* (ICALP '08). Springer-Verlag, Reykjavik, Iceland, 486–498. ISBN: 9783540705826. DOI: 10.1007/978-3-540-70583-3_40.

[19] Yehuda Lindell. 2020. Secure multiparty computation. *Commun. ACM*, 64, 1, (Dec. 2020), 86–96. DOI: 10.1145/3387108.

[20] Yehuda Lindell and Benny Pinkas. 2008. Secure multiparty computation for privacy-preserving data mining. *J. Priv. Confidentiality*, 1.

[21] Moni Naor, Benny Pinkas, and Reuban Sumner. 1999. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce* (EC '99). Association for Computing Machinery, Denver, Colorado, USA, 129–139. ISBN: 1581131763. DOI: 10.1145/336992.337028.

[22] B. Pinkas, T. Schneider, N.P. Smart, and S. Williams. 2009. Secure two-party computation is practical. Cryptology ePrint Archive, Report 2009/314. https://ia.cr/2009/314. (2009).

[23] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. 2018. Efficient circuit-based psi via cuckoo hashing. In *EUROCRYPT (3)*. Springer, 125–157. DOI: 10.1007/978-3-319-78372-7_5.

[24] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2018. Scalable private set intersection based on ot extension. *ACM Trans. Priv. Secur.*, 21, 2, Article 7, (Jan. 2018), 35 pages. DOI: 10.1145/3154794.

[25] P. Rogaway and T. Shrimpton. 2009. Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. English. *Fast Software Encryption(FSE 2004), Lecture Notes in Computer Science, Springer-Verlag*, Vol. 3017, (July 2009). /brokenurl#web.cs.ucdavis.edu/~rogaway/papers/relates.pdf.

[26] Mike Rosulek. 2015. A brief history of practical garbled circuit optimizations. (2015). https://simons.berkeley.edu/talks/mike-rosulek-2015-06-09.

[27] Mike Rosulek. [n. d.] The joy of cryptography. https://joyofcryptography.com. (). https://joyofcryptography.com.

[28] Mike Rosulek and Lawrence Roy. 2021. Three halves make a whole? beating the half-gates lower bound for garbled circuits. Cryptology ePrint Archive, Report 2021/749. https://ia.cr/2021/749. (2021).

[29] Thomas Schneider. 2012. *Engineering Secure Two-Party Computation Protocols: Design, Optimization, and Applications of Efficient Secure Function Evaluation*. Springer Publishing Company, Incorporated. ISBN: 3642300413.

[30] Peter Snyder. 2014. Yao ' s garbled circuits : recent directions and implementations. In.

[31] Heribert Vollmer. 2010. *Introduction to Circuit Complexity: A Uniform Approach*. (1st ed.). Springer Publishing Company, Incorporated. ISBN: 3642083986.

[32] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 162–167. DOI: 10.1109/SFCS.1986.25.

[33] Samee Zahur, Mike Rosulek, and David Evans. 2015. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT (2)*. Springer, 220–250. DOI: 10.1007/978-3-662-46803-6_8.

[34] Ruiyu Zhu, Yan Huang, Jonathan Katz, and Abhi Shelat. 2016. The cut-and-choose game and its application to cryptographic protocols. In *Proceedings of the 25th USENIX Conference on Security Symposium* (SEC'16). USENIX Association, Austin, TX, USA, 1085–1100. ISBN: 9781931971324.