Non-Interactive MPC with Trusted Hardware Secure Against Residual Function Attacks

Ryan Karl, Timothy Burchfield, Jonathan Takeshita, and Taeho Jung

University of Notre Dame, Notre Dame IN 46556, USA
{rkarl,tburchfi,jtakeshi,tjung}@nd.edu

Abstract. Secure multiparty computation (MPC) has been repeatedly optimized, and protocols with two communication rounds and strong security guarantees have been achieved. While progress has been made constructing non-interactive protocols with just one-round of online communication (*i.e.,* non-interactive MPC or NI-MPC), since correct evaluation must be guaranteed with only one round, these protocols are by their nature vulnerable to the residual function attack in the standard model. This is because a party that receives a garbled circuit may repeatedly evaluate the circuit locally, while varying their own inputs and fixing the inputs of others to learn the values entered by other participants. We present the first MPC protocol with a one-round online phase that is secure against the residual function attack. We also present rigorous proofs of correctness and security in the covert adversary model, a reduction of the malicious model that is stronger than the semi-honest model and better suited for modeling the behaviour of parties in the real world, for our protocol. Furthermore, we rigorously analyze the communication and computational complexity of current state of the art protocols which require two rounds of communication or one round during the online-phase with a reduced security requirement, and demonstrate that our protocol is comparable to or outperforms their complexity.

Keywords: Non-Interactive MPC · Communication round complexity · Trusted hardware

1 Introduction

Secure multiparty computation (MPC) is formally defined as functionality that allows a group of parties to jointly compute a function over their inputs, while keeping those inputs private. Two conditions must be satisfied: Correctness (the correct value must be computed from the given inputs) and Security (no information about the function's inputs should be gleaned after computation, other than the output). One of the primary tools for achieving this goal is the garbled circuit [36], where one party (the sender) encrypts a Boolean circuit and then assigns two randomly generated strings (labels) to each wire in the circuit: one each for 0 and 1. The sender also encrypts the output entry for each of the circuit's gate's truth tables so that the table can only be decrypted if a receiving party has the correct two input labels. A great deal of work has been invested into extending and optimizing MPC protocols to build more secure, efficient, and scalable MPC systems [27,37].Generally, modern MPC protocols are divided into three phases: the function-independent preprocessing phase, where parties do not need to know their inputs or the function to be computed, the function-dependent preprocessing phase, where parties know the function, but do not know their inputs, and the online phase, where parties evaluate the agreed function over their respective inputs [34].

Great progress has been made improving the computational complexity of these systems, but it is only recently that researchers have started to investigate improving these protocols' communication round complexity, or the minimum number of sets of parallel messages sent between parties in the protocol. For example, if during a protocol party A must wait to receive a message from party B before sending a followup message back to party B, we would consider this to be a two-round protocol. Note that in many cases, especially when MPC is conducted over the Internet, communication round complexity is the primary bottleneck, as network latency slows the delivery of packets necessary for continuing the protocol [5]. This problem becomes worse when parties are geographically distant, and is currently a major obstacle preventing MPC from being deployed in a global setting [35]. For several years, the total number of rounds needed has continued to decrease. To the best of our knowledge, the most efficient known protocols that satisfy security requirements in the standard model require two rounds of online communication [6,11–13,16,24,32].

In an effort to further reduce the number of rounds of communication, there has been a movement among MPC researchers to construct non-interactive MPC protocols (NI-MPC) which only require one-round of online communication [4,19–21]. Clearly, this would have many practical benefits, since it would allow participants in the protocol to immediately terminate communication as soon as they received the needed response from other participants, and not waste energy and other resources maintaining an Internet connection while awaiting further messages. However, all NI-MPC protocols with a one-round online phase are vulnerable to the residual function attack, and thus cannot guarantee input privacy of participants under the standard security model [26]. In this attack, because correct evaluation must be guaranteed with only one round, the party that receives a garbled circuit should be able to repeatedly evaluate the circuit locally on different inputs while fixing the inputs of others until they learn the values inputted by other participants.

Existing works [4,20,22] choose to relax the security of the standard model, and define a new model of security that allows adversaries to learn nothing more than what can be discovered via this attack, which they define as the "best-possible security" for any given set of corrupted parties. Intuitively, this means that the adversary is prevented from learning more than they can via the residual function attack, as this is, practically speaking, the most meaningful security that can be achieved in the standard model given the one-round constraint. In more formal terms, they assert that if the evaluator colludes with a set of corrupted parties, denoted T, it is allowed to learn the value of the original function on the honest parties inputs combined with every possible choice of inputs from set T. As long as nothing more than that is learned, their relaxed security model defines that the protocol is secure.

This paper aims to design an NI-MPC scheme constructed with only one communication round in the online phase without sacrificing security or privacy. Achieving this goal is challenging, because any MPC protocol with only one communication round in the standard model is vulnerable to the residual function attack described above. We address this by building our protocol using secure functionality available in trusted hardware (*e.g.,* TPM and Intel SGX). The TPM is a mature cryptoprocessor technology that has existed for over a decade and has been internationally standardized by the ISO, and Intel SGX is a set of instructions that protect application code and data from being disclosed or modified. Both types are widely available in consumer and enterprise systems. For the purposes of our protocol, we use various secure functionalities including a *monotonic counter, binding, sealing, and remote attestation. Monotonic counters* enable us to only permit certain steps of our protocol to be executed a finite number of times. Combining this with *binding and sealing* will limit the users' ability to perform arbitrary evaluation for launching the aforementioned residual function attack. We also make use of *remote attestation* to verify the integrity of the protocol. All functions are available in both TPM and Intel SGX.

Although there has been some controversy over the use of a TPM, which we address in a later section, TPMs are rapidly becoming a major part of the digital security and privacy ecosystem, having been deployed in hundreds of millions of devices and on almost all commercial PCs and servers [1]. The Intel SGX is a newer development, but it is maintained and supported by Intel, making it credible and trusted hardware for practical purposes.

We have the following contributions in this paper.

1. We propose the first NI-MPC protocol that is secure under standard security models even though there is only one communication round in the online phase.
2. We provide a comprehensive analysis of existing state-of-the-art MPC schemes which require two rounds of communication or one-round during the online-phase and demonstrate that our NI-MPC protocol is comparable to or outperforms their asymptotic complexity.
3. We prove that our NI-MPC protocol is secure by showing simulation-based security under standard models.

2 Related Work

Since BMR [3], there have been many advances made in improving the round complexity of MPC using garbled circuits. [29] combines the BMR protocol with the SPDZ protocol [10] to achieve a twelve-round protocol given certain assumptions concerning the adversary, which was later improved to six rounds after modifying the protocol to use SHE instead of SPDZ [30]. Later, [19] were able to achieve a four-round MPC protocol under standard polynomial-time hardness assumptions by utilizing a black-box proof of security. This was later combined with tamper-proof hardware tokens to achieve a three-round protocol [26]. Following this work, [5] presented a four-round protocol that can be optimized to three or two rounds after performing several precomputations.

Recently, there has been much interest in constructing two-round MPC protocols, as this was shown in [24] to be necessary to securely compute certain common functionalities. [11] achieved the first two-round MPC scheme by relying on indistinguishability obfuscation, but later this assumption was reduced to witness encryption [16]. Following this work, two-round protocols were achieved in [32] based on the learning with errors assumption (LWE), in [6] based on the DDH assumption, and in [13] based on bilinear maps, after using ideas from [9]. Currently, the protocol with the best known communication round complexity which makes the least assumptions is [14], which notably achieves a two-round protocol by only relying on oblivious transfer (OT). This protocol was later made more efficient by minimizing the number of public key operations required, but it still requires two communication rounds [12].

However, there has been less work constructing protocols with a one-round online phase. "One-round" protocols for the two-party setting [28] and the mobile agents setting [8] have been constructed, but their "one round" refers to sending two messages back and forth (*i.e.,* one round of exchange), so these protocols would be categorized as a two-round protocol with the current round definition.

Recently, there has been an interest in constructing Non-Interactive MPC (NI-MPC) protocols that have only one round of communication, but use a weaker security model that tolerates the residual function attack. [4] was the first to initiate study in this area, and notably achieved protocols for several special use cases such as group products, symmetric functions, etc. Later, [21] robustly studied the setting of NI-MPC to develop a unified framework for studying secure multi-party computation (MPC) under restricted interaction patterns, and went on to build more efficient NI-MPC protocols. These techniques were later improved [23] to concretely improve the communication and computational complexity of NI-MPC, after further developing a theory of the best-possible information theoretic security that could be achieved in this setting. This theory showed that for NI-MPC, since the communication strings for each player in a particular evaluation depend on each other, an adversary can prevent any simulator from generating views computationally indistinguishable from those in a real execution of the protocol by performing the attack. This makes proving security impossible for NI-MPC under standard security definitions using simulation proofs. Recently, [20] also proposed a protocol that notably achieved NI-MPC without the commonly assumed correlated randomness at the expense of relying on fully homomorphic encryption, which negatively impacts its overall efficiency. A summary of our comparison of different protocols is presented in Tables 1 and 2.[1]

[1] Note that the notation $O(1^\lambda)$ is used commonly in the literature to indicate the complexity grows linearly with respect to the security parameter [6, 12–14, 16, 32].

Table 1. Comparison of Existing Two-round MPC and our Protocol

Paper	Computation Complexity	Communication Complexity	Assumptions
[16]	$O((DL)^\omega)$ where L is the circuit depth, D is the dimension parameter of the matrix, and $\omega >= 2.3727$ is the matrix multiplication exponent	$O(q^\lambda)$ where q is the size of the input, and λ is the security parameter	Honest majority, a broadcast channel, point-to-point channels, and witness encryption
[6]	$O(M/\delta)$ where M is an upper bound on the difference between inputs and δ is an upper bound on the error probability	$O(S)+poly(\lambda)$ where S is the size of the circuit and λ is the security parameter	DDH Assumption, multiple servers, and Public Key Infrastructure
[12]	$O((nS\lambda)^k)$ where n is the number of parties, S is the size of the circuit, λ is the security parameter, and k is constant	$O(1^\lambda) + \Omega(S)$ where λ is the security parameter and S is the size of the circuit	2-round OT
[14]	$O((n\lambda)^k)$ where n is the number of parties, λ is the security parameter, and k is constant	$O(1^\lambda)$ where λ is the security parameter	2-round OT
[32]	$O((DL)^\omega)$ where L is the depth of the circuit, D is the dimension parameter of the matrix, and $\omega >= 2.3727$ is the matrix multiplication exponent	$O(Sm\lambda)^k$ where k is a constant, S is the size of the input, m is the size of the output, and λ is the security parameter	CRS model, broadcast channel, LWE, and NIZKs
[13]	$O((nS\lambda)^k)$ where n is the number of parties, S is the size of the circuit, λ is the security parameter, and k is constant	$O(1^\lambda) + \Omega(S)$ where λ is the security parameter and S is the size of the circuit	Standard Bilinear Map Assumptions
[11]	$O((DL)^\omega)$ where L is the depth of the circuit, D is the dimension parameter of the matrix, and $\omega >= 2.3727$ is the matrix multiplication exponent	$O(S^\lambda)$ where S is the size of the input, and λ is the security parameter	Indistinguishability obfuscation, CCA-secure public key encryption, NIZKs, and 1 honest party
Ours	$O(n^2S)$ where n is the number of parties and S is the size of the circuit	$O(1^\lambda)$ where λ is the security parameter	Trusted Hardware

3 Preliminaries

Both TPM and Intel SGX are equipped with the secure functionalities we need for constructing NI-MPC schemes that are secure under a standard (*i.e.*, not

Table 2. Comparison of Existing NI-MPC and our Protocol

Paper	Computation Complexity	Communication Complexity	Assumptions
[4]	Polynomial in the communication complexity	$O(n^t)$ where n is the number of parties and t is a constant $0 \leq t \leq n$	Correlated randomness
[23]	$O(\binom{n}{n/2}n)$ where n is the number of parties	$O(nsA)$ where n is the number of parties, s is a random vector in field F^k where k is a constant, and A is the number of AND gates	One-way functions for reusable correlated randomness and non-interactive key exchange for PKI setup
[21]	$O(2^n)$ where n is the number of parties	$O(n2^n)$ where n is the number of parties	Fully homomorphic encryption and indistinguishibility obfuscation for general circuits
[20]	$O((DL)^\omega)$ where L is the depth of the circuit, D is the dimension parameter of the matrix, and $\omega >= 2.3727$ is the matrix multiplication exponent	$O(Sm\lambda)^k$ where k is a constant, S is the size of the input, m is the size of the output, and λ is the security parameter	PKI and a common random string
Ours	$O(n^2S)$ where n is the number of parties and S is the size of the circuit	$O(1^\lambda)$ where λ is the security parameter	Trusted Hardware

* Ours is the only scheme secure against residual function attacks under standard security models.

relaxed) security model. We chose to rely on the TPM in this paper for the implementation, because of its availability. Almost all laptops and desktops are equipped with a TPM, and it is even compatible with embedded systems. Because our scheme relies on the functionality, rather than the hardware itself, it can be implemented using Intel SGX as well.

3.1 GNIOT for Non-interactivity and Covert Security

To achieve a protocol with a one-round online phase, our protocol relies on a special Oblivious Transfer (OT) called Generalized Non-Interactive Oblivious Transfer (GNIOT), proposed in [18], which makes use of the monotonic counter. Traditional OT allows a sender to safely transfer one of potentially many pieces of information to a recipient, but the sender cannot determine which piece was transferred. This idea was first proposed by Rabin in 1981 [33] for the two-party case, but in the years following has been extended to support multiple parties, and transferring more than one piece of information [7]. Traditional OT requires two or more communication rounds, but GNIOT requires just one round for the multiparty case.

Besides, with GNIOT, users are unable to receive more than one valid input for each input wire, making it impossible for malicious users to publish messages with fake inputs to others while locally evaluating the circuit with true inputs.

3.2 Justification of Using TPMs

TPMs have been underutilized when designing cryptographic protocols, due to impressions that they are insecure, an undesirable assumption to make, or simply too difficult to use [1]. TPMs have gained a reputation of being insecure, partially due to notable security breaches of TPM 1.2 [15]. However, a new and patched version (TMP 2.0) was released in 2015 with an updated specification that avoids the shortcomings of its predecessor. In 2017, an attack was reported against TPM 2.0, but this attack was only successful against an improperly implemented code library developed by Infineon , and did not exploit any underlying weakness in the TPM 2.0 specification itself. There are no known threats against the TPM 2.0.

While standard algorithmic assumptions (DDH, LWE, *etc.*) are preferable, since they do not impose hardware requirements, certain functionality cannot be supported in the standard model without relying on secure hardware. We argue it is worthwhile to make this assumption to support the computation of many useful functions in certain settings, *e.g.,* where a one-round online phase is desirable, if there are no known alternatives. Some of the functionality that cannot be supported without relying on secure hardware include: unconditional and non-interactive secure computation for one-time programs against malicious adversaries, interactive secure computation from stateless tokens based on one-way UC-secure functions, and program obfuscation from stateless tokens against malicious adversaries [17].

3.3 Definitions

Adversary Model When considering weaknesses in our protocol, we consider three types of adversarial behavior: a *semi-honest* adversary, *covert* adversary, and *fully malicious* adversary. A *semi-honest* adversary will not deviate from behavior prescribed by the protocol, though they may carry out local computation to attempt to gain information about other parties' private inputs. The semi-honest attacker model provides only weak guarantees of security (though in some situations more realistic), but allows more efficient cooperation. Conversely, a *fully malicious* adversary may deviate from a protocol in any way, and may attempt to carry out a wide range of malicious behavior. This behavior may include gaining information about other parties' private inputs, giving incorrect information to other parties, or even preventing the completion of the protocol. A protocol robust against fully malicious adversaries provides a strong security guarantee, but may be less efficient and more complex.

When discussing the security of our scheme, we use the *covert adversary* model as described in [2] to model users. Under the covert adversary model, while adversaries may behave in a fully malicious manner, they will refrain from deviating from the protocol if such an action would probably be noticed by other parties. In other words, a covert adversary will be only honest-but-curious unless they are likely to be able to behave maliciously with only a small chance of being detected.

Note that NI-MPC protocols cannot achieve active security against fully malicious adversaries as a result of the non-interactivity. Because each party sends all of their messages to the other parties in one round, if a malicious party chooses to send malformed data to other parties, the honest parties will not become aware of this until after they have sent their messages. In this way, the adversary can recover all of the data needed to complete the protocol successfully while preventing others from having access to enough valid data needed to complete the protocol. For our protocol, we thus find it most salient to consider the case where the computing parties may be covert adversaries and a trusted garbler is semi-honest. (If the trusted garbler is not semi-honest, then it becomes impossible to guarantee the security or correctness of the protocol.)

Simulation Correctness and Security We define correctness and security as a simulation as is commonly done in the literature [6, 12–14, 16, 32] so that we can use simulation based proof techniques later in Section 6. Note that because this is a protocol, and not an encryption scheme, techniques such as proving IND-CCA or IND-CPA do not directly demonstrate the security of the entire protocol.

Definition 1 (Correctness). *An MPC scheme π for a class of functions F is said to correctly compute F among players if, for any $f \in F$ and for any set of inputs $X := (x_1, \cdots, x_n)$ in the domain of f where the i-th player P_i controls x_i, all players receive $f(X)$ from the scheme with a probability not less than $1 - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$ and the security parameter λ.*

Definition 2 (Security). *An MPC scheme π for a class of functions F is said to be secure for F against covert adversary if, for any $f \in F$ and for any probabilistic polynomial time adversary $\mathcal{A}$ controlling a subset A of all players, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for any set of inputs $X := (x_1, \cdots, x_n)$ in the domain of f where the i-th player P_i controls x_i,*

$$\{\mathcal{S}(f(X), A, \{x_j \mid P_j \in A\})\}_\lambda \overset{c}{\equiv} \{\mathsf{View}^{\rho}_{A}(X)\}_\lambda$$

where $\overset{c}{\equiv}$ refers to computational indistinguishability, λ is the security parameter, and $\mathsf{View}^{\pi}_{A}(X)$ represents the messages received by members of A during the execution of protocol π and any cheating by a covert adversary can be detected with significant probability.

4 High-Level Description of Our Protocol

Figure 1 in Section 5 presents a formal description of the proposed protocol. We provide a high level description here. Our protocol relies on the monotonic counter functionality, a secure functionality that stores a non-negative integer which can only be read from or incremented [1]. With this, along with binding and sealing, we can limit the user's access to a public/private key pair stored on the trusted hardware to a finite number of times, after which the ability of users to make use of the keys to perform an action is revoked [26]. We also make use of the remote attestation functionality implemented with the Attestation Identity Key (AIK) in TPM. AIK is a special-purpose TPM-resident cryptographic key used to provide platform authentication and verify that users have not performed unauthorized changes to the software. By querying the TPM, we can certify that the software currently running on the device is in the presence of a cryptographic key that came from an identifiable piece of hardware that will function correctly. In the event this certification fails, we can deny a malicious party's access. Note that such a functionality is available in Intel SGX as well.

We assume that the parties have access to trusted hardware which supports a monotonic counter, and have agreed on the circuit to be evaluated C (we will say the circuit has $\mathcal{N}$ input wires). We denote the number of parties as n (we sometimes refer generally to party $\mathcal{P}_i$ for $i \in [1, n)$. To participate in our protocol, each party queries the on-board trusted hardware to generate a public/private key pair $(\mathcal{K}_{\mathbf{p}_i}, \mathcal{K}_{\mathbf{s}_i})$ stored in the secure memory that can only be used $\mathcal{W}_i$ times where $\mathcal{W}_i$ will denote the number of input wires a party controls. This behaviour can be enforced using the TPM by assigning an upper bound to the cryptographic keys that depends on the monotonic counter.

Each time the keys are used, the monotonic counter is incremented, but after the counter exceeds the assigned bound, users will no longer be able to use the keys. To certify that these keys were generated correctly, each party certifies it did not tamper with the key generation process by broadcasting a certification using the TPM's AIK of the public key. A semi-honest garbler also creates a symmetric key $\mathcal{R}^{(i)}$. After generating the garbled circuit, for each party $\mathcal{P}_i$ the garbler encodes both of $\mathcal{P}_i$'s possible inputs as wire labels (we work with Boolean circuits, so these labels correspond to 0 or 1) for each wire $w \in \mathcal{W}_i$ and encrypts using the symmetric key. Note that to an adversary, the wire labels appear to be random strings whose length is proportional to the security parameter, so the adversary cannot evaluate the garbled circuits without retrieving the correct keys from the TPM. The garbler then proceeds to split the symmetric key $\mathcal{R}^{(i)}$ into $\mathcal{W}_i$ secret shares. Following this, each encoded input is paired with a secret share of $\mathcal{R}^{(i)}$ as a tuple, referred to as an intermediate ciphertext. This intermediate ciphertext is encrypted using the public key $\mathcal{K}_{\mathbf{p}_i}$. Then each respective $\mathcal{P}_i$'s encrypted intermediate ciphertexts are then broadcast to them.

Because the number of decryptions permitted using the public key is equal to the number of input wires of the garbled circuit, due to the monotonic counter, each party can only decrypt one possible input for each wire. This means that no party can decrypt both encoded inputs corresponding to 0 and 1 for a wire and perform the residual function attack described above, as this will use up a decryption that they need to recover the input to one of the remaining input wires. They can only decrypt one encoded input per wire or they will be unable to complete the protocol. After decrypting the encrypted intermediate ciphertexts, the parties can locally combine the secret shares to recover the symmetric key $\mathcal{R}^{(i)}$ that was used to encrypt the encoded inputs to the circuit and recover the wire labels. Since the wire labels reveal nothing about a party's choice of input, they can be sent to the other parties, and be used by each party to locally evaluate the circuit to receive the output.

Note that NI-MPC protocols cannot achieve active security against fully malicious adversaries as a result of their non-interactivity as described above, because all parties send all of the data the other parties need to complete the protocol in one round simultaneously. The best that can be achieved is covert security, which models the situation where malicious adversaries are willing to cheat only if they are not caught. In our protocol, if an adversary sends malformed data to another honest party, the honest party will be unable to finish evaluating the circuit, but because all data sent can be traced back to the sending party with significant probability. The honest party will know who acted maliciously, and notify the other participating parties of the bad behaviour.

5 Our Protocol

Our protocol employs the following algorithms as building blocks: gen, enc_pub, enc_sec, dec_pub, dec_sec, and garble. Any algorithms that have the described input/output can be adopted.

- **gen**$(\lambda) \rightarrow \mathcal{R}, \mathcal{K}_\mathbf{p}, \mathcal{K}_\mathbf{s}$: this is an algorithm that takes the security parameter λ as input and outputs a symmetric key $\mathcal{R}$ and a public/private key pair ($\mathcal{K}_\mathbf{p}$, $\mathcal{K}_\mathbf{s}$). For example, RSA/ECC or AES key generation algorithms.
- **enc_pub**$(\mathcal{K}_\mathbf{p}, \mathcal{X}) \rightarrow \mathcal{PK}_{\mathcal{K}_\mathbf{p}}(\mathcal{X})$: this is a public key encryption algorithm that takes public key $\mathcal{K}_\mathbf{p}$ and plaintext $\mathcal{X}$ as input and returns ciphertext $\mathcal{PK}_{\mathcal{K}_\mathbf{p}}(\mathcal{X})$. For example, RSA or ECC encryption algorithms.
- **enc_sec**$(\mathcal{R}, \mathcal{X}) \rightarrow \mathcal{SK}_\mathcal{R}(\mathcal{X})$: this is a symmetric key encryption algorithm that takes symmetric key $\mathcal{R}$ and plaintext $\mathcal{X}$ as input and returns ciphertext $\mathcal{SK}_\mathcal{R}(\mathcal{X})$. For example, the AES encryption algorithm.
- **dec_pub**$(\mathcal{K}_\mathbf{s}, \mathcal{PK}_{\mathcal{K}_\mathbf{p}}(\mathcal{X})) \rightarrow \mathcal{X}$: this is a public key decryption algorithm that takes private key $\mathcal{K}_\mathbf{s}$ and ciphertext $\mathcal{PK}_{\mathcal{K}_\mathbf{p}}(\mathcal{X})$ as input and returns plaintext $\mathcal{X}$. For example, the RSA or ECC decryption algorithms.
- **dec_sec**$(\mathcal{R}, \mathcal{SK}_\mathcal{R}(\mathcal{X})) \rightarrow \mathcal{X}$: this is a decryption algorithm that takes symmetric key $\mathcal{R}$ and ciphertext $\mathcal{SK}_\mathcal{R}(\mathcal{X})$ as input and returns plaintext $\mathcal{X}$. For example, the AES decryption algorithm.
- **garble**$(C) \rightarrow GC$: this is a circuit garbling algorithm that takes as input a circuit C and returns a garbled circuit GC. For example, a garbling algorithm or related software tools (i.e. Frigate [31]) from the survey [25] may be used.

Our protocol is described in detail in Figure 1. Note that in the preprocessing phase, ordinarily our protocol would be vulnerable to the residual function attack, as an adversary could hypothetically decrypt more than one ciphertext pair ($\mathcal{C}_{w,0}$, $\mathcal{C}_{w,1}$) associated with an input wire to recover its associated symmetric key and gain access to both wire labels. This would allow them to evaluate the function repeatedly over both inputs while fixing the input of others, until they learn the values inputted by other participants. However, because the number of decryptions of the ciphertexts $\mathcal{C}_{w,0}$ or $\mathcal{C}_{w,1}$ is limited with the monotonic counter in the trusted hardware, if they attempt to perform more decryptions than specified for the circuit, they will be unable to access enough shares of the symmetric key to later recover $\mathcal{R}^{(i)}$ and complete the protocol. As a result, the residual function attack is blocked. Also, note that only steps 2 and 4 of the initialization phase and step 1 of the online phase require a communication round. However, the communication in the initialization phase only needs to occur once during setup. After this, communication only occurs during the online phase for each iteration of the protocol.

6 Proofs

6.1 Proof of Correctness and Security

Simulation Correctness Our protocol π correctly computes Boolean circuits as defined in Definition 1.

Proof. Recall that the circuit C is agreed upon prior to beginning the protocol. We note first that if the parties use their TPMs exactly as described in the protocol, they will use gen to generate a $\mathcal{W}_i$-time use count limited key pair $(\mathcal{K}_{\mathbf{p}_i}, \mathcal{K}_{\mathbf{s}_i})$. The parties will later send their $\mathcal{K}_{\mathbf{p}_i}$ to the "separate" garbler. (This step is verified during attestation as described in Figure 1.) The garbler is semi-honest, so they must correctly use the garbling function garble to convert the ordinary circuit C to a corresponding garbled circuit GC whose inputs map correctly to the corresponding outputs of the original circuit. Following this, the garbler takes the input wire labels $x_{w,(0,1)}$ corresponding to wires controlled by $\mathcal{P}_i$ and must encrypt them with symmetric key $\mathcal{R}^{(i)}$ using $\mathsf{enc\_sec}$, which is $\mathsf{enc\_sec}(x_{w,\mathbf{(0,1)}}) = \mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}})$. Then, the garbler must use the function $\mathsf{enc\_pub}$ to encrypt the wires with $\mathcal{P}_i$'s public key $\mathcal{K}_{\mathbf{p}_i}$ along with a perfectly secret share (*e.g.* Shamir's) of symmetric key $\mathcal{R}^{(i)}$ corresponding to the wire w, which is $\mathsf{enc\_pub}(\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}}), \mathcal{R}_w^{(i)}) = \mathcal{PK}_{\mathcal{K}p_i}(\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}}),\ \mathcal{R}_w^{(i)})$. The garbler then must broadcast this ciphertext as $\mathcal{C}_{w,(0,1)}$ to party $\mathcal{P}_i$ along with the garbled circuit GC. Note that some index information is encoded into the ciphertext so the recieving $\mathcal{P}_i$ will know which ciphertext corresponds to the boolean value 0 or 1. Because of the monotonic counter, the number of decryptions permitted using the public key is equal to the number of input wires of the garbled circuit, and each party can only decrypt one possible input for each wire. After receiving all ciphertexts $\mathcal{C}_{w,(0,1)}$, $\mathcal{P}_i$ chooses to decrypt the finite number they are permitted, that correspond to their desired input of 0 or 1 for each wire w using $\mathsf{dec\_pub}$, which is $\mathsf{dec\_pub}(\mathcal{C}_{w,(0,1)}) = (\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}}), \mathcal{R}_w^{(i)})$. From here $\mathcal{P}_i$ can combine the shares of $\mathcal{R}_w^{(i)}$ to recover $\mathcal{R}^{(i)}$ and use $\mathsf{dec\_sec}$ to decrypt $\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}})$ and correctly recover the proper wire label as $\mathsf{dec\_sec}(\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}})) = x_{w,\mathbf{(0,1)}}$.

Our Protocol π

Initialize: Given the security parameter 1^λ, this phase distributes garbled circuit GC and ciphertext pair $\mathcal{C}_{w,(0,1)}$ corresponding to each wire controlled by each player.

Preprocess: This phase can be run in advance of the online phase. All players $\mathcal{P}_i$ run this phase along with the garbler $\mathcal{G}$. Note steps 2 and 4 require one round each but this is a one time cost.

1. Each $\mathcal{P}_i$ calls gen and queries the TPM to generate a $\mathcal{W}_i$ time use count limited public/private key pair $(\mathcal{K}_{\mathbf{p}_i}, \mathcal{K}_{\mathbf{s}_i})$ where $\mathcal{W}_i$ is the number of input wires $\mathcal{P}_i$ controls for circuit C.
2. Each $\mathcal{P}_i$ certifies its $\mathcal{K}_{\mathbf{p}_i}$ by using an Attestation Identity Key via a Trusted Platform Module, and broadcasts $\mathcal{K}_{\mathbf{p}_i}$.
3. The semi-honest garbler $\mathcal{G}$, which is "separate" from the function evaluation, takes the previously agreed upon circuit C, calls garble, and computes a corresponding garbled circuit GC.
4. Using GNIOT, for every player $\mathcal{P}_i$ with $\mathcal{W}_i$ input wires and public key $\mathcal{K}_{\mathbf{p}_i}$, the semi-honest garbler $\mathcal{G}$ calculates a symmetric cipher key $\mathcal{R}^{(i)}$ and splits $\mathcal{R}^{(i)}$ into $\mathcal{W}_i$ shares $\mathcal{R}_w^{(i)}$ for $w \in [\mathcal{W}_i]$. Let the labels in the garbled circuit GC of an input wire $w \in [\mathcal{W}_i]$ be called $x_{w,0}$ and $x_{w,1}$ for the Boolean values 0 and 1 respectively. The garbler then calls enc_sec to encrypt the each label and enc_pub to encrypt each tuple of an encrypted label and a secret share, and computes and broadcasts $\mathcal{C}_{w,0} = \mathcal{PK}_{\mathcal{K}p_i}(\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{0}}),\ \mathcal{R}_w^{(i)})$ and $\mathcal{C}_{w,1} = \mathcal{PK}_{\mathcal{K}p_i}(\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{1}}),\ \mathcal{R}_w^{(i)})$ for all $w \in \mathcal{W}_i$.

Online: This phase communicates all parties' inputs to each individual party. All players $\mathcal{P}_i$ run this phase. Note this step requires one round.

1. Each $\mathcal{P}_i$ decrypts either $\mathcal{C}_{w,0}$ or $\mathcal{C}_{w,1}$ (for $w \in [\mathcal{W}_i]$) using their private key $\mathcal{K}_{\mathbf{s}_i}$ stored on their TPM to get intermediate ciphertexts $\mathcal{T}_{w,0} = (\ \mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{0}}),\ \mathcal{R}_w^{(i)})$ or $\mathcal{T}_{w,1} = (\ \mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{1}}),\ \mathcal{R}_w^{(i)})$, as part of the GNIOT, by calling dec_pub.
2. Then $\mathcal{P}_i$ extracts each $\mathcal{R}_w^{(i)}$ which are then recombined to recover the symmetric key $\mathcal{R}^{(i)}$.
3. Then $\mathcal{R}^{(i)}$ is used to decrypt either $\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{0}})$ or $\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{1}})$, by calling dec_sec, based on the choice of $\mathcal{P}_i$ to recover either $x_{w,0}$ or $x_{w,1}$, and complete the GNIOT.
4. The parties broadcast their chosen $x_{w,0}$ or $x_{w,1}$ to each party but the semi-honest garbler (wire labels appear as random strings to an adversary).

Evaluate (offline): This phase evaluates the garbled circuit. All players $\mathcal{P}_i$ run this phase locally.

1. Each party $\mathcal{P}_i$ inputs one of $x_{w,0}$ or $x_{w,1}$ for $w \in [\mathcal{W}_i]$ into the garbled circuit GC to reveal the output and return the plaintext circuit output.
2. Parties learn a specific input is corrupted when they input the $x_{w,0}$ or $x_{w,1}$ they receive from an adversarial party A to a wire A owns and evaluation fails for that wire. When this happens, they all abort the protocol and broadcast a notification to all other parties the $x_{w,0}$ or $x_{w,1}$ from A was corrupted.

Fig. 1. Our MPC protocol that evaluates a Boolean circuit among n players.

Then, as long as the garbler properly generated the garbled circuit GC, we know that $\mathcal{P}_i$ can input all $x_{w,(0,1)}$ into GC to recover the circuit's output.

Simulation Security Assuming $|\mathcal{C}| = O(\log \lambda)$, all parties use the TPM as the protocol describes, the public key and symmetric key encrypted ciphertexts supported by the TPM are indistinguishable, players properly perform attestation with their keys, and the semi-honest, noncolluding garbler correctly garbles circuits, our protocol π securely computes Boolean circuits as defined in Definition 2.

Proof. Without loss of generality, assume the adversary controls the first $m < n$ variables, where n is the total number of variables. We show that a probabilistic polynomial time simulator can generate an entire simulated view, given $y = f(z_1, \cdots, z_m, z_{m+1}, \cdots, z_n)$ and $z_1, \cdots z_m$ for an adversary indistinguishable from the view an adversary sees in a real execution of the protocol. Note the simulator is able to find $z'_{m+1}, \cdots, z'_n$ such that $y = f(z_1, \cdots, z_m, z'_{m+1}, \cdots, z'_n)$ in polynomial time since $|\mathcal{C}| = O(\log \lambda)$. Besides this, it follows the protocol as described in Figure 1, pretending to be the honest players. Note that in any world, the adversary can only evaluate what can be evaluated when other players' input is fixed (denoted as *intended evaluation point* hereafter), because the adversary can only access the wire labels for the intended evaluation point. Now the simulator $\mathcal{S}$ generates a view indistinguishable from that of a real execution, since all parameters broadcast i.e. $\mathcal{K}_{\mathbf{P}_i}$, $\mathcal{C}'_{w,(0,1)}$, $x'_{w,(0,1)}$ are indistinguishable from the corresponding ones in the real protocol i.e., $\mathcal{K}_{\mathbf{P}_i}$, $\mathcal{C}_{w,(0,1)}$, $x_{w,(0,1)}$ as they are generated identically and have the exact same distribution. However, note that each party broadcasts its $x'_{w,(0,1)}$ to all other parties in participating in the protocol. If a covert adversary transmits corrupted $x'_{w,(0,1)}$ to other players, the other parties will know precisely which $x'_{w,(0,1)}$ caused the protocol to abort, because during evaluation of the circuit the protocol can verify whether a $x'_{w,(0,1)}$ corresponds to a valid wire label for the wires of the circuit. Which wires are owned by which parties agreed upon beforehand, and so because all data sent can be traced back to the sending party who owns the wire, the honest party will know who cheated with significant probability. Recall from the protocol, every ciphertext $\mathcal{C}'_{w,(0,1)}$ is either opened or unopened. By the assumption of the security of the public key encryption provided by the TPM, unopened $\mathcal{C}'_{w,(0,1)}$ are indistinguishable. Recall that $\mathcal{K}_{\mathbf{P}_i}$ is only usable $\mathcal{W}_i$ times due to the monotonic counter. If the value of $\mathcal{W}_i$ is correctly calculated beforehand, each party will only be able to decrypt one intermediate ciphertext for each input wire $w \in \mathcal{W}_i$. Decrypting gives us the inner encryptions of wire labels $\mathcal{SK}_{\mathcal{R}^{(i)}}(x_{w,\mathbf{(0,1)}})'$ and R'_i. Note all R'_i are generated at random and will have the same random distribution as R_i. By the assumption of the security of the underlying symmetric key protocol, the encryptions of wire labels in both worlds will be indistinguishable from random, and thus indistinguishable from each other. Additionally, the wire labels of both worlds will also be indistinguishable by the assumption that the garbler correctly garbled the circuits and the fact that we chose inputs that give the same output.

This demonstrates that for the class of all functions F, our protocol is secure against covert adversaries A since:

$$\{\mathcal{S}(f(X), A, \{x_j \mid P_j \in A\})\}_\lambda \overset{c}{\equiv} \{\mathsf{View}^{\pi}_{A}(X)\}_\lambda$$

Therefore, the adversary cannot distinguish between real and simulated executions and our protocol securely computes Boolean circuits as defined in Definition 2.

7 Conclusions and Future Work

This paper demonstrates the first MPC scheme constructed with one communication round in the online phase that does not sacrifice security or privacy and can be proven secure in the standard model. Previous protocols subject to this one-round constraint in the standard model were vulnerable to the residual function attack, where a party that receives a garbled circuit may repeatedly evaluate the circuit locally while varying their own inputs and fixing the input of others to learn the values entered by other participants. We overcome this problem by building our protocol using a secure hardware primitive, specifically a Trusted Platform Module (TPM), a mature cryptoprocessor technology. We rigorously analyzed the communication and computational complexity of current state of the art protocols which require two rounds of communication or one-round during the online-phase with a reduced security requirement, and demonstrated that our protocol is comparable to or outperforms their complexity. Also, we provided rigorous proofs of correctness and security in the covert adversary model for our protocol. We are actively developing an implementation of the algorithms in our NI-MPC scheme with Microsoft's TPM 2.0 Simulator, and the MPIR, OpenSSL, and TPM.CPP libraries. Our code is available at `https://github.com/Ryan-Karl/one_round_mpc_with_tpm`. We hope that this further improves the viability of MPC as a practical solution for facilitating private communication, especially in global environments.

References

1. Arthur, W., Challener, D.: A practical guide to TPM 2.0: using the Trusted Platform Module in the new age of security. Apress (2015)
2. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Theory of Cryptography Conference. pp. 137–156. Springer (2007)
3. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: STOC. pp. 503–513. ACM (1990)
4. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: International Cryptology Conference. pp. 387–404. Springer (2014)
5. Ben-Efraim, A., Lindell, Y., Omri, E.: Optimizing semi-honest secure multiparty computation for the internet. In: CCS. pp. 578–590. ACM (2016)

6. Boyle, E., Gilboa, N., Ishai, Y.: Group-based secure computation: Optimizing rounds, communication, and computation. In: Eurocrypt. pp. 163–193. Springer (2017)
7. Brassard, G., Crépeau, C., Robert, J.M.: All-or-nothing disclosure of secrets. In: Eurocrypt. pp. 234–238. Springer (1986)
8. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-round secure computation and secure autonomous mobile agents. In: ICALP. pp. 512–523. Springer (2000)
9. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: CRYPTO. pp. 33–65. Springer (2017)
10. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO, pp. 643–662. Springer (2012)
11. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure mpc from indistinguishability obfuscation. In: TCC. pp. 74–94. Springer (2014)
12. Garg, S., Miao, P., Srinivasan, A.: Two-round multiparty secure computation minimizing public key operations. In: CRYPTO. pp. 273–301. Springer (2018)
13. Garg, S., Srinivasan, A.: Garbled protocols and two-round mpc from bilinear maps. In: FOCS. pp. 588–599. IEEE (2017)
14. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Eurocrypt. pp. 468–499. Springer (2018)
15. Goodin: Ex-army man cracks popular security chip. The Register (2010), `theregister.co.uk/2010/02/17/infineon_tpm_crack/`
16. Gordon, S.D., Liu, F.H., Shi, E.: Constant-round mpc with fairness and guarantee of output delivery. In: CRYPTO. pp. 63–82. Springer (2015)
17. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: TCC. pp. 308–326. Springer (2010)
18. Gunupudi, V., Tate, S.R.: Generalized non-interactive oblivious transfer using count-limited objects with applications to secure mobile agents. In: FC. pp. 98–112. Springer (2008)
19. Halevi, S., Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Round-optimal secure multi-party computation. In: CRYPTO. pp. 488–520. Springer (2018)
20. Halevi, S., Ishai, Y., Jain, A., Komargodski, I., Sahai, A., Yogev, E.: Non-interactive multiparty computation without correlated randomness. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 181–211. Springer (2017)
21. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Proceedings ACM Conference on Innovations in Theoretical Computer Science. pp. 157–168. ACM (2016)
22. Halevi, S., Ishai, Y., Kushilevitz, E., Rabin, T.: Best possible information-theoretic mpc. In: Theory of Cryptography Conference. pp. 255–281. Springer (2018)
23. Halevi, S., Ishai, Y., Kushilevitz, E., Rabin, T.: Best possible information-theoretic mpc. In: Theory of Cryptography. Theory of Cryptography, vol. 11240, pp. 255–281. Springer (2018). https://doi.org/10.1007/978-3-030-03810-6_10
24. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: CRYPTO. pp. 132–150. Springer (2011)
25. Hastings, M., Hemenway, B., Noble, D., Zdancewic, S.: Sok: General purpose compilers for secure multi-party computation. In: SoK: General Purpose Compilers for Secure Multi-Party Computation. p. 0. IEEE (2019)
26. Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Composable security in the tamper-proof hardware model under minimal complexity. In: TCC. pp. 367–399. Springer (2016)

27. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security. pp. 35–35. SEC'11, USENIX Association, Berkeley, CA, USA (2011)
28. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free xor gates and applications. In: ICALP. pp. 486–498. Springer (2008)
29. Lindell, Y., Pinkas, B., Smart, N.P., Yanai, A.: Efficient constant round multi-party computation combining bmr and spdz. In: CRYPTO. pp. 319–338. Springer (2015)
30. Lindell, Y., Smart, N.P., Soria-Vazquez, E.: More efficient constant-round multi-party computation from bmr and she. In: TCC. pp. 554–581. Springer (2016)
31. Mood, B., Gupta, D., Carter, H., Butler, K., Traynor, P.: Frigate: A validated, extensible, and efficient compiler and interpreter for secure computation. In: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. pp. 112–127. IEEE (2016)
32. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key fhe. In: Eurocrypt. vol. 9666, pp. 735–763 (2016)
33. Rabin, M.: How to exchange secrets with oblivious transfer. Harvard University Technical Report (1981)
34. Wang, X., Ranellucci, S., Katz, J.: Authenticated garbling and efficient maliciously secure two-party computation. In: CCS. pp. 21–37. ACM (2017)
35. Wang, X., Ranellucci, S., Katz, J.: Global-scale secure multiparty computation. In: CCS. pp. 39–56. ACM (2017)
36. Yao, A.C.C.: How to generate and exchange secrets. In: FOCS. pp. 162–167. IEEE (1986)
37. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole. In: Eurocrypt. pp. 220–250. Springer (2015)