Security Against Honorific Adversaries: Efficient MPC with Server-aided Public Verifiability

Li Duan
Paderborn University
liduan@mail.upb.de

Yufan Jiang
Karlsruhe Institute of Technology
yufan.jiang@partner.kit.edu

Yong Li
Ruhr University Bochum
Yong.Li@rub.de

Jörn Müller-Quade
Karlsruhe Institute of Technology
joern.mueller-quade@kit.edu

Andy Rupp
University of Luxembourg
andy.rupp@uni.lu

***Abstract*—Secure multiparty computation (MPC) allows distrustful parties to jointly compute some functions while keeping their private secrets unrevealed. MPC adversaries are often categorized as semi-honest and malicious, depending on whether they follow the protocol specifications or not. Covert security was first introduced by Aumann and Lindell in 2007, which models a third type of active adversaries who cheat but can be caught with a probability. However, this probability is predefined externally, and the misbehavior detection must be made by other honest participants with cut-and-choose in current constructions. In this paper, we propose a new security notion called security against *honorific adversaries*, who may cheat during the protocol execution but are extremely unwilling to be punished . Intuitively, honorific adversaries can cheat successfully, but decisive evidence of misbehavior will be left to honest parties with a probability close to one. By introducing an *independent but not trusted* auditor to the MPC ideal functionality in the universal composability framework (UC), we avoid heavy cryptographic machinery in detection and complicated discussion about the probability of being caught. With this new notion, we construct new provably secure protocols without cut-and-choose for garbled circuits that are much more efficient than those in the covert and malicious model, with slightly more overhead than passively secure protocols.**

***Index Terms*—MPC, security notion, efficient protocols, honorific adversaries**

I. Introduction

In secure multiparty computation (MPC), n parties are willing to jointly compute a function $\mathcal{C}()$ without revealing their private input $\{x_1, \cdots, x_n\}$ to others. MPC protocols should guarantee that besides the output of the given function $\{y_1, \cdots, y_n\} = \mathcal{C}(x_1, \cdots, x_n)$, nothing else can be learned (privacy), and the output $\{y_1, \cdots, y_n\}$ is distributed correctly (correctness). In different settings, MPC protocols are designed against various types of adversaries, making trade-offs between efficiency and security. Up to now, three main categories of adversaries are considered and most prior works achieved provable security against one of them.

Semi-honest adversaries: these adversaries can be seen as protocol participants who do not violate the protocols but attempt to learn more than predefined output of functions.

Malicious adversaries: these adversaries may deviate arbitrarily from the protocol. In addition to the power of semi-honest adversaries, they can take any action to manipulate the result and messages. Protocols secure against malicious adversaries ensure that the participants can always detect misbehavior of its partner, and may abort the execution.

It is not hard to see that protocols secure against *semi-honest* adversaries offers quite limited privacy, while the ones with *malicious* security [21] [26] [29] are usually too inefficient for large-scale applications in practice [13].

Covert adversaries: these adversaries may actively cheat like malicious adversaries, but they can be caught with a constant probability $\mathcal{E}$ and they are afraid of being caught. Protocols secure against such adversaries allow successful cheating, but guarantee that honest parties can detect such behaviors with the given probability $\mathcal{E}$.

This notion can lead to more efficient protocols constructions [5] than malicious security. However, the existing instantiations in prior works [4] [5] [10] [12] [16] [18] [23] are still much heavier than semi-honest ones. On the other hand, it is extremely non-trivial in real world to quantify the probability $\mathcal{E}$ of being caught as a reasonable externality in the theoretical model.

Therefore, it remains prominent to explore other alternatives between the weakest and strongest security guarantee.

A. Security Against Honorific Adversaries

Instead of adopting security-with-abort [25], we take a fresh look at misbehavior detection and its consequence. The incentive for a rational party to behave honestly is the fear of being caught and punished. Intuitively, if a rational party $\mathcal{A}$ cherishes its reputation so much that it cannot tolerate being publicly labeled as dishonest, and if the detection probability is overwhelming, then $\mathcal{A}$ will give up cheating with overwhelming probability. We call such a rational $\mathcal{A}$ as an ***honorific adversary***. Like any law or regulation, this notion itself does not technically prohibit any cheating during execution, but provides honest parties with the ability to detect and punish cheaters later on in public. Against such honorific adversaries, we introduce an *independent auditor* as an additional "external" protocol party.

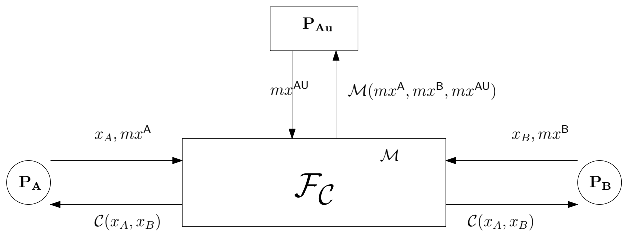

Fig. 1: Intuition of MPC with independent auditor $\mathbf{P_{au}}$. Party $\mathbf{P_A}$ and $\mathbf{P_B}$ give private inputs x_A and x_B to the ideal functionality $\mathcal{F_C}$ to compute $\mathcal{C}(x_A, x_B)$.

a) Modeling an Independent Auditor.: The idea of explicitly including an independent party $\mathbf{P_{au}}$ roots in Beaver's commodity based cryptography (CbC) [7]. In CbC, besides the normal participants Alice and Bob, there are a group of independent servers that do not actively take part in the core protocol but serve as *security resources* to Alice and Bob. An example of such resources is independently generated Beaver's multiplicative triples [7], which are widely used in new protocols such as Overdrive in the pre-processing model for statistical security [22]. MPC from hardware tokens [6] [15] also implies an independent party for attestation and sealing, which can persuade both Alice and Bob to believe in the correctness and non-malleability of logic in the hardware [1]. Although the hardware may actively participate in every protocol execution [27], the independent party itself does not always do.

We thus use an indepdent auditor $\mathbf{P_{au}}$ as a security resources. Technically, we allow the MPC ideal functionaltity to be parameterized with an auditing function as in Figure 1. This auditing function is a Turing Machine (TM) $\mathcal{M}$ which can take input ax from an independent auditor $\mathbf{P_{au}}$, who is NOT a fully trusted third party (TTP) but independent. On one hand, if $\mathbf{P_{au}}$ functions honestly, it can help other honest parties identify the cheater with the input mx^{A} or mx^{B} from both parties and an external evidence mx^{AU} from $\mathbf{P_{au}}$. On the other hand, we allow $\mathbf{P_{au}}$ to tamper with the input mx^{AU}, as long as $\mathbf{P_{au}}$ is not colluding with any other corrupted party. An algorithm given in Fig.1 could be, for example, proving whether the sender $\mathbf{P_A}$ has handed in the correct messages within any scope during an OT protocol (instantiated in protocol 7).

b) Smooth but Powerful Composition.: Suppose that a protocol π consists of functionalities ($\mathcal{F}_{\mathcal{C}_1}, \mathcal{F_C}$) and some other intermediate steps. What we would like to achieve for π is the auditing throughout the complete execution of π.

Intuitively, the ideal functionalities $\mathcal{F}_{\mathcal{C}_1}$ and $\mathcal{F_C}$ can be composed in the following way for π as shown in Figure 2.

(1) The sub-protocol $\mathcal{F}_{\mathcal{C}_1}$ produce outputs for party $\mathbf{P_A}$ and $\mathbf{P_B}$. $\mathbf{P_{au}}$ audits $\mathcal{F}_{\mathcal{C}_1}$ as defined and keeps the result $\mathcal{M}_1(ax_1)$.
(2) $\mathbf{P_A}$ and $\mathbf{P_B}$ send information to $\mathbf{P_{au}}$ based on the previous output for generating evidence for the next sub-protocol.
(3) $\mathbf{P_A}$ and $\mathbf{P_B}$ insert input for the next sub-protocol $\mathcal{F_C}$. $\mathbf{P_{au}}$ uses newly computed evidence ax to audit $\mathcal{F_C}$, and produces the final auditing results from $\mathcal{M}_1(ax_1)$ and $\mathcal{M}(ax)$.

This composition allows inheritance of evidences from previous $\mathcal{F}_{\mathcal{C}_1}$ smoothly, without interfering the pure two party part that $\mathbf{P_A}$ and $\mathbf{P_B}$ participate in $\mathcal{F_C}$. A concrete example of the power of this composition is Oblivious Transfer Extension (OTE) [2], which needs the receiver's input to be consistent in both base oblivious transfer and the extension phase. This requirement can be easily fulfilled in the above generic composition, which we will elaborate in Section VII technically.

c) Distinction from other Notions.: Bringing in an auditor has already made our model distinct from pure n-party MPC, where each party takes care of its own privacy. We do not take the probability of catching cheaters as an externality, but require it to be $1 - \mathsf{negl}(\kappa)$, where $\mathsf{negl}(\kappa)$ a negligible function of the security parameter κ.

The evidence mx^{AU} for auditors is also fundamentally different from the direct leakage adopted in the *acceptable security* paradigm proposed by Du *et al.* [11]. More specifically, MPC participants deem as *acceptable* the exposure of q bits in the private input in [11]. In contrast, without collusion, the auditor $\mathbf{P_{au}}$ in our model cannot infer a single bit of private input except the inherit leakage, such as input length and the origin of messages.

We also offer extra power to each adversary. By providing $\mathbf{P_{au}}$ with the participant generated evidence as auxiliary input besides $\mathcal{M}$, it can launch more unpredictable attacks in theory, while the hardware adversary has no external input from participants in the semi-trusted hardware model [27]. Malicious participants ($\mathbf{P_A}$ or $\mathbf{P_B}$) also have the potential to generate rogue information for evidence and observe the auditing result to learn more.

B. Our Contribution and Structure of the Paper

In this work, we introduce a new security notion called ***honorific security*** and present efficient garbled circuit and OT/OTE protocols with honorific security in the universal composability framework (UC) [9]. More specifically, we achieve the following goals.

- ***New Ideal Functionalities in UC and Constructions.*** We formalize fundamental ideal functionalities and provide concrete constructions provably secure in UC. More specifically, we formalize Oblivious Transfer with Auditing Algorithm ($\mathcal{F}^{OT}_{\mathcal{M}^{OT,A}}$), Oblivious Transfer Extension with honorific Security ($\mathcal{F}^{m\times OT}_{\mathcal{M}}$), and Two-Party Computation ($\mathcal{F}^{2pc}_{\mathcal{M}}$) based on $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$. Then we provide constructions from public key encryption, garble circuit and digital signatures and we prove that all of our constructions are secure in UC. The basic idea behind this notion is, participants are responsible to exchange encrypted evidence including all randomness they have used, along with a signature to each other. Receiving all these additional information will allow an independent auditor to find out any misbehavior, but this happens

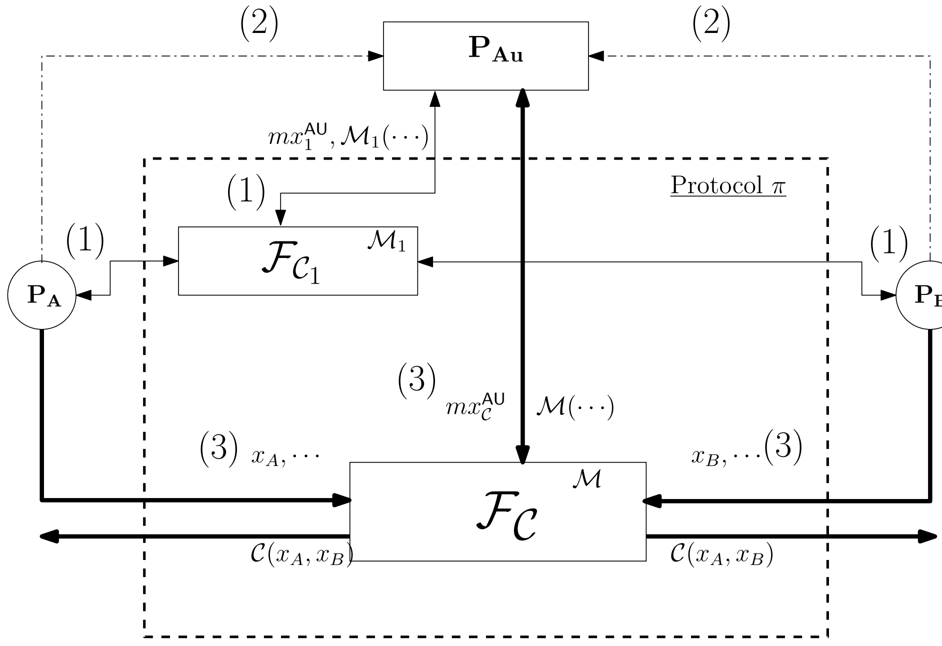

Fig. 2: Intuition of composition.

only if any party starts the auditing process. Since parties do not know the decision made by others, holding the evidence by its opponents will force parties to behave honestly, avoiding punishment by a potential auditing.

- ***High Efficiency.*** We show that our protocols are almost as efficient as those with semi-honest security. Another attractive part of our protocol is that participant do not have to prove misbehavior of other parties online with cut-and-choose or zero-knowledge proofs. Thus our protocols are significantly more efficient than protocols with covert security and malicious security in metrics such as computation and communication cost.
- ***Adaptive OT Protocol Options.*** In most of garbled circuit protocols against covert adversaries [4] [18], either signed OT or malicious OT is required. Within our protocol, besides using a trivial solution to run OT protocols with malicious security [3], we design a new secure OT protocol in the presence of honorific adversaries. We show that this new OT protocol is more efficient than the previous one against malicious adversaries, and meanwhile more powerful than those against semi-honest adversaries by letting garbler hold the evidence.
- ***Compatible for Fully Malicious Adversary.*** We show that our protocol can be constructed to achieve active security by letting participants send an auditing query after each break point of the protocol. We stress that this trade-off between security guarantee and efficiency is another highlight of our new notion. Since the adversaries does not know how honest parties will behave after receiving the evidence, this deterrence will force them to act honestly.

Following the introduction, we survey related work in Section II and introduce necessary notations and cryptographic primitives in Section III. Our new model, ideal functionalities and constructions of a generic honorific garbled circuit protocol are presented in Section IV and V, containing the complete security proof. New oblivious transfer (OT) and constructions can be found in Section VI, followed by oblivious transfer extension (OTE) ideal and constructions in Section VII. For all constructions we provide security proofs or sketches. We summarize the result and propose possible future research in Section VIII.

II. Related Work

A. Security Notions Beyond Semi-honest

The formal definition of *malicious security* can be found in Goldreich's seminal two volume classics [14]. Protocols with malicious security ensure that even if an adversary $\mathcal{A}$ deviates from protocol definition arbitrarily, $\mathcal{A}$ cannot learn anything about other parties' inputs, except that $\mathcal{A}$ may cause other parties to abort (Security with abort) [19]. Typical constructions and analysis can be found in [21] [26] [29].

Covert security was first introduced by Aumann and Lindell [5] in 2007, targeting the middle ground between passive and malicious security. Loosely speaking, in a covert-secure protocol, cheating behavior can succeed with probability $1-\mathcal{E}$, and will be detected by other parties with the remaining probability $\mathcal{E}$, which is also called the deterrence factor. Follow-up works till 2016 [4] [10] [16] [24] confirmed that protocols with reasonable $\mathcal{E}$ have a clear advantage in efficiency over the ones with malicious security.

Among them, authors of [4] highlighted another critical feature a covert-secure MPC protocol may need, the public verifiability (PVC). PVC means in addition to self-detection, the honest parties can also generate a certificate cert for the misbehavior of the cheater, and cert can be verified by anyone with the *Judge Algorithm* with public keys. Later in 2018 Hong *et al.* [18] proposed a PVC protocol with further improvement in efficiency.

B. Performance of Existing and Our Constructions

Lindell and Pinkas designed a protocol against malicious adversaries [26] based on *cut-and-choose* technique (CC). Let κ denote the security parameter, the garbler creates κ pieces of circuits. Both parties proceed with a *tossing-coin* protocol to jointly determine which circuits will be opened, the rest of them are the evaluation circuits set. In order to perform the consistency-check for garbler's input wire labels, the garbler computes commitments for all input wire labels of κ circuits, let the evaluator open some of them based on the *cut-and-choose* technique. A malicious garbler can still perform so called *selective-failure* attack, by providing one correct and another incorrect label during the OT protocol. The garbler is able to learn the evaluator's one-bit input based on whether the evaluator aborts or not. This attack is prevented in this paper by letting the evaluator's input bit split by new κ input bits, whose exclusive-or value replaces the original input bit (xor tree in [12]). For the evaluation part, evaluator evaluates the rest unopened circuits, takes the output that appears in the most evaluation circuits set.

Regarding protocols with $\mathcal{E}$-security for garbling circuits [18], the garbler is forced to garble the same circuit several times, while the evaluator opens the rest of the garbled circuits to verify whether the garbler has cheated during garbling, except one of those circuits remaining for evaluation. For *public verifiability*, the garbler has to additionally send its signature over all transcripts, which allows an honest evaluator to create some certificates proving if the garbler has cheated during the protocol execution. Within this protocol, a malicious OT protocol is required to prevent a malicious evaluator from cheating by handing inconsistent choice bit y. *Selective Failure Attacks* are prevented by performing OT protocols several times (corresponding to the number of opened garbled circuits), which allows the evaluator to check whether the rest of the received labels (except the one for evaluation) are invalid.

According to the analysis in the prior work [4] [5] [23] [18], to achieve a deterrence factor $\mathcal{E} = 1/2$, at least $\lambda = 2$ replication factor is needed. A more general function of $\mathcal{E}$ with expansion of the XOR-tree can be expressed as $\mathcal{E} = (1 - \frac{1}{\lambda})(1 - 2^{-\zeta+1})$, where ζ denotes the number of input wires of the XOR-tree that an original bit has been expanded. Regarding the protocol [26], $\mathbf{P_B}$ has to open the half of sent circuits and evaluate the rest of them. And meanwhile, $\mathbf{P_A}$ has to compute commitments for labels corresponding to all circuits and send them to $\mathbf{P_B}$. In the work [18], $\mathbf{P_A}$ and $\mathbf{P_B}$ run λ executions of Π_{OT} to transit the $\mathsf{seed}_i^{\mathsf{A}}$, avoiding sending the original copies of garbled circuits. Still, $\mathbf{P_B}$ is heavily stressed by proving the rest of sent circuits.

The works developed by Wang *et al.* [29] and specifically Katz *et al.* [21] get rid of the *cut-and-choose* by letting both $\mathbf{P_A}$ and $\mathbf{P_B}$ hold the secret share of each gate. They integrate an *information theoretic-MAC* on $\mathbf{P_A}$'s share of the masked bit to ensure that $\mathbf{P_A}$ cannot affect correctness of garbled table. This protocol is later improved by Katz *et al.* making the protocol compatible with the half-gate optimization and avoiding sending an information-theoretic MAC for each garbled row. Within their protocol, the execution of OT protocol is replaced by letting $\mathbf{P_A}$ and $\mathbf{P_B}$ open authenticated values and transit input wire labels based on their truth input bits masked by masking bits (recall that this masking bits are hidden from both parties). Labels delivery in this style is circuit dependent and must be done in online phase. Their approach requires a function-independent preprocessing protocol for both parties to prepare triples for AND gates. After $\mathbf{P_B}$ finishing evaluation, both parties need to prove that each AND gate has been computed correctly by locally computing the authenticated shares then authenticating in the online phase, as well as $\mathbf{P_A}$'s authenticated shares for output wires.

In general, among all protocols with malicious or covert security, parties are forced to prove guilt by themselves, and do not have evidence to challenge corrupted parties (except protocols with public verifiability). We argue that our protocol against honorific adversaries provides sufficient security guarantee in many cases and enables participants to perform a postponed verification of cheating behavior.

Meanwhile, we believe that our protocol is almost as efficient as the state-of-the-art protocols against semi-honest adversaries. We have compared our protocol to those state-of-the-art two-party computation protocols against covert adversaries [18] and malicious adversaries [29] [21] as well. The semi-honest protocol we compared with incorporates all existing optimizations. We have chosen the standard RSA algorithm provided by **openssl** as the public-key-encryption scheme and the signature scheme. All experiments are implemented on a Ubuntu 20.04 VM enabling 4 cores of Intel i7-8550U CPU, running at 1.80GHz each. The circuits used for evaluation are listed in the following:

Circuit	$\lvert\mathcal{C}\rvert$
AES-128	6800
SHA-256	22573
SHA-512	57947
Sorting	10.223 K
Multi	4.192 K

TABLE I: Circuits for evaluation. $|\mathcal{C}|$: number of AND gates.

1) Computation Complexity: The running time of our protocol for each circuit compared with that for semi-honest is shown below in Tab. II. All implemented result contains the running time of an OTE protocol [2]. We have also enclosed the performance comparison cited in paper [18]. In PVC, the OT protocol with malicious security is necessarily chosen to be implemented. As pointed out by the authors, the running time of the protocol for small circuits will be "dominated by the base OTs". We would like to stress that the slowdown factors of this paper are much smaller than those implementing malicious OTs, since the baseline semi-honest protocol we compared with requires less running time. As expected, our protocol is at most 37% slower than semi-honest protocol in LAN setting.

Circuit	Implemented		Slowdown [18]		
	This paper	Semi-honest	This paper	PVC	malicious
AES-128	19.36 ms	14.14 ms	37%	60%	541%
SHA-256	56.14 ms	44.69 ms	25%	27%	1160%
SHA-512	171.97 ms	154.91 ms	11%	-	-

TABLE II: Comparison of running times between our protocol and a covert/malicious protocol in LAN setting (performance of covert/malicious protocol cited from paper [18]).

2) Communication complexity: To consider comparison in WAN setting, we compared our protocol with those state-of-the-art two-party computation protocols for computing different circuits. We have chosen two representative protocols with covert security by Chen *et al.* [18] and with malicious security by Katz *et al.* [21]. The communication overhead of this paper against semi-honest only differs from the length of signature and the encrypted evidences (e.g. the seed to garble the circuit or to create random messages for base OTs). It is also possible to sign and encrypt only once in batch, which will extremely reduce the additionally transferred data size. For one single circuit, the communication complexity of our protocol almost the same as that in semi-honest setting as well, shown in Tab. III.

III. Notation and Preliminaries

We summarize basic notation in this paper in Table IV, and other terms will be introduced when necessary. We use "a party uses randomness derived from seed" as a convention for the action that a party uses seed as the key of a pseudo-random function (PRF) to obtain a sufficiently long series of pseudorandomness.

Definition 1 (**Garbling Scheme**). *A circuit garbling scheme* $\mathsf{GCS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *consists of the following algorithms.*

- $\mathsf{Gb}(1^\kappa, \mathcal{C})$ *denotes the garbling algorithm. It takes the security parameter* 1^κ*, the circuit* $\mathcal{C}$ *as input. It returns a garbled circuit* GC*, encoding information* e*, and decoding information* d*.*
- $\mathsf{En}(e, w)$ *denotes the encoding algorithm. It takes the input* w *and encoding information* e *as input. It returns the garbled input* $\{W_{i,b}\}$*.*

	AES-128	SHA-128	SHA-256	Sort.	Mult.
Semi-Honest	0.222	1.165	2.800	313.1	128.0
Covert (PVC) [18]	0.243	1.205	2.844	325.1	128.2
Malicious [21]	3.545	17.69	42.95	2953	1228
This paper	0.230	1.173	2.808	-	-

TABLE III: Communication complexity in MB (communication overhead of covert/malicious protocol cited from paper [18]).

$\{(\hat{x}_j^0, \hat{x}_j^1)\}_{j\in[m]}$	m pairs messages $\{x^0, x^1\}$ of party auditor
$\{(x_j^0, x_j^1)\}_{j\in[m]}$	m pairs messages $\{x^0, x^1\}$ of party A
$\{\mathsf{ek}_{au}, \mathsf{dk}_{au}\}$	encryption key and decryption key of party auditor for a PKE scheme
$\{\hat{B}_{i,j,b}\}$	a set of correct evaluator's input label $\hat{B}_{i,j,b}$ extracted from $\hat{\mathsf{GC}}_i$
$\{\mathsf{pk}_i, \mathsf{sk}_i\}$	public key and private key of party i for a signature scheme
$\{A_{i,j,b}\}$	a set of garbler's input label $A_{i,j,b}$ for ith circuit, jth wire, label b
$\{B_{i,j,b}\}$	a set of evaluator's input label $B_{i,j,b}$ for ith circuit, jth wire, label b
$\{k_0, k_1\}^l$	party B's input in base OT phase $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$
$\{O_{i,j,b}\}$	a set of output label $B_{i,j,b}$ for ith circuit, jth wire, label b
$\{y_j^0, y_j^1\}_{j\in[m]}$	m pairs encrypted messages
$\hat{\mathsf{GCS}}$	garbling scheme $\mathsf{GCS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$
$\hat{\mathsf{GC}}_i$	ith garbled circuit computed by party auditor using decrypted seed_i^A
GC_i	ith garbled circuit computed by party A
κ	the security parameter
$\mathbf{P}_i$	Party i
$\mathbf{q}^i$	ith column in matrix Q
$\mathbf{r}$	choice bit of party B
$\mathbf{s}$	party A's input in base OT phase $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$
$\mathbf{t}^i$	ith column in matrix T
$\mathbf{u}^i$	ith column in matrix U
$\mathcal{F}^{m\times OT}_{\mathcal{M}}$	ideal functionality $m \times OT$ without $\mathbf{P_A}$'s input consistency auditing algorithm
$\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$	ideal functionality $m \times OT$ with $\mathbf{P_A}$'s input consistency auditing algorithm $\mathcal{M}^{m\times OT,A}$
$\mathcal{F}^{2pc}_{\mathcal{M}}$	ideal functionality 2pc with honorific security
$\mathcal{M}^{m\times OT,i}$	auditing algorithm within $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,i}}$ for party i
$\mathcal{M}^i$	auditing algorithm for party i
$\mathsf{bad}_{\mathsf{ot}}$	event bad OT in main GC based 2PC protocol
$\mathsf{cheatParty}$	flag documenting cheat party
$\mathsf{ct}^{\mathsf{i,OT}}$	evidence computed by party i for OT protocol
$\mathsf{ct}_i^{\mathsf{A,GC_i}}$	evidence computed by party A for ith computation in GC_i
$\mathsf{seed}_i^{\mathsf{A}}$	seed used for garbling ith circuit by party A
$\mathsf{seed}^{\mathsf{B}}$	seed used to generate $\{k_0, k_1\}^l$ ith by party B
rnd	randomness used for $\mathsf{OT.ENC}$ by party A
$\Pi^{\mathsf{GC}}_{\mathcal{M}}$	GC Protocol with honorific security
$\Pi^{m\times OT}_{\mathcal{M}}$	$m \times OT$ Protocol with honoridic security
$\Pi^{OT}_{\mathcal{M}^{OT,A}}$	OT Protocol implementing $\mathbf{P_A}$'s input consistency auditing algorithm
$\Pr[A : B]$	the probability that event A happens if action B is taken
sid	session identifier
$\sigma^{\mathsf{i,OT}}$	signature computed by party i for OT protocol
$\sigma_i^{\mathsf{A,GC_i}}$	signature computed by party A for ith computation in GC_i
ξ	Party's invalid input
1^κ	unary string of κ ones
$a \xleftarrow{\$} S$	sampling an element from S uniformly at random
$m \xleftarrow{\$} \mathcal{A}^{\mathsf{O}(\cdot)}()$	$\mathcal{A}$ outputs m with the help of another algorithm (oracle) $\mathsf{O}(\cdot)$
$X \Vert Y$	the operation concatenating two binary strings X and Y
$\mathsf{bad}_{\mathsf{baseOT}}$	event bad base OT in main OTE protocol
G	pseudorandom generator
H	collision-resistent hash function
x, y, ax	Party's valid input

TABLE IV: Notations.

- $\mathsf{Ev}(\mathsf{GC}, W)$ *denotes the evaluation algorithm. It takes the garbled circuit* GC *and garbled input* $\{W_{i,b}\}$ *as input. It returns a garbled output* $\{O_i\}$.
- $\mathsf{De}(d, O)$ *denotes the decoding algorithm. It takes the decoding information* d *and garbled output* $\{O_i\}$ *as output. It returns the output* $\{o_i\}$

Definition 2 (Correctness [8]). *A garbling scheme* $\mathsf{GCS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is correct, if for all functions* $\mathcal{C}$ *and input* w*:*

$$\Pr[(\mathsf{GC}, e, d) \leftarrow \mathsf{Gb}(1^\kappa, \mathcal{C}) : \\ \mathsf{De}(d, \mathsf{Ev}(\mathsf{GC}, \mathsf{En}(e, w))) = \mathcal{C}(w)] = 1$$

Definition 3 (Simulatable Privacy [8] [27]). *A garbling scheme* $\mathsf{GCS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is simulatable private, if for all functions* $\mathcal{C}$ *and input* w*, there exists a probabilistic polynomial time (PPT) simulator* $\mathbf{Sim}$ *such that for all PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{c} b = b^\star : \\ (\mathsf{GC}_0, e_0, d_0) \leftarrow \mathsf{Gb}(1^\kappa, \mathcal{C}); W_0 \leftarrow \mathsf{En}(e, w); \\ (\mathsf{GC}_1, W_1, d_1) \leftarrow \mathbf{Sim}(1^\kappa, \mathcal{C}(w), \Phi(\mathcal{C})); \\ b \leftarrow \{0,1\}; b^\star \leftarrow \mathcal{A}(\mathsf{GC}_b, W_b, d_b); \end{array}\right] \leq \mathsf{negl}(\kappa)$$

where Φ *denotes the side-information function.*

Definition 4 (Signature scheme). *A signature scheme* $\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$ *consists of three algorithms* $\mathsf{SIG.Gen}$, $\mathsf{SIG.Sign}$ *and* $\mathsf{SIG.Vfy}$ *described as below.*

- $\mathsf{SIG.Gen}(1^\kappa) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$. *The non-deterministic key generation algorithm* $\mathsf{SIG.Gen}()$ *takes the security parameter* 1^κ *as the input and outputs the public key* pk *and the private key* sk.
- $\mathsf{SIG.Sign}(\mathsf{sk}, m) \xrightarrow{\$} \sigma$. *The (non-deterministic) message signing algorithm* $\mathsf{SIG.Sign}()$ *takes the private key* sk *and a message* m *as the input and outputs the signature* σ.
- $\mathsf{SIG.Vfy}(\mathsf{pk}, m, \sigma) = b$. *The deterministic signature verification algorithm* $\mathsf{SIG.Vfy}()$ *takes the public key* pk*, a message* m*, a signature* σ *as input and outputs a boolean value* b. b *is* TRUE *if* σ *is a valid signature on* m.

Definition 5 (Public key encryption scheme, PKE). *A public key encryption scheme* $\Pi = (\mathsf{KGen}, \mathsf{ENC}, \mathsf{DEC})$ *consists of three algorithms* KGen, ENC *and* DEC *described as below.*

- $\Pi.\mathsf{KGen}(1^\kappa) \xrightarrow{\$} (\mathsf{ek}, \mathsf{dk})$. *The non-deterministic key generation algorithm* $\mathsf{KGen}()$ *takes the security parameter* 1^κ *as the input and outputs the public encryption key* ek *and the private decryption key* dk.
- $\Pi.\mathsf{ENC}(\mathsf{ek}, m) \xrightarrow{\$} \mathsf{CT}$. *The non-deterministic encryption algorithm* $\mathsf{ENC}()$ *takes the encryption key* ek *and a message* m *as the input and outputs a ciphertext* CT.
- $\Pi.\mathsf{DEC}(\mathsf{dk}, \mathsf{CT}) = m'$. *The deterministic decryption algorithm* $\mathsf{DEC}()$ *takes the public key* pk*, a ciphertext* CT *as input and outputs a plaintext* m'.

Due to the page limitation, we refer the reader to cryptography texts such as [20] for definitions of correctness and security of SIG, hash function, PKE, PRF and other primitives.

IV. Honorific MPC Ideals and GC protocol

A. Ideals against Honorific Adversaries in UC

The Ideal $\mathcal{F}^{2pc}_{\mathcal{M}}$*:* Let $\mathbf{P_A}$, $\mathbf{P_B}$ and $\mathbf{P_{au}}$ denote the participating parties $\mathcal{P} = \{\mathbf{P_A}, \mathbf{P_B}, \mathbf{P_{au}}\}$, $\mathcal{P}_c \subseteq \{\mathbf{P_A}, \mathbf{P_B}, \mathbf{P_{au}}\}$ denote the corrupted parties controlled by an adversary $\mathcal{A}$. We include following basic queries in our new ideal.

Inputs: Let x denote party $\mathbf{P_A}$'s input, y denote party $\mathbf{P_B}$'s input, z denote adversary $\mathcal{A}$'s auxiliary input.

Send inputs to ideal functionality: The honest party P_j sends its input to the ideal functionality. The corrupted parties controlled by $\mathcal{A}$ may either send their predefined input, some other input of the same length, or abort (by replacing the input with a special $(\mathbf{abort}, \mathbf{P_i}, \mathsf{sid})$ to the ideal functionality.

Early Abort Option: If the ideal functionality receives the message $(\mathbf{abort}, \mathbf{P_i}, \mathsf{sid})$, it sends $(\mathbf{abort}, \mathbf{P_i}, \mathsf{sid})$ to all parties and the ideal execution terminates. Otherwise, the ideal execution proceeds.

Ideal functionality sends outputs to adversary: The ideal functionality computes $f_i(x', y')$ and sends $f_i(x', y')$ to party $\mathbf{P_i}$ for all $i \in P_c$ (i.e. to all corrupted parties).

Adversary instructs ideal functionality to continue or halt: $\mathcal{A}$ sends either $(\mathbf{continue}, \mathbf{P_i}, \mathsf{sid})$ or $(\mathbf{abort}, \mathbf{P_i}, \mathsf{sid})$ to the ideal functionality. If received $(\mathbf{continue}, \mathbf{P_i}, \mathsf{sid})$, the ideal functionality sends $f_j(x', y')$ to the honest party $\mathbf{P_j}$ for all $j \notin P_c$ (i.e. to all honest parties). Otherwise, the ideal functionality send $(\mathbf{abort}, \mathbf{P_i}, \mathsf{sid})$ to party $\mathbf{P_j}$.

Outputs An honest party always outputs what ideal functionality sends to it. The corrupted parties output nothing. The adversary $\mathcal{A}$ outputs any (probabilistic polynomial time computable) function of the initial inputs $\{x_i\}_{i \in P_c}$, the auxiliary input z, and the messages $\{f_i(x', y')\}_{i \in P_c}$ received from the ideal functionality.

The ideal is now parameterized with two additional flags:

- $\mathbf{cheatParty}$ is a set of cheated parties. Since we consider the honest majority setting in this paper, only one party could be corrupted and then cheats.
- $\mathcal{M}$ is implemented as an auditing algorithm. Considering an OT protocol with malicious security against the receiver, if both parties $\mathbf{P_A}$ and $\mathbf{P_B}$ agree that the input messages of the sender must be $x_i \in \mathcal{X}$. Then having $\mathbf{P_{au}}$ holding the external input $mx^{\mathsf{AU}} = \mathcal{X}$, and $mx^{\mathsf{A}} = \{x_i\}$, the auditing algorithm $\mathcal{M}(mx^{\mathsf{A}}, mx^{\mathsf{AU}})$ outputs the exact cheating behavior performed by $\mathbf{P_A}$ (when $\mathbf{P_A}$ is the sender).

We then extend the ideal by adding new instructions that the adversary can send to the ideal functionality. Similar to the ideals with covert security, the adversary is able to send a $\mathbf{cheat}$ option to the ideal functionality, and this cheat decision "must be made before the adversary learns anything" [5]. Besides, we extend this cheat option by letting the cheat party also include the input mx^{i} for auditing algorithm $\mathcal{M}$:

Cheat Option: If a corrupted party send $(\mathbf{cheat}, (mx^{\mathsf{i}}, \mathcal{M})$ $\mathbf{P_i}, \mathsf{sid})$ to the ideal functionality, then the ideal functionality sends all of the honest parties' inputs $\{x_j\}_{j \notin P_c}$ to $\mathbf{P_i}$ and stores the mx^{i}. If multiple $(\mathbf{cheat}, (mx^{\mathsf{i}}, \mathcal{M}), \mathbf{P_i}, \mathsf{sid})$ are sent, the ideal functionality ignores all but one.

Additionally, we allow any party sends an auditing query $((\mathbf{audit}, \mathbf{P_j}), \mathbf{P_i}, \mathsf{sid})$ to the ideal functionality. Note that parties are able to send this query at any time during the protocol. The ideal functionality will first require the input of $\mathbf{P_{au}}$ if needed, then it sends the auditing result (which is the output of $\mathcal{M}$) to $\mathbf{P_{au}}$. We stress that this $\mathbf{P_{au}}$ is not a part of the secure computation and come into force only when any $\mathbf{P_i}$ sends the auditing query.

Audit Option: If any party $\mathbf{P_i}$ sends $((\mathbf{audit}, \mathbf{P_j}), \mathbf{P_i}, \mathsf{sid})$ to the ideal functionality intended to audit $\mathbf{P_j}$, if $\mathcal{M}$ requires mx^{AU} from $\mathbf{P_{au}}$, then the ideal functionality sends $((\mathcal{M}, \mathbf{P_j}), \mathbf{P_i}, \mathsf{sid})$ to $\mathbf{P_{au}}$. Upon receiving $(mx^{\mathsf{AU}}, \mathbf{P_{au}}, \mathsf{sid})$ from $\mathbf{P_{au}}$, it sends $\mathcal{M}(mx^{\mathsf{i}}, mx^{\mathsf{AU}})$ to $\mathbf{P_{au}}$. Otherwise, it directly sends $\mathcal{M}(mx^{\mathsf{i}})$ to $\mathbf{P_{au}}$. It waits for the $(\mathbf{deliver}, \mathbf{P_{au}}, \mathsf{sid})$ and broadcasts the result of $\mathcal{M}$.

We let $\mathcal{F}^{OT}_{\mathcal{M}^{OT,A}}$ denote the ideal OT functionality with auditing algorithm $\mathcal{M}^{OT,A}$, and $\mathcal{F}^{m \times OT}_{\mathcal{M}^{m \times OT,A}}$ its the parallel version, as shown in Figure 4.

B. GC-based MPC with Honorific Security, the idea

In this section, we show a high-level overview of our protocol $\Pi^{\mathsf{GC}}_{\mathcal{M}}$. A formal definition is provided in Figure 5.

Using a signature scheme $\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$, both parties $\mathbf{P_A}$ and $\mathbf{P_B}$ run $\mathsf{SIG.Gen}$ to obtain their public-private key pairs. We assume that both parties and $\mathbf{P_{au}}$ know the public key of participants before running the protocol. Using a PKE $(\mathsf{KGen}, \mathsf{ENC}, \mathsf{DEC})$, $\mathbf{P_{au}}$ runs $\mathsf{KGen}()$ to obtain its public and private key pair. We assume that both parties know the public key of $\mathbf{P_{au}}$ as well before running the protocol. In the common reference string (CRS) model, this will allow the simulator to simulate the key pair for the PKE and the signature scheme.

To achieve security against honorific adversaries, both party $\mathbf{P_A}$ and $\mathbf{P_B}$ have to send an evidence and a signature along with part of the transcript to each other. If one party, say $\mathbf{P_B}$ would like to check whether $\mathbf{P_A}$ has cheated during the protocol, $\mathbf{P_B}$ can send the evidence, the signature, and the corresponding part of the transcript to auditor, who can open the evidence and verify all $\mathbf{P_A}$'s behaviors during the protocol. Note that since the auditor could be corrupted by a semi-honest adversary, both parties are not willing to expose their private input in the evidence.

In a garbled circuit based MPC, after garbling the ith circuit, $\mathbf{P_A}$ has to compute the evidence $\mathsf{ct}_i^{\mathsf{A},\mathsf{GC}_i}$ including all (pseudo)randomness derived from a corresponding seed $\mathsf{seed}_i^{\mathsf{A}}$. $\mathbf{P_A}$ sends all information above along with a signature $\sigma_i^{\mathsf{A},\mathsf{GC}_i}$ to $\mathbf{P_B}$. [1]

[1]Recall that if $\mathbf{P_{au}}$ opens $\mathbf{P_A}$'s evidence, $\mathbf{P_{au}}$ will have the whole garbled circuit, so $\mathbf{P_A}$'s input wire labels can not be included in the $\mathsf{ct}_i^{\mathsf{A},\mathsf{GC}_i}$.

The above idea will prevent an honorific $\mathbf{P_A}$ from cheating during the GC generation phase, since $\mathbf{P_B}$ can verify $\mathbf{P_A}$'s behavior at any time. But during the OT protocol, where $\mathbf{P_B}$ learns the receiver's input wire labels, $\mathbf{P_A}$ can still perform a selective failure attack, by providing a pair of true and false label observing if $\mathbf{P_B}$ aborts to obtain $\mathbf{P_B}$'s one-bit input. In our model, if $\mathbf{P_A}$ performs such attacks and $\mathbf{P_B}$ prosecutes, $\mathbf{P_{au}}$ should be able to detect such cheating behavior. Thus, we require an improved OT ideal functionality $\mathcal{F}^{m \times OT}_{\mathcal{M}^{m \times OT,A}}$ shown in Fig. 4 within our protocol. $\mathcal{F}^{m \times OT}_{\mathcal{M}^{m \times OT,A}}$ allows $\mathbf{P_{au}}$ to input reference OT labels retrieved from the receiving evidence within the GC protocol and will compare these with the real input of $\mathbf{P_A}$. By importing a *consistency check* algorithm $\mathcal{M}$, any protocol implementing conventional ideal OT functionality can easily be converted to implement $\mathcal{F}^{m \times OT}_{\mathcal{M}^{m \times OT,A}}$. The detailed description is in Section VI.

There still remains the issue that $\mathbf{P_B}$ may cheat during the OT protocol by using inconsistent choice bit $\mathbf{r}$ to obtain $\mathbf{P_A}$'s choice bit s and can thus decrypt both $\{y_j^0, y_j^1\}$, where $j \in [n_2]$. We can either preventing such a cheating behavior by deploying an OT protocol with malicious security (e.g. [28] and [3]), or extending an OT protocol (e.g. from [2]) of passive security against $\mathbf{P_B}$ to maintain *honorific security*. Following the same idea above, $\mathbf{P_B}$ as OT protocol receiver has to provide $\mathbf{P_A}$ with an evidence and a signature during the OT protocol, enabling $\mathbf{P_A}$ to audit $\mathbf{P_B}$'s behavior at any time. More details are explained in Sec. VII.

Finally, $\mathbf{P_A}$ sends the garbler's input $\{A_{i,j,b}\}$ to $\mathbf{P_B}$, allowing $\mathbf{P_B}$ to compute the output.

V. Protocols and Security Analysis

We now analyze the security of $\Pi^{\mathsf{GC}}_{\mathcal{M}}$. First we define the server-aided public verifiability.

Definition 6 (Server-aided Public Verifiability)**.** *If a corrupted party P_j outputs an evidence ct^{j}, its corresponding signature σ^{j} and sends them to an honest party P_i, then we know that an semi-honest auditor holding the private key dk_{au} can always output $\mathcal{M}^j(mx^{\mathsf{j}}, mx^{\mathsf{AU}})$ upon receiving the auditing query by P_i, except with negligible probability.*

Theorem 1. *Assume $\mathsf{GCS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ is a secure simulate private and correct garbling scheme, $\mathsf{H}()$ is a collision-resistant hash function, SIG is existentially unforgeable under a chosen-message attack, PKE has indistinguishable encryptions under a chosen-ciphertext attack. Protocol $\Pi^{\mathsf{GC}}_{\mathcal{M}}$ described in Fig.5 securely realizes $\mathcal{F}^{2pc}_{\mathcal{M}}$ described in Fig.3 with server-aided public verifiablity in the $\mathcal{F}^{m \times OT}_{\mathcal{M}^{m \times OT,A}}$-hybrid model, in the presence of an honorific adversary who can corrupt either $\mathbf{P_A}$ or $\mathbf{P_B}$, or in the presence of a semi-honest adversary who can corrupt $\mathbf{P_{au}}$, with static corruption.*

We prove Theorem 1 by analyzing the cases of static corruption of $\mathbf{P_B}$, $\mathbf{P_A}$ and independently corrupted $\mathbf{P_{au}}$.

A. Honorific Security — Malicious $\mathbf{P_B}$

Let $\mathcal{A}$ be an adversary corrupting $\mathbf{P_B}$. In CRS model, we build the following simulator $\mathcal{S}$, who simulates a key pair of

Functionality $\mathcal{F}_{2pc}$ interacts with players $\mathcal{P} := \{\mathbf{P_A}, \mathbf{P_B}, \mathbf{P_{au}}\}$ and the adversary $\mathcal{S}$. It is parameterized with the set of corrupted parties $\mathcal{P}_c$, a set of cheated parties $\mathbf{cheatParty} \in \{\mathbf{P_A}, \mathbf{P_B}\}$, and a set of PPT algorithms $\mathcal{M}^{\mathsf{2pc}} = \{\mathcal{M}^{\mathbf{A}}, \mathcal{M}^{\mathbf{B}}\}$. Initially, $\mathcal{P}_c = \emptyset$, $\mathbf{cheatParty} = \emptyset$.

Corruption handling:

- Upon receiving $(\mathbf{corrupt}, \mathbf{P_i}, \mathsf{sid})$ from the adversary $\mathcal{S}$:
  - If $\mathbf{P_i} \in \mathcal{P}$ and $\mathcal{P}_c = \emptyset$, set $\mathcal{P}_c := \{\mathbf{P_i}\}$. Send $(\mathbf{corrupt\_success}, \mathbf{P_i}, \mathsf{sid})$ to $\mathcal{S}$.
  - Otherwise, send $(\mathbf{corrupt\_failed}, \mathbf{P_i}, \mathsf{sid})$ to $\mathcal{S}$

Compute:

- Upon receiving $(\mathbf{input}, x, \mathbf{P_A}, \mathsf{sid})$ from party $\mathbf{P_A}$ and $(\mathbf{input}, y, \mathbf{P_B}, \mathsf{sid})$ from party $\mathbf{P_B}$:
  - Compute $f(x, y)$ and send it to $\mathbf{P_B}$.
- Upon receiving $(\mathbf{input}, \xi, \mathbf{P_i}, \mathsf{sid})$ from any party $\mathbf{P_i} \in \{\mathbf{P_A}, \mathbf{P_B}\}$, where ξ is a invalid input:
  - Send $\perp$ to $\mathbf{P_B}$.

Cheating handling:

- Upon receiving $(\mathbf{cheat}, (mx^{\mathsf{A}}, \mathcal{M}^{\mathbf{A}}), \mathbf{P_A}, \mathsf{sid})$ from party $\mathbf{P_A}$:
  - If $\mathbf{P_A} \in \mathcal{P}_c$, send a message $(\mathbf{cheat\_success}, y, \mathsf{sid})$ to $\mathcal{S}$, wait to receive o^B from $\mathbf{P_A}$ and send o^B to $\mathbf{P_B}$. Set $\mathbf{cheatParty} = \{\mathbf{P_A}\}$, and stores mx^{A};
    Otherwise, send a message $(\mathbf{cheat\_failed}, \mathsf{sid})$ to $\mathcal{S}$;
- Upon receiving $(\mathbf{cheat}, (mx^{\mathsf{B}}, \mathcal{M}^{\mathbf{B}}), \mathbf{P_B}, \mathsf{sid})$ from party $\mathbf{P_B}$:
  - If $\mathbf{P_B} \in \mathcal{P}_c$, send a message $(\mathbf{cheat\_success}, x, \mathsf{sid})$ to $\mathcal{S}$, wait to receive o^A from $\mathbf{P_B}$ and send o^A to $\mathbf{P_A}$. Set $\mathbf{cheatParty} = \{\mathbf{P_B}\}$, and stores mx^{B};
    Otherwise, send a message $(\mathbf{cheat\_failed}, \mathsf{sid})$ to $\mathcal{S}$;

Auditing:

- Upon receiving $((\mathbf{audit}, \mathbf{P_j}), \mathbf{P_i}, \mathsf{sid})$ from party $\mathbf{P_i}$ intended to audit $\mathbf{P_i}$:
  - Run $\mathcal{M}^j$ and sends the result to $\mathbf{P_{au}}$. Wait for response and broadcasts.

Fig. 3: Two Party Functionality $\mathcal{F}^{2pc}_{\mathcal{M}}$.

Private inputs: $\mathbf{P_A}$ has input $\{x^0, x^1\}^m$ and keys $\{\mathsf{pk}_A, \mathsf{sk}_A\}$ for a signature scheme. $\mathbf{P_B}$ has input $y \in \{0,1\}^m$ and keys $\{\mathsf{pk}_B, \mathsf{sk}_B\}$ for a signature scheme. $\mathbf{P_{au}}$ has input $\{\hat{x}^0, \hat{x}^1\}^m$.
Public inputs: Both party agree to compute a set of auditing algorithms $\mathcal{M}^{\mathsf{m\times OT}} = \{\mathcal{M}^{\mathsf{m\times OT,A}}, \mathcal{M}^{\mathsf{m\times OT,B}}\}$, know the public key pk_i of each other, and agree with a encryption key ek_{au} for the public-key-encryption scheme.
Initialization: $\mathcal{P}_c = \emptyset$, $\mathbf{cheatParty} = \emptyset$.

Compute: Upon receiving $(\{x^0, x^1\}^m, P_A, \mathsf{sid})$ from $\mathbf{P_A}$ and (y, P_B, sid) from $\mathbf{P_B}$, send $\{x^{y[i]}\}^m$ to P_B.

Cheating handling:

1) Upon receiving $(\mathbf{cheat}, (mx^{\mathsf{m\times OT,B}}, \mathcal{M}^{\mathsf{m\times OT,B}}), \mathbf{P_B}, \mathsf{sid})$ from party $\mathbf{P_B}$, send a message $(\mathbf{cheat\_success}, \{x^0, x^1\}^m, \mathsf{sid})$ to $\mathcal{S}$, wait to receive o^A from $\mathbf{P_B}$ and send o^A to $\mathbf{P_A}$. Set $\mathbf{cheatParty} = \{\mathbf{P_B}\}$ and stores $mx^{\mathsf{m\times OT,B}}$.
2) Upon receiving $(\mathbf{cheat}, (mx^{\mathsf{m\times OT,A}}, \mathcal{M}^{\mathsf{m\times OT,A}}), \mathbf{P_A}, \mathsf{sid})$ from party $\mathbf{P_A}$, send a message $(\mathbf{cheat\_success}, y, \mathsf{sid})$ to $\mathcal{S}$, wait to receive o^B from $\mathbf{P_A}$ and send o^B to $\mathbf{P_B}$. Set $\mathbf{cheatParty} = \{\mathbf{P_A}\}$ and stores $mx^{\mathsf{m\times OT,A}}$.

***Auditing algorithm $\mathcal{M}^{\mathsf{m\times OT,B}}$:**

- Upon receiving $((\mathbf{audit}, \mathbf{P_B}), \mathbf{P_A}, \mathsf{sid})$ from $\mathbf{P_A}$, send $\mathcal{M}^{\mathsf{m\times OT,B}}(mx^{\mathsf{B}})$ to $\mathbf{P_{au}}$. Wait for $(\mathbf{deliver}, \mathbf{P_{au}}, \mathsf{sid})$ and broadcast.

***Auditing algorithm (Input consistency check included) $\mathcal{M}^{\mathsf{m\times OT,A}}$:**

- Upon receiving $((\mathbf{audit}, \mathbf{P_A}), \mathbf{P_B}, \mathsf{sid})$ from $\mathbf{P_B}$ and $(mx^{\mathsf{AU}}, \mathbf{P_{au}}, \mathsf{sid})$ from $\mathbf{P_{au}}$ where $mx^{\mathsf{AU}} = \{\hat{x}^0, \hat{x}^1\}^m$, send $\mathcal{M}^{\mathsf{m\times OT,A}}(mx^{\mathsf{m\times OT,A}}, mx^{\mathsf{m\times OT,AU}})$ to $\mathbf{P_{au}}$. Wait for $(\mathbf{deliver}, \mathbf{P_{au}}, \mathsf{sid})$ and broadcast.

Fig. 4: OT ideal functionality $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$.

Private inputs: $\mathbf{P_A}$ has input $x \in \{0,1\}^{n_1}$ and a key pair $\{\mathsf{pk}_A, \mathsf{sk}_A\}$ for the signature scheme. $\mathbf{P_B}$ has input $y \in \{0,1\}^{n_2}$ and a key pair $\{\mathsf{pk}_B, \mathsf{sk}_B\}$ for the signature scheme. $\mathbf{P_{au}}$ has a key pair $\{\mathsf{ek}_{au}, \mathsf{dk}_{au}\}$ for the PKE scheme.
Public inputs: Both parties agree to compute the functionality $\mathcal{C}$ with parameters κ, λ and deploy an ideal functionality $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$. They both know the public key pk_i of each other and agree with the ek_{au} for a encryption scheme. $\mathbf{P_{au}}$ knows the pk_i of $\mathbf{P_A}$ and $\mathbf{P_B}$.

Protocol:

1) $\mathbf{P_A}$ garbles $\mathcal{C}$ using randomness derived from $\mathsf{seed}^{\mathsf{A}}_i$. The ith garbled circuit are denoted by GC_i, as well as $\mathbf{P_A}$'s input wire labels $\{A_{i,j,b}\}_{j\in[n_1],b\in\{0,1\}}$, $\mathbf{P_B}$'s input wire labels $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$, and output wire labels $\{O_{i,j,b}\}_{j\in[n_3],b\in\{0,1\}}$. $\mathbf{P_A}$ then computes a decoding table $\mathsf{table}_i \leftarrow \{\mathsf{Label}^0_{i,j}, \mathsf{Label}^1_{i,j}\}_{j\in[n_3]}$, where $\mathsf{Label}^0_{i,j} \leftarrow \mathsf{H}(O_{i,j,0})$ and $\mathsf{Label}^1_{i,j} \leftarrow \mathsf{H}(O_{i,j,1})$.
2) $\mathbf{P_A}$ and $\mathbf{P_B}$ call $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, where $\mathbf{P_A}$ uses $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$ as input and $\mathbf{P_B}$ uses y as input.
3) $\mathbf{P_A}$ computes an evidence $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i \leftarrow \mathsf{ENC}(\mathsf{ek}_{au}, \mathsf{seed}^{\mathsf{A}}_i, \mathsf{sid})$. Then $\mathbf{P_A}$ computes a signature of this message combined with transcript $\sigma^{\mathsf{A},\mathsf{GC}_i}_i \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_A, \mathcal{C}, \mathsf{GC}_i, \mathsf{table}_i, \mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i, \mathsf{sid})$. $\mathbf{P_A}$ then sends GC_i, table_i, $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ and $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$ to $\mathbf{P_B}$.
4) $\mathbf{P_B}$ checks whether $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$ is a valid signature for GC_i and $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$, aborts with output $\perp$ if not.
   * **Auditing first part:**
     a) $\mathbf{P_B}$ sends an auditing query $(\mathcal{C}, \mathsf{GC}_i, \mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i, \sigma^{\mathsf{A},\mathsf{GC}_i}_i, \mathsf{sid})$ to $\mathbf{P_{au}}$, who checks whether $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$ is valid, aborts with $\perp$ if not.
     b) $\mathbf{P_{au}}$ retrieves $\mathsf{seed}^{\mathsf{A}}_i \leftarrow \mathsf{DEC}(\mathsf{dk}_{au}, \mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i)$.
     c) $\mathbf{P_{au}}$ computes $\hat{\mathsf{GC}}_i$ and $\hat{\mathsf{table}}_i$ using $\mathsf{seed}^{\mathsf{A}}_i$, then compares whether $\mathsf{GC}_i = \hat{\mathsf{GC}}_i$ and $\mathsf{table}_i = \hat{\mathsf{table}}_i$, broadcasts the result.
   * **Auditing second part:**
     a) $\mathbf{P_B}$ sends an auditing query $((\mathbf{audit}, \mathbf{P_A}), \mathbf{P_B}, \mathsf{sid})$ to $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, which sends a notification $((\mathcal{M}^{\mathsf{m\times OT,A}}, \mathbf{P_A}), \mathbf{P_B}, \mathsf{sid})$ to $\mathbf{P_{au}}$.
     b) Upon receiving $(\{\hat{B}_{i,j,b}\}, \mathbf{P_{au}}, \mathsf{sid})$ from $\mathbf{P_{au}}$, $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ sends $\mathcal{M}^{\mathsf{m\times OT,A}}(\{B_{i,j,b}\}, \{\hat{B}_{i,j,b}\})$ to $\mathbf{P_{au}}$.
     c) Upon receiving $(\mathbf{deliver}, \mathbf{P_{au}}, \mathsf{sid})$ from $\mathbf{P_{au}}$, $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ broadcasts $\mathcal{M}^{\mathsf{m\times OT,A}}(\{B_{i,j,b}\}, \{\hat{B}_{i,j,b}\})$.
5) $\mathbf{P_A}$ sends $\{A_{i,j,x[j]}\}_{j\in[n_1]}$ to $\mathbf{P_B}$.
6) $\mathbf{P_B}$ evaluates GC_i using $\{A_{i,j,b}\}_{j\in[n_1]}$, $\{B_{i,j,b}\}_{j\in[n_2]}$, obtains $\{O_{i,j}\}_{j\in[n_3]}$. Then, $\mathbf{P_B}$ compute $\{\mathsf{H}(O_{i,j})\}_{j\in[n_3]}$. If any $\mathsf{H}(O_{i,j}) \notin \{\mathsf{Label}^0_{i,j}, \mathsf{Label}^1_{i,j}\}$, aborts with $\perp$, otherwise output o^B.

Fig. 5: Full description of $\Pi^{\mathsf{GC}}_{\mathcal{M}}$ against honorific adversaries.

$\mathbf{P_A}$ for the signature scheme, a key pair of $\mathbf{P_{au}}$ for the PKE, holds the public key of $\mathbf{P_B}$ and runs $\mathcal{A}$ as subroutine, while playing the role of $\mathbf{P_B}$ in the ideal world interacting with $\mathcal{F}^{2pc}_{\mathcal{M}}$ as follows.

1) Play the role of an honest $\mathbf{P_A}$, using pseudorandomness derived from uniform distributed κ-bit $\mathsf{seed}^{\mathsf{A}}_i$ to create pseudorandom keys $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$.
2) Play the $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ with $\mathcal{A}$ playing $\mathbf{P_B}$:
   a) If the input is $(\mathbf{cheat}, (mx^{\mathsf{B}}, \mathcal{M}^{\mathsf{m\times OT,B}}), \mathbf{P_B}, \mathsf{sid})$, send $(\mathbf{cheat}, (mx^{\mathsf{B}}, \mathcal{M}^{\mathbf{B}}), \mathbf{P_B}, \mathsf{sid})$ to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and receive back $\mathbf{P_A}$'s input x. Then use the input x of $\mathbf{P_A}$ to perfectly emulate $\mathbf{P_A}$ in the rest of execution. Send o^A to $\mathcal{F}^{2pc}_{\mathcal{M}}$, which will be received by $\mathbf{P_A}$ as output. The simulation ends here for this case.
   b) If the input is y, hands $\mathcal{A}$ the wire label $\{B_{i,j,y[j]}\}$ that are chosen by $\mathcal{A}$, proceed with the simulation below.
3) Send the $\mathcal{A}$'s input y as $\mathbf{P_B}$'s input to $\mathcal{F}^{2pc}_{\mathcal{M}}$, receive back the output o^B.
4) Using the appropriate $\mathbf{P_B}$'s input wire labels from above, generate a garbled GC_i and corresponding table_i, which is always evaluated to o^B. Send GC_i and table_i to $\mathcal{A}$.
5) Compute $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ using the $\mathsf{seed}^{\mathsf{A}}_i$ and a corresponding $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$, send to $\mathcal{A}$.
6) Hand $\mathcal{A}$ arbitrary $\mathbf{P_A}$'s input wire labels and halt.

We now prove the indistinguishability of views.

Proof. Denote Protocol 5 as $\Pi^{\mathsf{GC}}_{\mathcal{M}}$ and $\mathcal{P}_c = \{\mathbf{P_B}\}$ (i.e. $\mathbf{P_B}$ is corrupted), we show:

$$\left\{\mathrm{IDEAL}^h_{\mathcal{C},S(z),\mathcal{P}_c}((x,y),n)\right\} \overset{c}{\equiv} \left\{\mathrm{HYBRID}^{OT,h}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}((x,y),n)\right\} \quad (1)$$

To prove (1), we first consider the cases when $\mathcal{A}$ sends $(\mathbf{cheat}, (mx^{\mathsf{m\times OT,B}}, \mathcal{M}^{\mathsf{m\times OT,B}}), \mathbf{P_B}, \mathsf{sid})$ or behaves honestly, denoted by $\mathsf{bad_{ot}}$ and $\neg\mathsf{bad_{ot}}$, respectively. Note that the oblivious transfer is the first step of the protocol, and $\mathcal{A}$'s view in IDEAL is identical to its view in the real protocol execution with an ideal functionality $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, whether the event $\mathsf{bad_{ot}}$ happens depends on $\mathcal{A}$ itself. This indicates that the probability an event $\mathsf{bad_{ot}}$ happens is *identical* in the IDEAL and HYBRID execution. Thus we still have to show that the following equations hold:

$$\left\{\mathrm{IDEAL}^h_{\mathcal{C},S(z),\mathcal{P}_c}(x,y)|\mathsf{bad_{ot}}\right\} \equiv \left\{\mathrm{HYBRID}^{OT,h}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}(x,y)|\mathsf{bad_{ot}}\right\} \quad (2)$$

$$\left\{\text{IDEAL}^{h}_{\mathcal{C},S(z),\mathcal{P}_c}(x,y)|\neg\mathsf{bad}_{\mathsf{ot}}\right\} \overset{c}{\equiv} \left\{\text{HYBRID}^{OT,h}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}(x,y)|\neg\mathsf{bad}_{\mathsf{ot}}\right\} \quad (3)$$

To prove that Eq. (2) holds, we stress that after sending $(\mathbf{cheat}, (mx^{\mathsf{m}\times\mathsf{OT},\mathsf{B}}, \mathcal{M}^{\mathbf{B}}), \mathbf{P_B}, \mathsf{sid})$, $\mathcal{S}$ receives back honest $\mathbf{P_A}$'s input x. For both cases, $\mathbf{P_{au}}$'s output is exactly the same auditing result computed by $\mathcal{S}$ during HYBRID execution. Since $\mathcal{S}$ is able to emulates $\mathbf{P_A}$ perfectly when holding $\mathbf{P_A}$'s real input, and $\mathbf{P_A}$ has no output from Protocol 5, the joint distribution of $\mathcal{A}$'s view and output of $\mathbf{P_A}$ and $\mathbf{P_{au}}$ in IDEAL is thus indistinguishable from the joint distribution of $\mathcal{A}$'s view and output of $\mathbf{P_A}$ and $\mathbf{P_{au}}$ in HYBRID execution.
To prove Eq. (3), we recall that $\mathbf{P_A}$ has no output, and $\mathbf{P_{au}}$'s output is identical to the result computed by $\mathcal{S}$, so we just have to show that the $\mathcal{A}$'s view in IDEAL is computational indistinguishable from the $\mathcal{A}$'s view in HYBRID execution. Thus, we consider the following hybrid worlds.
Hybrid 1: We consider a simulator $\mathcal{S}^1$, which holds the real $\mathbf{P_A}$'s input x. This $\mathcal{S}^1$ garbles the circuit $\mathcal{C}$ honestly and send the corresponding input wire labels $\{A_{i,j,b}\}_{j\in[n_1],b\in\{0,1\}}$ to $\mathcal{A}$. This is trivial to verify that the distribution of $\mathcal{A}$'s view in the IDEAL is identical to the distribution of $\mathcal{A}$'s view in the real HYBRID execution. Thus we have

$$\left\{\text{IDEAL}^{h}_{\mathcal{C},S^1(x,z),\mathcal{P}_c}((x,y),n)\right\} \equiv \left\{\text{HYBRID}^{OT,h}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}((x,y),n)\right\}. \quad (4)$$

Hybrid 2: We consider a simulator $\mathcal{S}^2$ that works exactly as $\mathcal{S}^1$ except using an uniform distributed κ-bits $\mathbf{seed_i^{A'}}$ instead of using the correct $\mathsf{seed}^{\mathsf{A}}_i$ to compute the evidence $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ and the corresponding $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$. Due to the fact that $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ is computed by a PKE, which has indistinguishable encryptions under CCA, the distribution of $\mathcal{A}$'s view in the IDEAL is computational indistinguishable from the distribution of $\mathcal{A}$'s view in the *Hybrid 1*. Thus, we have

$$\left\{\text{IDEAL}^{h}_{\mathcal{C},S^2(x,z),\mathcal{P}_c}((x,y),n)\right\} \overset{c}{\equiv} \left\{\text{IDEAL}^{h}_{\mathcal{C},S^1(x,z),\mathcal{P}_c}((x,y),n)\right\}. \quad (5)$$

Hybrid 3: We consider a simulator $\mathcal{S}^3$ that works exactly as $\mathcal{S}^2$ except that $\mathcal{S}^3$ garbles the circuit $\mathcal{C}$ in such a way that this circuit will always be evaluated to o^B, using the appropriate input wire labels. Since Gb is a secure simulatable private garbling scheme, the distribution of $\mathcal{A}$'s view in the IDEAL is computational indistinguishable from the distribution of $\mathcal{A}$'s view in *Hybrid 2*. Thus, we have

$$\left\{\text{IDEAL}^{h}_{\mathcal{C},S^3(z),\mathcal{P}_c}((x,y),n)\right\} \overset{c}{\equiv} \left\{\text{IDEAL}^{h}_{\mathcal{C},S^2(x,z),\mathcal{P}_c}((x,y),n)\right\}. \quad (6)$$

This completes the proof. □

B. Honorific Security — Malicious $\mathbf{P_B}$

Let $\mathcal{A}$ be an adversary corrupting $\mathbf{P_A}$. In CRS model, we build the following simulator $\mathcal{S}$ which simulates a key pair of $\mathbf{P_A}$ for the signature scheme, a key pair of $\mathbf{P_{au}}$ for the PKE, holds the public key of $\mathbf{P_A}$ and runs $\mathcal{A}$ as subroutine, while playing the role of $\mathbf{P_A}$ in the ideal world interacting with $\mathcal{F}^{2pc}_{\mathcal{M}}$:

1) Play the $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ with $\mathcal{A}$ playing $\mathbf{P_A}$:
   a) If the input is $(\mathbf{cheat}, (mx^{\mathsf{m}\times\mathsf{OT},\mathsf{A}}, \mathcal{M}^{\mathsf{m}\times\mathsf{OT},\mathsf{A}}), \mathbf{P_A}, \mathsf{sid})$, send $(\mathbf{cheat}, (mx^{\mathsf{m}\times\mathsf{OT},\mathsf{A}}, \mathcal{M}^{\mathbf{A}}), \mathbf{P_A}, \mathsf{sid})$ to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and receive back $\mathbf{P_B}$'s input y. Then use the input y of $\mathbf{P_B}$ to perfectly emulate $\mathbf{P_B}$ in the rest of execution. Send o^B to $\mathcal{F}^{2pc}_{\mathcal{M}}$, which will be received by $\mathbf{P_B}$ as output. The simulation ends here for this case.
   b) If the input is $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$, proceed with the simulation below.
2) Play the role as an honest $\mathbf{P_B}$, receive GC_i, table_i, $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ and $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$, from $\mathcal{A}$
3) Check if the signature $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$ is invalid, send $\perp$ to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and halt.
4) Extract $(\mathsf{seed}^{\mathsf{A}}_i, \mathsf{sid})$ with $\mathsf{DEC}(\mathsf{dk}_{au}, \mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i)$. Compute $\hat{\mathsf{GC}}_i$ and $\hat{\mathsf{table}}_i$ using the pseudorandomness derived from $\mathsf{seed}^{\mathsf{A}}_i$.
5) For the following cases:
   a) If $\hat{\mathsf{GC}}_i \neq \mathsf{GC}_i$ or $\hat{\mathsf{table}}_i \neq \mathsf{table}_i$, send $(\mathbf{cheat}, (\mathsf{GC}_i, \mathsf{table}_i, \mathsf{seed}^{\mathsf{A}}_i, \mathcal{M}^{\mathbf{A}}), \mathbf{P_A}, \mathsf{sid})$ to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and receive back $\mathbf{P_B}$'s input y. Then use the input y of $\mathbf{P_B}$ to perfectly emulate $\mathbf{P_B}$ in the rest of execution. Send o^B to $\mathcal{F}^{2pc}_{\mathcal{M}}$, which will be received by $\mathbf{P_B}$ as output. The simulation ends here for this case.
   b) If any of $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$ is not a consistent label corresponding to $\hat{\mathsf{GC}}_i$, send $(\mathbf{cheat}, (\{B_{i,j,b}\}, \{\hat{B}_{i,j,b}\}, \mathcal{M}^{\mathbf{A}}), \mathbf{P_A}, \mathsf{sid})$ to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and receive back $\mathbf{P_B}$'s input y. Then run perfect emulation using $\mathbf{P_B}$'s input. Send o^B to $\mathcal{F}^{2pc}_{\mathcal{M}}$, which will be received by $\mathbf{P_B}$ as output. The simulation ends here for this case.
6) Upon receiving $\{A_{i,j,x[j]}\}$ from $\mathcal{A}$, check if any $\{A_{i,j,x[j]}\}$ are invalid, send ξ to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and halt (cause $\mathbf{P_B}$ to receive $\perp$). Otherwise, derive the exact $\mathcal{A}$'s input x, send to $\mathcal{F}^{2pc}_{\mathcal{M}}$ and halt.

Proof. Denote Protocol 5 as $\Pi^{\mathsf{GC}}_{\mathcal{M}}$ and $\mathcal{P}_c = \{\mathbf{P_A}\}$ (i.e. $\mathbf{P_A}$ is corrupted), we show:

$$\left\{\text{IDEAL}^{h}_{\mathcal{C},S(z),\mathcal{P}_c}(x,y)\right\} \overset{c}{\equiv} \left\{\text{HYBRID}^{OT,h}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}(x,y)\right\} \quad (7)$$

The analysis of the event $\mathsf{bad}_{\mathsf{ot}}$ and probability of both $\mathsf{bad}_{\mathsf{ot}}$ and $\neg\mathsf{bad}_{\mathsf{ot}}$ are identical to the case when $\mathbf{P_B}$ is corrupted and so we have:

$$\left\{\text{IDEAL}^{h}_{\mathcal{C},S(z),\mathcal{P}_c}(x,y)|\mathsf{bad}_{\mathsf{ot}}\right\} \equiv \left\{\text{HYBRID}^{OT,h}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}(x,y)|\mathsf{bad}_{\mathsf{ot}}\right\} \quad (8)$$

It remains to prove for the case $\neg\mathsf{bad}_{\mathsf{ot}}$. Note that after $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, $\mathcal{A}$ does not receive any message from $\mathbf{P_B}$. The $\mathcal{A}$'s view in the IDEAL is therefore indistinguishable from its view in the HYBRID execution. We point out, that this is not the case when we consider the joint distribution including $\mathbf{P_B}$'s output, because any cheating by $\mathcal{A}$ can cause $\mathbf{P_B}$ to output an incorrect value. Thus, we prove the joint distribution of $\mathbf{P_B}$'s output and $\mathbf{P_{au}}$'s output are identical in IDEAL and HYBRID execution, to show that if any cheating behavior occurs, the cheated party can cause other honest parties to output some incorrect value. However, it cannot prevent an honest auditor from having the correct auditing results.

Note that as long as $\mathcal{A}$ decides to cheat, the simulator $\mathcal{S}$ discovers cheating behavior directly by decrypting the evidence, receives $\mathbf{P_B}$'s real input y, and can thus perfectly emulate $\mathbf{P_B}$. The output o^B that received by $\mathbf{P_B}$ in IDEAL is exactly the same output that $\mathcal{S}$ has computed during the HYBRID execution. Besides, the simulator $\mathcal{S}$ extracts evidence exactly as an honest auditor will do in the HYBRID execution. Thus, the output received by $\mathbf{P_{au}}$ in IDEAL is identical to the auditing result that $\mathcal{S}$ has computed during the HYBRID execution. This completes the proof. □

C. *Honorific Security — Semi-honest $\mathbf{P_{au}}$*

For simplicity, we consider the case that $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ is implemented by a protocol with malicious security against both $\mathbf{P_A}$ and $\mathbf{P_B}$, indicating that $\mathbf{P_B}$'s cheating behavior will be detected by $\mathbf{P_A}$ already within the ideal functionality $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$. So regarding $\mathbf{P_{au}}$'s view, we will only focus on whether $\mathbf{P_A}$'s cheating behavior takes place in the rest of proofs. Let $\mathcal{A}$ be an adversary corrupting $\mathbf{P_{au}}$. In CRS model, we build the following simulator $\mathcal{S}$ who simulates the two key pairs of $\mathbf{P_A}$ and $\mathbf{P_B}$ for the signature scheme, knows the public key of $\mathbf{P_{au}}$ for PKE, and receives adversary $\mathcal{A}$'s output $\mathcal{M}^{\mathbf{A}}(mx^{\mathsf{A}})$. $\mathcal{S}$ runs $\mathcal{A}$ as subroutine, while playing the role of $\mathbf{P_{au}}$ in the ideal world interacting with $\mathcal{F}^{2pc}_{\mathcal{M}}$.

a. The output $\mathcal{M}^{\mathbf{A}}(mx^{\mathsf{A}})$ received by $\mathcal{S}$ indicates that $\mathbf{P_A}$ has behaved honestly:
   1) Play the role of an honest $\mathbf{P_A}$, choose uniform distributed κ-bit $\mathsf{seed}^{\mathsf{A}}_i$ to compute $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$.
   2) Play $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ between $\mathbf{P_A}$ and $\mathbf{P_B}$, where $\mathbf{P_A}$ uses $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$ and $\mathbf{P_B}$ uses y as input.
   3) Use $\mathsf{seed}^{\mathsf{A}}_i$ to garble circuit $\mathcal{C}$ and thus receive GC_i and table_i. Then compute $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ and $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$.
   4) Play role of $\mathbf{P_B}$, send the auditing query $(\mathcal{C},\mathsf{GC}_i,\mathsf{table}_i,\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i,\sigma^{\mathsf{A},\mathsf{GC}_i}_i,\mathsf{sid})$ to $\mathcal{A}$.
   5) Play the role of $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, send $((\mathcal{M}^{\mathsf{m\times OT,A}},\mathbf{P_A}),\mathbf{P_B},\mathsf{sid})$ to $\mathcal{A}$. After receiving $(\{\hat{B}_{i,j,b}\}$ from $\mathcal{A}$, send $\mathcal{M}^{\mathsf{m\times OT,A}}(\{B_{i,j,b}\},\{\hat{B}_{i,j,b}\},\mathsf{sid})$ to $\mathcal{A}$ and halt.

b. The output $\mathcal{M}^{\mathbf{A}}(mx^{\mathsf{A}})$ received by $\mathcal{S}$ indicates that $\mathbf{P_A}$ has cheated. With $\mathcal{M}^{\mathbf{A}}$, the simulator is able to restore the cheated behavior of $\mathbf{P_A}$ during the protocol execution:
   - We consider the first case when $\mathbf{P_A}$ has sent an incorrect GC_i or table_i.
     1) Play the role of $\mathbf{P_A}$, choose uniform distributed κ-bit $\mathsf{seed}^{\mathsf{A}}_i$ to compute $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$.
     2) Play $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ between $\mathbf{P_A}$ and $\mathbf{P_B}$, where $\mathbf{P_A}$ uses $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$ and $\mathbf{P_B}$ uses y as input.
     3) Use $\mathsf{seed}^{\mathsf{A}}_i$ to garble circuit $\mathcal{C}$ and thus receive GC_i. Make use of $\mathcal{M}^{\mathbf{A}}(\bar{\mathcal{C}}^{\mathsf{A}})$, replace the correct part of GC_i with incorrect strings, denoted by GC'_i. Or replace the correct table_i with table'_i. Then compute $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ and $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$ using the modified GC'_i and table'_i.
     4) Send $(\mathcal{C},\mathsf{GC}'_i,\mathsf{table}'_i,\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i,\sigma^{\mathsf{A},\mathsf{GC}_i}_i,\mathsf{sid})$ to $\mathcal{A}$.
     5) Play the role of $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, send $((\mathcal{M}^{\mathsf{m\times OT,A}},\mathbf{P_A}),\ \mathbf{P_B},\mathsf{sid})$ to $\mathcal{A}$. After receiving $(\{\hat{B}_{i,j,b}\},\mathbf{P_{au}},\mathsf{sid})$ from $\mathcal{A}$, send $\mathcal{M}^{\mathsf{m\times OT,A}}(\{B_{i,j,b}\},\{\hat{B}_{i,j,b}\},\mathsf{sid})$ and halt.
   - We consider the second case when $\mathbf{P_A}$ has handed an incorrect label to $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$.
     1) Play the role of $\mathbf{P_A}$, choose uniform distributed κ-bit $\mathsf{seed}^{\mathsf{A}}_i$ to compute $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$.
     2) Play $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ between $\mathbf{P_A}$ and $\mathbf{P_B}$, where $\mathbf{P_A}$ uses $\{B_{i,j,b}\}_{j\in[n_2],b\in\{0,1\}}$ and $\mathbf{P_B}$ uses y as input.
     3) Use $\mathsf{seed}^{\mathsf{A}}_i$ to garble circuit $\mathcal{C}$ and thus receive GC_i and table_i. Then compute $\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i$ and $\sigma^{\mathsf{A},\mathsf{GC}_i}_i$.
     4) Send $(\mathcal{C},\mathsf{GC}_i,\mathsf{table}_i,\mathsf{ct}^{\mathsf{A},\mathsf{GC}_i}_i,\sigma^{\mathsf{A},\mathsf{GC}_i}_i,\mathsf{sid})$ to $\mathcal{A}$.
     5) Play the role of $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$, send $((\mathcal{M}^{\mathsf{m\times OT,A}},\mathbf{P_A}),\mathbf{P_B},\mathsf{sid})$ to $\mathcal{A}$. After receiving $(\{\hat{B}_{i,j,b}\}$ from $\mathcal{A}$, send $\mathcal{M}^{\mathsf{m\times OT,A}}(\{B_{i,j,b}\},\{\hat{B}_{i,j,b}\},\mathsf{sid})$ and halt.

Proof. Denote Protocol in Figure 5 as $\Pi^{\mathsf{GC}}_{\mathcal{M}}$ and $\mathcal{P}_c = \{\mathbf{P_{au}}\}$ (i.e. $\mathbf{P_{au}}$ is corrupted), we show

$$\left\{\text{IDEAL}^{semi}_{\mathcal{C},S(z),\mathcal{P}_c}((x,y),n)\right\} \overset{c}{\equiv} \left\{\text{HYBRID}^{OT,semi}_{\Pi^{\mathsf{GC}}_{\mathcal{M}},A(z),\mathcal{P}_c}((x,y),n)\right\}. \quad (9)$$

Since the output received by $\mathcal{A}$ reflects already, in which part of protocol execution $\mathbf{P_A}$ has cheated, the simulator can thus emulate the cheating behavior exactly as a cheated $\mathbf{P_A}$ has done during the HYBRID execution. Besides, the evidence is the only message received by $\mathcal{A}$, the $\mathcal{A}$'s view in ideal world is indistinguishable from its view in HYBRID execution, this completes the proof. □

VI. Secure Oblivious Transfer with Auditing

In this section, we show how to convert an OT protocol (with malicious security) to a protocol implementing $\mathcal{F}^{OT}_{\mathcal{M}^{OT,A}}$. If we run parallel version of this OT protocol, we can easily see that this parallel version implements $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$. This is a security upgrade, since with our new notion the converted OT can now detect inconsistency in inputs referring to external evidence, which is not possible for standalone malicious OT.

We first take the OT protocol with malicious security [28] as an example, and provide a prove sketch that the modified protocol implements the $\mathcal{F}^{OT}_{\mathcal{M}^{OT,A}}$. Due to the space limitation, we do not provide a detailed description of the original protocol here and refer the reader to the formal description of the modified protocol in Figure 6 in Appendix.

Theorem 2. *Assume that* $\mathsf{H}()$ *is a collision-resistant hash function,* SIG *is existentially unforgeable under a chosen-message attack,* PKE *has indistinguishable encryptions under a chosen-ciphertext attack. Protocol* $\Pi^{OT}_{\mathcal{M}^{OT,A}}$ *described in Fig.6 securely realizes* $\mathcal{F}^{OT}_{\mathcal{M}^{OT,A}}$*, and thus its parallel version securely realizes* $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ *with public verifiability described in Fig.4, in the presence of an malicious adversary who can corrupt either* $\mathbf{P_A}$ *or* $\mathbf{P_B}$*, or in the presence of a semi-honest adversary who can corrupt* $\mathbf{P_{au}}$*, with static corruption.*

The security analysis is similar.

A. Honorific Security — Malicious $\boldsymbol{P_A}$*:*

Peikert *et. al* have already proved that the unmodified protocol (without * steps shown in Fig.6) is secure against the malicious $\mathbf{P_A}$. If we consider the joint distribution of the output of $\mathbf{P_{au}}$, since the internal cheat behaviors are already detected by $\mathbf{P_B}$, the only remaining auditing algorithm $\mathcal{M}^{\mathsf{OT,A}}$ will be to verify if $\mathbf{P_A}$ has handed the correct $\{x^0, x^1\}$. Recall that in CRS model, the simulator is able to forge the key pair of $\mathbf{P_{au}}$ and extract the encrypted rnd as mx^{A}, exactly as an honest $\mathbf{P_{au}}$ will do during the real protocol execution. When H is a collision-resistant hash function, the ciphertext of $\mathsf{OT.PKE}$ is a binding commitment of the randomness-message pair, a malicious $\mathbf{P_A}$ succeeds in finding a rnd' so that $\mathsf{H}(\mathsf{OT.ENC}(\{x^0,\mathsf{rnd}'\})||\mathsf{OT.ENC}(\{x^1,\mathsf{rnd}'\}))$ is identical with $\mathsf{H}(\mathsf{OT.ENC}(\{\hat{x}^0,\mathsf{rnd}'\})||\mathsf{OT.ENC}(\{\hat{x}^1,\mathsf{rnd}'\}))$, only with negligible probability. Thus, a malicious $\mathbf{P_A}$ cannot cause $\mathbf{P_{au}}$ to output an incorrect result. In the new protocol, we stress that $\mathbf{P_A}$ does not receive any more message from $\mathbf{P_B}$ and $\mathbf{P_{au}}$, and thus do not provide further proof and directly derive malicious security from the proof of original paper.

B. Honorific Security — Malicious $\boldsymbol{P_B}$*:*

We follow the same reasoning above and argue that receiving two additional messages $\mathsf{ct}^{\mathsf{A,OT}}$ and $\sigma^{\mathsf{A,OT}}$ does not provide a malicious $\mathbf{P_B}$ with any new ability. Besides, it can not cause $\mathbf{P_{au}}$ to output an incorrect result. When SIG is existentially unforgeable under CMA and PKE has indistinguishable encryptions under CCA, the output of $\mathbf{P_{au}}$ is not decided by $\mathbf{P_B}$, but by $\mathbf{P_A}$. Thus, we do not provide further proof and directly conclude with the malicious security from original paper [28].

C. Honorific Security — Semi-honest $\boldsymbol{P_{Au}}$*:*

We can easily prove that the adversary's view in ideal world is *identical* with its view in HYBRID execution, since the simulator $\mathcal{S}$ receives the input $(\hat{x}^0, \hat{x}^1)$ and the output $\mathcal{M}^{\mathsf{OT,A}}(\{x_0,x_1\},\{\hat{x}_0,\hat{x}_1\})$ obtained by adversary, which enables the simulator to perfectly emulate $\mathbf{P_A}$'s behavior. Recall that if a malicious $\mathbf{P_A}$ has encrypted "a wrong" rnd as evidence, the real rnd and $\{x_0, x_1\}$ are then uniform distributed to $\mathcal{A}$, which can easily be simulated. Since H is a collision-resistant hash function and PKE has CCA security, the adversary cannot distinguish its view in ideal generated by simulator and the view in the real protocol execution.

VII. OTE against Honorific Adversaries

In this section, we present our new protocol (extended from OTE in [2]) with improved security against a honorific receiver and a semi-honest auditor. A formal definition is provided in Fig.7 in Appendix. We stress that, this new protocol already realizes $\mathcal{F}^{m\times OT}_{\mathcal{M}}$. For conciseness, we only show how the original protocol can benefit from the presence of a semi-honest auditor, and discuss the improvement from $\mathcal{F}^{m\times OT}_{\mathcal{M}}$ to $\mathcal{F}^{m\times OT}_{\mathcal{M}^{m\times OT,A}}$ at the end of this section.

Theorem 3. *Assume* $\mathsf{H}()$ *is a collision-resistant hash function,* SIG *is existentially unforgeable under a chosen-message attack,* PKE *has indistinguishable encryptions under a chosen-ciphertext attack,* G *is a pseudo-random generator. Protocol* $\Pi^{m\times OT}_{\mathcal{M}}$ *described in Fig. 7 securely realizes* $\mathcal{F}^{m\times OT}_{\mathcal{M}}$ *with public verifiablity in the* $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$*-hybrid model, in the presence of a malicious adversary who can corrupt* $\mathbf{P_A}$*, or in the presence of an honorific adversary who can corrupt* $\mathbf{P_B}$*, or in the presence of a semi-honest adversary who can corrupt* $\mathbf{P_{au}}$*, with static corruption.*

We refer the reader to Appendix C for the complete proof of Theorem 3.

Remark on Auditor. To improve the implemented auditing algorithm $\mathcal{M}$ in $\Pi^{m\times OT}_{\mathcal{M}}$ so that the consistency of $\mathbf{P_A}$'s input could be proved by $\mathbf{P_{au}}$, we only have to force $\mathbf{P_A}$ to also include its randomness used within the OTE protocol as evidence and send this with a corresponding signature to $\mathbf{P_B}$. However, there is a subtlety in OTE here when $\mathcal{M}$ is defined as follows:

(1) When $\mathbf{P_A}$ sends an auditing query $(\{u^i \oplus u^j\}_{i,j\in[l]}, \mathsf{ct}^{\mathsf{B,OT}}, \sigma^{\mathsf{B,OT}}, \mathsf{sid})$ to $\mathbf{P_{au}}$, $\mathbf{P_{au}}$ is able to decrypt $(\mathsf{seed}^{\mathsf{B}}, \mathsf{sid}) \leftarrow \mathsf{DEC}(\mathsf{dk}_{au}, \mathsf{ct}^{\mathsf{B,OT}})$ and thus compute $\{k_0, k_1\}^l$.
(2) When $\mathbf{P_B}$ sends an auditing query $(\{u^i\}, \{\mathsf{H}(y^0_j), \mathsf{H}(y^1_j)\}^m, \mathsf{ct}^{\mathsf{A,OT}}, \sigma^{\mathsf{A,OT}}, \mathsf{sid})$ to $\mathbf{P_{au}}$ for a input consistency check.

And in case both parties try to prove misbehavior of each other by submitting the evidence, there exists an attack by $\mathbf{P_{au}}$ reconstructing the full receiver's view. Once $\mathbf{P_{au}}$ has both $\{u^i\}^l$ and $\{k_0, k_1\}^l$, it can compute $\mathbf{P_B}$'s private input by:

$$r \leftarrow u^i \oplus \mathsf{G}(k^i_0) \oplus \mathsf{G}(k^i_1)$$

For this case, we will have to separate both auditing algorithms.

VIII. Conclusion and Future Work

In this paper, we propose a new security notion *honorific security* in UC framework. We show this notion provides good security guarantee and implies high efficiency by constructing OT, OTE and efficient GC-based MPC protocols with provable security. We believe that it is extremely meaningful to construct sharing-based MPC protocols in our model in the future, which may involve non-interactive zero-knowledge proof.

References

[1] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13. Citeseer, 2013, p. 7.

[2] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 535–548.

[3] ——, "More efficient oblivious transfer extensions with security for malicious adversaries," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 673–701.

[4] G. Asharov and C. Orlandi, "Calling out cheaters: Covert security with public verifiability," in *Advances in Cryptology – ASIACRYPT 2012*, X. Wang and K. Sako, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 681–698.

[5] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," vol. 23, no. 2. Springer, 2010, pp. 281–343.

[6] S. Badrinarayanan, A. Jain, R. Ostrovsky, and I. Visconti, "Uc-secure multiparty computation from one-way functions using stateless tokens," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2019, pp. 577–605.

[7] D. Beaver, "Commodity-based cryptography," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 446–455.

[8] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," 2012, https://ia.cr/2012/265.

[9] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[10] I. Damgård, M. Geisler, and J. B. Nielsen, "From passive to covert security at low cost," 2009, https://ia.cr/2009/592.

[11] W. Du and Z. Zhan, "A practical approach to solve secure multi-party computation problems," in *Proceedings of the 2002 workshop on New security paradigms*, 2002, pp. 127–135.

[12] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2-3, 2017.

[13] D. Evans, V. Kolesnikov, M. Rosulek *et al.*, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2-3, pp. 70–246, 2018.

[14] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[15] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia, "Founding cryptography on tamper-proof hardware tokens," in *Theory of Cryptography Conference*. Springer, 2010, pp. 308–326.

[16] V. Goyal, P. Mohassel, and A. Smith, "Efficient two party and multi party computation against covert adversaries," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 289–306.

[17] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 415–432.

[18] C. Hong, J. Katz, V. Kolesnikov, W. jie Lu, and X. Wang, "Covert security with public verifiability: Faster, leaner, and simpler," 2018, https://ia.cr/2018/1108.

[19] Y. Ishai, R. Ostrovsky, and V. Zikas, "Secure multi-party computation with identifiable abort," in *Annual Cryptology Conference*. Springer, 2014, pp. 369–386.

[20] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[21] J. Katz, S. Ranellucci, M. Rosulek, and X. Wang, "Optimizing authenticated garbling for faster secure two-party computation," 2018, https://ia.cr/2018/578.

[22] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making spdz great again," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 158–189.

[23] V. Kolesnikov and A. J. Malozemoff, "Public verifiability in the covert model (almost) for free," in *Advances in Cryptology – ASIACRYPT 2015*, T. Iwata and J. H. Cheon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 210–235.

[24] Y. Lindell, "Fast cut-and-choose-based protocols for malicious and covert adversaries," *Journal of Cryptology*, vol. 29, no. 2, pp. 456–490, 2016.

[25] ——, "How to simulate it–a tutorial on the simulation proof technique," *Tutorials on the Foundations of Cryptography*, pp. 277–346, 2017.

[26] Y. Lindell and B. Pinkas, "An efficient protocol for secure two-party computation in the presence of malicious adversaries," 2008, https://ia.cr/2008/049.

[27] Y. Lu, B. Zhang, H.-S. Zhou, W. Liu, L. Zhang, and K. Ren, "Correlated randomness teleportation via semi-trusted hardware—enabling silent multi-party computation," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 699–720.

[28] C. Peikert, V. Vaikuntanathan, and B. Waters, "A framework for efficient and composable oblivious transfer," in *Annual international cryptology conference*. Springer, 2008, pp. 554–571.

[29] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," 2017, https://ia.cr/2017/030.

Appendix

A. OT protocol with auditing algorithm.

See Figure 6.

B. OTE protocol against honorific adversaries.

See Figure 7.

C. Honorific Security — Malicious $\mathbf{P_A}$ in OTE

The original paper [2] already provides security against a malicious $\mathbf{P_A}$ [3]. Since SIG is unforgeable under CMA, PKE has indistinguishable encryptions under CCA, receiving two additional messages $\mathsf{ct}^{\mathsf{B,OT}}$ and $\sigma^{\mathsf{B,OT}}$ does not provide a malicious $\mathbf{P_A}$ with more information or new ability. Besides, a malicious $\mathbf{P_A}$ cannot cause $\mathbf{P_{au}}$ to output an incorrect result, since this only depends on the $\mathsf{ct}^{\mathsf{B,OT}}$ and $\sigma^{\mathsf{B,OT}}$ sent by $\mathbf{P_B}$. Thus we do not further prove security but directly derive security against a malicious $\mathbf{P_A}$ from the original paper.

D. Honorific Security — Malicious $\mathbf{P_B}$ in OTE

Let $\mathcal{A}$ be an adversary corrupting $\mathbf{P_B}$. In CRS model, we build the following simulator $\mathcal{S}$ which forges a key pair of $\mathbf{P_A}$ for the signature scheme, a key pair of $\mathbf{P_{au}}$ for the PKE, holds the public key of $\mathbf{P_B}$ and runs $\mathcal{A}$ as subroutine, while playing the role of $\mathbf{P_B}$ in the ideal world interacting with $\mathcal{F}_{\mathcal{M}}^{m\times OT}$:

1) Play the $\mathcal{F}_{\mathcal{M}^{l\times OT,B}}^{l\times OT}$ with $\mathcal{A}$ playing $\mathbf{P_B}$:
   a) If input is $(\mathbf{cheat}, (mx^{\mathsf{l\times OT,B}}, \mathcal{M}^{\mathsf{l\times OT,B}}), \mathbf{P_B}, \mathsf{sid})$, send $(\mathbf{cheat}, (mx^{\mathsf{l\times OT,B}}, \mathcal{M}^{\mathsf{m\times OT,B}}), \mathbf{P_B}, \mathsf{sid})$ to $\mathcal{F}_{\mathcal{M}}^{m\times OT}$ and receive back $\mathbf{P_A}$'s input $\{x_j^0, x_j^1\}^m$. Then use the real $\mathbf{P_A}$'s input to perfectly emulate $\mathbf{P_A}$ in the rest of execution. Send o^A to $\mathcal{F}_{\mathcal{M}}^{m\times OT}$, which will be received by $\mathbf{P_A}$ as output. The simulation ends here for this case.

Private inputs: $\mathbf{P_A}$ has input $(\mathsf{sid}, \mathsf{ssid}, x^0, x^1)$ and a key pair $\{\mathsf{pk}_A, \mathsf{sk}_A\}$ for a signature scheme, $\mathbf{P_B}$ has input $(\mathsf{sid}, \mathsf{ssid}, \mathbf{r})$ and a key pair $\{\mathsf{pk}_B, \mathsf{sk}_B\}$ for a signature scheme. $\mathbf{P_{au}}$ has a key pair $\{\mathsf{ek}_{au}, \mathsf{dk}_{au}\}$ for a PKE scheme and conditionally reference input $(\hat{x}^0, \hat{x}^1)$.
Public inputs: Both parties know the public key pk_i of each other and the ek_{au} for the auditing PKE scheme. They both agree to deploy $\mathcal{F}_{CRS}^{\mathsf{mode}}$, OT.KeyGen, OT.ENC, OT.DEC. $\mathbf{P_{au}}$ knows the pk_i of $\mathbf{P_A}$ and $\mathbf{P_B}$.

Protocol:

1) $\mathbf{P_B}$ queries $\mathcal{F}_{CRS}^{\mathsf{mode}}$ with $(\mathsf{sid}, \mathbf{P_A}, \mathbf{P_B})$ and gets back $(\mathsf{sid}, \mathsf{crs})$.
2) $\mathbf{P_A}$ queries $\mathcal{F}_{CRS}^{\mathsf{mode}}$ with $(\mathsf{sid}, \mathbf{P_A}, \mathbf{P_B})$ and gets back $(\mathsf{sid}, \mathsf{crs})$.
3) $\mathbf{P_B}$ computes $(\mathsf{ek_{OT}}, \mathsf{dk_{OT}}) \leftarrow \mathsf{OT.KeyGen}(\mathsf{crs}, \mathbf{r})$, sends $(\mathsf{sid}, \mathsf{ssid}, \mathsf{ek_{OT}})$ to $\mathbf{P_A}$, and stores $(\mathsf{sid}, \mathsf{ssid}, \mathsf{dk_{OT}})$.
4) $\mathbf{P_A}$ computes $y_0 \leftarrow \mathsf{OT.ENC}(\mathsf{ek_{OT}}, 0, x_0)$ and $y_1 \leftarrow \mathsf{OT.ENC}(\mathsf{ek_{OT}}, 1, x_1)$ and sends $(\mathsf{sid}, \mathsf{ssid}, y_0, y_1)$ to $\mathbf{P_B}$.

* $\mathbf{P_A}$ compute $\mathsf{ct}^{\mathsf{A,OT}} \leftarrow \mathsf{ENC}(\mathsf{ek}_{au}, \mathsf{rnd})$ where the rnd denotes all randomness $\mathbf{P_A}$ has used for the OT.ENC, and a signature $\sigma^{\mathsf{A,OT}} \leftarrow \mathsf{SIG.Sign}(\mathsf{pk}_A, \mathsf{H}(y_0||y_1), \mathsf{ct}^{\mathsf{A,OT}}, \mathsf{ek_{OT}}, \mathsf{crs}, \mathsf{sid}, \mathsf{ssid})$. Then $\mathbf{P_A}$ sends $\mathsf{ct}^{\mathsf{A,OT}}$ and $\sigma^{\mathsf{A,OT}}$ to $\mathbf{P_B}$, who checks whether $\sigma^{\mathsf{A,OT}}$ is valid, aborts if not.

5) $\mathbf{P_B}$ outputs $(\mathsf{sid}, \mathsf{ssid}, x_r)$ where $x_r \leftarrow \mathsf{OT.DEC}(\mathsf{dk_{OT}}, y_r)$.

* **Input consistency auditing:**

a) $\mathbf{P_B}$ sends an auditing query $(\{y_0, y_1\}, \mathsf{ct}^{\mathsf{A,OT}}, \sigma^{\mathsf{A,OT}}, \mathsf{ek_{OT}}, \mathsf{crs}, \mathsf{sid}, \mathsf{ssid})$ to $\mathbf{P_{au}}$, who verifies whether $\sigma^{\mathsf{A,OT}}$ is valid, aborts with $\perp$ if not.
b) $\mathbf{P_{au}}$ computes $\mathsf{rnd} \leftarrow \mathsf{OT.DEC}(\mathsf{dk_{OT}}, \mathsf{ct}^{\mathsf{A,OT}})$ and then $\hat{y}_0 \leftarrow \mathsf{OT.ENC}(\mathsf{ek_{OT}}, 0, \hat{x}_0, \mathsf{rnd})$ and $\hat{y}_1 \leftarrow \mathsf{OT.ENC}(\mathsf{ek_{OT}}, 1, \hat{x}_1, \mathsf{rnd})$.
c) $\mathbf{P_{au}}$ verifies whether $\mathsf{H}(y_0||y_1) = \mathsf{H}(\hat{y_0}||\hat{y_1})$, broadcasts the result.

Fig. 6: A secure OT protocol (malicious security) with auditing algorithm

b) If input is $\{k_0, k_1\}^l$, proceed with the simulation below.

2) Play the role as an honest $\mathbf{P_A}$, receive $\{\mathbf{u}\}^l$, $\mathsf{ct}^{\mathsf{B,OT}}$, $\sigma^{\mathsf{B,OT}}$, check whether $\sigma^{\mathsf{B,OT}}$ is invalid, send $\perp$ to $\mathcal{F}_{\mathcal{M}}^{m\times OT}$ and halt.
3) Extract $(\mathsf{seed}^{\mathsf{B}}, \mathsf{sid}) \leftarrow \mathsf{DEC}(\mathsf{dk}_{au}, \mathsf{ct}^{\mathsf{B,OT}})$ and compute $\{\hat{k}_0, \hat{k}_1\}^l$.
4) For the following case:

a) If for any $i, j \in [l]$:
$$u^i \oplus u^j \oplus \hat{k}_0^i \oplus \hat{k}_1^i \oplus \hat{k}_0^j \oplus \hat{k}_1^j \neq 0$$
send $(\mathbf{cheat}, (mx^{\mathsf{m\times OT,B}}, \mathcal{M}^{\mathsf{m\times OT,B}}), \mathbf{P_B}, \mathsf{sid})$ to $\mathcal{F}_{\mathcal{M}^{m\times OT,A}}^{m\times OT}$ and receive back $\mathbf{P_A}$'s input $\{x_j^0, x_j^1\}^m$. Then use the real $\mathbf{P_A}$'s input to perfectly emulate $\mathbf{P_A}$ in the rest of execution. Send o^A to $\mathcal{F}_{\mathcal{M}}^{m\times OT}$, which will be received by $\mathbf{P_A}$ as output. The simulation ends here for this case.

b) If $\{\hat{k}_0, \hat{k}_1\}^l \neq \{k_0, k_1\}^l$, do the same as above. The simulation ends here for this case.

5) Extract $\mathcal{A}$'s input r and send to $\mathcal{F}_{\mathcal{M}}^{m\times OT}$. Upon receive $\{x^{r_j}\}^m$, use $\{u^i\}^l$, s, $\{k_{s_i}\}^l$ and received $\{x^{r_j}\}^m$ to compute $\{y^{r_j}\}^m$, set $\{y^{\overline{r_j}}\}^m$ uniformly random. Send $\{y^b\}^m$ to $\mathcal{A}$ and halt.

Proof. Denote protocol 7 as $\Pi_{\mathcal{M}}^{m\times OT}$ and set $\mathcal{P}_c = \{\mathbf{P_B}\}$ (i.e. $\mathbf{P_B}$ is corrupted), we show:
$$\left\{\mathrm{IDEAL}_{m\times OT,S(z),\mathcal{P}_c}^{h}((x,y),n)\right\} \overset{c}{\equiv} \left\{\mathrm{HYBRID}_{\Pi_{\mathcal{M}}^{m\times OT},A(z),\mathcal{P}_c}^{l\times OT,h}((x,y),n)\right\} \quad (10)$$
We still consider two events $\mathsf{bad_{baseOT}}$ and $\neg\mathsf{bad_{baseOT}}$. The analysis of the event $\mathsf{bad_{baseOT}}$ and probability of both $\mathsf{bad_{baseOT}}$ and $\neg\mathsf{bad_{baseOT}}$ are identical to the case when $\mathbf{P_B}$ is corrupted in GC protocol, so we have:
$$\left\{\mathrm{IDEAL}_{m\times OT,S(z),\mathcal{P}_c}^{h}(x,y)|\mathsf{bad_{baseOT}}\right\} \overset{c}{\equiv} \left\{\mathrm{HYBRID}_{\Pi_{\mathcal{M}}^{m\times OT},A(z),\mathcal{P}_c}^{l\times OT,h}(x,y)|\mathsf{bad_{baseOT}}\right\} \quad (11)$$
It remains to prove for the case $\neg\mathsf{bad_{baseOT}}$. For both cases when $\mathcal{A}$ decides to cheat by sending $\{u^i\}^l$ with inconsistent choice bits r or handing inconsistent $\{k_0, k_1\}^l$ in $\mathcal{F}_{\mathcal{M}^{l\times OT,B}}^{l\times OT}$ and $\mathsf{ct}^{\mathsf{B,OT}}$, the simulator $\mathcal{S}$ receives back the real $\mathbf{P_A}$'s input $\{x_j^0, x_j^1\}^m$ and can thus perfectly emulate $\mathbf{P_A}$'s behavior in the rest of HYBRID execution. Since $\mathbf{P_A}$ does not have output from protocol 7, and $\mathbf{P_{au}}$ outputs exactly what $\mathcal{S}$ outputs during the HYBRID execution, we have just shown that the joint distribution of $\mathcal{A}$'s view and both the outputs of $\mathbf{P_A}$ and $\mathbf{P_{au}}$ in IDEAL is identical to the joint distribution of $\mathcal{A}$'s view and the outputs of $\mathbf{P_A}$ and $\mathbf{P_{au}}$ in the HYBRID execution.

For the second case when $\mathcal{A}$ behaves honestly, we consider the following hybrid worlds:
Hybrid 1: We consider a simulator $\mathcal{S}^1$ holds $\mathbf{P_A}$'s input $\{x_j^0, x_j^1\}^m$. It is trivial to verify that the distribution of $\mathcal{A}$'s view in IDEAL is identical to the distribution of $\mathcal{A}$'s view in HYBRID execution. Thus we have shown that:
$$\left\{\mathrm{IDEAL}_{m\times OT,S^1(z,\{x_0,x_1\}^m),\mathcal{P}_c}^{h}((x,y),n)|\neg\mathsf{bad_{baseOT}}\right\} \overset{c}{\equiv} \left\{\mathrm{HYBRID}_{\Pi_{\mathcal{M}}^{m\times OT},A(z),\mathcal{P}_c}^{l\times OT,h}((x,y),n)|\neg\mathsf{bad_{baseOT}}\right\} \quad (12)$$
Hybrid 2: We consider a simulator $\mathcal{S}^2$ works exactly as $\mathcal{S}^1$ except holding only $\{x^{r_j}\}$, not $\{x_0, x_1\}$. Now we show that the distribution generated by $\mathcal{S}^2$ and $\mathcal{S}^1$ are computationally indistinguishable. Because $\mathbf{A}$ does not know the choice bit s.

Private inputs: $\mathbf{P_A}$ has input $\{(x_j^0, x_j^1)\}_{j\in[m]}$ and key pair $\{\mathsf{pk}_A, \mathsf{sk}_A\}$ for a signature scheme, $\mathbf{P_B}$ has input $\mathbf{r} = (r_1, ..., r_m)$ and key pair $\{\mathsf{pk}_B, \mathsf{sk}_B\}$ for a signature scheme. $\mathbf{P_{au}}$ has a key pair $\{\mathsf{ek}_{au}, \mathsf{dk}_{au}\}$ for a PKE scheme.
Public inputs: Both parties know the public key pk_i of each other and the ek for a encryption scheme. They agree to deploy an ideal functionality $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$. $\mathbf{P_{au}}$ know the public key pk_i of $\mathbf{P_A}$ and $\mathbf{P_B}$.

Protocol:

1) $\mathbf{P_A}$ initializes a random vector $\mathbf{s} \in \{0,1\}^l$ and $\mathbf{P_B}$ use randomness derived from $\mathsf{seed}^{\mathsf{B}}$ to generate $\{k_0, k_1\}^l$ each of size κ.
2) Parties proceed with a $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$ where $\mathbf{P_A}$ acts as the receiver with input $\mathbf{s}$ and $\mathbf{P_B}$ acts as the sender with input $\{k_0, k_1\}^l$.
3) For $i \in [l]$, let $\mathbf{t^i} = \mathsf{G}(k_0^i)$. Let $T = [t^1|...|t^l]$ denote the $m \times l$ matrix where ith column is $\mathbf{t^i}$. Let $\mathbf{t_j}$ denote the jth row of T. $\mathbf{P_B}$ uses input choice bits $\mathbf{r}$ to compute $\mathbf{t^i} = \mathsf{G}(k_0^i)$ and $\mathbf{u^i} = \mathbf{t^i} \oplus \mathsf{G}(k_1^i) \oplus r$.
4) $\mathbf{P_B}$ then computes an evidence $\mathsf{ct}^{\mathsf{B,OT}} \leftarrow \mathsf{ENC}(\mathsf{ek}, \mathsf{seed}^{\mathsf{B}}, \mathsf{sid})$ and a signature $\sigma^{\mathsf{B,OT}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_B, \{u^i \oplus u^j\}_{i,j\in[l]}, \mathsf{ct}^{\mathsf{B,OT}}, \mathsf{sid})$.
5) $\mathbf{P_B}$ sends $\{u^i\}^l$, $\mathsf{ct}^{\mathsf{B,OT}}$, $\sigma^{\mathsf{B,OT}}$ to $\mathbf{P_A}$, who checks if the signature $\sigma^{\mathsf{B,OT}}$ is valid, aborts with output $\perp$ if not.
 * **Receiver's choice-bits auditing first part:**
   a) $\mathbf{P_A}$ sends an auditing query $(\{u^i \oplus u^j\}_{i,j\in[l]}, \mathsf{ct}^{\mathsf{B,OT}}, \sigma^{\mathsf{B,OT}}, \mathsf{sid})$ to $\mathbf{P_{au}}$, who checks whether $\sigma^{\mathsf{B,OT}}$ is valid, aborts with $\perp$ if not.
   b) $\mathbf{P_{au}}$ retrieves $(\mathsf{seed}^{\mathsf{B}}, \mathsf{sid}) \leftarrow \mathsf{DEC}(\mathsf{dk}_{au}, \mathsf{ct}^{\mathsf{B,OT}})$ and can thus compute $\{\hat{k}_0, \hat{k}_1\}^l$.
   c) $\mathbf{P_{au}}$ verifies if for every $i, j \in [l]$:
$$u^i \oplus u^j \oplus \hat{k}_0^i \oplus \hat{k}_1^i \oplus \hat{k}_0^j \oplus \hat{k}_1^j = 0$$
   d) $\mathbf{P_{au}}$ broadcasts the result.
 * **Receiver's choice-bits auditing second part:**
   a) $\mathbf{P_A}$ sends an auditing query $((\mathbf{audit}, \mathbf{P_B}), \mathbf{P_A}, \mathsf{sid})$ to $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$, which sends a notification $((\mathcal{M}^{\mathsf{l\times OT,B}}, \mathbf{P_B}), \mathbf{P_A}, \mathsf{sid})$ to $\mathbf{P_{au}}$.
   b) Upon receiving $(\{k_0, k_1\}^l, \mathbf{P_{au}}, \mathsf{sid})$ from $\mathbf{P_{au}}$, $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$ sends $\mathcal{M}^{\mathsf{l\times OT,B}}(\{k_0, k_1\}^l, \{\hat{k}_0, \hat{k}_1\}^l)$ to $\mathbf{P_{au}}$.
   c) Upon receiving $(\mathbf{deliver}, \mathbf{P_{au}}, \mathsf{sid})$ from $\mathbf{P_{au}}$, $\mathcal{F}^{l\times OT}_{\mathcal{M}^{l\times OT,B}}$ broadcasts $\mathcal{M}^{\mathsf{l\times OT,B}}(\{k_0, k_1\}^l, \{\hat{k}_0, \hat{k}_1\}^l)$.
6) For $i \in [l]$, $\mathbf{P_A}$ defines $\mathbf{q^i} = (s_i \cdot \mathbf{u^i}) \oplus \mathsf{G}(k_{s_i}^i)$. Let $Q = [q^1|...|q^l]$ denote the $m \times l$ bit matrix with $\mathbf{q^i}$ represents its ith column. Let $\mathbf{q_j}$ denote the jth row of the matrix Q. Note that $\mathbf{q^i} = (s_i \cdot \mathbf{r}) \oplus \mathbf{t^i}$ and $\mathbf{q_j} = (r_j \cdot \mathbf{s}) \oplus \mathbf{t_j}$.
7) $\mathbf{P_A}$ computes and sends $\{y_j^0, y_j^1\}_{j\in[m]}$, where:
$$y_j^0 = x_j^0 \oplus \mathsf{H}(j, q_j) \quad \text{and} \quad y_j^1 = x_j^1 \oplus \mathsf{H}(j, q_j \oplus s)$$
8) $\mathbf{P_B}$ outputs $(x_1^{r_1}, ..., x_m^{r_m})$ with $x_j^{r_j} = y_j^{r_j} \oplus \mathsf{H}(j, \mathbf{t}_i)$.

Fig. 7: OTE protocol against honorific adversaries.

For each $j \in [l]$, due to the functionality of protocol, $\mathcal{A}$ can only compute one of $H(q_j)$ or $H(q_j \oplus s)$ depends on its choice bit r_j, and the other one is uniform distributed to $\mathcal{A}$. Since $\mathcal{A}$ does not know $x^{\overline{r_j}}$ either, $y^{\overline{r_j}} \leftarrow x^{\overline{r_j}} \oplus H^{\overline{r_j}}$ is uniform distributed to $\mathcal{A}$ as well, which shows that:

$$\begin{aligned}&\left\{\mathrm{IDEAL}_{\mathcal{C},S^2(z,\{x^{r_j}\}),\mathcal{P}_c}((x,y),n)|\neg\mathsf{bad}_{\mathsf{baseOT}}\right\}\\ &\overset{c}{\equiv} \left\{\mathrm{IDEAL}_{\mathcal{C},S^1(z,\{x_0,x_1\}^m),\mathcal{P}_c}((x,y),n)|\neg\mathsf{bad}_{\mathsf{baseOT}}\right\} \quad (13)\end{aligned}$$

This completes the proof. □

E. Honorific Security — Semi-honest $\mathbf{P_{au}}$ in OTE

The proof for the security against a semi-honest $\mathbf{P_{au}}$ is similar to the proof provided in GC protocol. The simulator $\mathcal{S}$ receives the output $\mathcal{M}^{\mathsf{m\times OT,B}}$ of $\mathcal{A}$. And in CRS model, the simulator S can forge $\mathbf{P_A}$'s signature and thus perfectly emulates an honest or cheated $\mathbf{P_B}$ providing the corresponding evidence to a simulated $\mathbf{P_A}$ and eventually to $\mathcal{A}$.