Non-Malleable Multi-Party Computation

Fuchun Lin*

Abstract. We study a *tamper-tolerant* implementation security notion for general purpose Multi-Party Computation (MPC) protocols, as an analogue of the *leakage-tolerant* notion in the MPC literature. An MPC protocol is tamper-tolerant, or more specifically, *non-malleable* (with respect to a certain type of tampering) if the processing of the protocol under corruption of parties (and tampering of some ideal resource assumed by the protocol) can be simulated by an ideal world adversary who, after the trusted party spit out the output, further decides how the output for honest parties should be tampered with. Intuitively, we relax the correctness of secure computation in a privacy-preserving way, decoupling the two entangled properties that define secure computation. The rationale behind this relaxation is that even the strongest notion of correctness in MPC allows corrupt parties to substitute *wrong* inputs to the trusted party and the output is *incorrect* anyway, maybe the importance of insisting on that the adversary does not further tamper with the *incorrect* output is overrated, at least for some applications. Various weak privacy notions against malicious adversary play an important role in the study of *two-party computation*, where full security is hard to achieve efficiently.

We begin with the honest majority setting, where *efficient* constructions for general purpose MPC protocols with full security are well understood assuming secure point-to-point channels. We then focus on non-malleability with respect to tampered secure point-to-point channels. (1) We show achievability of non-malleable MPC against the *bounded state tampering* adversary in the *joint tampering* model through a naive compiler approach, exploiting a known construction of *interactive non-malleable codes*. The construction is currently not efficient and should be understood as showing feasibility in a rather strong tampering model. (2) We show efficient constructions of non-malleable MPC protocols against weaker variants of bounded state tampering adversary in the *independent tampering* model, where the protocol obtained have the same asymptotic communication complexity as best MPC protocols against honest-but-curious adversary. These are all information-theoretic results and are to be contrasted against impossibility of *secure* MPC when secure point-to-point channels are compromised.

Though general non-malleable MPC in no honest majority setting is beyond the scope of this work, we discuss interesting applications of honest majority non-malleable MPC in the celebrated *MPC-in-the-head* paradigm. Other than an abstract result concerning non-malleability, we also derive, in standard model where there is no tampering, that strong

* Department of Electrical and Electronic Engineering, Imperial College London, UK. E-mail: flin@ic.ac.uk

(ideal/real world) privacy against malicious adversary can be achieved in a conceptually very simple way.

1 Introduction

In secure Multi-Party Computation (MPC), a set of players wish to evaluate a function f on their private inputs without revealing information about their private inputs beyond what is contained in the output. The function f is publicly known to all players and is assumed to be an arithmetic circuit C over some finite field. The task can be trivially accomplished given a *trusted party*: every player gives his/her input to the trusted party and the trusted party does the computation, returns the result. The study of secure MPC is about replacing the trusted party with a protocol that works exactly the same as the trusted party, despite possible active/passive attacks from bounded number of players. Both *correctness* and *privacy* against corruption of parties are formulated as a simulation based notion, in an entangled way, that involves a real world and an ideal world (in ideal world there is a trusted party). The security is formulated as the existence of an efficient simulator that, through oracle call to the trusted party in the ideal world and may *substitute the inputs* of the corrupt parties, simulates a set of views of the corrupt parties that have the same distribution as the views they would get if they were to run the protocol in the real world where the corrupt parties may deviate from the protocol.

MPC was introduced in the work of Yao [Yao82]. Feasibility results in the computational setting on MPC were obtained by Yao [Yao82] and Goldreich et al. [GMW87], where the adversary can corrupt all but one players but is assumed to have bounded computational resources. Our focus in this work is information-theoretic security. Feasibility results in information-theoretic setting were shown, for up to less than $1/3$ corrupt parties assuming secure point-to-point communication channels [BGW88,CCD88], and for up to less than $1/2$ corrupt parties assuming secure point-to-point communication channels and a broadcast channel [RB89,Bea89]. A construction of secure MPC with a slightly weaker correctness notion called *security with abort* for up to less than $1/2$ corrupt parties under the sole assumption of secure point-to-point communication channels was given that has communication complexity asymptotically the same as best MPC protocols for passive adversary [GIP+14].

The above feasibility results were proved in the *theoretic security* (*black-box security*) model, where every cryptography primitive is an abstract object with ideal functionality. In real life, every cryptographic algorithm is ultimately implemented on a physical device that affects, and being affected by, the environment around it. Security models that take this fact into account are called the *implementation security* models. The real-life adversary studied in implementation security models can be divided into two groups: *leakage adversary* (passive attack) and *tampering adversary* (active attack). Implementation security models against leakage attacks for general purpose MPC protocols were first studied in

[BGJ+13,BGJK12]. The adversary considered in [BGJ+13] can corrupt an arbitrary subset of parties and, in addition, can learn arbitrary auxiliary information on the entire states of all honest parties (including their inputs and random coins), in an adaptive manner, throughout the protocol execution. The above standard notion of secure computation is impossible (the adversary can simply leak the private input of an honest party) and a weaker notion called *leakage-tolerant* was shown to be achievable, which guarantees that for any amount of information the real world adversary is able to (adaptively) acquire throughout the protocol, this same amount of auxiliary information is given to the ideal world simulator. In contrast to [BGJ+13], [BGJK12] constructed MPC protocols that achieve standard ideal world security (where no leakage is allowed in the ideal world) against real world adversaries that may leak repeatedly from the secret state of each honest player separately, assuming a one-time leak-free preprocessing phase, and assuming the number of parties is large enough. Intuitively, the one-time leak-free preprocessing phase is exploited by the honest parties to secret share their private inputs (private inputs are erased once the shares are stored) and with the independent leakage assumption, it is possible to prevent the adversary from obtaining information about the private inputs. The recent result on Leakage-Resilient MPC (LR-MPC) [BDIR18] belongs to the latter category.

Formulating meaningful security notions for MPC against tampering is more delicate than against leakage. We draw ideas from the tamper-resilient cryptography literature. *Non-malleability* as a coding goal was proposed by Dziembowski, Pietrzak and Wichs [DPW18], which offers an abstract solution: encoding the sensitive state using a Non-Malleable Code (NMC) and decoding it before the state is needed for an execution. An NMC with respect to a class $\mathcal{F}$ of tampering functions guarantees that for any tampering function $f \in \mathcal{F}$, there is a distribution $\mathcal{D}_f$ solely defined by f such that decoding under the influence of f can be simulated using $\mathcal{D}_f$ and interacting with a trusted party who has the secret state, which in effect turns a possibly harmful tampering f into a harmless one. The phrases "harmful" and "harmless" are best illustrated by the following example that is usually used as a motivating example for NMC. It was shown in [BDL01] that if the state to protect is a secret signing key of a signature scheme used in an implementation of RSA based on the Chinese remainder theorem, and the application of a tampering function to the storage always results in flipping a single bit of the secret signing key, then the RSA modulus can be factored to recover the signing key, through observing an incorrectly generated signature. Here, the influence that turns an unknown value into a *related* unknown value is a harmful influence. On the other hand, turning an unknown value into a different value without knowing the relation between the two is harmless influence. Although special purpose protocols such as non-malleable commitments [AGM+15b,AGM+15a,GPR16] were constructed using NMC, it is not clear whether NMC can be used to secure a general purpose MPC protocol against tampering.

The next object closer to providing a solution is the currently active research area of Non-Malleable Secret Sharing (NM-SS) [GK18] (see Definition 5). Secret sharing, introduced independently by Blakley [Bla79] and Shamir [Sha79], is a major tool in MPC constructions (cf. [CDN15]) and threshold cryptography [Des94]. The goal in secret sharing is to encode a secret s into a number of *shares* $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_\mathsf{n}$ that are distributed among a set $[\mathsf{n}] = \{1, \ldots, \mathsf{n}\}$ of parties such that for a privacy threshold t, any set $\mathcal{R} \subseteq [\mathsf{n}]$ with $|\mathcal{R}| > \mathsf{t}$ can reconstruct the secret, while any set $\mathcal{A} \subset [\mathsf{n}]$ with $|\mathcal{A}| \leq \mathsf{t}$ do not have information about the secret. For special purpose protocols such as threshold signature schemes, compilers based on NM-SS that transform a black-box security threshold signature scheme into one that is resilient against independent tampering of all parties were constructed in [ADN+19,FV19]. Given the ubiquitous presence of secret sharing in the construction of general purpose MPC protocols, one would expect a more prominent role of NM-SS in securing MPC protocols against tampering. One bottle neck here is that NM-SS was proposed as the opposite of secret sharing with homomorphic properties, which are exactly the kind of secret sharing used in MPC protocols.

The object closest to providing a general solution is the following. Similar to generalising block codes to interactive codes, NMC was recently generalised to Interactive NMC (INMC) [FGJ+19]. It is shown that the interactive setting allows INMC to be constructed in many powerful tampering models that would have been impossible in non-interactive setting. For example, they considered a Bounded State Tampering (BST) model, where an adversary can keep a state that stores information about past messages and use it in tampering with the current message (the current message is given to the adversary in full) and showed that a rather strong security called *protocol non-malleability* is achievable. However, the two parties executing the protocol are both honest and the tampering is in an outsider model. Encoding a secure two-party computation protocol using an INMC will not provide any protection, as the adversary of two-party computation is executing the protocol as an insider. We will return to this in **Related works.**

Our contributions. We propose a *tamper-tolerant* notion for general purpose MPC protocols, as an analogue of the leakage-tolerant notion studied in [BGJ+13] in the sense that we allow the ideal world adversary to tamper with the output of the computation, to be compared with that the leakage ideal world adversary in [BGJ+13] gets to leak same amount of information as the real world adversary. On the other hand, in order to define a useful tamper-tolerant notion, we insist that the ideal world adversary should only be allowed to *harmlessly* tamper with the output of the computation, in the sense of [DPW18].

Our motivating example. We continue with the signature scheme motivating example for NMC and take it to a setting of *threshold signature scheme*, where the secret signing key remains in the distributed form held by n servers all through the signing process. Moreover, consider a natural situation where the secret signing key itself is to be generated on the spot, from private values that are held by honest but mutually distrusting clients (for instance, computing a

session key from global private keys). A standard solution that kills two birds with one stone is to have the servers and the clients run an MPC protocol in the so-called *client-server model*, where the clients provide private inputs and the servers compute. If the MPC protocol employed is based on a secret sharing scheme, the servers naturally have the distributed form of the secret signing key upon the completion of circuit evaluation as each server's *last message* to be delivered for output reconstruction [1]. The robustness of MPC protocol prevents a black-box adversary that corrupts bounded number of servers (servers do not have input hence can not do input substitution) from changing the value of the being computed secret signing key. But if a real life adversary exploits flaws in the implementation of the MPC protocol and is capable of inflicting a bit-flip in the computed signing key, the same attack of [BDL01] can be used to factor the underlying RSA modulus. The application scenario motivates the study of a *non-malleable MPC*, where tampered protocol should not compute an output related to private inputs of honest parties.

In order to achieve the non-malleability described in the motivating example, we must adapt at least some minimum restrictions, as otherwise, a tampering adversary can trivially make the output depend on the private inputs of honest parties if we were to allow the adversary access to everything following [BGJ$^+$13]. But instead of assuming a harm-free phase as done in [BGJK12] (seems too much to ask in real life), we begin with MPC protocols in honest majority setting and go for an *adversarial channel tampering* model (cf. [FGJ$^+$19]), where the adversary can tamper with all the messages traveling between parties, but has to follow certain patterns that define a structured type of tampering. We explicitly model the execution of such an MPC protocol under corruption and a type $\mathcal{F}$ of adversarial channel tampering (see the beginning of Section 3).

Definition[NM-MPC] (informal summary of Definition 6) An MPC protocol evaluating a circuit is non-malleable if for any active adversary corrupting a bounded number of parties (and any choice of a tampering function from tampering class $\mathcal{F}$), there exists a simulator that takes the adversary's corruption strategy (and tampering function) as inputs and simulate in the ideal world, where there is an incorruptible trusted party evaluating the circuit, the real execution of the protocol under corruption (and tampering): the simulator is

[1] The client-server model with the output of computation remains in secret-shared form is common practice and in particular useful (see [MR18] and references therein). Machine learning is widely used to produce models that classify images, authenticate biometric information, recommend products, choose which Ads to show, and identify fraudulent transactions, etc. Internet companies regularly collect users' online activities and browsing behaviour to train more accurate recommender systems. The healthcare sector envisions a future where patients' clinical and genomic data can be used to produce new diagnostic models. There are efforts to share security incidents and threat data, to improve future attack prediction. In all above application scenarios, the data being classified or used for training is often sensitive and may come from multiple sources with different privacy requirements. MPC is an important tool for supporting machine learning on private data held by mutually distrusting entities and the end product of data training is stored in distributed form for privacy.

allowed to modify the incorruptible trusted party's output for honest parties, if the real world adversary deviates from the protocol.

Non-malleability as defined above is a relaxed notion of security for MPC protocols which suggests an interesting way of decoupling the two entangled properties (correctness and privacy) that define secure computation. Intuitively, we no longer insist on correctness (we allow the ideal world adversary to tamper with the output if the real world adversary deviates from the protocol), but preserve the privacy notion in its strongest possible form (the view of the adversary is simulated without information beyond the corrupted parties' input and what is implied in an output). The rationale behind this relaxation is that, since the correctness of secure computation in its strongest possible form still allows malicious adversary to substitute wrong inputs on behalf of the corrupted parties leading to an unintended output, the importance of insisting on preventing the adversary from further tampering with the unintended output may be overrated, at least in some applications. For the two example NM-MPC constructions, we in fact achieve a stronger security that allows the modified output for honest parties to be either remains the same as the incorruptible trusted party has outputted it or following a distribution determined by the adversary (the probability of remaining the same and the fixed distribution are both independent of the incorruptible trusted party's output). This makes them sufficient for the application in *Our motivating example*.

We begin with showing that with this relaxation of security (privacy against malicious adversary), information-theoretic honest majority MPC protocols can be constructed even when their secure point-to-point channels are tampered with. *Bounded State Tampering* is a well-established adversarial tampering model since [CM97]. We follow the recent formulation in [FGJ+19]. The adversary keeps a state of bounded size (at most s bits) storing information about past messages that can be used in the tampering of the current message. For a 2-round protocol, $f = (f_1, f_2) \in \mathcal{F}^s_{\mathsf{BST}}$ can produce $\tilde{m}_1 = g_1(m_1)$ and $\tilde{m}_2 = f_2(m_1, m_2) = g_2(m_2, h_1(m_1))$ depending on $h_1(m_1)$ and m_2, where the range of h_1 is $\{0,1\}^s$. Our first result of feasibility nature is that once the amount of information of past messages the adversary is allowed to "memorise" is limited: $\mathcal{F} = \mathcal{F}^s_{\mathsf{BST}}$, NM-MPC can be constructed even in the *joint-tampering* model, where an adversary can jointly tamper with messages intended for different receivers. We emphasize that the adversary has full control over the current messages and there is no information-theoretic approach to secure the channel against such adversary (not even to detect its presence).

Theorem (informal summary of Theorem 1) *For channel tampering class $\mathcal{F}$ over which a pair of independent keys can be generated by two communicants, there is a compiler that turns a secure MPC protocol into an NM-MPC protocol in the joint tampering model and the communication complexity increases by $(3/\rho + 1)$ times, where ρ is the rate in which such independent keys can be generated.*

We will discuss the notion of *independent keys* further in **technical overview**. Substituting in a known construction for generating such independent keys against $\mathcal{F}^s_{\mathsf{BST}}$-tampering [FGJ+19], we obtain NM-MPC with respect to cor-

ruption of parties and $\mathcal{F}^s_{\mathsf{BST}}$-tampering of secure channels in the joint tampering model. In particular, the level of non-malleability achieved in this construction is only marginally weaker than the secure with abort notion in [GIP+14] (see Definition 1), where the adversary is allowed to individually decide, after learning its own outputs, whether each honest party receives its correct output from the functionality or a special $\perp$ message which the party outputs. The level of non-malleability achieved above further relaxes secure with abort notion and allows each honest party to receive its correct output with a probability p and output $\perp$ with probability $1-\mathsf{p}$ ("secure with probabilistic abort"). On the other hand, the shortcoming of the above approach is the excessive cost (the generated key is of length a close to zero fraction of total communication). The known construction for generating the independent keys against $\mathcal{F}^s_{\mathsf{BST}}$-tampering is based on a *split-state non-malleable extractor* [CG17], whose efficient construction is a notoriously hard bottle-neck problem in NMC and NM-SS literature [Li18].

Motivated by finding more efficient approaches that circumvent NM-KE, we consider relying on the secret sharing scheme implementing the protocol. Without generating a long key to mask the transmitted messages, we can not hope to defeat a joint tampering adversary. We fall back to *independent tampering* model and also consider a weakened bounded state tampering $\mathcal{F}^s_{\mathsf{weakBST}}$, where the state size bound is effective for the current message and all past messages. For a 2-round protocol, $f = (f_1, f_2) \in \mathcal{F}^s_{\mathsf{weakBST}}$ can produce $\tilde{m}_1 = g_1(h_1(m_1))$ depending on $h_1(m_1)$ only and $\tilde{m}_2 = f_2(m_1, m_2) = g_2(h_2(m_2, h_1(m_1)))$ depending on $h_2(m_2, h_1(m_1))$ only, where the range of h_1, h_2 is $\{0,1\}^s$. Though much weaker than $\mathcal{F}^s_{\mathsf{BST}}$-tampering, here $\mathcal{F}^s_{\mathsf{weakBST}}$-tampering still allows the adversary to selectively overwrite the whole current message (again, impossible to detect adversary's presence).

Theorem (informal summary of Theorem 2) *For constant integers s, θ, and big enough prime number p, there exist a MPC protocol that non-malleably computes an arithmetic circuit over $\mathbb{F}_p$ against an active adversary corrupting at most θ servers, and $\mathcal{F}^s_{\mathsf{weakBST}}$-tampering of the secure channel in the independent model. The protocol has the same asymptotic communication complexity as best passive security MPC.*

It is fair to say that the combination of $\mathcal{F}^s_{\mathsf{weakBST}}$-tampering and the independent tampering model defines a rather contrived adversary that is difficult to find applications in real life. The more interesting message carried in this construction is the fact that weaker notion of security (privacy against active adversary) for MPC is efficiently achievable without assuming secure point-to-point channels.

Using the technical framework initiated in [BDIR18], we are able to construct NM-SS that provides flexible choice of parameters ranging from two extremes (parameters of NM-SS directly translate into those of NM-MPC). One extreme is the maximum state case, where for every share the adversary stores all information except one bit (essentially the adversary can arbitrarily tamper with the share, independently). It is still possible to obtain NM-SS as the non-malleability error vanishes exponentially fast in the number $\mathsf{t} - \theta$ of uncorrupted shares in a reconstruction set of size $\mathsf{t} + 1$. The other extreme is the minimum state case,

where for every share the adversary stores one bit information. In this case, for a 10 bits prime p ($\log p = 10$), choosing $\mathsf{n} = p - 1$, $\mathsf{t} = 300$ (approximately $\mathsf{n}/3$) allows for $\varepsilon = 2^{-50}$ against up to $\theta = 175$ fully tampered shares.

Allowing non-explicit Monte-Carlo constructions and further restricting to a yet smaller class of tampering functions (the information to store in the state is through reading physical bits only, called *physical-bit* s-BST), we are able to use the more recent results of [MPSW21,MNP+21] to achieve another dimension of extreme parameters: the reconstruction threshold can be set as low as $\mathsf{t}+1 = 2$ (privacy threshold $\mathsf{t} = 1$) , and considering varying number of parties $\mathsf{n} = 10, 100,$ and 1000, the non-malleability error as small as 2^{-50} against physical-bit 1-bounded state tampering can be achieved with success probability $1 - 2^{-50}$ (over choosing the evaluation places), using a prime number p with more than $\lambda = 430, 4800,$ and 62000 bits, respectively.

As the applications of NM-MPC in honest majority setting, we discuss NM-MPC in no honest majority setting making use of the *MPC-in-the-head* paradigm [IPS08,IPS09]. One reason for this choice is that these protocols are described in the Oblivious Transfer (OT)-hybrid model, where an ideal OT functionality is assumed that the protocol can make repeated calls as an incorruptible trusted party. Information-theoretic secure MPC protocols in no honest majority setting can be efficiently constructed in OT-hybrid model similar to the honest majority counterparts efficiently constructed assuming secure point-to-point channels. Another reason is that the MPC-in-the-head framework gives us a quick way to transfer the findings in one setting to another. The MPC-in-the-head framework uses an inner semi-honestly secure two-party protocol π^{OT} to emulate the execution of an outer MPC protocol Π in honest majority setting among a large set of virtual servers computing the given circuit.

Theorem 4 Let $\mathcal{T}$ be a tampering class for OT functionality. Let $\mathcal{F}_{\mathcal{T}}$ be the tampering class for the communication channels between virtual servers induced by executing π^{OT} under $\mathcal{T}$-tampered OT. Let Π be an NM-MPC with respect to $\mathcal{F}_{\mathcal{T}}$. There is a compiler for Π and semi-honest π^{OT} that gives a general purpose non-malleable two-party computation protocol with respect to $\mathcal{T}$-tampering of OT.

We do not consider explicit tampering models for ideal OT functionality in this work. When there is no tampering, non-malleability becomes privacy against malicious adversary. The above result suggests a conceptually very simple way to obtain real/ideal world privacy against malicious adversary (the strongest form of privacy) for general purpose two-party protocols: construct an NM-MPC protocol Π with respect to $\mathcal{F}_{\emptyset}$ and use semi-honestly secure two-party protocol π^{OT} to emulate it.

Related works and open questions.

The leakage-tolerant notion of [BGJ+13] relaxes privacy of secure computation in a correctness-preserving way. This notion allows clean modeling of the leakage and clean quantitive formulation: the amount of leakage given to the ideal world adversary is the same as the amount allowed in the real world.

The tampering counterpart is more difficult to capture both conceptually and quantitively. We propose a tamper-tolerant notion motivated by relaxing the correctness of secure computation in a privacy-preserving way. Conceptually, we draw inspirations from the non-malleability notion in tamper-resilient cryptography and merge it with the simulation based formulation in MPC literature. This generalizes the idea of harmless tampering, which is a tampering defined independent of any information that would breach the privacy, to secure computation. Quantitively, the "amount" of tampering allowed in the real world does not translate into the "amount" of tampering in the ideal world (as a symbol-by-symbol analogy to [BGJ+13] would suggest), but into the amount of overhead (e.g. communication complexity) required for turning the harmful real world tampering into harmless ideal world tampering.

The secure with abort MPC protocols in [GIP+14] represent the maximum corruption (up to 1/2), minimum assumptions (secure point-to-point channels only) and highest efficiency (asymptotically same as best passive security MPC) in the honest majority setting. The high efficiency is the result of an unusual two-step approach to active security through first constructing an intermediate protocol that computes an *additively corruptible functionality/circuit* (in fact, most of the celebrated passive security MPC constructions suffice), and then apply the intermediate protocol to compute an encoded version of the functionality/circuit, instead of the functionality/circuit itself. Intuitively, the first step, through protocol design and the secure channels assumption, reduces a full-fledged malicious adversary to an *additive adversary*, who is then efficiently (negligible overhead) defeated in the second step through a novel circuit encoding technique and conventional input encoding against additive attack. We study a weakening of security (privacy against malicious adversary) for MPC protocols that is defined by allowing the output of the incorruptible functionality/circuit to be corruptible and mainly show constructions that remove the secure point-to-point channel assumption. To put this new notion of security in the right context, we note here that malicious privacy in honest majority setting may not be easy to achieve (much harder than semi-honest privacy), even assuming secure point-to-point channels. The first step of the above construction does not guarantee malicious privacy, since the functionality itself is corruptible, not only the output of the functionality. Even after the circuit encoding (before input encoding) in the second step above, the input to the functionality is still corruptible, rendering it not maliciously private. Achieving malicious privacy using the above construction takes the same amount of efforts as achieving security with abort. Finally, our conceptually very simple approach to privacy against malicious adversary using MPC-in-the-head framework can be interpreted as an example of circuit encoding such that running the encoded circuit using semi-honestly secure two-party protocol yields privacy against malicious adversary.

The study of *Interactive Non-Malleable Codes (INMC)* [FGJ+19] considers encoding of two-party protocols for achieving a strong non-malleability notion called *protocol non-malleable* against an outsider tampering adversary (an instance of outsider adversarial tampering is defined by a set of restrictions that

distinguish it from the insider tampering where one party is corrupted and fully controlled by adversary). In [FGJ+19], three adversarial channel tampering classes: bounded state tampering, *unbalanced split-state* tampering and *fragmented sliding window* tampering were studied. The descriptions of the latter two tampering models depend on the round number of the protocols to be encoded, which makes them not suitable for MPC (the round number of a general purpose MPC protocol may depend on the depth of the circuit to be computed). We first show that INMC against bounded state tampering can be used to construct general purpose NM-MPC in the honest majority setting (at least three parties). We then show that NM-MPC in the honest majority setting can be used to construct a two-party protocol through the MPC-in-the-head paradigm. Combining these two results, we have a *correct* way of obtaining non-malleability against an *insider* adversary in two-party setting (as opposed to the naive direct encoding approach).

This preliminary study of NM-MPC leaves behind many interesting open questions. Many cryptographic primitives (e.g. non-malleable commitments) have been studied in non-synchronizing man-in-the-middle setting, we leave non-malleability of MPC protocols against richer tampering models as future works.

The NM-MPC we define is corresponding to static (v.s. adaptive) adversary MPC, where the adversary decides on a fixed set of parties to corrupt and stick to the set all through the execution. We do include partial adaptiveness in defining the tampering, for instance, the joint tampering model and the BST functions. The more challenging fully adaptive NM-MPC is open. The non-malleability security discussed in this work is a *stand-alone* (v.s. universally composible) notion, where only a single protocol session runs in isolation. Such stand-alone notion provides only limited guarantees regarding the security of systems that involve the execution of two or more protocols. Although the absence of correctness guarantee may limit its applications (the motivating example describes an application in composition of protocols), universally composible NM-MPC is open.

Lastly and most importantly, a systematic study of NM-MPC in no honest majority setting is the most interesting open area to explore. Given the importance of weak security notions in this setting in the literature, one would most likely find non-trivial applications of NM-MPC here.

Technical overview. The ideal functionality of secure point-to-point channel assumed in the study of honest majority secure MPC is implemented either using public key cryptography [DH76] under computational hardness assumptions or using imperfect correlated randomness [Mau92,Mau93] obtained from some assumed resource, by having the two communicants first agree on a shared key private from outsiders and then mask their outgoing messages with a One-Time-Pad (OTP) followed by a Message Authentication Code (MAC) tag to establish an abstract secure channel. Through exchanging independent uniform messages using the given imperfect channel controlled by a BST adversary, one obtains correlated randomness in the latter model. An important observation here is these correlated randomness are possibly not correlated at all, since any BST

adversary has full writing capability (in the sense of completely overwriting the messages) and in effect cutting off the communication. To make it worse, there is no public discussion channel usually assumed in the later model [Mau92,Mau93] for *information reconciliation* that results in a shared imperfect secret (correctness) and *privacy amplification* that generates a secret key (privacy) using a *randomness extractor*. Intuitively, the channel controlled by a BST adversary is too weak a resource for establishing secure communication. A novel idea of generating *independent* keys that are sufficient for establishing non-malleable communication was proposed in [FGJ+19] and such a protocol was termed *Non-Malleable Key Exchange (NM-KE)* (see Definition 9 for an exact definition). It was shown that NM-KE can be constructed by having both communicants applying a 2-*split-state non-malleable extractor* [CG17] to their correlated randomness mentioned above. Intuitively, independent keys does not need the information reconciliation (correctness) and can be generated from the privacy amplification (non-malleability is malicious privacy).

The remaining technicalities for this idea to work in MPC setting (especially in the joint tampering model) are carefully analyzing the type of information available for a BST adversary at the tampering of a current message. Note that each pair of parties are to independently run an NM-KE protocol and observe that the messages exchanged are fresh uniform messages independent from previous messages within one protocol and across multiple protocols.

In the independent tampering model, we observe that a *Non-Malleable Secret Sharing (NM-SS)* (see Definition 5) with n shares has, in particular, an implicit n-*split-state non-malleable extractor* [CG17] in its reconstruction function, which guarantees that (roughly speaking) for any n independently chosen tampering functions applied to the n shares, reconstructing from the tampered shares is independent from reconstructing from the n clean shares. This suggests that one could skip the independent keys generation step (and the corresponding OTP followed by MAC) and simply rely on the reconstruction function of NM-SS to provide non-malleability. Unfortunately, it is trivial to show that such NM-SS implies that its reconstruction function is not a linear function (define a tampering that adds a constant share vector for the secret 1, share by share independently, the linear reconstruction function returns a related secret). But for a secret sharing scheme to be useful in constructing general purpose MPC protocols, linearity is the least property required to enable privacy-preserving evaluation of secret values. For the efficient protocols in the honest majority setting, a stronger *multiplication* property is required for the underlying secret sharing scheme, which allows the reconstruction of the product of two secrets from the share-wise multiplication of their share vectors (effectively requires that the reconstruction of each secret should take less than $\mathsf{n}/2$ shares). These make relying on NM-SS for constructing general purpose NM-MPC protocols highly technical, though efficiency-wise highly attractive.

The obstacles discussed above are the main reasons that the NM-MPC protocols constructed in this approach can only tolerate a rather limited BST variant. Firstly, to overcome the impossibility of NM-SS with linear reconstruction func-

tion, we restrict the type of functions the adversary is allowed to tamper with each share, resulting in the $\mathcal{F}^s_{\mathsf{weakBST}}$-tampering class. Secondly, we are able to show that without the need to further restrict the tampering adversary, there are linear NM-SS against the above restricted share tampering that at the same time has (strong) multiplication property.

2 Preliminary

The *statistical distance* of two random variables (their corresponding distributions) is defined as follows. For $\mathsf{X}, \mathsf{Y} \leftarrow \Omega$,

$$\mathsf{SD}(\mathsf{X};\mathsf{Y}) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr(\mathsf{X}=\omega) - \Pr(\mathsf{Y}=\omega)|.$$

We say X and Y are ε-close (denoted $\mathsf{X} \overset{\varepsilon}{\sim} \mathsf{Y}$) if $\mathsf{SD}(\mathsf{X},\mathsf{Y}) \leq \varepsilon$.

A secure computation task is defined by a function specifying the desired mapping from the inputs to the final output. We consider arithmetic circuits over some finite field and will identify a circuit C with the functionality it computes.

Formalizing the real world computation. We use the so-called *client-server model* refinement of MPC protocols in this work. The inputs are provided by the *Clients* $\{C_1,\ldots,C_{\mathsf{m}}\}$, each client C_i holds an input x_i, and the computation is processed by the *Servers* $\{S_1,\ldots,S_{\mathsf{n}}\}$, who do not have inputs. An n-party m-client protocol Π over a finite field $\mathbb{F}$ proceeds in rounds where in each round j in circuit evaluation phase, the protocol's description contains n *next message functions* $\mathsf{nextMSG}_i^j$ for $i = 1,\ldots,\mathsf{n}$ that can be represented as arithmetic circuits over $\mathbb{F}$. The next message function $\mathsf{nextMSG}_i^j$ of server S_i for the j-th round gets as input all the messages that S_i received until the j-th round and S_i's local randomness, and outputs S_i's messages in the j-th round. The view of S_i during an execution of a protocol Π on inputs $\mathbf{x}$, denoted by $\mathsf{view}_i^{\Pi}(\mathbf{x})$, contains the random input $\mathbf{R}_i$ and all the messages received from the clients and other servers. For every $\mathcal{S} = \{i_1,\ldots,i_{\mathsf{t}}\} \subset [\mathsf{n}]$, we denote by $\mathsf{view}_{\mathcal{S}}^{\Pi}(\mathbf{x}) = (\mathsf{view}_{i_1}^{\Pi}(\mathbf{x})),\ldots,(\mathsf{view}_{i_{\mathsf{t}}}^{\Pi}(\mathbf{x}))$. The input sharing phase and output reconstruction phase happen before and after the circuit evaluation phase, respectively. For every $\mathcal{C} \subset [\mathsf{m}]$, we denote the view of clients in $\mathcal{C}$ by $\mathsf{view}_{\mathcal{C}}^{\Pi}(\mathbf{x})$. Let $\mathsf{out}_{\mathcal{C}}^{\Pi}(\mathbf{x})$ denote the output of the clients in $\mathcal{C}$. The real world adversary Adv corrupts a set $\mathcal{S}$ of servers and a set $\mathcal{C}$ of clients, which means the adversary act on the corrupted parties' behalf in the protocol. The execution of Π in the presence of Adv who corrupts $\mathcal{S}\cup\mathcal{C}$ is characterized by a random variable

$$\mathsf{Real}_{\Pi,\mathsf{Adv},\mathcal{S}\cup\mathcal{C}}(\mathbf{x}) = \left(\mathsf{view}_{\mathcal{S}\cup\mathcal{C}}^{\Pi,\mathsf{Adv}}(\mathbf{x}), \mathsf{out}_{\overline{\mathcal{C}}}^{\Pi,\mathsf{Adv}}(\mathbf{x})\right),$$

where the superscript $^{,\mathsf{Adv}}$ highlights the presence of Adv. The first component is adversary's view, which can be divided into the truncated views of $\mathcal{S}\cup\mathcal{C}$ and the last communication round messages from honest servers to corrupted clients: $\mathsf{view}_{\mathcal{S}\cup\mathcal{C}}^{\Pi,\mathsf{Adv}}(\mathbf{x}) = \left(\mathsf{truncview}_{\mathcal{S}\cup\mathcal{C}}^{\Pi,\mathsf{Adv}}(\mathbf{x}), \mathsf{lastmview}_{\overline{\mathcal{S}}\to\mathcal{C}}^{\Pi,\mathsf{Adv}}(\mathbf{x})\right)$. The second component is

the output of honest clients, appending which is crucial for a unified formation of privacy and correctness.

Formalizing the ideal world computation. There is an incorruptible trusted party in the ideal world, who evaluates the circuit C on m inputs provided by the clients and provides the outputs. The ideal world adversary Sim corrupts a set $\mathcal{S} \cup \mathcal{C}$ of parties, which means that Sim gets to substitute the inputs of the corrupted clients in $\mathcal{C}$ before they are given to the trusted party, and simulate views for $\mathcal{S} \cup \mathcal{C}$. The ideal world computation is characterized by a random variable

$$\mathsf{Ideal}^{\mathsf{abort}}_{\mathsf{C},\mathsf{Sim},\mathcal{S}\cup\mathcal{C}}(\mathbf{x}) = \left(\mathsf{view}^{\mathsf{C},\mathsf{Sim}}_{\mathcal{S}\cup\mathcal{C}}(\mathbf{x}), \mathsf{out}^{\mathsf{C},\mathsf{Sim}}_{\overline{\mathcal{C}}}(\mathbf{x})\right),$$

where the superscript $^{\mathsf{abort}}$ indicates that here Sim is allowed to individually decide, after learning its own outputs, whether each honest party receives its correct output from the functionality or a special $\perp$ message which the party outputs.

Comparing real/ideal world. The combination of privacy and correctness are captured by requiring that given any real world adversary Adv corrupting $\mathcal{S} \cup \mathcal{C}$, there exist an ideal world Sim that (also corrupts $\mathcal{S} \cup \mathcal{C}$) simulates Adv in the sense that the two random variables $\mathsf{Real}_{\Pi,\mathsf{Adv},\mathcal{S}\cup\mathcal{C}}(\mathbf{x})$ and $\mathsf{Ideal}^{\mathsf{abort}}_{\mathsf{C},\mathsf{Sim},\mathcal{S}\cup\mathcal{C}}(\mathbf{x})$ are indistinguishable.

Definition 1. *Let C be a m-input functionality and let Π be an m-client n-server protocol. We say that Π (t, ϵ)-*securely computes *C if for every probabilistic adversary Adv in the real world controlling a set $\mathcal{S} \subset [\mathsf{n}]$ of servers such that $|\mathcal{S}| \leq \mathsf{t}$ and a set $\mathcal{C} \subset [\mathsf{m}]$, there exists a probabilistic simulator Sim in the ideal world such that for every input $\mathbf{x}$, it holds that*

$$\mathsf{SD}\left(\mathsf{Real}_{\Pi,\mathsf{Adv},\mathcal{S}\cup\mathcal{C}}(\mathbf{x}); \mathsf{Ideal}^{\mathsf{abort}}_{\mathsf{C},\mathsf{Sim},\mathcal{S}\cup\mathcal{C}}(\mathbf{x})\right) \leq \epsilon.$$

The security defined above is sometimes referred to as security with "selective abort", which does not require honest parties to simultaneously abort. In the stronger notion of "unanimous abort", the ideal world adversary needs to decide whether all honest parties receive their correct output or all of them abort. The advantage of using the weaker variant is that it supports feasibility results which only use secure point-to-point channels.

An encoding Π' of an interactive protocol Π between Alice and Bob is defined by two simulators S^A, S^B with black-box access to *stateful oracles* encapsulating the next-message functions of Alice and Bob, respectively. Let $\mathsf{Trans}^{\Pi}(x, y)$ denote the function mapping inputs x, y to the transcript of an honest execution of Π between Alice holding input x and Bob holding input y. The protocol $\Pi' = (S^A, S^B)$ is an ε-correct encoding of protocol Π if for all inputs $x \in \mathcal{X}$, $y \in \mathcal{Y}$, protocol Π' ε-correctly evaluate the functionality $(\mathsf{Trans}^{\Pi}(x, y), \mathsf{Trans}^{\Pi}(x, y))$.

Definition 2 ([FGJ+19]). *An encoding $\Pi' = (S^A, S^B)$, of an interactive protocol Π between Alice and Bob is ε-protocol-non-malleable for a family $\mathcal{F}$ of tampering functions if the following holds: For each tampering function $f \in \mathcal{F}$,*

there exists a distribution $\mathcal{D}_f$ over $\{\perp, \mathsf{same}\}^2$ such that for all x, y, the product distribution of $S^A(A, x)$'s and $S^B(B, y)$'s outputs is ε-close to the distribution $\mathsf{Patch}(\mathcal{D}_f, \mathsf{Trans}(x, y))$.

Definition 3. *A pair of algorithms* $(\mathsf{Share}^{\mathsf{n},\mathsf{t}}, \mathsf{Recover}^{\mathsf{n},\mathsf{t}})$*, where* $\mathsf{Share}^{\mathsf{n},\mathsf{t}} \colon \mathbb{F} \to \mathbb{F}^{\mathsf{n}}$ *is randomised, mapping a secret to* n *shares, and* $\mathsf{Recover}^{\mathsf{n},\mathsf{t}}$ *is deterministic, mapping a reconstruction set* $\mathcal{R} \subset [\mathsf{n}]$ *and the corresponding shares to a secret, are said to be a secret sharing scheme if the following conditions hold for every* $\mathsf{n}, \mathsf{t} \in \mathbb{N}$.

- *Reconstruction. For any set* $\mathcal{R} \subset [\mathsf{n}]$ *such that* $|\mathcal{R}| > \mathsf{t}$ *and for any* $\mathsf{s} \in \mathbb{F}$ *it holds that*
$$\mathsf{Pr}[\mathsf{Recover}^{\mathsf{n},\mathsf{t}}(\mathsf{Share}^{\mathsf{n},\mathsf{t}}(\mathsf{s})_{\mathcal{R}}, \mathcal{R}) = \mathsf{s}] = 1,$$
where the subscript $\mathcal{R}$ *denotes the projection of a vector to the components in* $\mathcal{R}$.
- *Privacy. For any set* $\mathcal{A} \subset [\mathsf{n}]$ *such that* $|\mathcal{A}| \leq \mathsf{t}$ *and for any* $\mathsf{s}, \mathsf{s}' \in \mathbb{F}$*, it holds that*
$$\mathsf{Share}^{\mathsf{n},\mathsf{t}}(\mathsf{s})_{\mathcal{A}} \overset{0}{\sim} \mathsf{Share}^{\mathsf{n},\mathsf{t}}(\mathsf{s}')_{\mathcal{A}}.$$

When the parameters are clear from the context, we simply write $(\mathsf{Share}, \mathsf{Recover})$.

Linear secret sharing schemes are closely related to linear codes.

Definition 4. *A subset* $\mathcal{C} \subset \mathbb{F}^{\mathsf{n}}$ *is an* $[\mathsf{n}, \mathsf{k}, d]$*-linear code over finite field* $\mathbb{F}$ *if* $\mathcal{C}$ *is a subspace of* $\mathbb{F}^n$ *of dimension* k *such that: for all* $\mathbf{c} \in \mathcal{C} \backslash \{\mathbf{0}\}$*, the Hamming weight* $w_H(\mathbf{c}) > d$ *(i.e., the minimum Hamming distance between two elements of the code is at least* d*). A code is called Maximum Distance Separable (MDS) if* $\mathsf{n} - \mathsf{k} + 1 = d$.

For $\mathcal{R} \subset [\mathsf{n}]$, let $\mathcal{C}_{\mathcal{R}} \subset \mathbb{F}^{|\mathcal{R}|}$ be the projection of the code $\mathcal{C}$ on $\mathcal{R}$:

$$\mathcal{C}_{\mathcal{R}} \colon = \{\mathbf{c}_{\mathcal{R}} | \mathbf{c} \in \mathcal{C}\}.$$

It can be shown that if $\mathcal{C}$ is an $[\mathsf{n}, \mathsf{k}, \mathsf{n} - \mathsf{k} + 1]$-linear MDS code, then for any $\mathcal{R} \subset [\mathsf{n}]$ with $|\mathcal{R}| \geq \mathsf{k}$, we always have that $\mathcal{C}_{\mathcal{R}}$ is $[|\mathcal{R}|, \mathsf{k}, |\mathcal{R}| - \mathsf{k} + 1]$-linear MDS code.

An $[\mathsf{n}, \mathsf{k}, d]$-linear code $\mathcal{C}$ over finite field $\mathbb{F}$ can be represented by its *generator matrix* $G \in \mathbb{F}^{\mathsf{k} \times \mathsf{n}}$, whose rows form a basis of $\mathcal{C}$. Let $\mathbf{R} \leftarrow \mathbb{F}^{\mathsf{t}}$ be a uniform t-tuple. The sharing algorithm of Shamir's secret sharing scheme can be described as follows.

$$\mathsf{Share}(\mathsf{s}) = (\mathsf{s}, \mathbf{R}) \begin{bmatrix} 1 & 1 & \dots & 1 \\ a_1 & a_2 & \dots & a_{\mathsf{n}} \\ \vdots & \vdots & \dots & \vdots \\ a_1^{\mathsf{t}} & a_2^{\mathsf{t}} & \dots & a_{\mathsf{n}}^{\mathsf{t}} \end{bmatrix},$$

where $a_1, \dots, a_{\mathsf{n}}$ are distinct non-zero elements in $\mathbb{F}$. It can be seen that the support of the random variable $\mathsf{Share}(0)$ is an $[\mathsf{n}, \mathsf{t}, \mathsf{n} - \mathsf{t} + 1]$-linear MDS code.

Definition 5 ([GK18]). *A secret sharing scheme* $(\mathsf{Share}, \mathsf{Recover})$ *with* n *shares and privacy threshold* $\mathtt{t}$ *is non-malleable with respect to a class* $\mathcal{F}$ *of tampering functions (*$\mathcal{F}$*-NM for short) if for any secret* $\mathsf{s} \in \mathcal{S}$*, any* $f \in \mathcal{F}$ *and any* $\mathcal{R} \subset [\mathsf{n}]$ *of size* $|\mathcal{R}| = \mathtt{t} + 1$*, there is a distribution* $\mathcal{D}_{f,\mathcal{R}}$ *over the set* $\mathcal{S} \cup \{\perp\} \cup \{\mathsf{same}^*\}$ *determined solely by* f *and* $\mathcal{R}$*, such that the following real tampering experiment* $\mathrm{Tamper}_{\mathsf{s}}^{f,\mathcal{R}}$ *and the simulation* $\mathsf{Patch}(\mathsf{s}, \mathcal{D}_{f,\mathcal{R}})$ *using* $\mathcal{D}_{f,\mathcal{R}}$ *are distinguishable with a negligible advantage* ε*.*

$$\mathrm{Tamper}_{\mathsf{s}}^{f,\mathcal{R}} \overset{\varepsilon}{\sim} \mathsf{Patch}(\mathsf{s}, \mathcal{D}_{f,\mathcal{R}}),$$

where the real tampering experiment $\mathrm{Tamper}_{\mathsf{s}}^{f,\mathcal{R}}$ *and the simulation* $\mathsf{Patch}(\mathsf{s}, \mathcal{D}_{f,\mathcal{R}})$ *are defined as follows.*

- *The real tampering experiment is a random variable with randomness from the randomised sharing algorithm* Share*:*

$$\mathrm{Tamper}_{\mathsf{s}}^{f,\mathcal{R}} = \left\{ \begin{array}{c} \mathbf{v} \leftarrow \mathsf{Share}(\mathsf{s}) \\ \tilde{\mathbf{v}} = f(\mathbf{v}) \\ \tilde{\mathsf{s}} = \mathsf{Recover}(\tilde{\mathbf{v}}_{\mathcal{R}}, \mathcal{R}) \\ \text{Output } \tilde{\mathsf{s}}. \end{array} \right\}.$$

- *The simulation is a random variable defined from the distribution* $\mathcal{D}_{f,\mathcal{R}}$*:*

$$\mathsf{Patch}(\mathsf{s}, \mathcal{D}_{f,\mathcal{R}}) = \left\{ \begin{array}{c} \tilde{\mathsf{s}} \leftarrow \mathcal{D}_{f,\mathcal{R}} \\ \left\{ \begin{array}{l} \textit{output } \mathsf{s}, \textit{ if } \tilde{\mathsf{s}} = \mathsf{same}^*; \\ \textit{output } \tilde{\mathsf{s}}, \textit{ otherwise.} \end{array} \right. \end{array} \right\}.$$

3 NM-MPC in Honest Majority Setting

We follow the standard three-step procedure of defining a security notion for MPC protocols.

Formalizing the real world computation. We follow the *clock-driven* execution modeling of synchronous MPC protocols (c.f. [CDN15, Chapter 4]). It is then natural to model a synchronous tampering of such protocols, where the adversary cannot drop or delay messages or desynchronize the parties. We allow the tampering of the messages transmitted by the currently active agent be possibly related to the messages transmitted in the past. However, we do not allow the tampering of current messages to depend on future messages. In the case the adversary is allowed to tamper with all the messages transmitted by an agent jointly using a tampering function that takes all the messages transmitted during an activation as input and output the tampered messages, we call it the *joint tampering* model. In the case the adversary must tamper with each message independent of the messages transmitted through other channels, we call it the *independent tampering* model. We formerly describe the process of executing an MPC protocol under corruption *and tampering* in Appendix A. We formalize the tampering of all messages transmitted between two agents (say, a_1 and a_2) as an

interactive tampering function $f_{\mathsf{Ch}(a_1,a_2)}$. Without loss of generality, we assume the adversary does not tamper with channels connecting at least one corrupted agent (this does not mean that corruption and tampering are separated). In this way, we define a tampering function for an active adversary corrupting a set $\mathcal{S}$ of servers and a set $\mathcal{C}$ of clients in an MPC protocol Π computing a circuit C as a sequence of channel tampering functions as follows.

$$f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}} := \left(\{f_{\mathsf{Ch}(i,i')}\}_{i,i'\in\bar{\mathcal{S}},\ i<i'}; \{f_{\mathsf{Ch}(i,\mathsf{n}+i')}\}_{i\in\bar{\mathcal{S}},i'\in\bar{\mathcal{C}}}\right). \tag{1}$$

To summarize, the real world computation is the same as standard model where there is no tampering (we recycle some of the notations from Definition 1 and indicate the changes), except that the views of honest parties are subject to a tampering denoted by $f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}$. The global view in real world contains the view of corrupted parties (including all other information the adversary manage to maintain through tampering, e.g. a state of bounded size) and the output of honest clients. The real world computation is characterized by a random variable

$$\mathsf{Real}^{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}_{\Pi,\mathsf{Adv},\mathcal{S}\cup\mathcal{C}}(\mathbf{x}) = \left(\mathsf{view}^{\Pi,\mathsf{Adv}^*}_{\mathcal{S}\cup\mathcal{C}}(\mathbf{x}), \mathsf{out}^{\Pi,\mathsf{Adv}^*}_{\bar{\mathcal{C}}}(\mathbf{x})\right),$$

where the superscript $^{,\mathsf{Adv}^*}$ highlights the presence of Adv and $f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}$.

Formalizing the ideal world computation. There is an incorruptible trusted party in the ideal world, who evaluates the circuit C on m inputs provided by the clients and provides the outputs. The ideal world adversary Sim corrupts a set $\mathcal{S}\cup\mathcal{C}$ of parties, which means that Sim gets to substitute the inputs of the corrupted clients in $\mathcal{C}$ before they are given to the trusted party, and simulate views for $\mathcal{S}\cup\mathcal{C}$. Moreover, Sim is allowed to choose a function f and apply it to the output of honest clients. We clarify at this point that the application of f to the output happens after the circuit evaluation (without interfering with the working of the incorruptible trusted party). The ideal world computation is characterized by a random variable

$$\mathsf{Ideal}^{f\leftarrow\mathcal{D}_{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}}_{\mathsf{C},\mathsf{Sim},\mathcal{S}\cup\mathcal{C}}(\mathbf{x}) = \left(\mathsf{view}^{\mathsf{C},\mathsf{Sim}}_{\mathcal{S}\cup\mathcal{C}}(\mathbf{x}), f\left(\mathsf{out}^{\mathsf{C},\mathsf{Sim}}_{\bar{\mathcal{C}}}(\mathbf{x})\right)\right),$$

where the superscript $^{f\leftarrow\mathcal{D}_{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}}$ indicates that here Sim is allowed to individually modify, after learning its own outputs, the output that each honest party receives.

Comparing real/ideal world. The privacy (decoupled from correctness) against malicious adversary is captured by requiring that given any real world adversary Adv corrupting $\mathcal{S}\cup\mathcal{C}$ and tampering $f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}$, there exists an ideal world Sim that (also corrupts $\mathcal{S}\cup\mathcal{C}$) simulates the adversary in the sense that the two random variables $\mathsf{Real}^{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}_{\Pi,\mathsf{Adv},\mathcal{S}\cup\mathcal{C}}(\mathbf{x})$ and $\mathsf{Ideal}^{f\leftarrow\mathcal{D}_{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}}_{\mathsf{C},\mathsf{Sim},\mathcal{S}\cup\mathcal{C}}(\mathbf{x})$ are indistinguishable.

Definition 6 (NM-MPC). *Let C be a m-input functionality and let Π be an m-client n-server protocol that securely computes C when all parties follow the protocol specifications. We say that Π* $(\mathsf{t},\mathcal{F},\epsilon)$-non-malleablly computes C *if for*

every probabilistic adversary Adv *in the real world controlling a set* $\mathcal{S} \subset [\mathsf{n}]$ *of servers such that* $|\mathcal{S}| \leq \mathsf{t}$*, a set* $\mathcal{C} \subset [\mathsf{m}]$ *of clients and a sequence* $f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}$ *of* $\mathcal{F}$*-tampering functions, there exists a probabilistic simulator* Sim *in the ideal world such that for every input* $\mathbf{x}$*, it holds that*

$$\mathsf{SD}\left(\mathsf{Real}^{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}_{\Pi,\mathsf{Adv},\mathcal{S}\cup\mathcal{C}}(\mathbf{x}); \mathsf{Ideal}^{f\leftarrow\mathcal{D}_{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}}_{\mathsf{C},\mathsf{Sim},\mathcal{S}\cup\mathcal{C}}(\mathbf{x})\right) \leq \epsilon. \tag{2}$$

Specially, the protocol is a detection NM-MPC *if the distribution* $\mathcal{D}_{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}$ *is supported on* $\{\perp, \mathsf{same}^*\}^{|\overline{\mathcal{C}}|}$*.*

We only require the protocol to securely computes C when all parties follow the protocol specifications in Definition 6, which is the weakest form of *usefulness* that successfully rules out vacuous private protocols (e.g. simply ignore all parties and output a constant, this protocol is private against malicious adversary although it does not compute anything). Another natural way to define NM-MPC with a stronger form of usefulness is to further require (2) to hold for $f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}} = \mathsf{Id}$ (no tampering of secure point-to-point channels) with $\mathcal{D}_{\mathsf{Id}}$ solely supported on some $\Delta \in \{\perp, \mathsf{same}^*\}^{|\overline{\mathcal{C}}|}$ when there is deviation from protocol specifications. Secure (with abort) MPC protocols in Definition 1 satisfy this stronger form of usefulness.

The class $\mathcal{F}$ of interactive tampering functions considered in this preliminary study are the Bounded State Tampering (BST) functions defined in [FGJ+19] and its variants.

Definition 7 ([FGJ+19]). *Functions of the class of* s*-bounded state tampering functions* $\mathcal{F}^s_{\mathsf{BST}}$ *for an* r*-round interactive protocol are defined by an* r*-tuple of pairs of functions* $((g_1, h_1), \ldots, (g_r, h_r))$ *where the range of the functions* h_i *is* $\{0,1\}^s$*. Let* $m_1, \ldots, m_i$ *be the messages sent by the participants of the protocol in a partial execution. The tampering function for the* i*th message is then defined as*

$$f_i(m_1, \ldots, m_i) \colon = g_i(m_i, h_{i-1}(m_{i-1}, h_{i-2}(m_{i-2}, \ldots))). \tag{3}$$

For the constructions in Section 3.1, we consider a slightly stronger tampering class called *Bounded State Tampering with auxiliary information* $\mathcal{F}^{s,auxi}_{\mathsf{BST}}$. They are BST functions that can be described as follows.

$$f_i(m_1, \ldots, m_i) \colon = g_i(\mathsf{auxi}_i, m_i, h_{i-1}(\mathsf{auxi}_{i-1}, m_{i-1}, h_{i-2}(\mathsf{auxi}_{i-2}, m_{i-2}, \ldots))). \tag{3'}$$

For the constructions in Section 3.2, we consider a *weak Bounded State Tampering* $\mathcal{F}^s_{\mathsf{weakBST}}$, where the state size bound s is effective for the current message as well as past messages. The class $\mathcal{F}^s_{\mathsf{weakBST}}$ is strictly weaker than $\mathcal{F}^s_{\mathsf{BST}}$. The tampering access of the $\mathcal{F}^s_{\mathsf{weakBST}}$ adversary to a message is bounded by the number of values it is allowed to use to replace the original message. Suppose the message is a vector in the space $\mathbb{F}^u$. The admissible current message tampering function over $\mathbb{F}^u$ is given by

$$\{g_j \colon \mathbb{F}^u \to \mathbb{F}^u || \mathsf{Range}(g_j)| \leq 2^s\},$$

where $\mathsf{Range}(g_j)$ denotes the range of the function g_j.

Definition 8. *The class $\mathcal{F}^s_{\mathsf{SS},\theta}$ of independent tampering functions for secret sharing scheme over $\mathbb{F}^u$ induced by a θ-bounded (corrupting at most θ servers) adversary of MPC protocol with $\mathcal{F}^s_{\mathsf{weakBST}}$ channel tampering access is defined as follows. Any function $f \in \mathcal{F}^s_{\mathsf{SS},\theta}$ is written as*

$$f\colon (\mathbb{F}^u)^{\mathsf{n}} \to (\mathbb{F}^u)^{\mathsf{n}},\ f = (f_1, \ldots, f_{\mathsf{n}}),$$

where at most θ components $f_i\colon \mathbb{F}^u \to \mathbb{F}^u$ are arbitrary functions and the rest of the $\mathsf{n} - \theta$ components $f_i\colon \mathbb{F}^u \to \mathbb{F}^u$ should satisfy

$$|f_i(\mathbb{F}^u)| \leq 2^s.$$

3.1 Feasibility Results with Strong Tampering

Consider an interactive protocol Π executed by Alice and Bob on a communication channel that is partially controlled by an adversary Eve, whose capability is characterised by a set $\mathcal{F}$ of tampering functions, from which the adversary can arbitrarily choose one to tamper with all the messages communicated between Alice and Bob. At the end of protocol Π, Alice and Bob privately (locally) generate $K_A, K_B \in \{0,1\}^k \cup \{\perp\}$ from their respective views. For a function $f \in \mathcal{F}$, let $\mathsf{view}^f_{\mathsf{Adv}}$ denote the adversary's view when f is chosen to tamper with the execution. Given a string $\mathsf{v} \in \{0,1\}^k \cup \{\perp\}$, let $\mathsf{purify}(\mathsf{v})$ be $\perp$ if $\mathsf{v} = \perp$, and otherwise replace $\mathsf{v} \neq \perp$ by a fresh k-bit random string: $\mathsf{purify}(\mathsf{v}) \leftarrow \mathsf{U}_k$. Intuitively, we want the guarantee that the uniform keys generated by Alice and Bob are either the same or independent of each other, and, in all cases, independent from the view of the tampering adversary.

Definition 9 (implicit in [FGJ+19]). *An interactive protocol Π executed by Alice and Bob is a ε-non-malleable key exchange (ε-NM-KE) protocol with respect to $\mathcal{F}$ if it satisfies the following properties.*

1. *The keys K_A, K_B are close to uniform conditioned on adversary's view.*

$$\mathsf{SD}\left((K_A, \mathsf{view}^f_{\mathsf{Adv}}); (\mathsf{U}_k, \mathsf{view}^f_{\mathsf{Adv}})\right) \leq \varepsilon; \mathsf{SD}\left((K_B, \mathsf{view}^f_{\mathsf{Adv}}); (\mathsf{U}_k, \mathsf{view}^f_{\mathsf{Adv}})\right) \leq \varepsilon.$$

2. *If the adversary chooses a function $f \in \mathcal{F}$ that does not alter any message (we say the adversary is passive in this case), then*

$$\mathsf{Pr}[K_A = K_B \wedge K_A \neq \perp \wedge K_B \neq \perp] = 1.$$

3. *If the adversary chooses a function $f \in \mathcal{F}$ that alters messages (we say the adversary is active in this case), then there exists a probability p_f such that*

$$\mathsf{Pr}[K_A = K_B \wedge K_A \neq \perp \wedge K_B \neq \perp] = \mathsf{p}_f,$$

and (when K_A and K_B are not equal) at least one of the following must hold.

$$\mathsf{SD}\left((K_A, \mathsf{view}^f_{\mathsf{Adv}}, K_B); (\mathsf{purify}(K_A), \mathsf{view}^f_{\mathsf{Adv}}, K_B)\right) \leq \varepsilon;$$
$$\mathsf{SD}\left((K_A, \mathsf{view}^f_{\mathsf{Adv}}, K_B); (K_A, \mathsf{view}^f_{\mathsf{Adv}}, \mathsf{purify}(K_B))\right) \leq \varepsilon.$$

The rate of a non-malleable key exchange is the ratio $\frac{k}{|\mathsf{Trans}^{\Pi}|}$*, where* Trans^{Π} *denotes the transcript of the protocol in the case that no abort occurs. A rate* $0 < \rho < 1$ *is achievable by NM-KE with respect to* $\mathcal{F}$*, if there exists* ε*-non-malleable key exchange protocols with respect to* $\mathcal{F}$ *with rate approaching* ρ *and* ε *goes to zero as the transcript size grows.*

The high level idea of the construction in Theorem 1 is to encode the communication between each pair of parties using an INMC with respect to bounded state tampering. In order to argue non-malleability in the joint tampering model, which is a natural extension as we move from interactive coding to multi-party coding, we need a careful analysis of how the auxiliary information available at the tampering of one message can affect the execution.

Theorem 1. *Let* Π *be an MPC protocol* $(\mathsf{t}, \varepsilon)$*-securely computes a circuit* C *with* r *rounds and a communication complexity of* Σ *bits. Let* $\Pi_{\mathsf{NM\text{-}KE}}$ *be the* $r_{\mathsf{NM\text{-}KE}}$*-round* ε*-non-malleable key exchange protocol with respect to* $\mathcal{F}^{s}_{\mathsf{BST}}$*-channel tampering with rate* $\rho_{\mathsf{NM\text{-}KE}}$ *[FGJ+19]. Let* $\mathsf{MAC}\colon \{0,1\}^{2\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ *be a* $2^{-\lambda}$*-secure information theoretic message authentication code, where* λ *is a big enough constant (assume each message transmitted in* Π *is also of length* λ*). There is an* $(r + r_{\mathsf{NM\text{-}KE}} + 1)$*-round MPC protocol* Π_{NM} $(\mathsf{t}, \mathcal{F}^{s}_{\mathsf{BST}}, O(\varepsilon, 2^{-\lambda}))$*-non-malleably computes a circuit* C *in the joint tampering model, which has communication complexity* $\left(\frac{3}{\rho_{\mathsf{NM\text{-}KE}}} + 1\right)\Sigma$*. In particular,* Π_{NM} *is a detection NM-MPC.*

Proof. The NM-MPC protocol Π_{NM} is described as follows.

1. Key generation phase.
   Each server S_i runs a key exchange Π_{KE} with any other server $S_{i'}$ to generate a key $\mathsf{key}_{i,i'}$ of length $|\mathsf{key}_{i,i'}| = r_{i,i'} \cdot 3\lambda$, where $r_{i,i'}$ denotes the number of messages exchanged between the two parties, and split the key $\mathsf{key}_{i,i'}$ into substrings
   $$\mathsf{key}_{i,i'} = \mathsf{MACkey}^{1}_{i,i'} || \mathsf{OTPkey}^{1}_{i,i'} || \ldots || \mathsf{MACkey}^{r_{i,i'}}_{i,i'} || \mathsf{OTPkey}^{r_{i,i'}}_{i,i'},$$
   where $|\mathsf{MACkey}^{j}_{i,i'}| = 2\lambda$ and $|\mathsf{OTPkey}^{j}_{i,i'}| = \lambda$ for $j = 1, \ldots, r_{i,i'}$.
   Each server S_i runs a key exchange Π_{KE} with any client $C_{i'}$ (considered as the $(\mathsf{n}+i')$th party) to generate a key $\mathsf{key}_{i,\mathsf{n}+i'}$ of length $|\mathsf{key}_{i,\mathsf{n}+i'}| = r_{i,\mathsf{n}+i'} \cdot 3\lambda$, where $r_{i,\mathsf{n}+i'}$ denotes the number of messages exchanged between the two parties, and split the key $\mathsf{key}_{i,\mathsf{n}+i'}$ into substrings
   $$\mathsf{key}_{i,\mathsf{n}+i'} = \mathsf{MACkey}^{1}_{i,\mathsf{n}+i'} || \mathsf{OTPkey}^{1}_{i,\mathsf{n}+i'} || \ldots || \mathsf{MACkey}^{r_{i,\mathsf{n}+i'}}_{i,\mathsf{n}+i'} || \mathsf{OTPkey}^{r_{i,\mathsf{n}+i'}}_{i,\mathsf{n}+i'},$$
   where $|\mathsf{MACkey}^{j}_{i,\mathsf{n}+i'}| = 2\lambda$ and $|\mathsf{OTPkey}^{j}_{i,\mathsf{n}+i'}| = \lambda$ for $j = 1, \ldots, r_{i,\mathsf{n}+i'}$.
   In this phase, all $(\mathsf{n}-\mathsf{t})(\mathsf{n}-\mathsf{t}-1+2\mathsf{m})/2$ independent copies of Π_{KE} are run in parallel. Since different copies of Π_{KE} use independent randomness, the non-malleability of the keys against joint tampering adversary reduces to the non-malleability against the independent tampering adversary. Moreover, since the tampering class $\mathcal{F}$ is round number insensitive, combining the key exchange phase with the protocol evaluation phase does not affect the non-malleability of the keys.

2. Protocol evaluation phase.
   - If the jth round is not a round of input sharing gate for Π,
     (a) When it is an honest server S_i's turn to send messages, the party invokes the next-message function $\mathsf{nextMSG}_i^j$ of Π to compute

$$(m_{i,1}^j, \ldots, m_{i,\mathsf{n}}^j) \colon = \mathsf{nextMSG}_i^j(\mathbf{R}_i^j, \mathsf{view}_i^\Pi).$$

     S_i appends the local randomness $\mathbf{R}_i^j$ and the message $m_{i,i}^j$ to the party's own view view_i^Π.

     (b) Next for every receiver $S_{i'}$ of S_i in the jth round of Π, the party S_i computes the one-time pad encryption as well as authentication tag

$$c_{i,i'}^j = m_{i,i'}^j \oplus \mathsf{OTPkey}_{i,i'}^{j'} \text{ and } t_{i,i'}^j = \mathsf{MAC}\left(\mathsf{MACkey}_{i,i'}^{j'}, c_{i,i'}^j\right),$$

     where j' denotes that this is the j'th message transmitted between S_i and $S_{i'}$. The masked messages together with their corresponding tags are then transmitted in place of the plain messages.

     (c) When every server that sends messages in the jth round of Π completes the transmission, each honest server $S_{i'}$ verifies the tag of the received messages $(c_{i,i'}^j, t_{i,i'}^j)$ from each server S_i using the corresponding authentication key $\mathsf{MACkey}_{i,i'}^{j'}$ then decrypts

$$\mathsf{Vf}(\mathsf{MACkey}_{i,i'}^{j'}, c_{i,i'}^j, t_{i,i'}^j) = 1,\ m_{i,i'}^j = c_{i,i'}^j \oplus \mathsf{OTPkey}_{i,i'}^{j'},$$

     and finally appends $m_{i,i'}^j$ to the party's view $\mathsf{view}_{i'}^\Pi$. If the protocol is executed under tampering, then the MAC key of server $S_{i'}$ may be different from the MAC key of server S_i, or the messages received by server $S_{i'}$ may no longer be equal to $(c_{i,i'}^j, t_{i,i'}^j)$, and the verification may fail causing the server $S_{i'}$ to abort.

   - If the jth round is a round of input sharing or output reconstruction gate for Π, do the following.
     (a) When it is a client C_i's (considered as the $(\mathsf{n}+i)$th party) turn to send messages, the party invokes $\mathsf{nextMSG}_{\mathsf{n}+i}^j$ of Π to compute

$$(m_{\mathsf{n}+i,1}^j, \ldots, m_{\mathsf{n}+i,\mathsf{n}}^j) \colon = \mathsf{nextMSG}_i^j(\mathbf{R}_{\mathsf{n}+i}^j, \mathsf{view}_{\mathsf{n}+i}^\Pi).$$

     (b) Next for every receiver $S_{i'}$ of C_i in the jth round of Π, the party C_i computes the one-time pad encryption as well as authentication tag

$$c_{\mathsf{n}+i,i'}^j = m_{\mathsf{n}+i,i'}^j \oplus \mathsf{OTPkey}_{i',\mathsf{n}+i}^{j'} \text{ and } t_{\mathsf{n}+i,i'}^j = \mathsf{MAC}\left(\mathsf{MACkey}_{i',\mathsf{n}+i}^{j'}, c_{\mathsf{n}+i,i'}^j\right),$$

     where j' denotes that this is the j'th message transmitted between S_i and $C_{i'}$. The masked messages together with their corresponding tags are then transmitted instead of the plain messages.

(c) When every client that sends messages in the jth round of Π completes the transmission, each honest server $S_{i'}$ verifies the tag of the received messages $(c^j_{\mathsf{n}+i,i'}, t^j_{\mathsf{n}+i,i'})$ from each client C_i using the corresponding authentication key $\mathsf{MACkey}^{j'}_{i',\mathsf{n}+i}$ then decrypts

$$\mathsf{Vf}(\mathsf{MACkey}^{j'}_{i',\mathsf{n}+i}, c^j_{\mathsf{n}+i,i'}, t^j_{\mathsf{n}+i,i'}) = 1,\ m^j_{i,i'} = c^j_{i,i'} \oplus \mathsf{OTPkey}^{j'}_{i',\mathsf{n}+i},$$

and finally appends $m^j_{\mathsf{n}+i,i'}$ to the party's view $\mathsf{view}^{\Pi}_{i'}$. If the protocol is executed under tampering, then the MAC key of server $S_{i'}$ may be different from the MAC key of server S_i, or the messages received by server $S_{i'}$ may no longer be equal to $(c^j_{\mathsf{n}+i,i'}, t^j_{\mathsf{n}+i,i'})$, and the verification may fail causing the server $S_{i'}$ to abort.

We define the non-malleability simulator Sim for Π_{NM}. We begin with simulating the corrupted parties' view. In the key exchange phase, the corrupted parties receive two uniform messages from each honest party (the length depends on how many messages are to be communicated between a corrupted party and the honest party during execution of the underlying MPC protocol). The corrupted parties' view after the key exchange phase is simulated with the help of the simulator of the underlying MPC protocol (before harden for NM) until when there are honest parties abort. Here the NM-MPC simulator first invokes the MPC simulator to obtain corrupted parties' "plain-text" view and then uses the keys generated in key exchange phase to encrypt (OTP and MAC) the "plain-text" view. Whenever an honest party aborts before completing his/her role in the protocol execution, the NM-MPC simulator stops calling the MPC simulator (because in the real world, the honest party stops sending messages). We now discuss how to simulate whether or not honest parties abort and, if they do, when exactly does each of them abort. The NM-MPC simulator uses the message topology of the underlying MPC protocol (before hardened for NM), to proceed with simulating. It starts with the first round in the message topology when we have an honest sender and do the following for each of his/her honest receivers. In the case a pair of honest parties have different keys (according to the corresponding NM-KE simulator), the honest receiver aborts. Here the NM-MPC simulator invokes several copies of NM-KE simulators, each NM-KE simulator for one honest receiver, which simulates whether the sender and receiver have the same key according to the corresponding tampering function provided by the adversary. In the case when the keys of a pair of honest parties are the same, the NM-MPC simulator decide whether or not the receiver should abort according to whether or not there is tampering at the transmission from the sender to the receiver (adversary provides this information to NM-MPC simulator). In this way, following the message topology, when the next time an honest party is the sender, if the sender has already aborted in previous rounds, the NM-MPC simulator let all honest receivers in the current round abort; if the sender has not aborted in previous rounds, the NM-MPC simulator repeat calling NM-KE simulator (if all pairs of honest parties are checked, can skip calling NM-KE simulator in the future) and adversary to decide whether or not each receiver should

abort. The fixed message topology agreed on by all parties before hand allows the simulation to match the real protocol execution. Note that it is possible that some honest parties may not abort if only their last messages are tampered with.

Without loss of generality, we describe the simulation of honest party's output for single output C. If an honest party aborts before completing his/her role in the full MPC protocol execution, when he/she is the sender in the round after he/she aborts, the parties expecting his/her messages will abort. In this way, more and more parties abort until the party (reconstructor) who is supposed to compute the final output aborts. This leads to an $\perp$ output of the NM-MPC simulator. It remains to consider the case when the last message (the sender has no further role in the execution after the current round) of an honest party is tampered with. In the honest majority MPC protocols, these rounds are collectively called the "last message round", where each party sends a share to the reconstructor, who reconstructs the final output value. If tampering happens when an honest party is sending his/her share to the reconstructor, he/she will not abort. But according to the hardened protocol the reconstructor verifies the last message of each party before using a subset of them for reconstruction, hence would catch a tampering and abort. To summarize, the MPC simulator will output $\perp$ with a probability computed from the tampering functions or a value from the trusted party. The concrete simulation algorithm is given below, followed by analysis.

First step, non-malleability simulator reads the sequence $f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}$ of channel tampering functions for honest parties. Extracts the tampering functions for the rounds corresponding to Π_{KE} and sample the reaction of key exchange phase. If all keys are correctly generated, move on to next step, otherwise output $\perp$. Let p_1 be the probability that Sim continues. Second step, non-malleability simulator extracts the tampering functions for the remaining rounds and apply them to uniform messages. In this step, compute the probability p_2 that all tampering functions fix all messages simultaneously. This defines, till here, the distribution $\mathcal{D}_{f_{\Pi(\mathsf{C}),\mathcal{S}\cup\mathcal{C}}}$, which is supported on $\{\perp, \mathsf{same}^*\}$. Third step, non-malleability simulator invoke the simulator of Π on the adversary strategy Adv to obtain either $\perp$ or an output provided by the trusted party. In the first step, the tampering functions receives messages of other parallel executions of Π_{NM} as auxiliary information in the joint tampering model. Note that, in particular, the Π_{NM} considered here sends independent uniform strings each round and there is no dependence across parallel execution. These auxiliary information does not affect the security of the generated keys. In the second step, since all messages are masked by the keys generated in the first step, independence of auxiliary information in the joint tampering model reduces to security of the keys. In the third step, if the real world protocol did not abort up to this point, with overwhelming probability the adversary did not tamper with secure point-to-point channels and the analysis reduces to corruption only adversary case.

3.2 Optimal Efficiency with Weak Tampering

Theorem 2. *Let* $c_s = \frac{2^s \sin(\pi/2^s)}{p \sin(\pi/p)} < 1$ *(when* $2^s < p$*). Let* $\mathsf{C}\colon \mathbb{F}_p \times \ldots \times \mathbb{F}_p \to \mathbb{F}_p$ *be an* m*-input functionality. There is an* m*-client* n*-server MPC protocol* Π_{NM} *(basing on a secret sharing scheme* $(\mathsf{Share}, \mathsf{Recover})$ *over* $\mathbb{F}_p^u$*) that* $(\theta, \mathcal{F}^s_{\mathsf{weakBST}}, \big(O(\mathsf{m}, \mathsf{n} - \theta) \cdot \frac{u}{2} \cdot 2^s \cdot {c_s}^{\mathsf{n}/2-\theta-1} + \epsilon\big))$*-non-malleably computes* C *in the independent tampering model. Moreover, the protocol* Π_{NM} *has asymptotic communication complexity same as best passive security MPC.*

The protocol Π_{NM} in Theorem 2 is a special instantiation of the secure (with abort) MPC construction proposed in [GIP+14] (achieving same asymptotic communication complexity as best passive security MPC) that adds one more layer of algebraic structure to the construction for achieving the non-malleability against $\mathcal{F}^s_{\mathsf{weakBST}}$-tampering. This algebraic structure comes from the choice of a finite field $\mathbb{F}_p$ of large prime order p, which in effect turns the Shamir secret sharing scheme over a field of characteristic p into a NM-SS with respect to the tampering class $\mathcal{F}^s_{\mathsf{SS},\theta}$ in Definition 8 (this explains the constant c_s in the statement). Since our construction uses the construction of [GIP+14] in a black-box manner, we will only mention here that the construction is based on a *redundant dense linear* secret sharing scheme (see Appendix B for the definition of linear-based MPC, which captures the typical structure of many information-theoretic secure MPC protocols such as the BGW protocol [BGW88,AL17] and the DN protocol [DN07]). For completeness, we include a brief introduction of the full construction of [GIP+14] in Appendix C. The proof of Theorem 2 uses notations introduced in Appendix C, hence is given in Appendix D. In the rest of this subsection, we discuss the construction of NM-SS with linear reconstruction function, which is the core of the technicality behind Theorem 2.

Properties of Shamir's scheme over large prime fields

With the application to securing general purpose MPC protocols against passive implementation attacks in mind, the pioneering work [BDIR18] and various follow-up works [BDIR21,MPSW21,MNP+21] analyze a special property of Shamir's scheme over large prime fields that makes them Leakage-Resilient Secret Sharing (LR-SS) schemes. In order to give a unified exposition of these recent findings for both passive and active implementation security applications (leakage functions and tampering functions have different output length), we describe a function $f = (f_1, \ldots, f_{\mathsf{n}})$ (here f can be an s-bit leakage function or an s-BST function) in an abstract fashion using a sequence of partitions $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_{\mathsf{n}})$, where each partition $\mathcal{P}_i$ is defined by the pre-image sets of all elements in the range of f_i. We continue with this abstract characterization of functions and capture the state size bound of BST as follows (here we deliberately use parameter k instead of n to emphasize that it is possible that only a subset of the n components of f appear in the analysis).

Definition 10. *Let* $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_{\mathsf{k}})$ *be a sequence of partitions of* $\mathbb{F}_p$*. We say that the partition sequence* $\mathcal{P}$ *is* s*-bounded if each partition* $\mathcal{P}_i$ *divides* $\mathbb{F}_p$ *into no more than* 2^s *parts.*

When $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_{\mathsf{k}})$ is s-bounded, we simply write (in the case when a partition contains strictly less than 2^s subsets, we pad empty sets in the back)

$$\mathcal{P} = (\mathcal{P}_1^1 \cup \ldots \cup \mathcal{P}_1^{2^s}, \ldots, \mathcal{P}_{\mathsf{k}}^1 \cup \ldots \cup \mathcal{P}_{\mathsf{k}}^{2^s}),$$

where for each $i \in [\mathsf{k}]$, subsets $\mathcal{P}_i^1, \ldots, \mathcal{P}_i^{2^s}$ are disjoint and $\mathcal{P}_i^1 \cup \ldots \cup \mathcal{P}_i^{2^s} = \mathbb{F}_p$. In this way, an element $\mathbf{y} = (y_1, \ldots, y_{\mathsf{k}})$ in the range of $f = (f_1, \ldots, f_{\mathsf{k}})$ is labeled by a tuple $(j_1, \ldots, j_{\mathsf{k}}) \in [2^s]^{\mathsf{k}}$ and $f(\mathbf{x}) = \mathbf{y}$ means $(x_1 \in \mathcal{P}_1^{j_1}) \wedge \ldots \wedge (x_{\mathsf{k}} \in \mathcal{P}_{\mathsf{k}}^{j_{\mathsf{k}}})$.

Using the connection between linear secret sharing schemes and linear MDS codes (see Section 2), the problem is reduced to the study of the distribution of the random variable $f(C)$, where $C \leftarrow \mathfrak{C}$ is sampled uniformly from a linear MDS code $\mathfrak{C}$. The Fourier analysis approach transforms the probability of $f(C)$ taking a certain value into a sum concerning products of Fourier coefficients, and then, through bounding the Fourier coefficients, analyses the probability distribution of $f(C)$. The probability of $f(C)$ taking the value labeled by $(j_1, \ldots, j_{\mathsf{k}})$ is computed by summing, over the codeword space $\mathfrak{C}$, the probability of $C = (c_1, \ldots, c_{\mathsf{k}})$ and $(c_1 \in \mathcal{P}_1^{j_1}) \wedge \ldots \wedge (c_{\mathsf{k}} \in \mathcal{P}_{\mathsf{k}}^{j_{\mathsf{k}}})$, which is either 0 or $\frac{1}{|\mathfrak{C}|}$. Using the *Poisson Summation Formula*, the sum over the linear space $\mathfrak{C}$ is transformed into a sum over the dual space $\mathfrak{C}^\perp$ concerning products of Fourier coefficients. Consider another random variable $f(U)$, where $U \leftarrow \mathbb{F}_p^{\mathsf{k}}$ is sampled uniformly from the full space $\mathbb{F}_p^{\mathsf{k}}$. The probability of $f(U)$ taking the same value labeled by $(j_1, \ldots, j_{\mathsf{k}})$ (a sum over $\mathbb{F}_p^{\mathsf{k}}$) is transformed using the *Poisson Summation Formula* into a sum over the dual space $\left(\mathbb{F}_p^{\mathsf{k}}\right)^\perp = \{\mathbf{0}\}$, which is the zero space consisting of an all-0 vector. This means that the difference of the probability of the random variables $f(C)$ and $f(U)$ taking the same value labeled by $(j_1, \ldots, j_{\mathsf{k}})$ is expressed as a sum over $\mathfrak{C}^\perp \setminus \left(\mathbb{F}_p^{\mathsf{k}}\right)^\perp = \mathfrak{C}^\perp \setminus \{\mathbf{0}\}$ of products concerning Fourier coefficients (the quantity inside the $|\cdot|$ below), and hence the statistical distance between $f(C)$ and $f(U)$ is expressed as follows.

$$\mathsf{SD}(f(C); f(U)) = \frac{1}{2} \sum_{(j_1,\ldots,j_{\mathsf{k}}) \in [2^s]^{\mathsf{k}}} \left| \sum_{(\alpha_1,\ldots,\alpha_{\mathsf{k}}) \in \mathfrak{C}^\perp \setminus \{\mathbf{0}\}} \prod_{i \in [\mathsf{k}]} \hat{\mathbf{1}}_{\mathcal{P}_i^{j_i}}(\alpha_i) \right|.$$

Applying the triangle inequality and rearrange, we have an upper bound

$$\mathsf{SD}(f(C); f(U)) \leq \frac{1}{2} \sum_{(\alpha_1,\ldots,\alpha_{\mathsf{k}}) \in \mathfrak{C}^\perp \setminus \{\mathbf{0}\}} \prod_{i \in [\mathsf{k}]} \left(\sum_{j_i \in [2^s]} |\hat{\mathbf{1}}_{\mathcal{P}_i^{j_i}}(\alpha_i)| \right). \tag{4}$$

It is shown (see [BDIR21, Lemma 4.16]) that the quantity inside the $(\cdot)$ in (4) can be bounded as follows.

$$\begin{cases} \sum_{j_i \in [2^s]} |\hat{\mathbf{1}}_{\mathcal{P}_i^{j_i}}(\alpha_i)| \leq c_s, \text{ if } \alpha_i \neq 0; \\ \sum_{j_i \in [2^s]} |\hat{\mathbf{1}}_{\mathcal{P}_i^{j_i}}(\alpha_i)| = 1, \text{ if } \alpha_i = 0, \end{cases}$$

where the constant c_s is defined and bounded as follows (see [BDIR21, Lemma 4.10])

$$c_s = \frac{2^s \sin(\pi/2^s)}{p \sin(\pi/p)} \leq 1 - 2^{-2s}, \ 1 \leq s \leq \log p - 1. \tag{5}$$

Now for a given s, through choosing a big enough dimension of $\mathfrak{C}$, one can make the sum over $\mathfrak{C}^{\perp}\backslash\{\mathbf{0}\}$ of the product concerning Fourier coefficients (each product in (4) is upper bounded by $c_s^{d^{\perp}}$, where $d^{\perp}$ denotes the minimum distance of $\mathfrak{C}^{\perp}$) smaller than a given error parameter. There is an unnecessary undesirable dependence on the cardinality $|\mathfrak{C}^{\perp}\backslash\{\mathbf{0}\}|$ of the dual code space ($|\mathfrak{C}^{\perp}\backslash\{\mathbf{0}\}|$ increases as p increases), which can be removed (through more sophisticated analysis) yielding the bound in Lemma 1.

Definition 11. *Let* $\mathsf{MDS}[\mathsf{k},\mathsf{k}-1,2]_p$ *be an MDS code over alphabet* $\mathbb{F}_p$ *with code parameter* $[\mathsf{k},\mathsf{k}-1,2]$. *Let* $C \leftarrow \mathsf{MDS}[\mathsf{k},\mathsf{k}-1,2]_p$ *denote a random codeword of* $\mathsf{MDS}[\mathsf{k},\mathsf{k}-1,2]_p$ *chosen uniformly from the codebook. Let* $U \leftarrow \mathbb{F}_p^{\mathsf{k}}$ *be the random variable uniformly distributed over* $\mathbb{F}_p^{\mathsf{k}}$. *We say that* $C \leftarrow \mathsf{MDS}[\mathsf{k},\mathsf{k}-1,2]_p$ *is* ε*-indistinguishable from uniform by* s*-bounded partitions if for any* s*-bounded* $\mathcal{P} = (\mathcal{P}_1^1 \cup \ldots \cup \mathcal{P}_1^{2^s}, \ldots, \mathcal{P}_{\mathsf{k}}^1 \cup \ldots \cup \mathcal{P}_{\mathsf{k}}^{2^s})$,

$$\frac{1}{2} \sum_{(j_1,\ldots,j_{\mathsf{k}})\in[2^s]^{\mathsf{k}}} \left| \prod_{i=1}^{\mathsf{k}} \mathsf{Pr}[C_i \in \mathcal{P}_i^{j_i}] - \prod_{i=1}^{\mathsf{k}} \mathsf{Pr}[U_i \in \mathcal{P}_i^{j_i}] \right| \leq \varepsilon.$$

Lemma 1 ([BDIR21] Th 4.6 with $\mathsf{n}=\mathsf{k}$ and $\mathsf{t}=\mathsf{k}-1$). *Let* $c_s = \frac{2^s \sin(\pi/2^s)}{p \sin(\pi/p)} < 1$ *(when* $2^s < p$*). The random variable* $C \leftarrow \mathsf{MDS}[\mathsf{k},\mathsf{k}-1,2]_p$ *is* ε*-indistinguishable from uniform by* s*-bounded partitions for* $\varepsilon = \frac{1}{2} \cdot 2^s \cdot (c_s + 2^{-2s-1})^{\mathsf{k}-2}$.

Multiplicative NM-SS by overcoming n/2 threshold barrier

In terms of explicit parameters, the LR-SS schemes obtained in this line of works [BDIR18,BDIR21,MPSW21,MNP+21] can only have reconstruction threshold $0.8675\mathsf{n}$ [MPSW21] and $0.85\mathsf{n}$ [BDIR21] for Shamir's secret sharing scheme. Having a large reconstruction threshold renders that they can only be applied to restricted types of MPC protocols. Secure multiplication of two secrets requires a secret sharing reconstruction threshold $\mathsf{k} < \mathsf{n}/2$, even against passive adversary.

The reason why we can overcome the $\mathsf{n}/2$ threshold barrier originates from the distinction between security notions against passive attacks and active attacks (in this case, LR-SS v.s. NM-SS). Although tampering adversaries are in general more difficult to handle, there is one aspect in the security requirement for tampering attacks that could be manipulated in the designer's favour. Let $f \in \mathcal{F}$ be an attack function (be it a leakage function or a tampering function). The notion of leakage-resilience requires that $f(\mathsf{Share}(\mathbf{s}_0)) \sim f(\mathsf{Share}(\mathbf{s}_1))$ for any pair of distinct secrets $\mathbf{s}_0$ and $\mathbf{s}_1$. The notion of non-malleability, roughly speaking, requires that $\mathsf{Recover}_{\mathcal{R}}(f(\mathsf{Share}(\mathbf{s}_0))) \sim \mathsf{Recover}_{\mathcal{R}}(f(\mathsf{Share}(\mathbf{s}_1)))$ [2], for any reconstruction set $\mathcal{R}$. The aspect that designers can exploit is the fact that although $f(\mathsf{Share}(\mathbf{s}_0))$ and $f(\mathsf{Share}(\mathbf{s}_1))$ may have distinguishable distributions (over the randomness of the sharing algorithm), it is still possible that the transformation $\mathsf{Recover}_{\mathcal{R}}(\cdot)$ turns them into indistinguishable distributions. This is

[2] This is called *strong non-malleability* in [DPW18].

exactly what we exploit to prove that Shamir's secret sharing scheme with reconstruction threshold below $\mathsf{n}/2$ can be a NM-SS.

Theorem 3. *Let* $c_s = \frac{2^s \sin(\pi/2^s)}{p \sin(\pi/p)} < 1$ *(when* $2^s < p$*). Shamir's secret sharing scheme over* $\mathbb{F}_p$ *with privacy threshold* t *is* $\mathcal{F}^s_{\mathsf{SS},\theta}$*-NM with error* $\varepsilon = \frac{1}{2} \cdot 2^s \cdot (c_s + 2^{-2s-1})^{\mathsf{t}-\theta-1}$.

Proof. We first prove the theorem for the special case of $\theta = 0$ using Lemma 1 and then show a reduction of the $\theta > 0$ case to an instance of the $\theta = 0$ case with shortened code lengthen.

Assume $\theta = 0$. Let $f = (f_1, \ldots, f_{\mathsf{n}}) \in \mathcal{F}^s_{\mathsf{SS},\theta}$ be a secret sharing tampering function. In particular, let $\mathsf{Range}(f_i) = \{\tilde{c}_i^1, \ldots, \tilde{c}_i^{2^s}\}$, $i = 1, \ldots, \mathsf{n}$. In the case when $|\mathsf{Range}(f_i)| < 2^s$, we pad values not in $\mathsf{Range}(f_i)$ and let the pre-mage sets of the padded values be empty. Let $\mathcal{P}_i = (\mathcal{P}_i^1, \ldots, \mathcal{P}_i^{2^s})$ be defined by the f_i pre-image sets of $\tilde{c}_i^1, \ldots, \tilde{c}_i^{2^s}$. By definition, we have

$$\begin{cases} \mathsf{Pr}[C_i \in \mathcal{P}_i^{j_i}] = \mathsf{Pr}[f_i(C_i) = \tilde{c}_i^{j_i}],\ i = 1, \ldots, \mathsf{n}; \\ \mathsf{Pr}[U_i \in \mathcal{P}_i^{j_i}] = \mathsf{Pr}[f_i(U_i) = \tilde{c}_i^{j_i}],\ i = 1, \ldots, \mathsf{n}. \end{cases}$$

According to the linearity of Shamir's secret sharing scheme, there is constant vector $\Delta^{\mathsf{s}} \in \mathbb{F}_p^{\mathsf{n}}$ such that $\mathsf{Share}(\mathsf{s}) = \mathsf{Share}(0) + \Delta^{\mathsf{s}}$. Now, for any reconstruction set $\mathcal{R} = \{i_1, \ldots, i_{\mathsf{t}+1}\}$, define the simulation

$$\mathcal{D}_{f,\mathcal{R}} = \left\{ \begin{array}{c} U \leftarrow \mathbb{F}_p^{\mathsf{n}} \\ \tilde{\mathbf{v}}_{\mathcal{R}} \leftarrow \left(f_{i_1}(U_{i_1} + \Delta^{\mathsf{s}}_{i_1}), \ldots, f_{i_{\mathsf{t}+1}}(U_{i_{\mathsf{t}+1}} + \Delta^{\mathsf{s}}_{i_{\mathsf{t}+1}})\right) \\ \tilde{\mathsf{s}} = \mathsf{Recover}(\tilde{\mathbf{v}}_{\mathcal{R}}, \mathcal{R}) \\ \text{Output } \tilde{\mathsf{s}}. \end{array} \right\},$$

where the distributions of $U_{i_1} + \Delta^{\mathsf{s}}_{i_1}, \ldots, U_{i_{\mathsf{t}+1}} + \Delta^{\mathsf{s}}_{i_{\mathsf{t}+1}}$ are in fact independent of the secret $\mathbf{s}$. On the other hand, the real tampering experiment is

$$\mathrm{Tamper}^{f,\mathcal{R}}_{\mathsf{s}} = \left\{ \begin{array}{c} \mathbf{v} + \Delta^{\mathsf{s}} \leftarrow \mathsf{Share}(0) + \Delta^{\mathsf{s}} \\ \tilde{\mathbf{v}}_{\mathcal{R}} = \left(f_{i_1}(v_{i_1} + \Delta^{\mathsf{s}}_{i_1}), \ldots, f_{i_{\mathsf{t}+1}}(v_{i_{\mathsf{t}+1}} + \Delta^{\mathsf{s}}_{i_{\mathsf{t}+1}})\right) \\ \tilde{\mathsf{s}} = \mathsf{Recover}(\tilde{\mathbf{v}}_{\mathcal{R}}, \mathcal{R}) \\ \text{Output } \tilde{\mathsf{s}}. \end{array} \right\}.$$

Firstly, the offset Δ^{s} appears in both $\mathcal{D}_{f,\mathcal{R}}$ and $\mathrm{Tamper}^{f,\mathcal{R}}_{\mathsf{s}}$ can be dropped without affecting $\mathsf{SD}(\mathrm{Tamper}^{f,\mathcal{R}}_{\mathsf{s}}; \mathcal{D}_{f,\mathcal{R}})$. Secondly, $\mathsf{Share}(0)_{\mathcal{R}}$ has the same distribution as $C \leftarrow \mathsf{MDS}[\mathsf{t}+1, \mathsf{t}, 2]_p$, where $\mathsf{MDS}[\mathsf{t}+1, \mathsf{t}, 2]_p$ is defined by puncturing the components in $[\mathsf{n}] \backslash \mathcal{R}$ in the MDS code corresponding to the support of $\mathsf{Share}(0)$. Finally, we invoke Lemma 1 to claim

$$\begin{array}{l} \mathsf{SD}(\mathrm{Tamper}^{f,\mathcal{R}}_{\mathsf{s}}; \mathcal{D}_{f,\mathcal{R}}) \\ \leq \frac{1}{2} \sum_{(j_{i_1}, \ldots, j_{i_{\mathsf{t}+1}}) \in [2^s]^{\mathsf{t}+1}} \left| \prod_{k=1}^{\mathsf{t}+1} \mathsf{Pr}[C_k \in \mathcal{P}_{i_k}^{j_{i_k}}] - \prod_{k=1}^{\mathsf{t}+1} \mathsf{Pr}[U_{i_k} \in \mathcal{P}_{i_k}^{j_{i_k}}] \right| \\ \leq \frac{1}{2} \cdot 2^s \cdot (c_s + 2^{-2s-1})^{\mathsf{t}-1} \end{array}$$

where the first inequality is due to the fact that we are bounding the statistical distance between the outputs of $\mathsf{Recover}(\cdot, \mathcal{R})$ using that of the inputs.

In the case when $\theta > 0$, let $\Theta \subset [\mathsf{n}]$ with $|\Theta| = \theta$ denote the set of shares that are arbitrarily tampered with by f. A simple observation to begin with is that if $\mathcal{R} \cap \Theta = \emptyset$, the arguments above for $\theta = 0$ case still go through without any modification. It is only when $\mathcal{R} \cap \Theta \neq \emptyset$, we need adjustments. The tampering at the shares in $\mathcal{R} \cap \Theta$ are not s-bounded. We will exclude these shares from the analysis and construct a new instance where we can apply Lemma 1. It suffices to bound the statistical distance for the worst-case scenario $\Theta \subset \mathcal{R}$.

We have used the fact that $\mathsf{Share}(0)_{\mathcal{R}}$ has the same distribution as $C \leftarrow \mathsf{MDS}[\mathsf{t}+1, \mathsf{t}, 2]_p$. Now conditioned on $\mathsf{Share}(0)_\Theta = \mathbf{w}$, for a constant vector $\mathbf{w} \in \mathbb{F}_p^\theta$, the distribution

$$\mathsf{Share}(0)_{\mathcal{R}} | (\mathsf{Share}(0)_\Theta = \mathbf{w}) \;\equiv\; C | (C_{\Theta'} = \mathbf{w}),$$

where Θ' denotes the corresponding indices in $[\mathsf{t}+1]$ of Θ in $\mathcal{R} \subset [\mathsf{n}]$. Using properties of linear MDS codes, the support of random variable $C|(C_{\Theta'} = 0^\theta)$ is a sub-code of $\mathsf{MDS}[\mathsf{t}+1, \mathsf{t}, 2]_p$ of dimension $\mathsf{t} - \theta$ and minimum distance 2. Since the components in Θ' of the codewords in this sub-code are all zeros, the components in $[\mathsf{t}+1] \backslash \Theta'$ form a linear code of a shorter length $\mathsf{t}+1-\theta$, which is also MDS. We then have

$$C_{[\mathsf{t}+1]\backslash\Theta'} | (C_{\Theta'} = 0^\theta) \;\equiv\; C',\; C' \leftarrow \mathsf{MDS}[\mathsf{t}+1-\theta, \mathsf{t}-\theta, 2]_p.$$

Finally, the support of $C|(C_{\Theta'} = \mathbf{w})$ is a coset of $\mathsf{MDS}[\mathsf{t}+1, \mathsf{t}, 2]_p$ with respect to the (sub-space) support of $C|(C_{\Theta'} = 0^\theta)$. There is a codeword $\mathbf{c}^{\mathbf{w}}$ of $\mathsf{MDS}[\mathsf{t}+1, \mathsf{t}, 2]_p$ such that

$$C_{[\mathsf{t}+1]\backslash\Theta'} | (C_{\Theta'} = \mathbf{w}) \;\equiv\; C' + \mathbf{c}^{\mathbf{w}}_{[\mathsf{t}+1]\backslash\Theta'},\; C' \leftarrow \mathsf{MDS}[\mathsf{t}+1-\theta, \mathsf{t}-\theta, 2]_p.$$

On the other hand, we trivialy have

$$U_{\mathcal{R}\backslash\Theta} | (U_\Theta = \mathbf{w}) \;\equiv\; U' + \mathbf{c}^{\mathbf{w}}_{[\mathsf{t}+1]\backslash\Theta'}, U' \leftarrow \mathbb{F}_p^{\mathsf{t}+1-\theta}.$$

Similar to the $\theta = 0$ case, the offset $\mathbf{c}^{\mathbf{w}}_{[\mathsf{t}+1]\backslash\Theta'}$ can be dropped when computing statistical distance. Invoking Lemma 1 on the code $\mathsf{MDS}[\mathsf{t}+1-\theta, \mathsf{t}-\theta, 2]_p$, we have

$$\mathsf{SD}\left((\mathrm{Tamper}^{f,\mathcal{R}}_{\mathsf{s}} | (\mathsf{Share}(\mathsf{s})_\Theta = \mathbf{w})); (\mathcal{D}_{f,\mathcal{R}} | (U_\Theta = \mathbf{w}) \right) \leq \frac{1}{2} \cdot 2^s \cdot (c_s + 2^{-2s-1})^{\mathsf{t}-\theta-1}.$$

The claim of the theorem follows from averaging over all $\mathbf{w} \in \mathbb{F}_p^\theta$.

In order to give some idea of what Theorem 3 implies, we consider a few extreme parameter settings. Since the constant c_s satisfies $c_s \leq 1 - 2^{-2s}$, the error bound in Theorem 3 becomes

$$\varepsilon = \frac{1}{2} \cdot 2^s \cdot (c_s + 2^{-2s-1})^{\mathsf{t}-\theta-1} \leq \frac{1}{2} \cdot 2^s \cdot (1 - 2^{-2s-1})^{\mathsf{t}-\theta-1}.$$

One extreme is the maximum state bound $s = \lfloor \log p - 1 \rfloor$ case. The above bound means that once we fix $s = \lfloor \log p - 1 \rfloor$ to a constant, the non-malleability error vanishes exponentially fast in $\mathsf{t} - \theta$, which implies non-malleability is possible even for state bounds close to $\log p - 1$ [3].

Another extreme in the opposite direction is the minimum state bound $s = 1$ case. We want to know how small can we choose t and p while still have reasonable indistinguishability error ε. We substitute concrete values and estimate that for a 10 bits prime p ($\log p = 10$), choosing $\mathsf{n} = p - 1$, $\mathsf{t} = 300$ (approximately $\mathsf{n}/3$) allows for $\varepsilon = 2^{-50}$ against up to $\theta = 125$ fully tampered shares.

Obtaining constant threshold or share size via Monte-Carlo

There are application scenarios that require certain parameters of the underlying secret sharing scheme to be fixed to a constant, without being affected by the choice of other parameters. We first discuss the *constant reconstruction threshold* scenario to complement the results in previous subsection, where the non-malleability error crucially relies on a big enough reconstruction threshold.

Intuitively, finding tighter bounds for (4) leads to more flexible choice of parameters. Note that the bounds proved in [BDIR21] hold for any linear MDS codes, which characterises the worst case performance of linear MDS codes. It is natural to ask whether there are some linear MDS codes that provide better bounds than others. This idea was exploited in [MPSW21,MNP+21] to yield *Monte-Carlo* constructions of linear secret sharing schemes. Such schemes contain a randomised step in the generation of the sharing algorithm (and corresponding recovering algorithm), after which they function the same way as deterministic schemes where the sharing algorithm and recovering algorithm are fixed from the beginning. The randomised process of generating the sharing algorithm (and corresponding recovering algorithm) is efficient, but there is no efficient way to verify whether the generated algorithms do provide the desired security guarantee. The meaningfulness of such schemes rely on the fact that a random choice successful leads to working algorithms with overwhelming probability (so one could as well skip the inefficient verification). The results of [MPSW21,MNP+21] are shown in the setting of large alphabet size ($\mathsf{n} = \mathit{poly}(\log p)$) and motivated by providing a better bound of (4) (for the majority of the respective set of linear MDS codes) while minimising the code dimension (reconstruction threshold). Let $\lambda = \log p$ be the security parameter. It is shown in [MPSW21] that if $\frac{\mathsf{k}}{\mathsf{n}} > \frac{1}{2}$, a random linear $[\mathsf{n}, \mathsf{k}, \mathsf{n} - \mathsf{k} + 1]_p$ MDS code admits an exponentially (in λ) small bound for (4) with exponentially (in λ) small failure probability. The state-of-the-art results achieved by deterministic constructions only allow $\mathsf{k} = 0.85\mathsf{n}$ [BDIR21]. In order to prove results for constant reconstruction threshold k (for example $\mathsf{k} = 2$), by restricting the attack functions f in (4) to the *physical-bit* functions, where the partition is defined by the binary representation of the prime number p, it is shown in [MNP+21] that a

[3] Here the field size p and t are not independent, due to the fact that we are using an $[\mathsf{n}, \mathsf{t} + 1, \mathsf{n} - \mathsf{t}]_p$ MDS code in the construction.

random $[\mathsf{n}, \mathsf{k}, \mathsf{n}-\mathsf{k}+1]_p$ punctured Reed-Solomon code over finite field $\mathbb{F}_p$ admits an exponentially small bound with exponentially small failure probability (see Lemma 2 below).

An undesirable consequence of the Monte-Carlo nature of these constructions is that we can not directly combine it with the technique we used in previous subsection that exploits the difference between LR-SS and NM-SS. Assume we were to apply the results of [MPSW21] in combination with the technique in previous subsection, we should have the guarantee that for each reconstruction set, the non-malleability error is exponentially small except with exponentially small probability. But NM-SS requires non-malleability error to be negligible for all reconstruction sets simultaneously. A naive union bound argument seems to be not sufficient for keeping success probability close to 1, when the number of reconstruction sets is large, which is typically the case when $\mathsf{k} = \mathit{poly}(\log p)$. We then only discuss the implication of the results concerning the physical-bit attack functions, for which k can be chosen as small as a constant 2.

Lemma 2 ([MNP+21] Cor 3). *Let $0 < \delta < \ln 2$ be an arbitrary constant. Let $\mathsf{RS}[\mathsf{n}, \mathsf{k}; \mathbf{X}]_p$ denote a random $[\mathsf{n}, \mathsf{k}]$ punctured Reed-Solomon code over finite field $\mathbb{F}_p$ of prime order with evaluation places $\mathbf{X} \leftarrow (\mathbb{F}_p^*)^{\mathsf{n}}$. There exists a (slightly) super-linear function $P(\cdot, \cdot)$ such that the following holds. For any block length $\mathsf{n} \in \mathbb{N}$, code dimension $2 \leq \mathsf{k} \in \mathbb{N}$, physical-bit state bound $s \in \mathbb{N}$, and indistinguishability error parameter $\varepsilon = 2^{-\kappa}$, there exists $\lambda_0 = P(s\mathsf{n}/\mathsf{k}, \kappa/\mathsf{k})$ such that if the number of bits λ needed to represent the order of the prime field $\mathbb{F}_p$ satisfies $\lambda > \lambda_0$, then $C \leftarrow \mathsf{RS}[\mathsf{n}, \mathsf{k}; \mathbf{X}]_p$ is ε-indistinguishable from uniform by physical-bit s-bounded partitions with probability (over the randomness of choosing the evaluation places $\mathbf{X} \leftarrow (\mathbb{F}_p^*)^{\mathsf{n}}$) at least $1 - \exp(-\delta \cdot (\kappa - 1)\lambda)$.*

In particular, a function $P(s\mathsf{n}/\mathsf{k}, \kappa/\mathsf{k}) = \delta' \cdot \left(\frac{s\mathsf{n}}{\mathsf{k}} + \frac{\kappa}{\mathsf{k}}\right) \cdot \log^2\left(\frac{s\mathsf{n}}{\mathsf{k}} + \frac{\kappa}{\mathsf{k}}\right)$, for an appropriate universal positive constant δ', suffices.

Applying Lemma 2 to Theorem 3 instead of Lemma 1 gives NM-SS with respect to physical-bit bounded state tampering. With this extremely weak tampering (and relaxing from an explicit construction to a Monte-Carlo construction), the reconstruction threshold of the NM-SS can be an arbitrarily small constant independent of other parameters. Using the estimations from [MNP+21], for the extreme case privacy threshold $\mathsf{t} = 1$ (equivalently, reconstruction threshold $\mathsf{k} = 2$), and considering varying number of parties $\mathsf{n} = 10, 100$, and 1000, the non-malleability error as small as 2^{-50} against physical-bit 1-bounded state tampering can be achieved with success probability $1 - 2^{-50}$ (over choosing the evaluation places), using a prime number p with more than $\lambda = 430, 4800$, and 62000 bits, respectively.

There are MPC protocols that use a linear *ramp* secret sharing scheme with constant share size and unbounded number of shares, which can be constructed from Algebraic Geometric (AG) codes. A ramp secret sharing scheme allows a gap $\mathsf{g} = \mathsf{k} - \mathsf{t} > 1$ between the reconstruction threshold k and privacy threshold t. The Fourier analysis techniques applied to MDS codes was recently extended in [TX21] to provide bounds for (4) with respect to AG codes. Since we focus on

techniques that are applicable for most of the MPC protocols, we refer interested readers to read their work.

4 Applications in MPC-in-the-head Paradigm

We first recall the definitions of an outer protocol and an inner protocol in the MPC-in-the-head paradigm [IPS08].

The goal is a two-party computation protocol (between Alice and Bob) that computes an algebraic circuit C.

– *Outer protocol.* The outer protocol is an n-server MPC protocol Π evaluating C with the following protocol structure. The protocol Π proceeds in rounds where in each round each server sends messages to the other servers and updates its state by computing on its current state, and then also incorporates the messages it receives into its state. More concretely, each server $\overline{S}_i$ maintains a state

$$\overline{\Sigma}_i = (\overline{\sigma}_i, \overline{\mu}_{i\leftrightarrow A}, \overline{\mu}_{i\leftrightarrow B}),$$

where $\overline{\mu}_{i\leftrightarrow A}, \overline{\mu}_{i\leftrightarrow B}$ denote the collection of messages between client Alice and server $\overline{S}_i$, client Bob and server $\overline{S}_i$, respectively; and the rest of the state $\overline{\sigma}_i$ is called "non-local" part of the state. The next message function $\mathsf{nextMSG}_i^j$ in the jth round is evaluated on its current state and incoming messages, by server S_i, to update its state and compute the next messages $\overline{u}_{i\to 1}^j, \ldots, \overline{u}_{i\to n}^j$ to be sent to other servers:

$$\begin{aligned}\mathsf{nextMSG}_i^j &\left(\overline{\sigma}_i^{j-1}, \overline{\mu}_{i\leftrightarrow A}^{[j-1]}, \overline{\mu}_{i\leftrightarrow B}^{[j-1]}; \overline{\mu}_{i\leftarrow A}^j, \overline{\mu}_{i\leftarrow B}^j; \overline{u}_{i\leftarrow 1}^{j-1}, \ldots, \overline{u}_{i\leftarrow n}^{j-1}\right) \\ &= \left(\overline{\sigma}_i^j, \overline{\mu}_{i\leftrightarrow A}^{[j]}, \overline{\mu}_{i\leftrightarrow B}^{[j]}; \overline{u}_{i\to 1}^j, \ldots, \overline{u}_{i\to n}^j\right).\end{aligned}$$

– *Inner protocol.* The action of the abstract server $\overline{S}_i$ above is in fact emulated by Alice and Bob running an inner protocol to compute the functionality $\mathcal{G}_i^j$. Alice has

$$\left(\mathsf{Sh}_A(\overline{\sigma}_i^{j-1}), \overline{\mu}_{i\leftrightarrow A}^{[j-1]}; \overline{\mu}_{i\leftarrow A}^j; \mathsf{Sh}_A(\overline{u}_{i\leftarrow 1}^{j-1}), \ldots, \mathsf{Sh}_A(\overline{u}_{i\leftarrow n}^{j-1})\right).$$

Bob has

$$\left(\mathsf{Sh}_B(\overline{\sigma}_i^{j-1}), \overline{\mu}_{i\leftrightarrow B}^{[j-1]}; \overline{\mu}_{i\leftarrow B}^j; \mathsf{Sh}_B(\overline{u}_{i\leftarrow 1}^{j-1}), \ldots, \mathsf{Sh}_B(\overline{u}_{i\leftarrow n}^{j-1})\right).$$

The functionality $\mathcal{G}_i^j$ takes inputs from Alice and Bob, and proceeds by first reconstructing $\overline{\sigma}_i^{j-1}$ and $\overline{u}_{i\leftarrow 1}^{j-1}, \ldots, \overline{u}_{i\leftarrow n}^{j-1}$, and then evaluating a circuit defined by $\mathsf{nextMSG}_i^j$. The functionality divides each of the private values ($\overline{\sigma}_i^j$ and $\overline{u}_{i\to 1}^j, \ldots, \overline{u}_{i\to n}^j$) into two shares and outputs to Alice and Bob.
The inner protocol π^{OT} is an OT-hybrid two-party computation protocol that computes the collection of $\{\mathcal{G}_i^j\}_{i,j}$.

An *oblivious watch list* technique was then used to build a compiler out of Π and π^{OT} such that if Π is secure against active adversary in honest majority setting and π^{OT} is private against semi-honest adversary, then the compiled protocol is secure against active adversary. Intuitively, the semi-honest π^{OT} guarantees private communication between the virtual servers and the oblivious watch list technique enforces authenticated communication between the virtual servers, which collectively reduces the security of the compiled protocol to the security of Π with secure point-to-point channels.

We propose a general (abstract) framework of constructing non-malleable two-party computation with/without tampering of the assumed ideal functionality OT.

Theorem 4. *Let $\mathcal{T}$ be a tampering class for OT functionality. Let $\mathcal{F}_{\mathcal{T}}$ be the tampering class for the communication channels between virtual servers induced by executing π^{OT} under $\mathcal{T}$-tampered OT. Let Π be an NM-MPC with respect to $\mathcal{F}_{\mathcal{T}}$. There is a compiler for Π and semi-honest π^{OT} that gives a general purpose non-malleable two-party computation protocol with respect to $\mathcal{T}$-tampering of OT.*

The proof of Theorem 4 is straightforward and is omitted. Interesting tampering classes studied in the implementation security literature are those capturing powerful tampering adversaries against whom it is impossible to recover the ideal functionality. The types of imperfectness studied in the line of works on *OT combiners* [HKN+05] and more generally *OT extractors* [IKOS09] may serve as examples of weak tampering adversary (they are weak tampering because it is possible to extract ideal OT from the tampered version). Here we briefly discuss the special case of $\mathcal{T} = \emptyset$ (no tampering, OT is an ideal functionality). In this special case, non-malleability becomes privacy against malicious adversary in the OT-hybrid model. We argue that Theorem 4 provides a conceptually very simple way of achieving privacy against malicious adversary for general purpose two-party computation. The compiler can skip the oblivious watch list construction and naively compile Π and π^{OT}, which is amount to Alice and Bob running semi-honest π^{OT}. Intuitively, the emulation of the NM-MPC protocol Π (with respect to $\mathcal{F}_{\emptyset}$-tampering) servers as an encoding of the circuit C (in the sense of [GIP+14]) such that semi-honestly evaluate the encoded circuit is private against malicious adversary. Note that we do not have explicit construction of NM-MPC Π (with respect to $\mathcal{F}_{\emptyset}$-tampering) computing a circuit C and it is not clear if this approach yields efficient protocols.

5 Conclusion

We extended the non-malleability notion in tamper-resilient *storage* to tamper-resilient *computation* and defined Non-Malleable Multi-Party Computation (NM-MPC) using standard ideal/real world formulation: the ideal world adversary is allowed to tamper with the output of the trusted party, yielding a way to relax correctness of secure computation in a privacy-preserving way. For MPC protocols in honest majority setting, where efficient constructions with full security

assuming secure point-to-point channels are well understood and no security is known without the assumption, we showed non-malleability is achievable when the assumed channels are severely tampered. For MPC protocols in no honest majority setting, where weak secure computation notions play important roles, we discussed the implications of NM-MPC in honest majority setting in two-party computation, via MPC-in-the-head paradigm.

References

ADN+19. Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João L. Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures. In *Advances in Cryptology - CRYPTO*, pages 510–539, 2019.

AGM+15a. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology - CRYPTO 2015*, pages 538–557, 2015.

AGM+15b. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In *Theory of Cryptography Conference, TCC 2015*, pages 375–397, 2015.

AL17. Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptol.*, 30(1):58–151, 2017.

BDIR18. Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *Advances in Cryptology - CRYPTO 2018*, pages 531–561, 2018.

BDIR21. Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. *J. Cryptol.*, 34(2):10, 2021.

BDL01. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptol.*, 14(2):101–119, 2001.

Bea89. Donald Beaver. Multiparty protocols tolerating half faulty processors. In *Advances in Cryptology - CRYPTO '89*, volume 435, pages 560–572, 1989.

BGJ+13. Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In *Advances in Cryptology - CRYPTO*, pages 316–334, 2013.

BGJK12. Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In *Symposium on Theory of Computing Conference, STOC*, pages 1235–1254, 2012.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

Bla79. George R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, 1979.

CCD88. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Symposium on Theory of Computing, STOC*, pages 11–19, 1988.

CDF+08. Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Advances in Cryptology - EUROCRYPT*, volume 4965, pages 471–488, 2008.

CDN15. Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

CG16. Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. *IEEE Trans. Information Theory*, 62(3):1097–1118, 2016.

CG17. Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. *J. Cryptology*, 30(1):191–241, 2017.

CM97. Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In *Advances in Cryptology - CRYPTO '97*, pages 292–306, 1997.

Des94. Yvo Desmedt. Threshold cryptography. In *Eur. Trans. Telecommun.*, volume 5, pages 449–458, 1994.

DH76. Whitfield Diffie and Martin E Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DKRS06. Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *Advances in Cryptology-CRYPTO 2006*, pages 232–250. Springer, 2006.

DN07. Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Advances in Cryptology - CRYPTO 2007*, volume 4622, pages 572–590, 2007.

DPW18. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

FGJ+19. Nils Fleischhacker, Vipul Goyal, Abhishek Jain, Anat Paskin-Cherniavsky, and Slava Radune. Interactive non-malleable codes. In *Theory of Cryptography TCC*, pages 233–263, 2019.

FV19. Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In *Advances in Cryptology - CRYPTO*, pages 448–479, 2019.

GIP+14. Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *Symposium on Theory of Computing, STOC*, pages 495–504, 2014.

GK18. Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 685–698, 2018.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Symposium on Theory of Computing STOC*, pages 218–229, 1987.

GPR16. Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1128–1141, 2016.

HKN+05. Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Advances in Cryptology - EUROCRYPT*, pages 96–113, 2005.

IKOS09. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Extracting correlations. In *Symposium on Foundations of Computer Science, FOCS*, number 261–270, 2009.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology - CRYPTO*, pages 572–591, 2008.

IPS09. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *Theory of Cryptography, TCC*, pages 294–314, 2009.

Li18. Xin Li. Pseudorandom correlation breakers, independence preserving mergers and their applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:28, 2018.

Mau92. Ueli M. Maurer. Protocols for secret key agreement by public discussion based on common information. In *Advances in Cryptology - CRYPTO '92*, volume 740, pages 461–470, 1992.

Mau93. Ueli M Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39(3):733–742, 1993.

MNP+21. Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Leakage-resilience of the shamir secret-sharing scheme against physical-bit leakages. In *Advances in Cryptology - EUROCRYPT*, pages 344–374, 2021.

MPSW21. Hemanta K. Maji, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Constructing locally leakage-resilient linear secret-sharing schemes. In *Advances in Cryptology - CRYPTO*, pages 779–808, 2021.

MR18. Payman Mohassel and Peter Rindal. Aby^3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 35–52, 2018.

RB89. Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Symposium on Theory of Computing, STOC*, pages 73–85, 1989.

Sha79. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

TX21. Ivan Tjuawinata and Chaoping Xing. Leakage-resilient secret sharing with constant share size. 2021.

Yao82. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *Foundations of Computer Science, FOCS*, pages 160–164, 1982.

Supplementary Materials

A Execution of MPC protocol under tampering

An MPC protocol Π for evaluating an arithmetic circuit C evaluates different types of gates in topological order determined by the design of the protocol and

the circuit structure. Each gate typically contains several rounds of communications and in each round [4] a party transmits one message to other parties. Here we enforce a sequential order for the rounds from beginning of the protocol till the end of the protocol. Most gates are evaluated by the n servers, except the input sharing gates, where the m clients also take part. We first formally describe the processing of one round evaluated by the n servers under tampering. Assume it is server S_i's turn to activate his/her next message function $\mathsf{nextMSG}_i^j$ in the jth round. The function $\mathsf{nextMSG}_i^j$ is evaluated to compute the next messages for server S_i to be sent to other servers

$$(m_{i,1}^j, \ldots, m_{i,\mathsf{n}}^j) \colon = \mathsf{nextMSG}_i^j(\mathbf{R}_i^j, \mathsf{view}_i^{\Pi}),$$

where $\mathbf{R}_i^j$ is the local randomness of the server S_i generated in the jth round and view_i^{Π} denotes the collection of the view of S_i up to this point. The tampering function f_i^j for the current transmitted messages of server S_i is applied and results in

$$\begin{aligned}&(\tilde{m}_{i,1}^j, \ldots, \tilde{m}_{i,i-1}^j, \tilde{m}_{i,i+1}^j, \ldots, \tilde{m}_{i,\mathsf{n}}^j) \colon = \\ &f_i^j \left(\mathsf{previousMSG}, (m_{i,1}^j, \ldots, m_{i,i-1}^j, m_{i,i+1}^j, \ldots, m_{i,\mathsf{n}}^j)\right),\end{aligned}$$

where $\mathsf{previousMSG}$ denotes the collection of messages transmitted before the current batch of messages. The n next message functions $\mathsf{nextMSG}_i^j$, $i \in [\mathsf{n}]$ are activated in certain order (for example, rushing adversary is assumed to be able to have all honest servers complete their transmissions before the corrupt servers). As the activation token traverses from party to party followed by the application of the corresponding tampering function, all parties update their views at the end of the jth round as follows.

$$\mathsf{view}_i^{\Pi} \colon = \left(\mathsf{view}_i^{\Pi} || \mathbf{R}_i^j || (\tilde{m}_{1,i}^j, \ldots, \tilde{m}_{i-1,i}^j, m_{i,i}^j, \tilde{m}_{i+1,i}^j, \ldots, \tilde{m}_{\mathsf{n},i}^j)\right),$$

where the local randomness $\mathbf{R}_i^j$ of the server S_i and the message $m_{i,i}^j$ that S_i keeps on his/her own are not tampered with. In the input sharing phase of an MPC protocol, the m clients provide the input in the form of secret shares to the servers. We model the input sharing phase to be interactive and multiple-round as in the rest of the protocol. The communication only happens between the clients and the servers (the servers do not communicate with each other and the clients do not communicate with each other). We still denote the tampering function for the messages transmitted by the server S_i by f_i^{j}, although the messages of S_i in this case are consist of m components intended for m clients. Let the tampering function for the messages transmitted by the client C_i be denoted by $f_{\mathsf{n}+i}^{\mathsf{j}}$ (we consider C_i as the $(\mathsf{n}+i)$th party). The messages transmitted by the client C_i are consist of n components intended for n servers.

[4] Here the term round is to be distinguished from its usual usage in honest majority MPC literature where it allows multiple senders to send messages simultaneously.

In general, the tampering of an MPC protocol with a total of r rounds can be described by the following sequence of tampering functions.

$$\left\{ \begin{bmatrix} \{f_i^{\mathsf{j}}\}_{i\in[\mathsf{n}]}, & j\text{th round is not a round of input sharing gate} \\ \{f_i^{\mathsf{j}}\}_{i\in[\mathsf{n}+\mathsf{m}]}, & j\text{th round is a round of input sharing gate} \end{bmatrix} \right\}_{j=1,\ldots,r}$$

The execution of an MPC protocol under corruption means there are a set $\mathcal{S}$ of $\mathtt{t}$ corrupt servers that the adversary can read their views real time and can make these servers coordinately modify their next message functions. When analysing the combined influence, through controlling the corrupt servers and through tampering with the transmitted messages, we sometimes assume that the adversary does not tamper with the channels of the corrupt servers. This is without loss of generality because the adversary can always make the corrupt servers further modify their next message functions in order to achieve what the adversary could have additionally achieved through tampering with the messages corrupt servers receive/send. In the sequel, we only need to consider the tampering of the communication between honest parties,

$$\left\{ \begin{bmatrix} \{f_i^{\mathsf{j}}\}_{i\in\bar{\mathcal{S}}}, & j\text{th round is not a round of input sharing gate} \\ \{f_i^{\mathsf{j}}\}_{i\in\bar{\mathcal{S}}\cup\{\mathsf{n}+1,\ldots,\mathsf{n}+\mathsf{m}\}}, & j\text{th round is a round of input sharing gate} \end{bmatrix} \right\}_{j=1,\ldots,r}$$

We next formulate these tampering functions into adversarial channels, one channel for each pair of honest parties. We first define the adversarial channels connecting honest servers. Let $f_{i,i'}^j$ denote the projection of f_i^j on the component corresponding to the message to be sent to the server $S_{i'}$. That is

$$\tilde{m}_{i,i'}^j = f_{i,i'}^j\left(\mathsf{previousMSG}, (m_{i,1}^j, \ldots, m_{i,i-1}^j, m_{i,i+1}^j, \ldots, m_{i,\mathsf{n}}^j)\right).$$

In the special case of independent tampering, we have

$$\tilde{m}_{i,i'}^j = f_{i,i'}^j\left(\mathsf{previousMSG}_{\{i,i'\}}, m_{i,i'}^j\right),$$

where $\mathsf{previousMSG}_{\{i,i'\}}$ denotes the subset of previous messages that are transmitted between the server S_i and the server $S_{i'}$. We collect the tampering functions that are applied to the messages transmitted between the server S_i and the server $S_{i'}$, and define a tampering function $f_{\mathsf{Ch}(i,i')}$ for the channel $\mathsf{Ch}(i,i')$.

$$f_{\mathsf{Ch}(i,i')} = \left\{(f_{i,i'}^j, f_{i',i}^j)\right\}_{j\text{th round is not a round of input sharing gate}},$$

where some of these message tampering functions could be empty if there is no message transmitted from the server S_i to the server $S_{i'}$ or from the server $S_{i'}$ to the server S_i. Similarly, we also define the channel tampering functions for the input sharing phase. The channel tampering function for the server S_i and the client $C_{i'}$ is

$$f_{\mathsf{Ch}(i,\mathsf{n}+i')} = \left\{(f_{i,\mathsf{n}+i'}^j, f_{\mathsf{n}+i',i}^j)\right\}_{j\text{th round is a round of input sharing gate}}.$$

B Linear-based protocol with illustration

Definition 12 ([GIP+14]). *Let* $\mathsf{n} = 2\mathsf{t}+1$ *and let* $(\mathsf{Share}^{\mathsf{n},\mathsf{t}}, \mathsf{Recover}^{\mathsf{n},\mathsf{t}})$ *be a redundant dense linear secret sharing scheme. An* n*-server* m*-client protocol* Π *for computing a single-output* m*-client circuit* $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$ *is said to be linear-based with respect to* $(\mathsf{Share}^{\mathsf{n},\mathsf{t}}, \mathsf{Recover}^{\mathsf{n},\mathsf{t}})$ *if* Π *has the following structure with* linear protocols *(defined immediately afterwards) as internal components.*

1. ***Setup phase.*** *During this phase all servers participate in some linear protocol* Π_{setup} *that gets no auxiliary inputs. At the end of this phase every server* S_i *holds a vector of shares* $\mathsf{setup}\mathbf{Sh}_i^{g^k}$ *for every multiplication gate* $g^k \in \mathcal{G}_\Pi^{\mathsf{mult}}$ *in* C*.*
2. ***Randomness generation phase.*** *During this phase all servers participate in some linear protocol* Π_{random} *that gets no auxiliary inputs. At the end of this phase every server* S_i *holds a share* $\mathsf{randSh}_i^{g^k}$ *for every randomness gate* $g^k \in \mathcal{G}_\Pi^{\mathsf{rand}}$ *in* C*.*
3. ***Input sharing phase.*** Π *processes every input gate* $g^k \in \mathcal{G}_\Pi^{\mathsf{input}}$ *in* C *belonging to a client as follows. The client shares its input* x *for* g^k *using* $(\mathsf{Share}, \mathsf{Recover})$ *and then sends each server* S_i *its corresponding share* $\mathsf{Sh}_i^{g^k}$*.*
4. ***Circuit evaluation phase.*** Π *computes* C *in stages. During the* k*-th stage in an honest execution, the* k*-th gate,* g^k*, inside* C *is evaluated (in some topological order) and at the end of the stage the servers hold a sharing of the output of* g^k *with a distribution induced by* Share*. The evaluation of each gate is done as follows.*
   (a) *If* g^k *is an addition gate with inputs* g^a *and* g^b*,* Π *evaluates* g^k *by having each server* S_i *sum its shares corresponding to the outputs of* g^a *and* g^b*. Similarly, for a subtraction gate,* S_i *subtracts its shares corresponding to the outputs of* g^a *and* g^b*. There is no communication during these rounds. Each server* S_i *holds a share* $\mathsf{Sh}_i^{g^k}$ *of the output of gate* g^k*.*
   (b) *If* g^k *is a multiplication gate with inputs* g^a *and* g^b*,* Π *evaluates* g^k *using some* n*-party linear protocol* Π_{mult} *such that the main inputs of the* i*-th server* S_i *to* Π_{mult} *are its shares* $\mathsf{Sh}_i^{g^a}$ *and* $\mathsf{Sh}_i^{g^b}$ *corresponding to the outputs of* g^a *and* g^b*. The auxiliary input of* S_i *to* Π_{mult} *is* $\mathbf{Sh}_i^{g^k}$*, which is the results of the setup phase associated with* g^k*. Each server* S_i *holds a share* $\mathsf{Sh}_i^{g^k}$ *of the output of gate* g^k*.*
5. *The protocol finishes before the output gate* g^{out} *in* C *is processed and each server* S_i *holds a share* $\mathsf{Sh}_i^{g^{\mathsf{out}}}$ *of the output of gate* g^{out}*.*

The notion of *linear protocols* is an abstraction of the internal components Π_{setup}, Π_{random} and Π_{mult} of linear-based MPC protocols.

Definition 13 ([GIP+14]). *An* n*-party protocol* Π *is said to be a linear protocol, over some finite field* $\mathbb{F}$ *if* Π *has the following properties.*

1. ***Inputs.*** *The input of every server* S_i *is a vector of field elements from* $\mathbb{F}$*. Moreover,* S_i*'s inputs can be divided into two distinct types, the main inputs and auxiliary inputs.*

2. ***Messages.*** *Recall that each message in* Π *is a vector of field elements from* $\mathbb{F}$*. We require that every message* $\mathbf{m}$ *of* Π*, sent by some server* S_i*, belongs to one of the following categories:*
   (a) $\mathbf{m}$ *is some fixed arbitrary function of* S_i*'s main inputs (and is independent of its auxiliary inputs).*
   (b) *every entry* m_j *of* $\mathbf{m}$ *is generated as some fixed linear combination of* S_i*'s auxiliary inputs and elements of previous messages received by* S_i*.*
3. *The output of every party* S_i *is a linear function of its incoming messages.*

Let $(\mathsf{Share}, \mathsf{Recover})$ be the Shamir secret sharing scheme. The semi-honest DN [DN07] n-server m-client protocol Π for computing a single-output m-client circuit $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_m} \to \mathbb{F}^u$, where $n = 2\mathsf{t}+1$ is given as follows ($\mathsf{double\text{-}random}(\cdot)$ and $\mathsf{random}(\cdot)$ are defined immediately afterwards).

1. **Setup phase.** During this phase all servers participate in the linear protocol $\Pi_{\mathsf{setup}} = \mathsf{double\text{-}random}(\ell_{\mathsf{mult}})$ in order to generate the randomness needed for evaluation of multiplication gates during the protocol, where ℓ_{mult} denotes the number of multiplication gates. At the end of this phase every server S_i holds a vector of shares $\mathsf{setup}\mathbf{Sh}_i^{g^k} = \left(r_i^{g^k}, R_i^{g^k}\right)$ for every multiplication gate $g^k \in \mathcal{G}_{\Pi}^{\mathsf{mult}}$ in C.
2. **Randomness generation phase.** During this phase all servers participate in the linear protocol $\Pi_{\mathsf{random}} = \mathsf{random}(\ell_{\mathsf{rand}})$ in order to generate the shares corresponding to the outputs of the randomness gates inside C, where ℓ_{rand} denotes the number of randomness gates. At the end of this phase every server S_i holds a share $\mathsf{randSh}_i^{g^k} = r_i^{g^k}$ for every randomness gate $g^k \in \mathcal{G}_{\Pi}^{\mathsf{rand}}$ in C.
3. **Input sharing phase.** Π processes every input gate $g^k \in \mathcal{G}_{\Pi}^{\mathsf{input}}$ in C belonging to a client as follows. The client shares its input x for g^k using $(\mathsf{Share}, \mathsf{Recover})$ and then sends each server S_i its corresponding share $\mathsf{Sh}_i^{g^k}$.
4. **Circuit evaluation phase.** Π computes C in stages. During the k-th stage in an honest execution, the k-th gate, g^k, inside C is evaluated (in some topological order) and at the end of the stage the servers hold a sharing of the output of g^k with a distribution induced by Share. The evaluation of each gate is done as follows.
   (a) If g^k is an addition gate with inputs g^a and g^b, Π evaluates g^k by having each server S_i sum its shares corresponding to the outputs of g^a and g^b. Similarly, for a subtraction gate, S_i subtracts its shares corresponding to the outputs of g^a and g^b. There is no communication during these rounds. Each server S_i holds a share $\mathsf{Sh}_i^{g^k}$ of the output of gate g^k.
   (b) If g^k is a multiplication gate with inputs g^a and g^b, Π evaluates g^k using the following n-party linear protocol Π_{mult} as follows. The main inputs of the i-th server S_i to Π_{mult} are its shares $\mathsf{Sh}_i^{g^a}$ and $\mathsf{Sh}_i^{g^b}$ corresponding to the outputs of g^a and g^b. The auxiliary input of S_i to Π_{mult} is $\mathsf{setup}\mathbf{Sh}_i^{g^k} = \left(r_i^{g^k}, R_i^{g^k}\right)$, which is the results of the setup phase associated with g^k. Server S_i then does the following.

i. Compute $\mathsf{Sh}_i = \mathsf{Sh}_i^{g^a} \cdot \mathsf{Sh}_i^{g^b} + R_i^{g^k}$ and send Sh_i to S_1.
ii. S_1 upon receiving the shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$ from all the servers computes $D = \mathsf{Recover}_{[n]}(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$ and sends D to all the servers.
iii. Each server S_i upon receiving a value D from S_1 computes $\mathsf{Sh}_i^{g^k} = D - r_i^{g^k}$.

Note that $\mathsf{Share}(D) = (D, \ldots, D)$ when the random polynomial chosen by the share algorithm has zero coefficients for all positive degree terms. Each server S_i then holds a share $\mathsf{Sh}_i^{g^k}$ of the output of gate g^k.

5. The protocol finishes before the output gate g^{out} in C is processed and each server S_i holds a share $\mathsf{Sh}_i^{g^{\mathsf{out}}}$ of the output of gate g^{out}.

Definition 14. *Let $\mathbb{F}$ be a finite field. Let $M \in \mathbb{F}^{r \times c}$ be a matrix with the number r of rows bigger than the number c of columns. The matrix M is said to be super invertible if any sub-matrix formed by c rows of M is invertible.*

Let $M \in \mathbb{F}^{(\mathsf{t}+1) \times n}$ be a super invertible matrix and let $(\mathsf{Share}, \mathsf{Recover})$ be the Shamir secret sharing scheme. The following n-party protocol $\Pi_{setup} = \mathsf{double\text{-}random}(\ell_{\mathsf{mult}})$ on input ℓ_{mult}, each party P_i performs the following steps $\lceil \frac{\ell_{\mathsf{mult}}}{\mathsf{t}+1} \rceil$ times:

1. Generate a uniformly random value $\mathsf{s}^i \in \mathbb{F}$.
2. Compute $(\mathsf{Sh}_1^i, \ldots, \mathsf{Sh}_n^i) \leftarrow \mathsf{Share}(\mathsf{s}^i, t, n)$.
3. Compute $(\mathsf{Sh}_1'^i, \ldots, \mathsf{Sh}_n'^i) \leftarrow \mathsf{Share}(\mathsf{s}^i, 2t, n)$.
4. Send each party P_j the shares $(\mathsf{Sh}_j^i, \mathsf{Sh}_j'^i)$.
5. Upon receiving from all the parties the shares $(\mathsf{Sh}_i^1, \ldots, \mathsf{Sh}_i^n)$ and $(\mathsf{Sh}_i'^1, \ldots, \mathsf{Sh}_i'^n)$, the party P_i performs the following:
   (a) Compute $(r_i^1, \ldots, r_i^{\mathsf{t}+1}) = M(\mathsf{Sh}_i^1, \ldots, \mathsf{Sh}_i^n)$.
   (b) Compute $(R_i^1, \ldots, R_i^{\mathsf{t}+1}) = M(\mathsf{Sh}_i'^1, \ldots, \mathsf{Sh}_i'^n)$.
6. The output of the i-th party P_i is $\big((r_i^1, R_i^1), \ldots, (r_i^{\mathsf{t}+1}, R_i^{\mathsf{t}+1})\big)$.

The n-party protocol $\Pi_{random} = \mathsf{random}(\ell_{\mathsf{rand}})$ is the same as $\mathsf{double\text{-}random}(\ell_{\mathsf{mult}})$ except that Steps 3 and 5(b) are skipped.

C Secure with abort MPC protocol of [GIP+14]

Definition 15. *Let $\mathbb{F}$ be a finite field and let $\mathcal{G} = \{+, -, \times\}$, where gates labeled by $+$, $-$ and $\times$ indicate the addition, subtraction and multiplication over $\mathbb{F}$, respectively. An arithmetic circuit C over the gate set $\mathcal{G}$ and a set of variables $\mathcal{X} = \{x_1, \ldots, x_{\mathsf{m}}\}$ is a directed acyclic graph whose vertices are called gates and whose edges are called wires. Every gate in C of in-degree 0 is labeled by a variable from $\mathcal{X}$ and is referred to as an input gate. Every gate of out-degree 0 is called an output gate. All other gates are labeled by functions from $\mathcal{G}$. In some cases we also allow in-degree 0 gates labeled by rand and referred to as randomness gates. A circuit containing rand gates is called a randomized circuit and a circuit that does not contain rand gates is called a deterministic circuit. We*

write $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$ *to indicate that* C *is an arithmetic circuit over* $\mathbb{F}$ *with* m *inputs and one single output. We denote by* $|\mathsf{C}|$ *the number of gates in* C*. For an input* $\mathbf{x} \in \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}}$ *we denote by* $\mathsf{C}(\mathbf{x})$ *the result of evaluating* C *on* $\mathbf{x}$ *if* C *is deterministic and the resulting distribution if* C *is randomised.*

An *additive attack* $\mathcal{A}$ on a deterministic or randomised circuit C assigns an element of $\mathbb{F}^u$ to each of its internal wires as well as to each of its outputs. We denote by $\mathcal{A}_{u,v}$ the attack $\mathcal{A}$ restricted to the wire (u, v). For every wire (u, v) in C, the value $\mathcal{A}_{u,v}$ is added to the output of u before it enters the inputs of v. Similarly we denote by $\mathcal{A}_{\mathsf{out}}$ the restriction of $\mathcal{A}$ to the outputs of C and the value $\mathcal{A}_{\mathsf{out}}$ is added to the outputs of C. For simplicity, we assume $u_1 = \ldots = u_\mathsf{m} = u$.

Definition 16 (Additively corruptible version of a circuit). *Let* $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$ *be a circuit containing* w *wires. The additively corruptible version of* C *is the functionality*

$$\tilde{f}_\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \times \mathbb{F}^{wu} \to \mathbb{F}^u$$

that takes additional input from the adversary $\mathcal{A}$ *which indicates an additive corruption for every wire of* C*. For all* $(\mathbf{x}, \mathcal{A})$*,* $\tilde{f}_C(\mathbf{x}, \mathcal{A})$ *outputs the result of the additively corrupted* C *as specified by the additive attack* $\mathcal{A}$ *when invoked on the inputs* $\mathbf{x}$*.*

Definition 17 ([GIP+14]). *A randomised circuit* $\hat{\mathsf{C}}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$ *is an* ε*-secure implementation of a function* $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$ *against additive attacks if the following holds:*

- *Completeness. For all* $\mathbf{x} \in \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}}$*, it holds that* $\mathsf{Pr}[\hat{\mathsf{C}}(\mathbf{x}) = \mathsf{C}(\mathbf{x})] = 1$*.*
- *Additive-attack security. For any circuit* $\tilde{\mathsf{C}}$ *obtained by subjecting* $\hat{\mathsf{C}}$ *to an additive attack, there exists* $\mathbf{a} \in \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}}$ *and a distribution* $\mathbf{A}$ *over* $\mathbb{F}^u$ *such that for any* $\mathbf{x} \in \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}}$*, it holds that*

$$\mathsf{SD}\left(\hat{\mathsf{C}}(\mathbf{x}); \mathsf{C}(\mathbf{x} + \mathbf{a}) + \mathbf{A}\right) \leq \varepsilon.$$

The definition naturally extends to the case when the functionality computed by C *is randomised.*

Lemma 3 ([GIP+14]). *For any finite field* $\mathbb{F}$ *and arithmetic circuit* $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$*, there exists a randomised circuit* $\hat{\mathsf{C}}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_\mathsf{m}} \to \mathbb{F}^u$ *of size* $O(|\mathsf{C}|)$ *such that* $\hat{\mathsf{C}}$ *is* $O(|\mathsf{C}|/|\mathbb{F}|)$*-secure implementation of* C *against additive attacks.*

The following private version of AMD code [CDF+08] is due to [GIP+14]

Definition 18. *An* (u, u', ε)*-AMD code is a pair of circuits* $(\mathsf{AMDEnc}, \mathsf{AMDDec})$*, where* $\mathsf{AMDEnc}\colon \mathbb{F}^u \to \mathbb{F}^{u'}$ *is randomised and* $\mathsf{AMDDec}\colon \mathbb{F}^{u'} \to \mathbb{F} \times \mathbb{F}^u$ *is deterministic such that the following properties hold:*

- *Perfect completeness. For all* $\mathbf{x} \in \mathbb{F}^u$*, it holds that* $\mathsf{Pr}[\mathsf{AMDDec}(\mathsf{AMDEnc}(\mathbf{x})) = (0, \mathbf{x})] = 1$*.*

– *Additive robustness. For any* $\Delta \in \mathbb{F}^{u'}$, $\Delta \neq 0^{u'}$, *and for any* $\mathbf{x} \in \mathbb{F}^u$, *it holds that*

$$\mathsf{Pr}[\mathsf{AMDDec}(\mathsf{AMDEnc}(\mathbf{x} + \Delta)) \notin \mathsf{ERR}] \leq \varepsilon,$$

where $\mathsf{ERR} = \mathbb{F}^* \times \mathbb{F}^u$ *denotes detection of error.*

Moreover, $(\mathsf{AMDEnc}, \mathsf{AMDDec})$ *is called a private* (u, u', ε)*-AMD code if for any* $\Delta \in \mathbb{F}^{u'}$, $\Delta \neq 0^{u'}$, $\mathbf{y} \in \mathbb{F}^* \times \mathbb{F}^u$ *and for any* $\mathbf{x}^0, \mathbf{x}^1 \in \mathbb{F}^u$, *it holds that*

$$\begin{aligned}&\mathsf{Pr}[\mathsf{AMDDec}(\mathsf{AMDEnc}(\mathbf{x}^0 + \Delta)) = \mathbf{y} | \mathsf{AMDDec}(\mathsf{AMDEnc}(\mathbf{x}^0 + \Delta)) \in \mathsf{ERR}] \\ &\quad = \mathsf{Pr}[\mathsf{AMDDec}(\mathsf{AMDEnc}(\mathbf{x}^1 + \Delta)) = \mathbf{y} | \mathsf{AMDDec}(\mathsf{AMDEnc}(\mathbf{x}^1 + \Delta)) \in \mathsf{ERR}].\end{aligned}$$

Private AMD codes can be constructed from plain AMD codes $(\mathsf{AMDEnc}, \mathsf{AMDDec})$ through modifying its decoder as follows.

1. Compute $(b, \mathbf{z})$ from $\mathsf{AMDDec}(\mathbf{c})$.
2. Output $(0, \mathbf{z}) + b\mathbf{r}$, where $\mathbf{r}$ is generated uniformly from $\mathbb{F} \times \mathbb{F}^u$.

The above trick together with the known construction of asymptotically optimal constructions of AMD codes [DKRS06,CDF+08] immediately yields the following.

Lemma 4. *For any positive integers* u *and* σ, *there exists a pair of circuits* $(\mathsf{AMDEnc}, \mathsf{AMDDec})$ *s.t. for any finite field* $\mathbb{F}$ *it holds that* $(\mathsf{AMDEnc}, \mathsf{AMDDec})$ *is a private* $(u, O(u + \sigma), \frac{1}{|\mathbb{F}|^\sigma})$*-AMD code. Moreover, the size of* AMDEnc *and* AMDDec *is* $O(u + \sigma)$.

The construction of [GIP+14] (see Construction 1) starts with private MPC protocols and strengthens them for protection against the deviation of active Adv from the protocol. Intuitively, the linear-based protocols privately evaluate a circuit through sharing the inputs using the same linear secret sharing scheme (with independent randomness each time invoking the sharing algorithm) and, with the help of the homomorphic property of the secret sharing scheme, having the n servers operating on the secrets through operating on the shares. The homomorphic property of the underlying secret sharing also plays a crucial role in analysing what form of influence the deviation of malicious adversary has over the execution of these protocols, which are designed for privacy only. Our focus is the role of the assumption of secure point-to-point communication channels. A secure communication channel is both private and authenticated.

The authenticity of the communication channel guarantees that among the n shares for a secret held by a client or an honest server, $n - t$ shares are correctly received by honest servers, who, according to the definition of honest servers, will follow the protocol and correctly operate on these correct shares. Roughly speaking, since the $n - t$ shares out of the total n shares contain full information about the shared secret (honest majority), the execution of the protocol will not deviate from its course "too much", no matter how the corrupt servers may deviate from the protocol. In particular, the authenticity guarantee allows for an analysis that interprets the difference between what corrupt servers ought to do and what they actually do as a *blind additive* offset for the circuit. More

concretely, the analysis basically "ignore" the shares of the corrupt servers $\mathcal{S}$ and only consider the shares in $\bar{\mathcal{S}}$ (the honest servers). By doing this, the influence of Adv is limited to the gates that require communication among servers (usually the input gate and multiplication gate), where corrupt servers can influence the $\bar{\mathcal{S}}$ part through sending "wrong" messages to the honest servers $\bar{\mathcal{S}}$. For each corrupt server, the difference between the $n-t$ "wrong" shares and $n-t$ "correct shares" (if the server were to follow the protocol) determines an offset in the secret.

The privacy of the communication channel guarantees that only the t shares of $\mathcal{S}$ are seen by the adversary. According to the privacy of the secret sharing scheme, these t shares do not contain any information about the secret, and hence have a distribution independent of the other $n-t$ shares. This allows one to claim that the offset described above is only *blindly* chosen. Finally, the fact that the offset is *additive* follows from the homomorphic property we mentioned earlier, which is necessary for the protocol even there is no adversary.

Upon interpreting the influence of a deviation of active Adv as a set of blind additive offsets added to wires of the circuit being evaluated, one can seek a circuit protection solution to achieving robustness of MPC protocols. This solution takes two steps (see Construction 1, Circuit $\Pi(\mathsf{C})$ construction). The first step is to protect the inputs and the output of the circuit against additive attack using an AMD code. In particular, the encoding of the input and decoding of the output are done by clients locally while the decoding of the input (before computing) and encoding of the output (before outputting the result to the receiver client) are processed collectively by the n servers through privately evaluating some augmenting gates of the circuit. The second step is to compile the augmented circuit in the precious step into a protected version such that any blind additive corruption of the wires is turned into additive attacks at the input and output only (see Definition 17).

D Proof of Theorem 2

To combine the NM-SS over prime fields with other building blocks, we need a $\mathbb{F}_p$-linear $\mathcal{F}^s_{\mathsf{SS},\theta}$-NM-SS over an extension field $\mathbb{F}_{p^u}$.

Theorem 5. *Let p be a prime number and $s < \log p$. There is a $\mathbb{F}_p$-linear $\mathcal{F}^s_{\mathsf{SS},\theta}$-NM-SS over $\mathbb{F}^u_p$ with privacy threshold t and non-malleability error $\varepsilon = \frac{u}{2} \cdot \frac{1}{2} \cdot 2^s \cdot (c_s + 2^{-2s-1})^{\mathsf{t}-\theta-1}$, where $c_s = \frac{2^s \sin(\pi/2^s)}{p \sin(\pi/p)}$.*

Proof. Let $(\mathsf{Share}, \mathsf{Recover})$ be Shamir's secret sharing scheme over $\mathbb{F}_p$ with privacy threshold t for n players. We construct $(u\text{-}\mathsf{Share}, u\text{-}\mathsf{Recover})$ over $\mathbb{F}^u_p$.

- u-$\mathsf{Share}()$: On the input of a secret $(\mathsf{s}_1, \ldots, \mathsf{s}_u) \in \mathbb{F}^u_p$, the sharing algorithm generate, using fresh independent randomness,

$$\begin{cases} \mathsf{Share}(\mathsf{s}_1) = (\mathsf{Sh}^1_1, \ldots, \mathsf{Sh}^1_n); \\ \qquad \vdots \\ \mathsf{Share}(\mathsf{s}_u) = (\mathsf{Sh}^u_1, \ldots, \mathsf{Sh}^u_n). \end{cases}$$

The output of the sharing algorithm is then

$$u\text{-}\mathsf{Share}(\mathsf{s}_1,\ldots,\mathsf{s}_u) = \left((\mathsf{Sh}_1^1\ldots\mathsf{Sh}_1^u),\ \ldots\ ,(\mathsf{Sh}_n^1\ldots\mathsf{Sh}_n^u)\right).$$

– u-$\mathsf{Recover}(,)$: On the input of a reconstruction set $\mathcal{R} = \{i_1,\ldots,i_{\mathsf{t}+1}\}$ and the shares of the corresponding players, the reconstruction algorithm does the following.

$$\begin{cases} \mathsf{Recover}\left((\tilde{\mathsf{Sh}}_{i_1}^1,\ldots,\tilde{\mathsf{Sh}}_{i_{\mathsf{t}+1}}^1),\mathcal{R}\right) = \tilde{\mathsf{s}}_1; \\ \qquad\qquad\vdots \\ \mathsf{Recover}\left((\tilde{\mathsf{Sh}}_{i_1}^u,\ldots,\tilde{\mathsf{Sh}}_{i_{\mathsf{t}+1}}^u),\mathcal{R}\right) = \tilde{\mathsf{s}}_u. \end{cases}$$

The output of the reconstruction algorithm is $(\tilde{\mathsf{s}}_1,\ldots,\tilde{\mathsf{s}}_u)$.

The correctness and privacy of (u-Share, u-$\mathsf{Recover}$) follows straightforwardly by construction. So is the fact that the scheme is $\mathbb{F}_p$-linear. We now show the non-malleability of the scheme with respect to $\mathcal{F}^s_{\mathsf{SS},\theta}$.

We begin with the observation that a tampering function f over $\mathbb{F}_p^u$ induces u *random* tampering functions over $\mathbb{F}_p$ in a natural sense. Let $\mathsf{Range}(f) = \{\tilde{\mathbf{c}}^1,\ldots,\tilde{\mathbf{c}}^{2^s}\}$, where $\tilde{\mathbf{c}}^j = (\tilde{c}_1^j,\ldots,\tilde{c}_u^j) \in \mathbb{F}_p^u$ for $j = 1,\ldots,2^s$. For each (say the ith) of the u components of a vector in $\mathbb{F}_p^u$, f will only map it to at most 2^s different values ($\tilde{c}_i^j$, $j = 1,\ldots,2^s$). The tricky part is that the tampering function on the ith component induced by f can depend on the other input components. This fortunately does not create problem for using the non-malleability of the ($\mathsf{Share}, \mathsf{Recover}$) to argue the non-malleability of the ith component of the secret. The idea is since each component of the secret is shared using fresh independent randomness, of all the u components in one share of (u-Share, u-$\mathsf{Recover}$), only the ith component is chosen according to the randomness that is used for sharing the ith component of the secret; all other components in that share are independent. We then can interpret this tampering on the ith component of a share as a random s-bounded tampering with randomness coming from a distribution that is independent of the randomness that generates the ith component of the share. The non-malleability of the ($\mathsf{Share}, \mathsf{Recover}$) asserts that the tampered version $\tilde{\mathsf{s}}_i$ of the ith component of secret is independent of the original value s_i. A second tricky part is then to assert that the tampered version $\tilde{\mathsf{s}}_i$ of the ith component of the secret is independent of the the original values of other components $\{\mathsf{s}_j | j \neq i\}$ of the secret. We have seen the value of the tampered ith component of a share can depend on other components of the share. But the decision of j index of $\tilde{c}_i^j$ induces a partition of $\mathbb{F}_p^{u-1}$. All the possible influence that other components of the share have over the tampering of ith component of a share is captured in this induced partition of $\mathbb{F}_p^{u-1}$. Starting from the basis case of $u = 2$, we can use Lemma 1 to directly claim that this induced partition is independent of the other components of the secret. Building on this basis case of $u = 2$, we can argue by mathematical induction that this independence holds for any u.

Proof (Theorem 2). The construction closely follows the secure with abort MPC construction of [GIP+14]. We introduce a *virtual output extractor* to simplify the proof (this also captures the type of applications where the output remains in distributed form as shares of the underlying secret sharing scheme).

Construction 1 *Let* $\mathsf{C}\colon \mathbb{F}^{u_1} \times \ldots \times \mathbb{F}^{u_m} \to \mathbb{F}^{u}$ *be an* m*-client circuit. Let* Π^{Priv} *be a linear-based* t*-private* m*-client* n*-party protocol with respect to a redundant dense secret sharing scheme* $(\mathsf{NMShare}, \mathsf{NMRecst})$ *that is* $\mathcal{F}^{s}_{\mathsf{SS},\theta}$*-NM with error parameter* $\varepsilon_{\mathsf{NM}}$*. Let* $(\mathsf{AMDEnc}, \mathsf{AMDDec})$ *be an* $(u, u', \varepsilon_{\mathsf{AMD}})$*-AMD code. We construct a protocol* Π *for computing* C*.*

- *Circuit* $\Pi(\mathsf{C})$ *construction.*
  1. Augmented circuit $\mathsf{C}^{\mathsf{AMD}}$ for enabling input/output detection.
     Modify C *into the randomised* m*-client circuit* $\mathsf{C}^{\mathsf{AMD}}\colon \mathbb{F}^{u'} \times \ldots \times \mathbb{F}^{u'} \to \mathbb{F} \times \mathbb{F}^{u'}$ *that on inputs* $(x'_1, \ldots, x'_m)$ *performs the following:*
     (a) For all $1 \le i \le m$*, compute* $(b_i, x_i) \leftarrow \mathsf{AMDDec}(x'_i)$*.*
     (b) Compute $b \leftarrow \sum_{i=1}^{m} r_i b_i$*, where* r_i *is a random field element.*
     (c) Output $(b, \mathsf{AMDEnc}(\mathsf{C}(x_1, \ldots, x_m)) + br')$*.*
  2. Compiled circuit $\hat{\mathsf{C}}^{\mathsf{AMD}}$ secure against additive corruptions.
     Compile the augmented circuit $\mathsf{C}^{\mathsf{AMD}}$ *into* $\hat{\mathsf{C}}^{\mathsf{AMD}}$ *according to Lemma 3 to turn any additive circuit corruption into additive attack to circuit input/out.*
- *Protocol* Π*.*
  1. *Client side local pre-computation:*
     (a) Each client C_i *locally computes* $x'_i \leftarrow \mathsf{AMDEnc}(x_i)$*.*
     (b) Each client C_i *sends its encoded input* x'_i *to protocol* Π^{Priv}*.*
  2. *Circuit evaluation:*
     (a) Run Π^{Priv} *on inputs* $(x'_1, \ldots, x'_m)$ *evaluating* $\Pi(\mathsf{C}) = \hat{\mathsf{C}}^{\mathsf{AMD}}$*.*
     (b) Each server S_i *exits* Π^{Priv} *with a share* $\mathsf{Sh}_i^{g^{\mathsf{out}}}$ *of the output gate of* $\Pi(\mathsf{C}) = \hat{\mathsf{C}}^{\mathsf{AMD}}$*.*

  Output extractor (virtual)
  1. *On input a reconstruction set* $\mathcal{R} \subset [n]$ *of the underlying secret sharing scheme of* Π^{Priv}*, reconstruct the secret* (b, z') *from* $\mathsf{Sh}_{\mathcal{R}}^{g^{\mathsf{out}}}$*.*
  2. *If* $b \neq 0$*, then output aborts.*
  3. *Otherwise, compute* $(b', y) \leftarrow \mathsf{AMDDec}(z')$*.*
  4. *If* $b' \neq 0$*, then output aborts.*
  5. *Otherwise, output* y*.*

The fact that Π (t, ε)-securely computes C with abort was shown in [GIP+14]. In the sequel, we prove non-malleability of Π against channel tampering using the non-malleability of the underlying secret sharing.

We start with an observation that greatly simplifies the construction of non-malleability simulator (the simulation of the honest party's output) for Π. This observation is in fact a consequence of non-malleability of secret sharing and the definition of $\mathcal{F}^{s}_{\mathsf{SS},\theta}$. It can be shown using an argument similar to [CG16, Theorem 5.3] that if a secret sharing scheme is non-malleable with respect to

$\mathcal{F}^s_{\mathsf{SS},\theta}$, then the tampering experiment $\mathsf{Tamper}^{f,\mathcal{R}}_{\mathsf{s}}$ of the secret sharing scheme is strictly independent of the secret s, for any tampering function $f \in \mathcal{F}^s_{\mathsf{SS},\theta}$ and any reconstruction set $\mathcal{R}$. This is slightly stronger than the plain NM-SS definition, where the tampering experiment $\mathsf{Tamper}^{f,\mathcal{R}}_{\mathsf{s}}$ is allowed to depend on the secret s if its simulation $\mathcal{D}_{f,\mathcal{R}}$ outputs the symbol same^* with non-zero probability.

Let $\mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$ be the set of gates in the circuit for computing C according to Π that require secure point-to-point channels in processing them. With the above observation, it follows that once the shares of a secret are tampered with using a function from $\mathcal{F}^s_{\mathsf{SS},\theta}$, non-malleability of $(\mathsf{NMShare}, \mathsf{NMRecst})$ guarantees that the information about the original secret is destroyed in the sense that given a reconstruction set $\mathcal{R}$, the new secret contained in the tampered shares corresponding to $\mathcal{R}$ can be simulated without knowing the original secret. This in particular means that one does not need to keep track of the output values (in general they are random variables with randomness from the randomised sharing algorithm $\mathsf{NMShare}$) of the erroneously evaluated intermediate gates, since they will not affect the output of $g^{\mathsf{out}} \in \Pi(\mathsf{C})$ if there is a gate $g^k \in \mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$ lying between g^{out} and the intermediate gates. Our non-malleability simulator for Π works in reverse topological order starting from the output gate $g^{\mathsf{out}} \in \Pi(\mathsf{C})$. Let $\mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$ be the set of gates in the circuit for computing C according to Π that require secure point-to-point channels in processing them. If the output gate $g^{\mathsf{out}} \in \mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$, then $\tilde{\mathsf{Sh}}^{g^{\mathsf{out}}}_{\mathcal{R}}$, which determines the value (distribution) of the output of the virtual output extractor, can be directly simulated using the non-malleability simulator of $(\mathsf{NMShare}, \mathsf{NMRecst})$ and the deviation of the corrupt servers. If $g^{\mathsf{out}} \notin \mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$, we need to look at the gates in $\mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$ that are evaluated before the output gate g^{out} and do not have gates in $\mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$ lying between g^{out} and them. Let us denote these gates by $\mathcal{G}^{\Pi(\mathsf{C}),\mathsf{last}}_{|||\vdash}$ and the vector of shares corresponding to these gates by $\tilde{\mathbf{Sh}}^{\mathcal{G}^{\Pi(\mathsf{C}),\mathsf{last}}_{|||\vdash}}_i$, for server S_i, $i \in \mathcal{R}$. The $\tilde{\mathsf{Sh}}^{g^{\mathsf{out}}}_{\mathcal{R}}$ can be simulated through computing for each server S_i in $\mathcal{R}$ a share $\tilde{\mathsf{Sh}}^{g^{\mathsf{out}}}_i$ from $\tilde{\mathbf{Sh}}^{\mathcal{G}^{\Pi(\mathsf{C}),\mathsf{last}}_{|||\vdash}}_i$. Indeed, for severs in $\mathcal{R}\backslash\mathcal{S}$, since these servers are honest, we compute $\tilde{\mathsf{Sh}}^{g^{\mathsf{out}}}_i$ from $\tilde{\mathbf{Sh}}^{\mathcal{G}^{\Pi(\mathsf{C}),\mathsf{last}}_{|||\vdash}}_i$ according to $\Pi(\mathsf{C})$. For severs in $\mathcal{R} \cap \mathcal{S}$, since these servers are controlled by Adv, we compute $\tilde{\mathsf{Sh}}^{g^{\mathsf{out}}}_i$ from $\tilde{\mathbf{Sh}}^{\mathcal{G}^{\Pi(\mathsf{C}),\mathsf{last}}_{|||\vdash}}_i$ according to how Adv would deviate from the protocol. Depending on the specific construction of Π^{Priv}, the set $\mathcal{G}^{\Pi(\mathsf{C}),\mathsf{last}}_{|||\vdash}$ can be different. We next explicitly describe our non-malleability simulator for Π.

1. Simulating shares $\tilde{\mathsf{Sh}}^{g^{\mathsf{out}}}_{\mathcal{R}}$ of the output gate g^{out}:
   - $g^{\mathsf{out}} \in \mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}$. Directly simulate
     (a) Read from $\mathbf{f}^{\bar{\mathcal{S}},\mathcal{G}^{\Pi(\mathsf{C})}_{|||\vdash}}$ the subset $\{f^{j,g^{\mathsf{out}}} | j \in \bar{\mathcal{S}}\}$ of tampering functions corresponding to the honest servers $\bar{\mathcal{S}}$ for the output gate g^{out} and, for each $j \in \bar{\mathcal{S}}$, call the non-malleability simulator of $(\mathsf{NMShare}, \mathsf{NMRecst})$ with tampering function $f^{j,g^{\mathsf{out}}}$ and reconstruction set $\mathcal{R}$.
        i. $\tilde{\mathsf{s}} \leftarrow \mathcal{D}_{f^{j,g^{\mathsf{out}}},\mathcal{R}}$

ii. $(\tilde{\mathsf{Sh}}_1^{j,g^{\mathsf{out}}}, \ldots, \tilde{\mathsf{Sh}}_n^{j,g^{\mathsf{out}}}) \leftarrow \mathsf{NMShare}(\tilde{\mathsf{s}})$

iii. output $\tilde{\mathsf{Sh}}_{\mathcal{R}\backslash\mathcal{S}}^{j,g^{\mathsf{out}}}$

(b) Read from Adv the messages that it instructs the corrupt servers $\mathcal{S}$ to send to the honest servers in $\mathcal{R}\backslash\mathcal{S}$ and the final share for output gate if the corrupt server is in $\mathcal{R} \cap \mathcal{S}$. For each $j \in \mathcal{S}$,

i. output $\tilde{\mathsf{Sh}}_{\mathcal{R}\backslash\mathcal{S}}^{j,g^{\mathsf{out}}}$

ii. if $j \in \mathcal{S} \cap \mathcal{R}$, output $\tilde{\mathsf{Sh}}_j^{g^{\mathsf{out}}}$

(c) For each $j \in \mathcal{R}\backslash\mathcal{S}$, compute according to the protocol for the gate g^{out} the share $\tilde{\mathsf{Sh}}_j^{g^{\mathsf{out}}}$ from the received messages $(\tilde{\mathsf{Sh}}_j^{1,g^{\mathsf{out}}}, \ldots, \tilde{\mathsf{Sh}}_j^{n,g^{\mathsf{out}}})$ obtained in steps (a) and (b).

– $g^{\mathsf{out}} \notin \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C})}$. Find $\mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C}),\mathsf{last}}$ according to construction of Π^{Priv} and simulate the followig.

- Π^{Priv} without setup phase (e.g. [BGW88]): the gates $\mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C})}$ that require communication to process are the multiplication gates and input gates.

  (a) Simulating shares $\tilde{\mathsf{Sh}}_{\mathcal{R}}^{g^k}$ for each $g^k \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C}),\mathsf{last}}$.

  * If $g^k \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C}),\mathsf{last}}$ is an input gate, similar to the "Directly simulate" steps described for $g^{\mathsf{out}} \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C})}$ special case above with simplification. The non-malleability simulator of $(\mathsf{NMShare}, \mathsf{NMRecst})$ is only called for once and the item (c) is empty.
  * If $g^k \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C}),\mathsf{last}}$ is a multiplication gate, same as the "Directly simulate" steps described for $g^{\mathsf{out}} \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C})}$ special case above.

  (b) Computing shares $\tilde{\mathsf{Sh}}_{\mathcal{R}}^{g^{\mathsf{out}}}$ of the output gate g^{out}.
  For each $j \in \mathcal{R}\backslash\mathcal{S}$, compute according to the protocol the share $\tilde{\mathsf{Sh}}_j^{g^{\mathsf{out}}}$ from the received messages in previous steps. Read from Adv the final share $\tilde{\mathsf{Sh}}_j^{g^{\mathsf{out}}}$ of a corrupt server j for output gate if the corrupt server is in $\mathcal{R} \cap \mathcal{S}$

- Π^{Priv} with setup phase (e.g.[DN07]): the gates $\mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C})}$ that require communication to process are the setup gates and input gates. Moreover, the input gates are lying between the output gate and setup gates.

  (a) Simulating shares $\tilde{\mathsf{Sh}}_{\mathcal{R}}^{g^k}$ for each $g^k \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C}),\mathsf{last}}$.
  $g^k \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C}),\mathsf{last}}$ is always an input gate, similar to the "Directly simulate" steps described for $g^{\mathsf{out}} \in \mathcal{G}_{|||\vdash}^{\Pi(\mathsf{C})}$ special case above with simplification. The non-malleability simulator of $(\mathsf{NMShare}, \mathsf{NMRecst})$ is only called for once and the item (c) is empty.

  (b) Computing shares $\tilde{\mathsf{Sh}}_{\mathcal{R}}^{g^{\mathsf{out}}}$ of the output gate g^{out}.
  For each $j \in \mathcal{R}\backslash\mathcal{S}$, compute according to the protocol the share $\tilde{\mathsf{Sh}}_j^{g^{\mathsf{out}}}$ from the received messages in previous steps. Read from Adv the final share $\tilde{\mathsf{Sh}}_j^{g^{\mathsf{out}}}$ of a corrupt server j for output gate if the corrupt server is in $\mathcal{R} \cap \mathcal{S}$

2. Virtual output extractor:
   (a) On input a reconstruction set $\mathcal{R} \subset [n]$ of the underlying secret sharing scheme of Π^{Priv}, reconstruct the secret (b, z') from $\tilde{\mathsf{Sh}}_{\mathcal{R}}^{g^{\mathsf{out}}}$.
   (b) If $b \neq 0$, then output aborts.
   (c) Otherwise, compute $(b', y) \leftarrow \mathsf{AMDDec}(z')$.
   (d) If $b' \neq 0$, then output aborts.
   (e) Otherwise, output y.