Reverse Firewalls for Adaptively Secure MPC without Setup

Suvradip Chakraborty[1] *, Chaya Ganesh[2], Mahak Pancholi[3], Pratik Sarkar[4] **,

[1] IST Austria
[2] IISc Bangalore
[3] Aarhus University
[4] Boston University

Abstract. We study Multi-party computation (MPC) in the setting of subversion, where the adversary tampers with the machines of honest parties. Our goal is to construct actively secure MPC protocols where parties are corrupted adaptively by an adversary (as in the standard adaptive security setting), and in addition, honest parties' machines are compromised.

The idea of reverse firewalls (RF) was introduced at EUROCRYPT'15 by Mironov and Stephens-Davidowitz as an approach to protecting protocols against corruption of honest parties' devices. Intuitively, an RF for a party $\mathcal{P}$ is an external entity that sits between $\mathcal{P}$ and the outside world and whose scope is to sanitize $\mathcal{P}$'s incoming and outgoing messages in the face of subversion of their computer. Mironov and Stephens-Davidowitz constructed a protocol for passively-secure two-party computation. At CRYPTO'20, Chakraborty, Dziembowski and Nielsen constructed a protocol for secure computation with firewalls that improved on this result, both by extending it to *multi*-party computation protocol, and considering *active* security in the presence of *static* corruptions.

In this paper, we initiate the study of RF for MPC in the *adaptive* setting. We put forward a definition for adaptively secure MPC in the reverse firewall setting, explore relationships among the security notions, and then construct reverse firewalls for MPC in this stronger setting of adaptive security. We also resolve the open question of Chakraborty, Dziembowski and Nielsen by removing the need for a trusted setup in constructing RF for MPC.

Towards this end, we construct reverse firewalls for adaptively secure augmented coin tossing and adaptively secure zero-knowledge protocols and obtain a constant round adaptively secure MPC protocol in the reverse firewall setting without setup. Along the way, we propose a new multi-party adaptively secure coin tossing protocol in the plain model, that is of independent interest.

* Suvradip Chakraborty received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (682815 - TOCNeT).

** Pratik Sarkar received funding from NSF grants 1931714, 1414119.

1 Introduction

The standard definitions of security in cryptographic protocols are under the assumption that honest parties can completely trust the machines that implement their algorithms. However, such an assumption may be unwarranted in the real world. The security guarantees of cryptosystems depend on the adversarial model which, however, often makes idealized assumptions that are not always realized in actual implementations. Several practical attacks in the real-world exploit *implementation* details of an algorithm rather then treating it as a "black-box". In addition, users may be forced to use hardware built by companies with expertise, and software that are mandated by standardization agencies. The capability of the adversary to "tamper" with the implementation is not captured by security models in classical cryptography. This model is not overkill, as we now know by Snowden [4] revelations that one of potential mechanisms for large scale mass surveillance is compromise of security by subversion of cryptographic standards, and tampering of hardware. The threat of an adversary modifying the implementation so that the subverted algorithm remains indistinguishable from the specification in black-box behavior, while leaking secrets was originally studied by Young and Yung as kleptography [33], and in the setting of subliminal channels by Simmons [32]. Since Snowden revelations brought to light actual deployment of such attacks, there is renewed attention, and has led cryptographers to model such tampering in the security definition in order to closely capture real-world concerns.

Reverse Firewalls. The *cryptographic reverse firewall* (RF) framework was introduced by Mironov and Stephens-Davidowitz [28] in the context of designing protocols secure against adversaries that can corrupt users' machines in order to compromise their security. A reverse firewall for a party $\mathcal{P}$ is an external intermediate machine that modifies the incoming and outgoing messages sent by $\mathcal{P}$'s machine. In essence, a reverse firewall sits between a party $\mathcal{P}$ and the external world, and "sanitizes" the messages that are sent and received by $\mathcal{P}$. Note that the party does not put any trust in the RF, meaning that it does not share any secrets with the firewall. This rules out trivial solutions like a trusted RF that simply keeps $\mathcal{P}$'s secrets and runs on $\mathcal{P}$'s behalf. Instead, the goal is for an uncorrupted[5] RF to provide meaningful security guarantees even in the case that an honest party's machine has been tampered with. Consider an arbitrary protocol that satisfies some notions of functionality and security. A reverse firewall for a protocol is said to *functionality-maintaining* if the resulting protocol (protocol with a firewall for party $\mathcal{P}$) achieves the same functionality as the original protocol. Roughly, the RF should not ruin the functionality of the underlying protocol, in the sense that the protocol with an RF for a party should still work as expected in case no subversion takes place. At the same time, the RF is expected to preserve security. An RF is said to preserve security if the protocol *with* the firewall is secure even when an honest party's implementation

[5] The RF being corrupt is not interesting in the active setting, since the corrupt RF and the other party together can be thought of as the adversary.

is tampered with to behave in an arbitrarily corrupt way. Finally, an RF should provide *exfiltration-resistance*, i.e., regardless of how the user's machine behaves, the presence of the RF will prevent the machine from leaking any information to the outside world.

The work of [28] provides a construction of a two-party passively secure computation protocol with a reverse firewall. The recent work of [12] generalizes reverse firewalls for secure computation by showing feasibility of reverse firewalls for Multi Party Computation (MPC). They give a construction of reverse firewalls for secure computation in a stronger and general setting that handles multiple parties, and consider protocols in the malicious security model.

RFs in other settings have been constructed including key exchange and secure message transmission [20, 14], oblivious transfer [20, 14], digital signatures [2], and zero-knowledge proofs (ZK) [23]. Reverse firewalls has also been used in a practical context in the design of *True2F* [16], a system that is based on a firewalled key generation and ECDSA signature generation with potential applications in cryptocurrency wallets.

1.1 Our Results

In this work, we take forward the study of reverse firewalls in the setting of MPC. We begin by proposing definitions that capture the requirements of an RF for MPC in the presence of *adaptive* corruptions. We then explore relationships among them the notions. Next, we turn our attention to constructing RFs for maliciously secure protocols in the presence of adaptive corruptions. Towards this end, we construct protocols with reverse firewalls for multi-party augmented coin tossing, zero-knowledge, and coin tossing, all in the presence of adaptive corruptions. We then use the above building blocks to construct a maliciously secure MPC in the presence of adaptive corruptions together with a reverse firewall. We further elaborate on the contributions in this work.

On the relationship between definitions. As our first contribution, we revisit the different notions of subversion security for MPC protocols in the presence of RF. The work of [28] defined the notions of *security preservation* (SP) and *exfiltration resistance* (ER) as the properties required from an RF. SP asks that an RF preserve the security properties of the underlying protocol for an honest party even when the honest party's implementation is tampered with. ER is concerned with a type of attack called exfiltration, where an honest party's tampered implementation attempts to leak secrets. A reverse firewall that is exfiltration resistant prevents an adversary from learning secrets even when the honest party's machine is tampered with. Roughly, exfiltration resistance for a party P_i asks that the transcripts produced in the following two ways are *indistinguishable*: (i) by running the protocol with the RF for P_i whose implementation has been arbitrarily subverted and in the presence of other malicious parties, (ii) by running the protocol with the RF for honest implementation of P_i in the presence of other malicious parties. In [28], it was shown that for certain indistinguishability-based security notions like semantic security

of an encryption scheme, an exfiltration resistant RF is also security preserving. It was postulated in [28] that, in general, when security requirements are simulation-based, ER *does not* imply SP. Surprisingly, we establish that exfiltration resistance *implies* security preservation for a reverse firewall when the security of the underlying protocol is simulation-based (computational) MPC security. For simulation-based security, this implication was only known for special functionalities like zero-knowledge. Our definitional implication shows that ER is the "right" notion for RF in the MPC setting; for new constructions, we need only construct RFs that are exfiltration resistant for each of the parties, and when all honest parties have an RF, security preservation for the protocol follows in the presence of malicious parties and arbitrary tampering of honest parties' implementations. In the other direction, [28] showed that a security preserving RF is not necessarily ER when the underlying security does not promise privacy.

Reverse firewalls for adaptively secure MPC. The adaptive security notion for an MPC protocol models the realistic threat that an adversary can corrupt a party during the execution of a protocol. Adaptive security is much harder to achieve than static security for MPC. In the reverse firewall setting, capturing a technical formulation of the adaptive security notion requires some care. When a party gets adaptively corrupted, the adversary can learn all of that party's inputs and internal random coins. Consider an MPC protocol where an honest party deploys a firewall; now adaptively corrupting this party amounts to the adversary learning the composed state of the party with its reverse firewall. Typically, for reverse firewalls, security preservation means that the underlying security properties hold even under subversion. In the adaptive security case, we ask that adaptive security holds under subversion, where the adaptive adversary can learn the *composed state* of an adaptively corrupt party. Defining exfiltration resistance in the adaptive case needs some care. Here, we ask that the adversary not be able to distinguish between a tampered implementation of party P and an honest implementation, where the adversary can specify tampered implementations for initially honest parties and corrupt parties adaptively in the execution. While exfiltration resistance is not meaningful anymore once P gets corrupt in the middle of the protocol, our definition asks that up *until the point* that P gets corrupted, exfiltration resistance hold. Intuitively, the definition says that if P gets corrupted in the middle of execution, the adversary can see the composed state of P (the state of P composed with the state of the RF). Even given this state, the adversary should not be able to say if until corruption it was interacting with P composed with RF or $\tilde{P}$ composed with RF, where $\tilde{P}$ is a tampered implementation for P.

We construct reverse firewalls for *maliciously* secure MPC protocols in the presence of *adaptive* corruptions in the plain model. Similar to [12], we consider RFs for functionality-maintaining tampering (see Sec. 3).

Theorem 1. *(Informal) Assuming DDH and LWE assumptions, there exists an $\mathcal{O}(1)$ round actively secure MPC protocol with reverse firewalls that is secure against adaptive corruptions in the* urs *model.*

Later, we generate (Thm. 3) the urs using an adaptively secure coin tossing protocol in the *plain* model based on the Discrete Logarithm (DLOG) and the Knowledge of Exponent (KEA) assumption in a different group. We consider this to be a result of independent interest, and further elaborate on the coin tossing protocol in the technical overview section.

Our approach is to construct an MPC protocol along the lines of GMW [26], and add reverse firewall to this protocol. That is, our construction is essentially an *adaptive compiler*: it takes a semi-honest adaptively secure MPC protocol and runs [26]-like steps in the reverse firewall setting to yield an adaptively secure MPC protocol with reverse firewalls. Towards this, we design adaptively secure protocols for augmented coin tossing and zero-knowledge, and construct reverse firewalls for each of the sub-protocols used in the compiler. Finally, we show that the compiled MPC protocol is adaptively secure in the presence of tampering of honest parties. We state each of the results below.

– *Reverse firewall for ZK:* Zero-knowledge in the presence of subversion have been studied in the form of parameter subversion for NIZK [5], and in the RF setting for a class of interactive protocols called malleable sigma protocols [23]. In this work, we consider interactive ZK since we aim for protocols without setup. Our protocol is a variant of the adaptively secure ZK protocol of [11] which is in the Uniform Random String (urs) model. Finally, we show how to design an RF for this protocol.

  Theorem 2. *(Informal) Assuming LWE, there exists a three round actively secure ZK protocol with reverse firewalls that is secure against adaptive corruption of parties in the* urs *model.*

– *Reverse firewall for augmented coin-tossing:* We provide a construction of an adaptively-secure multi-party augmented coin-tossing protocol. Similar to our ZK protocol, our augmented coin-tossing protocol is also in the urs model. The main building block of our augmented coin tossing protocol is an adaptively-secure commitment scheme (in the urs model) which is additively homomorphic over the message and randomness spaces. We then show how to construct an RF for this protocol.

Since our adaptively-secure augmented coin-tossing and ZK protocols are in the urs model, the compiled MPC protocol is also in the urs model. However, in the subversion setting we consider, a trusted setup is not available since a setup is susceptible to subversion too. For instance, the security guarantees of NIZKs completely break down in the face of subversion of the CRS [5]. To circumvent the need for a trusted setup, we show how to generate the urs needed by our adaptively secure MPC protocol securely in the presence of subversion by presenting a multi-party coin tossing protocol with a reverse firewall in the plain model.

Adaptively secure coin tossing in plain model. As a contribution of independent interest, we construct an adaptively secure multi-party coin tossing protocol in the *plain model* under the knowledge of exponent (KEA) assumption. Our use of non-black-box assumptions seems justified, in light of the result

of [24] that shows that it is not possible to construct an adaptively secure multi-party protocol with respect to black-box simulators without giving up on round efficiency in the plain model[6]. We use our coin-tossing protocol to generate the urs of our MPC protocol.

Theorem 3. *(Informal) Assuming DLOG, KEA and LWE assumptions, there exists a $\mathcal{O}(1)$ actively secure multi-party coin-tossing protocol that is secure against adaptive corruptions in the plain model.*

We then show how to add reverse firewalls to our adaptively secure coin tossing protocol. Finally, putting everything together, we obtain an *adaptively* secure MPC protocol with reverse firewall in the *plain* model. This resolves the open question posed in [12] of removing the trusted setup assumption in constructing MPC protocols with reverse firewalls.

1.2 Technical Overview

We provide a high-level overview of our construction, which can be viewed as an adaptive compiler for MPC protocols in the RF setting following the blueprint of [26]. The main idea of the [26] compiler is as follows: Each party (i) runs an instance of an augmented multi-party coin-tossing protocol to obtain a uniformly random string that it is committed to, (ii) commits to its input and broadcasts the input commitment to every other party, (iii) runs the underlying semi-honest adaptively secure MPC protocol, while proving in zero-knowledge that the computations have been done correctly. Since our goal is adaptive security, we start with an adaptively secure semi-honest protocol. Our compiler will use adaptively secure augmented coin-tossing and adaptively-secure ZK protocols in the plain model.

Adding reverse firewalls. The protocol outlined above requires randomness in the augmented coin-tossing protocol and the ZK protocol. The rest of the MPC protocol is deterministic given the coins and the randomness of the ZK protocol. We propose an adaptively secure multi-party augmented coin-tossing protocol Π_{coin} and an adaptively secure (input-delayed) ZK protocol Π_{zk}. We then design reverse firewalls for these protocols and show that they provide exfiltration resistance for tampered parties. Then, by invoking our theorem that an exfiltration resistant RF is security preserving, we get that the RFs preserve security of the above protocols. We now explain them in more detail below.

– *$\Pi_{\mathsf{a\text{-}coin}}$ using reverse firewalls*: Our augmented coin-tossing uses the "commit-then-open" paradigm. At the end of this protocol, the initiating party (say, P_i) obtains a random string r_i along with the appropriate decommitment information, whereas all other parties $\{P_j\}_{j\in[n]\setminus i}$ obtain the (same) commitment to r_i. We assume that the message and randomness spaces of the commitment scheme

[6] If we had a coin tossing protocol with black-box simulation, we could use it to transform a two round adaptively secure MPC protocol in the URS model [10] to a protocol in the plain model by generating the URS via the coin toss protocol.

form an additive group and the commitment scheme is *additively homomorphic* over these spaces. In the first round, each party $\{P_j\}_{j\in[n]\setminus i}$ sample their own randomness r_j and s_j, commits to the random coin r_j using s_j and broadcasts the commitment $c_j = \mathsf{Com}(r_j; s_j)$. In the second round, party P_i samples its own randomness (r_i, s_i), and broadcasts the commitment $c_i = \mathsf{Com}(r_i; s_i)$ to all other parties. Finally, in the third round all parties $\{P_j\}_{j\in[n]\setminus i}$ broadcast their respective openings (r_j, s_j). Party P_i then obtains the final string as $R = \sum_{k\in[n]} r_k$, and locally computes the commitment c_i as $c = \mathsf{Com}(R; S)$, where $S = \sum_{k\in[n]} s_k$. All other parties can compute the same commitment c using the commitment c_i (broadcast by P_i) and the decommitment information of all other parties (broadcast in the final round) exploiting the homomorphic property of Com. We show that the above protocol is adaptively secure if the underlying commitment scheme Com is adaptively secure.

Consider the case when the initiating party P_i is tampered. In this case, the other malicious parties can launch an *input trigger attack* by sending a malformed commitment string which may serve as a wake up message to P_i. Besides, in the second round, tampered P_i can sample bad randomness and exfiltrate secrets via the commitment string c_i. When the receiving parties are corrupt, the commitment strings and their openings could also serve as a subliminal channel. The main idea of the RF is to exploit the homomorphic properties of Com to *sanitize* the incoming and outgoing messages. However, it must ensure that this mauling is consistent with the views of all parties. In particular, RF_i for P_i rerandomizes the commitment c_i to a fresh commitment $\hat{c}_i$ by choosing fresh randomness (r'_i, s'_i), computing $c'_i = \mathsf{Com}(r'_i; s'_i)$ and homomorphically adding them. In the final round, when all the parties send their openings (r_j, s_j), RF_i computes an additive secret sharing of r'_i and s'_i (sampled in the above step) and sanitizes each of these openings using the appropriate shares. Thus, the views of all the parties are consistent in this firewalled protocol and the final coin is also guaranteed to be random (since the offsets r'_i and s'_i were sampled randomly). Note that, the final commitment C computed by all the parties does not provide any channel to exfiltrate (since both R and S are random at the end of the firewalled execution). The detailed protocol together with the RF is in Section 5.1.

– Π_{zk} *using reverse firewalls* : Next, we need a ZK protocol to show conformance of each step of the protocol specification. We construct a reverse firewall for (a variant of) the adaptively secure ZK protocol of [11]. The protocol of [11] is based on the Sigma protocol of [21] where the prover sends a first message, the verifier sends a random bit string as a challenge, and the prover sends a response in a third message. Towards constructing a reverse firewall, we observe that the prover's messages can be re-randomized if the underlying primitives are homomorphic. However, the challenge string cannot be re-randomized, without also mauling the response provided by the prover. The ZK protocol of [11] does not seem to have this malleable property. Therefore, we modify the protocol, where the verifier's challenge is generated as the result of a coin-tossing protocol. This ensures that the challenge is indeed random, and after the firewall sanitizes, both the prover and the verifier have the same challenge string. Therefore, the

firewall can sanitize the protocol without the need to explicitly maul the response from the prover. The modified protocol remains adaptively secure. Note that the protocol also retains the input-delayed property – only the last round of the ZK protocol depends on the statement being proven and the corresponding witness. This allows running the first two rounds of the protocol before the inputs in the MPC protocol are defined. During the MPC protocol, the parties compute the input commitments and the protocol messages which define the statement and the witness. The last round of the ZK protocol is run after this is defined, thus helping to preserve the round complexity of the underlying semi-honest adaptively-secure MPC protocol.
The idea behind a firewall for a tampered party in this modified ZK protocol, is to re-randomize the prover's first message in the coin tossing homomorphically, thus ensuring that the verifier's challenge in the sigma protocol is random. We show reverse firewalls for the prover and the verifier, and prove exfiltration resistance.

We obtain the above protocols in the urs model by instantiating the semi-honest MPC protocol and the underlying primitives in the urs model based on DDH and LWE (see Sections 5.1 and 5.3). Next, we generate the urs using an adaptively-secure coin tossing protocol to remove the setup assumption.

Adaptively secure coin tossing in the plain model. In order to remove the setup, we construct a constant round multi-party coin tossing protocol Π_{coin} in the plain model that generates the urs required by the MPC protocol. In addition to Discrete Log (DL) and Learning With Errors (LWE) assumption, we rely on knowledge of exponent assumption (KEA) assumption in pairing groups. Since it is *impossible* to construct a constant round adaptively secure coin-tossing protocol in the plain model from black-box simulation techniques [24], our reliance on the KEA assumption seems justified. The only other adaptively secure coin-tossing protocol in the plain model by Garg and Sahai [24] uses Barak's non-blackbox technique, and it is non-trivial to extend this protocol in the RF setting. Therefore, we craft a new RF-compatible protocol from scratch.

The high-level idea behind our Π_{coin} protocol is as follows: There is an initial coin-tossing phase that sets up a public key of a homomorphic, obliviously sampleable encryption scheme. In subsequent steps, there is another coin-tossing phase where parties exchange commitments to their coins together with encryption of the commitment randomness under the public key generated in the previous coin-toss. The protocol uses the Pedersen commitment scheme – an *equivocal*, *perfectly hiding* commitment scheme, and a public key encryption scheme with additional properties. Π_{coin} consists of four phases - parameter generation phase, commitment generation phase, commitment opening phase and output phase.

In the first phase, the parties generate pairwise Pedersen commitment parameters and pairwise encryption key. For the commitment parameter, one party is the committer and the other party is the verifier; and the verifier additionally proves knowledge of the commitment trapdoor. The public key is of an encryption scheme that satisfies the following properties: oblivious ciphertext sampling, oblivious public key sampling and additive homomorphism of ciphertexts and public keys. The parameter generation is repeated by reversing the roles. In the

commitment generation phase, each party generates its random coin and commits (as the committer) to it pairwise using the pairwise commitment parameters generated in the previous phase. Each party also sends two encryptions e_0 and e_1: if the committed coin is $b \in \{0,1\}$, then e_b is an encryption of the randomness used to commit to the coin, and e_{1-b} is sampled obliviously. Upon obtaining pairwise commitments to the random coins, the parties open their commitments pairwise by sending the decommitment randomness and encryption randomness for b to the pairwise verifiers. e_{1-b} is claimed to be obliviously sampled. Each party also broadcasts its random coin b. In the output phase, each party verifies the pairwise commitment openings and that correct ciphertext is an encryption of the commitment randomness. If all the openings are correct and they are consistent with the broadcasted coins then the parties output the final coin by summing up all the broadcasted coins.

This protocol is adaptively secure if the commitment is equivocal and perfectly hiding. The simulator needs to bias the output coin to a simulated coin. It is performed as follows: In the parameter generation phase, the verifier proves knowledge of trapdoor using a sub-protocol. When the verifier is corrupt, a non-black-box assumption allows extraction of the trapdoor. When the committer is corrupt, the simulator receives the commitment and the public key, samples a key pair, rewinds the committer and sets its own oblivious key such that they homomorphically combine to the honestly sampled key. Now, the simulator knows the corresponding secret key. The simulator extracts the committed coins of the malicious parties in the commitment generation phase. In the opening phase the simulator equivocates (using the extracted trapdoors) the pairwise commitments and the coins broadcasted on behalf of the honest parties such that the final output coin is equal to the simulated output coin. Once the committer opens its public key in the first coin-toss, the simulator can rewind and force the output of this coin-toss phase to be a public key for which the simulator knows the secret key. In the subsequent coin-tossing phase where parties exchange commitment to their coins together with encryption of the commitment randomness, the simulator can extract the value committed. Crucially, the simulator can extract the committed coin of the corrupt committer *before* the adversary can see the output of the coin toss allowing it to simulate. When the committer is honest, the simulator can explain the ciphertexts as encrypting the correct values.

Adding reverse firewalls to the coin tossing protocol. We exploit the homomorphism property of the underlying commitment and public-key encryption scheme to sanitize round messages. In addition to this, the RF computes pairing equations in order to verify validity of messages.

On the setup assumption. The work of [12] required a structured setup due its augmented coin-tossing protocol. In their coin tossing protocol the receiving parties obtain commitments to the sender's coin, which is different from the commitment generated by the sender. As a result, during the later part of the protocol the RF needs to maul the proofs using a controlled-malleable NIZK (cm-NIZK) as the statement being proven by the sender is different from the one being verified by the receiving parties. Unfortunately, cm-NIZKs are not

known in the adaptive setting. We modify the coin-tossing protocol such that every party obtains the *same* commitment string, and hence the proof statement remains unchanged. Thus, we can use an interactive ZK protocol without needing controlled malleability (re-randomizability suffices), and this allows us to rely on urs instead of crs. Finally, we can use the coin-tossing protocol (in the plain model) to remove the need for urs.

Finally, in all our protocols we rely on the existence of broadcast channels in the RF setting. We implicitly use the protocol of [12], who showed how to implement broadcast channels in the RF setting.

1.3 Other Related Work

Besides the reverse firewall framework, other directions that address the challenge of protecting cryptosystems against different forms of subversion are reviewed below.

Algorithm Substitution Attacks. Bellare, Patterson, and Rogaway [7] initiated the study of subversion of symmetric encryption schemes in the form of algorithm-substitution attacks (ASAs). They show that such subversion of the encryption algorithm is possible in a way that is undetectable. They also show that deterministic, stateful, ciphers are secure against this type of ASAs. Subsequent works redefined and strengthened the notion in several aspects [17, 6], and extended the ASA model to other contexts, like digital signatures schemes [2], public key encryption [13].

Backdooring. Motivated by the backdooring of the DUAL EC DRBG [31], a formal study of backdooring of PRGs was initiated in [19], where public parameters are surreptitiously generated together with secret backdoors by a saboteur that allows to bypass security while remaining secure to any adversary that does not know the backdoor. Parameter subversion has been considered for several primitives, including pseudorandom generators [19, 18], non-interactive zero knowledge [5], and public-key encryption [3].

Watchdogs and Self-guarding. Another approach taken in [29, 30, 8] is to consider an external entity called a watchdog that is trusted to test whether a given cryptographic implementation is compliant with its specification via black-box access. Self-guarding is another approach to counter subversion [22]. The idea here is to not depend on external entities, instead assume a trusted initialization phase where the cryptosystem is unsubverted.

2 Preliminaries

Notation. We write PPT to denote a probabilistic polynomial time machine. We denote the security parameter by λ. For an integer $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, 2, \cdots, n\}$ and for any pair of integers $1 < i < j \leq n$, we denote by $[i, j]$ the set $\{i, i+1, \cdots, j\}$. For a distribution or random variable X, we denote $x \leftarrow X$ the action of sampling an element x according to X. For any integer $m \in \mathbb{N}$, we write U_m to denote the uniform distribution over all m-bit strings. We denote

by $\mathbb{G}$ the multiplicative group where DDH assumption holds. The corresponding field is denoted by $\mathbb{Z}_q$. We denote a negligible function in λ as $\mathsf{neg}(\lambda)$.

Required Primitives. A public key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies *oblivious ciphertext sampling* if there exists a polynomial time algorithm oEnc which obliviously samples a ciphertext s.t. it looks indistinguishable from a real ciphertext. Additionally, we require the PKE to satisfy *additive homomorphism* over message and randomness space, i.e. $\mathsf{Enc}(\mathsf{pk}, m; r) \cdot \mathsf{Enc}(\mathsf{pk}, m'; r') = \mathsf{Enc}(\mathsf{pk}, m + m'; r + r')$. We denote a commitment scheme as $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Verify})$. It is *equivocal* if there exists a polynomial time algorithm Equiv that equivocates a commitment to open to any message, given the trapdoor of the commitment parameters. We need an adaptively secure commitment scheme and we use the definition of [10]. We also need the commitment scheme to satisfy additively homomorphic property like the PKE scheme. We present a version of Elgamal commitment scheme without setup as follows. Given a generator $g \in \mathbb{G}$, the committer commits to a field element $m \in \mathbb{Z}_q$ by sampling randomness $x, r \leftarrow \mathbb{Z}_q$ and sets $c = (c_1, c_2, c_3) = (g^x, g^r, g^m g^{rx}) = (h, g^r, g^m h^r)$. The tuple (x, r) serves as the decommitment information. It is *perfectly* binding and computationally hiding due to the DDH assumption. We also require an *input-delayed* (interactive) ZK protocol $\Pi_{\mathsf{zk}} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, i.e., only the last message from the prover to the verifier should depend on the statement. We use the ZK definitions of [11].

Bilinear Groups and Knowledge of Exponent Assumption [1]. Let $\mathcal{BGG}$ denote a bilinear group generator. It takes in input the security parameter λ and outputs $(\mathbb{G}, \mathbb{H}, q, g, e)$ where $\mathbb{G}$ and $\mathbb{H}$ is a pair of groups of prime order q where g is a generator of group $\mathbb{G}$, and e is a non-degenerate bilinear map defined as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{H}$ for which $e(g^a, g^b) = e(g, g)^{ab}$ for $a, b \in \mathbb{Z}_q$ and $e(g, g) \neq 1_{\mathbb{H}}$.

Definition 1. (Discrete Log Assumption) *For every non-uniform poly-time algorithm $\mathcal{A}$, the following holds:*

$$\Pr[pub \leftarrow \mathcal{BGG}(1^\lambda), h \leftarrow \mathbb{G}, w \leftarrow \mathcal{A}(pub, h) : g^w = h] \leq \mathit{neg}(\lambda)$$

Definition 2. (Knowledge of Exponent Assumption). *For every non uniform poly-time algorithm $\mathcal{A}$ there exists a non-uniform poly-time algorithm $\mathcal{X}_\mathcal{A}$, the extractor such that:*

$$\Pr[pub \leftarrow \mathcal{BGG}(1^\lambda), x \leftarrow \mathbb{Z}_q, (A, \hat{A}; a) \leftarrow (\mathcal{A}||\mathcal{X}_\mathcal{A})(pub, g^x) :$$

$$\hat{A} = A^x \wedge A \neq g^a] \leq \mathit{neg}(\lambda)$$

where $(A, \hat{A}; a) \leftarrow (\mathcal{A}||\mathcal{X}_\mathcal{A})(pub, g^x)$ means that $\mathcal{A}$ and $\mathcal{X}_\mathcal{A}$ are executed on the same input (pub, g^x) and the same random tape, and $\mathcal{A}$ outputs $(A, \hat{A})$ whereas $\mathcal{X}_\mathcal{A}$ outputs a.

Definition 3 ([26] Multi-party Parallel Coin-Tossing into the Well). *An n-party* augmented coin-tossing into the well *protocol is an n-party protocol for securely computing the functionality $(1^\lambda, \cdots, 1^\lambda) \to \big(U_t, U_t, \ldots, U_t\big)$, where U_t denotes the uniform distribution over t-bit strings.*

Definition 4 ([26] Multi-party Augmented Parallel Coin-Tossing into the Well). *An n-party* augmented coin-tossing into the well *protocol is an n-party protocol for securely computing the functionality* $(1^\lambda, \cdots, 1^\lambda) \to \big((U_t, U_{t\cdot\lambda}), \mathsf{Com}(U_t; U_{t\cdot\lambda}), \cdots, \mathsf{Com}(U_t; U_{t\cdot\lambda})\big)$ *with respect to a fixed commitment scheme* $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Verify})$ *which requires λ random bits to commit to each bit, and U_t denotes the uniform distribution over t-bit strings.*

Next, we define adaptive security for MPC in the stand-alone setting. Let us assume that an adversary $\mathcal{A}$ runs a protocol Π with parties $(P_1, \ldots, P_n)$ to compute function f on inputs $\vec{x}$. $\mathcal{A}$ adaptively corrupts parties P_i in the set C, for $i \in \mathsf{C}$. We denote the real world adversary view of the protocol as $\mathsf{REAL}_{\Pi,(\mathsf{C},\mathcal{A})}(\lambda, \vec{x}, z)$. Let us denote $\mathsf{REAL}_{\Pi,(\mathsf{C},\mathcal{A})}$ as the distribution ensemble $\{\mathsf{REAL}_{\Pi,(\mathsf{C},\mathcal{A})}(\lambda, \vec{x}, z)\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^*)^n,z\in\{0,1\}^*}$. Let Sim be an ideal world adversary who interacts with the ideal functionality f and we denote the ideal world adversary view as $\mathsf{IDEAL}_{f,(\mathsf{C},\mathsf{Sim})}(\lambda, \vec{x}, \vec{r}, z)$. let $\mathsf{IDEAL}_{f,(\mathsf{C},\mathsf{Sim})}$ denote the distribution ensemble $\{\mathsf{IDEAL}_{f,(\mathsf{C},\mathsf{Sim})}(\lambda, \vec{x}, z)\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^*)^n,z\in\{0,1\}^*}$.We say that Π adaptively securely evaluates f if for every real world adversary $\mathcal{A}$ there exists an ideal world adversary Sim, s.t. $\mathsf{REAL}_{\Pi,(\mathsf{C},\mathcal{A})}(\lambda, \vec{x}, z) \approx_c \mathsf{IDEAL}_{f,(\mathsf{C},\mathsf{Sim})}(\lambda, \vec{x}, z)$.

3 Reverse Firewalls for Adaptively secure MPCs

In this section, we present definitions of reverse firewalls for adaptively secure MPC protocols. The existing definitions of security preservation and exfiltration-resistance for reverse firewalls are for a static adversary [12]. In the adaptive setting, while the security preservation is defined as before, exfiltration resistance now has to incorporate the adaptive power of the adversary. We first introduce some notation that will be used throughout the paper.

Notation. Let Π denote a ℓ-round MPC protocol, for some arbitrary polynomial $\ell(\cdot)$ in the security parameter λ. Let H and C denote the indices of the honest and maliciously corrupted parties respectively in the protocol Π. For a party P and reverse firewall RF we define $\mathsf{RF} \circ P$ as the "composed" party in which the incoming and outgoing messages of A are "sanitized" by RF. The firewall RF is a *stateful* algorithm that is only allowed to see the public parameters of the system, and does not get to see the inputs and outputs of the party P. We denote the tampered implementation of a party P by $\overline{P}$.

We denote the view of a party P_i by View_{P_i}, which consists of the input of P_i, its random tape and the messages received so far. We also denote the view of a party P_i till some round $k(\leq \ell)$ as $\mathsf{View}_{P_i}^{\leq k}$. We denote the reverse firewall for party P_i as RF_i and the internal state of RF_i by $\mathsf{st}_{\mathsf{RF}_i}$. We write $\mathsf{View}_{\mathsf{RF}_i \circ P_i}$ to denote the composed view of a party P_i and its RF RF_i. Let $\mathsf{Transform}(\cdot)$ be a polynomial time algorithm that takes as input the random tape r_i of a party P_i and the internal state (or randomness) $\mathsf{st}_{\mathsf{RF}_i}$ of RF_i and returns a sanitized random tape $\mathsf{Transform}(r_i, \mathsf{st}_{\mathsf{RF}_i})$[7]. Note that, the composed view $\mathsf{View}_{\mathsf{RF}_i \circ P_i}$ of

[7] Looking ahead, in all our constructions the function $\mathsf{Transform}$ will typically be a very simple function like addition or field multiplication.

P_i can be efficiently constructed from the view View_{P_i} of P_i and the state $\mathsf{st}_{\mathsf{RF}_i}$ of RF_i using the $\mathsf{Transform}$ function as a subroutine. We write $\Pi_{\mathsf{RF}_i \circ P_i}$ (resp. $\Pi_{\overline{P_i}}$) to represent the protocol Π in which the role of a party P_i is replaced by the composed party $\mathsf{RF}_i \circ P_i$ (resp. the tampered implementation $\overline{P_i}$).

Definition 5. (Functionality-maintaining RF). *For any reverse firewall* RF *and a party* P*, let* $\mathsf{RF}^1 \circ P = \mathsf{RF} \circ P$*, and* $\mathsf{RF}^k \circ P = \underbrace{\mathsf{RF} \circ \cdots \circ \mathsf{RF}}_{k \text{ times}} \circ P$*. For a protocol* Π *that satisfies some functionality requirements* $\mathcal{F}$*, we say that a reverse firewall* RF maintains functionality $\mathcal{F}$ *for a party* P *in protocol* Π *if* $\Pi_{\mathsf{RF}^k \circ P}$ *also satisfies* $\mathcal{F}$*, for any polynomially bounded* $k \geq 1$*.*

Definition 6. (Security-preserving RF for Malicious Adaptively secure MPCs). *Let* Π *be a multi-party protocol run between parties* $P_1, \ldots, P_n$ *satisfying functionality requirement* $\mathcal{F}$ *and is secure against adaptive malicious adversaries. We assume that each honest party* $\{P_i\}_{i \in \mathsf{H}}$ *is equipped with its corresponding reverse firewall* $\{\mathsf{RF}_i\}_{i \in \mathsf{H}}$*. When the adversary (adaptively) corrupts a party* P_i*, it receives* $\mathsf{View}_{\mathsf{RF}_i \circ P_i}$ *as the view of* P_i*. Then, we say that the reverse firewalls* RF_i *for parties* $\{P_i\}_{i \in \mathsf{H}}$ strongly (resp. weakly) *preserves security of the protocol* Π*, if there exists a polynomial-time computable transformation of polynomial-size circuit families* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ *for the real world into polynomial-size circuit families* $\mathsf{Sim} = \{\mathsf{Sim}_\lambda\}_{\lambda \in \mathbb{N}}$ *for the ideal model such that for every* $\lambda \in \mathbb{N}$*, every subset* $\mathsf{H} \subset [n]$*, every input sequence* $\vec{x} = (x_1, \ldots, x_n) \in (\{0,1\}^\lambda)^n$*, every auxiliary information* $z \in \{0,1\}^*$ *and every arbitrary (resp. functionality-maintaining) tampered implementation* $\{\overline{P_i}\}_{i \in \mathsf{H}}$ *we have the following:* $\mathsf{REAL}_{\Pi_{\{\mathsf{RF}_i \circ \overline{P_i}\}_{i \in \mathsf{H}}}, (\mathsf{C}, \mathcal{A})}(\lambda, \vec{x}, z) \approx_c \mathsf{IDEAL}_{f, (\mathsf{C}, \mathsf{Sim})}(\lambda, \vec{x}, z)$.

We now define exfiltration resistance in terms of the game LEAK that asks the adversary to distinguish between a tampered implementation of party P_i and an honest implementation, even *given* the composed state of P_i and its RF if P_i gets adaptively corrupt in the middle of execution. [8]

Definition 7. (Exfiltration-resistant RF in the presence of adaptive corruptions). *Let* Π *be a multi-party protocol run between the parties* $P_1, \ldots, P_n$ *satisfying functionality* $\mathcal{F}$ *and having reverse firewalls* RF_i *for the set of honest parties* $\{P_i\}_{i \in \mathsf{H}}$*. When the adversary (adaptively) corrupts a party* P_i*, it receives* $\mathsf{View}_{\mathsf{RF}_i \circ P_i}$ *as the view of* P_i*. Then* $\forall i \in \mathsf{H}$*, we say that the firewall* RF_i *is exfiltration-resistant for party* P_i *against all other parties* $\{P_j\}_{j \in [n] \setminus i}$*, if for any PPT adversary* $\mathcal{A}_{\mathsf{ER}}$*, the advantage* $\mathsf{Adv}^{\mathsf{LEAK}}_{\mathcal{A}_{\mathsf{ER}}, \mathsf{RF}_i}(\lambda)$ *of* $\mathcal{A}_{\mathsf{ER}}$ *(defined below) in the game* LEAK *(see Fig. 1) is negligible in the security parameter* λ*.*
The advantage of any adversary $\mathcal{A}_{\mathsf{ER}}$ *in the game* LEAK *is defined as:*

$$\mathsf{Adv}^{\mathsf{LEAK}}_{\mathcal{A}_{ER}, \mathsf{RF}_i}(\lambda) = \left| \mathsf{Pr}[\mathsf{LEAK}(\Pi, i, \{P_1, \ldots, P_n\}, \mathsf{RF}_i, \lambda) = 1] - \frac{1}{2} \right|.$$

[8] Note that, if we were to give P_i's internal state when it gets adaptively corrupt instead of the composed state, the adversary can trivially distinguish since the party's state does not explain the sanitized transcript.

Fig. 1. Exfiltration-resistance game LEAK for a party P_i for adaptively secure MPCs

$\underline{\mathsf{LEAK}(\Pi, i, \{P_1, \cdots, P_n\}, \mathsf{RF}_i, \lambda)}$

W.l.o.g, let $P_{\mathsf{H}} = (P_1, \cdots, P_h)$ denote the set of honest parties at the onset of the protocol Π, where $h = |\mathsf{H}|$. The exfiltration-resistance game LEAK for a party $P_i \in P_{\mathsf{H}}$ is modelled as an interactive game between a challenger $\mathcal{C}_{\mathsf{ER}}$ and an adversary $\mathcal{A}_{\mathsf{ER}}$ described as follows:

1. The adversary $\mathcal{A}_{\mathsf{ER}}$ provides the tampered implementations of all the honest parties along with their inputs $\{(\overline{P_1}, \cdots, \overline{P_h}), I_{\mathsf{H}}\}$ to the challenger $\mathcal{C}_{\mathsf{ER}}$.
2. The challenger $\mathcal{C}_{\mathsf{ER}}$ samples a bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random and does the following:
   - If $b = 1$, define $P_i^* \leftarrow \mathsf{RF}_i \circ \overline{P_i}$.
   - If $b = 0$, define $P_i^* \leftarrow \mathsf{RF}_i \circ P_i$.
3. $\mathcal{C}_{\mathsf{ER}}$ and $\mathcal{A}_{\mathsf{ER}}$ then engage in an execution of the MPC protocol Π, where the challenger $\mathcal{C}_{\mathsf{ER}}$ plays the role of all the honest parties P_{H} (with inputs I_{H}) and the adversary can adaptively corrupt parties in the set P_{H}. The $\mathcal{C}_{\mathsf{ER}}$ then returns the transcript $\mathcal{T}^*$ of Π to $\mathcal{A}_{\mathsf{ER}}$.
4. If $\mathcal{A}_{\mathsf{ER}}$ adaptively corrupts the party P_i at some point during the execution of Π (say at round k) the challenger $\mathcal{C}_{\mathsf{ER}}$ returns the composed view of P_i till round k, i.e., $\mathsf{View}^{\leq k}_{\mathsf{RF}_i \circ P_i}$ to $\mathcal{A}_{\mathsf{ER}}$. Note that, $\mathsf{View}^{\leq k}_{\mathsf{RF}_i \circ P_i}$ can be efficiently constructed from $\mathsf{View}^{\leq k}_{P_i}$ and the state $\mathsf{st}_{\mathsf{RF}_i}$ of RF_i using the $\mathsf{Transform}$ function as a subroutine.
5. The challenger $\mathcal{C}_{\mathsf{ER}}$ also returns the views of all the other uncorrupted parties $\{\mathsf{View}_{P_k}\}_{k \in [h \setminus i]}$ to $\mathcal{A}_{\mathsf{ER}}$.
6. The game ends when $\mathcal{A}_{\mathsf{ER}}$ returns a bit b' as a guess for the bit b. Output 1 if $b' = b$.

As in prior works on RF, we consider the notion of functionality-maintaining tampering attacks. Informally, such an attack excludes all conspicuous tamperings, which would otherwise be detected by honest parties. We provide a formal definition in the full version. We also define *transparency* of reverse firewalls which was informally introduced in [20], which means that the behavior of $\mathsf{RF} \circ P$ is identical to the behavior of P if P is the honest implementation. We will also need the notion of *valid transcripts* and *detectable failures* of reverse firewalls, as presented in [20]. In this work we do not consider *input replacement tampering* attack - tampering attacks work by substituting the actual input of the honest parties with a different (possibly (un)related) value. We defer these definitions to the full version.

4 Relations between Security Preservation and Exfiltration Resistance

In this section, we explore the relation between the notions of security preservation (SP) and exfiltration resistance (ER) for reverse firewalls in the MPC setting. Specifically, we show that ER implies SP for MPC protocols for adaptive corruptions; whereas the relation in the other direction is much less clear.

For all the implications we show in this section, we assume that security preservation be defined by the existence of a black-box simulator. We also assume that the adversaries are not computationally unbounded, and do not have access to additional oracles. Looking ahead, all of our constructions will satisfy the above requirements.

4.1 Exfiltration-Resistance implies Security Preservation

In this section, we show that ER implies SP for adaptively-secure MPC protocols by proving the following theorem.

Theorem 4. *Let f be an n-ary functionality and let Π be a an n-party protocol that securely computes f with abort in presence of malicious adaptive adversaries. Let $\mathsf{C} \subset [n]$ denote the indices of adaptively corrupt the corrupted parties in the protocol Π, and let $\mathsf{H} = [n] \setminus \mathsf{C}$ denote the indices of the honest parties at the outset of Π. Let $(\overline{P}_i)_{i\in\mathsf{H}}$ denote the tampered implementations of the honest parties provided by the adversary. Also, let RF_i denote the RF corresponding to party P_i. Then for all $i \in \mathsf{H}$, if RF_i is* functionality maintaining, (strongly/weakly) exfiltration resistant with adaptive security for P_i against all other parties $\{P_j\}_{[n]\setminus i}$ *and* transparent*, then for all PPT adversaries $\mathcal{A}$ and all PPT tamperings $(\overline{P_i})_{i\in\mathsf{H}}$ provided by $\mathcal{A}$, the firewalls RF_i for parties $\{P_i\}_{i\in\mathsf{H}}$ (strongly/weakly) preserve security of the protocol Π according to Def. 6 in the presence of adaptive corruptions.*

Proof. We need to show that security of the MPC protocol Π is (strongly/weakly) preserved by the reverse firewalls RF_i for parties $\{P_i\}_{i\in\mathsf{H}}$ by relying on (strong/ weak) exfiltration-resistance of the firewalls RF_i, transparency of RF_i and the adaptive security of the underlying MPC protocol Π. More formally, we will show that there exists a simulator/ideal-world adversary Sim such that for any real-world adversary $\mathcal{A}$ participating in the protocol Π, adaptively (maliciously) corrupting parties during the execution, for all $\lambda \in \mathbb{N}$, inputs $\vec{x} \in (\{0,1\}^\lambda)^n$ and auxiliary input $z \in \{0,1\}^*$ the following two random variables are computationally indistinguishable:

$$\{\mathsf{REAL}_{\Pi_{\{\mathsf{RF}_i\circ\overline{P_i}\}_{i\in\mathsf{H}}},(\mathsf{C},\mathcal{A})}(\lambda,\vec{x},z)\} \approx_c \{\mathsf{IDEAL}_{f,(\mathsf{C},\mathsf{Sim})}(\lambda,\vec{x},z)\}, \tag{1}$$

Note that, in the above, C denotes the set of parties adaptively corrupted by the adversary. This set is allowed to grow during the execution of the protocol. H denotes the indices of honest parties at the outset of the protocol Π, i.e the initial set of honest parties.
We prove the above theorem via a sequence of hybrids, as described below.

- Hyb_0 : This is the first hybrid which corresponds to the left hand side of Eq. 1. In particular, Hyb_0 corresponds to the real world view of the adversary $\mathcal{A}$ in the MPC protocol Π, who adaptively corrupts the subset P_C of parties. When the adversary $\mathcal{A}$ corrupts some party $P_i \in P_\mathsf{H}$, return $\mathsf{View}_{\mathsf{RF}_i\circ P_i} = \mathsf{Transform}(\mathsf{View}_{P_i}, \mathsf{st}_{\mathsf{RF}_i})$ to $\mathcal{A}$, and move P_i to the corrupt set. All the honest parties in H are replaced with their corresponding tampered implementations

composed with their firewalls, i.e., for all $i \in \mathsf{H}$, P_i is replaced with $\mathsf{RF}_i \circ \overline{P_i}$ in the protocol Π. The view of the real world adversary $\mathcal{A}$ consists of the following:

$$\{\mathsf{REAL}_{\Pi_{\{\mathsf{RF}_i \circ \overline{P_i}\}_{i\in\mathsf{H}}},(\mathsf{C},\mathcal{A})}(\vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^n}\} \tag{2}$$

– Hyb_1 : Hyb_1 is same as Hyb_0, except that, in the protocol Π the implementation of the first party P_1 is replaced by its honest implementation composed with its firewall RF_1. The rest of the honest parties remain tampered, that is, $\{P_j\}_{j\in\mathsf{H}\wedge j\in\{2,\cdots,n\}}$ are still replaced by $\mathsf{RF}_j \circ \overline{P_j}$, and the corrupt parties remain as in Hyb_0. In particular, the view of the real world adversary is as follows:

$$\{\mathsf{REAL}_{\Pi_{(\mathsf{RF}_1\circ P_1,\{\mathsf{RF}_j\circ\overline{P_j}\}_{j\in\mathsf{H}\wedge j\in\{2,\cdots,n\}})},(\mathsf{C},\mathcal{A})}(\vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^n}$$

We now present the general description of the ℓ-th hybrid for all $1 \leq \ell \leq n$ as follows:

– Hyb_ℓ : In Hyb_ℓ, in the protocol Π the implementations of the first ℓ parties $\{P_1, P_2, \cdots, P_\ell\}$ are replaced by their corresponding honest implementations composed with their firewalls. The other honest parties $\{P_j\}_{j\in\mathsf{H}\wedge j\in\{\ell+1,\cdots,n\}}$ are still replaced by $\mathsf{RF}_j \circ \overline{P_j}$ in the protocol Π, as in Hyb_0. When the adversary $\mathcal{A}$ corrupts a currently honest party P_j, return $\mathsf{View}_{\mathsf{RF}_j\circ P_j} = \mathsf{Transform}(\mathsf{View}_{P_j}, \mathsf{st}_{\mathsf{RF}_j})$ to $\mathcal{A}$, and move P_j to the set of corrupt parties as before. In particular, the adversary $\mathcal{A}$ obtains the following view:

$$\{\mathsf{REAL}_{\Pi_{(\{\mathsf{RF}_j\circ P_j\}_{j\in[\ell]},\{\mathsf{RF}_j\circ\overline{P_j}\}_{j\in\mathsf{H}\wedge j\in\{\ell+1,\cdots,n\}})},(\mathsf{C},\mathcal{A})}(\vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^n} \tag{3}$$

Note that, when $\ell = 0$, we are in Hyb_0, i.e., when the implementations of all the honest parties in P_H are replaced by their corresponding tampered implementations composed with their firewalls in the protocol Π. On the other hand, when $\ell = n$, we are in Hyb_n where the implementations of all the honest parties are replaced by their honest implementations composed with their firewalls. For the sake of completeness, we present the n-th hybrid as follows:

– Hyb_n : In Hyb_n, in the protocol Π the implementations of all the honest parties $\{P_j\}_{j\in\mathsf{H}}$ are replaced by their corresponding honest implementations composed with their firewalls. In particular, the adversary $\mathcal{A}$ obtains the following view:

$$\{\mathsf{REAL}_{\Pi_{(\{\mathsf{RF}_j\circ P_j\}_{j\in\mathsf{H}})},(\mathsf{C},\mathcal{A})}(\vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^n} \tag{4}$$

Note that, in each subsequent hybrid we replace each party (honest and corrupt) with the fire-walled honest implementation. However this does not mean that the corrupt parties are forced to behave with honest implementation. As will be clear in the indistinguishability proof below, this approach does not enforce any restriction on the way the corrupt parties behave. We take this approach for readability and ease of proving indistinguishability.
Now, we prove the indistinguishability of consecutive hybrids.

Claim. $\forall 1 \leq \ell \leq n$, $\mathsf{Hyb}_{\ell-1} \approx_c \mathsf{Hyb}_\ell$

Proof. Note that, the two hybrids $\mathsf{Hyb}_{\ell-1}$ and Hyb_ℓ differ in the implementation of the party P_ℓ. In particular, the only change from $\mathsf{Hyb}_{\ell-1}$ to Hyb_ℓ is that $\mathsf{RF}_\ell \circ \overline{P_\ell}$ in the former is replaced by $\mathsf{RF}_\ell \circ P_\ell$ in the latter. Let $\mathcal{D}_\ell$ be an adversary that distinguishes between these two hybrids. Since the adversary is allowed to corrupt parties adaptively, assume that party P_ℓ is corrupted by $\mathcal{D}_\ell$ in round k_ℓ. Using $\mathcal{D}_\ell$, we construct an exfiltration resistant adversary $\mathcal{A}_{\mathsf{ER}}$ such that if the advantage of $\mathcal{D}_\ell$ is non-negligible, then the advantage of $\mathcal{A}_{\mathsf{ER}}$ in breaking the exfiltration-resistance game (Def. 7) is also non-negligible. At a high level, $\mathcal{A}_{\mathsf{ER}}$ interacts with $\mathcal{D}_\ell$ as the challenger for the indistinguishibility game for $\mathcal{D}_\ell$ so that ultimately $\mathcal{D}_\ell$ either sees views from $\mathsf{Hyb}_{\ell-1}$ or Hyb_l. If party P_ℓ is already in the corrupt set at the start of the protocol, then exfiltration-resistance is trivially satisfied for P_ℓ. Otherwise, for the case when P_ℓ gets adaptively corrupt in round k_ℓ, by the exfiltration guarantee, $\mathcal{D}_\ell$ cannot distinguish views till round k_ℓ.
The reduction is as follows:

- The adversary $\mathcal{A}_{\mathsf{ER}}$ receives the initial indices of honest parties (H), and the tampered implementations $\{\overline{P_j}\}_{j\in\mathsf{H}\wedge j\in\{\ell,\cdots n\}}$ corresponding to the last $(|\mathsf{H}| - \ell + 1)$ honest parties in the set P_{H} from the distinguisher $\mathcal{D}_\ell$.
- $\mathcal{A}_{\mathsf{ER}}$ then sets (1) $\overline{P_j} = \mathsf{RF}_j \circ P_j$ for all $j \in \mathsf{H} \wedge j \in [\ell - 1]$, and (2) randomly samples inputs for the parties in the honest set to define I and sends it to $\mathcal{C}_{\mathsf{ER}}$. It forwards the set $\{\overline{P_j}\}_{j\in\mathsf{H}}$ (here H is set of honest parties at the outset of Π) to the challenger $\mathcal{C}_{\mathsf{ER}}$ of the exfiltration-resistance (ER) game (see the LEAK game in Fig. 1). In other words, $\mathcal{A}_{\mathsf{ER}}$ sets the tampered implementations of the first $\ell - 1$ honest parties in the set P_{H} to be simply their corresponding honest implementations with a wrapper of firewall on top of it and sets the implementations of the remaining $(|\mathsf{H}| - \ell + 1)$ honest parties as received from $\mathcal{D}_\ell$.
- Now $\mathcal{A}_{\mathsf{ER}}$ interacts with the challenger $\mathcal{C}_{\mathsf{ER}}$ and the distinguisher $\mathcal{D}_\ell$ to execute Π. $\mathcal{C}_{\mathsf{ER}}$ executes Π on the behalf of the currently honest parties, and $\mathcal{D}_\ell$ executes on behalf of the currently dishonest parties. $\mathcal{A}_{\mathsf{ER}}$ passes round messages between $\mathcal{C}_{\mathsf{ER}}$ and $\mathcal{D}_\ell$. If $\mathcal{D}_\ell$ adaptively corrupts an honest party, $\mathcal{A}_{\mathsf{ER}}$ too corrupts the same party and receives a transformed view from $\mathcal{C}_{\mathsf{ER}}$ which it passes on to $\mathcal{D}_\ell$. On corruption of an honest party, $\mathcal{C}_{\mathsf{ER}}$ moves it from the set of honest to dishonest parties. Note that if party P_ℓ is statically corrupt, the indistinguishability in the ER game trivially follows.
- Upon receiving the final views of parties from $\mathcal{C}_{\mathsf{ER}}$ as described in the LEAK game in Fig. 1, $\mathcal{A}_{\mathsf{ER}}$ constructs the view as in Eq. 3. If $\mathcal{D}_\ell$ corrupts any party post execution, $\mathcal{A}_{\mathsf{ER}}$ forwards relevant view. Note that, $\mathcal{A}_{\mathsf{ER}}$ only needs to know the views corresponding to the corrupt parties to construct the view as in Eq. 3.
- If $\mathcal{D}_\ell$ outputs a bit b', the adversary $\mathcal{A}_{\mathsf{ER}}$ outputs the same bit b'. Note that, if the challenger $\mathcal{C}_{\mathsf{ER}}$ of the ER game sampled the bit $b = 1$, then

we are in $\mathsf{Hyb}_{\ell-1}$; whereas if $b = 0$, we are in Hyb_{ℓ}. Hence, if the advantage of $\mathcal{D}_\ell$ in distinguishing between these hybrids is non-negligible, the advantage of $\mathcal{A}_{\mathsf{ER}}$ in breaking the ER game is also non-negligible.

□

Note that, at the end of Hyb_n, all the honest parties $\{P_j\}_{j\in\mathsf{H}}$ in the set P_{H} are replaced by $\mathsf{RF}_j \circ P_j$.

– Hyb_{n+1} : Hyb_{n+1} is same as Hyb_n, except that, in the protocol Π all the honest parties in H have honest implementations and there is no RF for the honest parties. In particular, the adversary $\mathcal{A}$ obtains the following view:

$$\{\mathsf{REAL}_{\Pi_{(\{P_j\}_{j\in\mathsf{H}})},(\mathsf{C},\mathcal{A})}(\vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^n}.$$

Claim. $\mathsf{Hyb}_n \approx_c \mathsf{Hyb}_{n+1}$.

Proof: It is easy to see that the hybrids Hyb_n and Hyb_{n+1} are identically distributed by relying on the transparency of the firewalls $\{\mathsf{RF}_j\}_{j\in\mathsf{H}}$. More formally, we can define a set of n hybrids (similar to the claim earlier in this section) and show that the consecutive hybrids are indistinguishable by the transparency property of each of the reverse firewalls. □

– Hyb_{n+2} : This is the final hybrid. This hybrid corresponds to the the ideal world adversary view for the MPC protocol Π where the set of corrupted parties is $\{P_i\}_{i\in\mathsf{C}}$. All the honest parties P_{H} have honest implementations and there is no RF for the honest parties. In particular, the adversary $\mathcal{A}$ obtains the following view:

$$\{\mathsf{IDEAL}_{f,(\mathsf{C},\mathsf{Sim})}(\vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^n} \tag{5}$$

Claim. $\mathsf{Hyb}_{n+1} \approx_c \mathsf{Hyb}_{n+2}$

Proof: Hyb_{n+2} is indistinguishable from Hyb_{n+1} due to the security of the protocol Π. Hyb_{n+1} corresponds to the real world adversary view of Π (without any RF) and Hyb_{n+2} corresponds to the ideal world adversary view of Π (without any RF). □

Thus, combining the above three claims, we obtain Eq. 1. This completes the proof of Thm. 4. □

This implication holds in the static corruption case as well when the exfiltration resistant game, transparency game and the security preservation games are defined for static corruptions. We defer this proof to the full version.

5 Adaptively-Secure Compiler using Reverse Firewalls in the urs model

In this section, we show a compiler that transforms any semi-honest adaptively secure MPC protocol to a maliciously-secure MPC protocol in the urs model, which withstands adaptive corruptions and admits reverse firewalls. As a building block, we first present the adaptively-secure multiparty augmented coin tossing protocol using reverse firewalls.

5.1 Adaptively-Secure Augmented Coin-Tossing using Reverse Firewalls in the urs model

The adaptively-secure augmented coin tossing protocol $\Pi_{\mathsf{a\text{-}coin}}$ is used to generate *random bits* (according to Def. 4) for all the parties participating in the adaptively-secure MPC protocol. The initiating party receives a random tuple (S, R) and all other parties receive a commitment $\mathsf{Com}(S; R)$ (under the urs of party P_i) of S using commitment randomness R, where $S \in \{0,1\}^\lambda$, Com is an adaptively secure homomorphic commitment and $R \leftarrow \mathcal{R}_{\mathsf{Com}}$. The protocol $\Pi_{\mathsf{a\text{-}coin}}$ is presented in Fig. 2.

Fig. 2. Adaptively-Secure Multi-party Augmented Coin-Tossing Protocol $\Pi_{\mathsf{a\text{-}coin}}$ for P_i

- **Primitives:** $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Verify})$ is an adaptively secure homomorphic commitment scheme where $\mathsf{Com}(\mathsf{urs}, a_1; b_1) \cdot \mathsf{Com}(\mathsf{urs}, a_2; b_2) = \mathsf{Com}(\mathsf{urs}, a_1 + a_2; b_1 + b_2)$ for $a_1, a_2 \in \{0,1\}^\lambda$ and $b_1, b_2 \in \mathcal{R}_{\mathsf{Com}}$ respectively. We denote $\mathsf{Com}_i(m; r) = \mathsf{Com}(\mathsf{urs}_i, m; r)$ under $\mathsf{urs}_i \leftarrow \mathsf{Gen}(1^\lambda)$.
- **Public Inputs:** Each party gets as input $\mathsf{urs}_{\mathsf{a\text{-}coin}} = \{\mathsf{urs}_i\}_{i\in[n]}$ where $\mathsf{urs}_i \leftarrow \mathsf{Com.Gen}(i, 1^\lambda)$. Party P_i commits using Com_i in the protocol.

Round 1: For $j \in [n] \setminus i$, every party P_j samples $s_j \leftarrow \{0,1\}^\lambda$ and $r_j \leftarrow \mathcal{R}_{\mathsf{Com}}$ respectively. It computes $c_j = \mathsf{Com}_j(s_j; r_j)$, and broadcasts c_j .

Round 2: Party P_i chooses a random $s_i \leftarrow \{0,1\}^\lambda$. It then computes $c_i = \mathsf{Com}_i(s_i; r_i)$ for a random $r_i \leftarrow \mathcal{R}_{\mathsf{Com}}$, and broadcasts c_i.

Round 3: For $j \in [n] \setminus i$, every party P_j broadcasts (s_j, r_j) as the opening of c_j.

Local Computation:

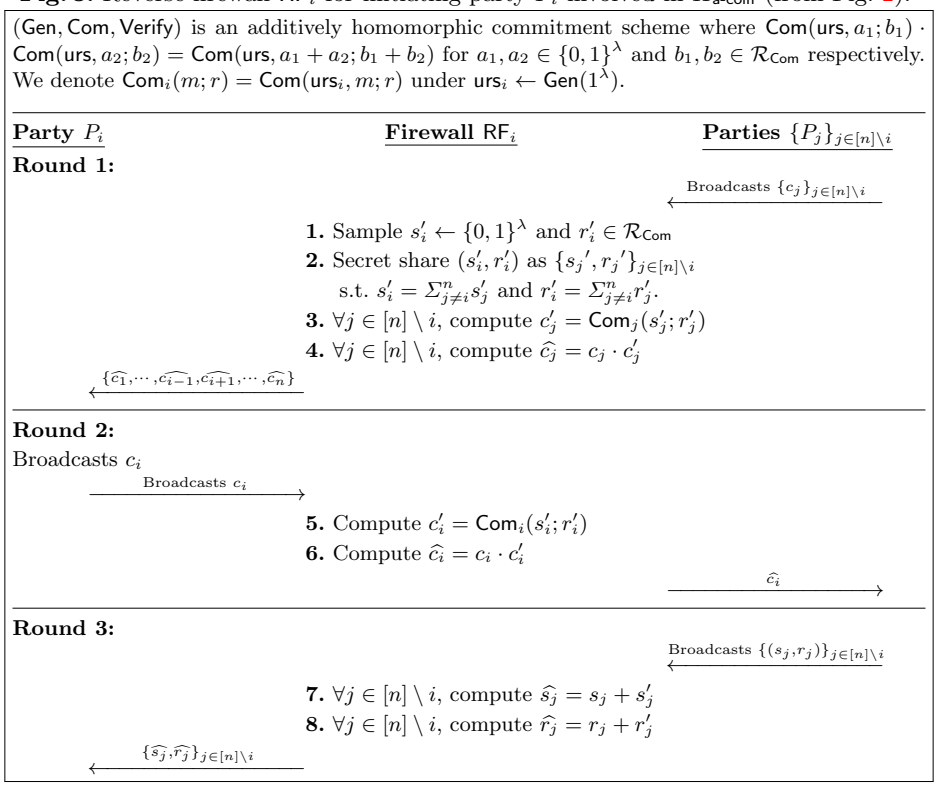
- For $j \in [n]$, P_j aborts if $\exists k \in [n] \setminus i$ s.t. $\mathsf{Verify}(\mathsf{urs}_k, c_k, s_k, r_k) = \perp$.
- Party P_i sets $S = \Sigma_{i\in[n]} s_i$ and $R = \Sigma_{i\in[n]} r_i$. P_i outputs $C = \mathsf{Com}_i(S; R)$.
- For $j \in [n] \setminus i$, Party P_j sets $S_j = \Sigma_{k\in[n]\setminus i} s_k$ and $R_j = \Sigma_{k\in[n]\setminus i} r_k$. P_j outputs $C = c_i \cdot \mathsf{Com}_i(S_j; R_j)$.

Theorem 5. *Assuming* Com *is an adaptively secure homomorphic commitment in the* urs *model, $\Pi_{\mathsf{a\text{-}coin}}$ securely implements the augmented coin-tossing functionality (Def. 4) against adaptive corruption of parties in the* urs *model.*

Next, we consider security of the protocol when honest parties are tampered. Our firewall RF_i for a tampered initiating party P_i and firewall $\{\mathsf{RF}_k\}_{k\in[n]\setminus i}$ for a tampered receiving party $\{\mathcal{P}_k\}_{k\in[n]\setminus i}$ is presented in Fig. 3 and Fig. 4 respectively. We prove that the firewalls provide weak exfiltration resistance and preserve security.

Theorem 6. *If the commitment scheme* Com *is adaptively secure in the* urs *model and is additively homomorphic, the firewall RF_i, (resp. RF_k) is transparent, functionality-maintaining, and provides weak exfiltration resistance for initiating P_i (resp. receiving party P_k) against other parties in $\Pi_{\mathsf{a\text{-}coin}}$ with valid transcripts, and detects failure for P_i (resp. P_k).*

Fig. 3. Reverse firewall RF_i for initiating party P_i involved in $\Pi_{\mathsf{a\text{-}coin}}$ (from Fig. 2).



Now, consider $\Pi^{\mathsf{RF}}_{\mathsf{a\text{-}coin}}$, a firewalled version of the protocol where all honest parties P_i have their respective firewalls RF_i attached to them. Now, from Thm. 6, and our implication from Thm. 4, we conclude weak security preservation; we have that $\Pi^{\mathsf{RF}}_{\mathsf{a\text{-}coin}}$ securely implements augmented coin tossing in the presence of adaptive corruptions.

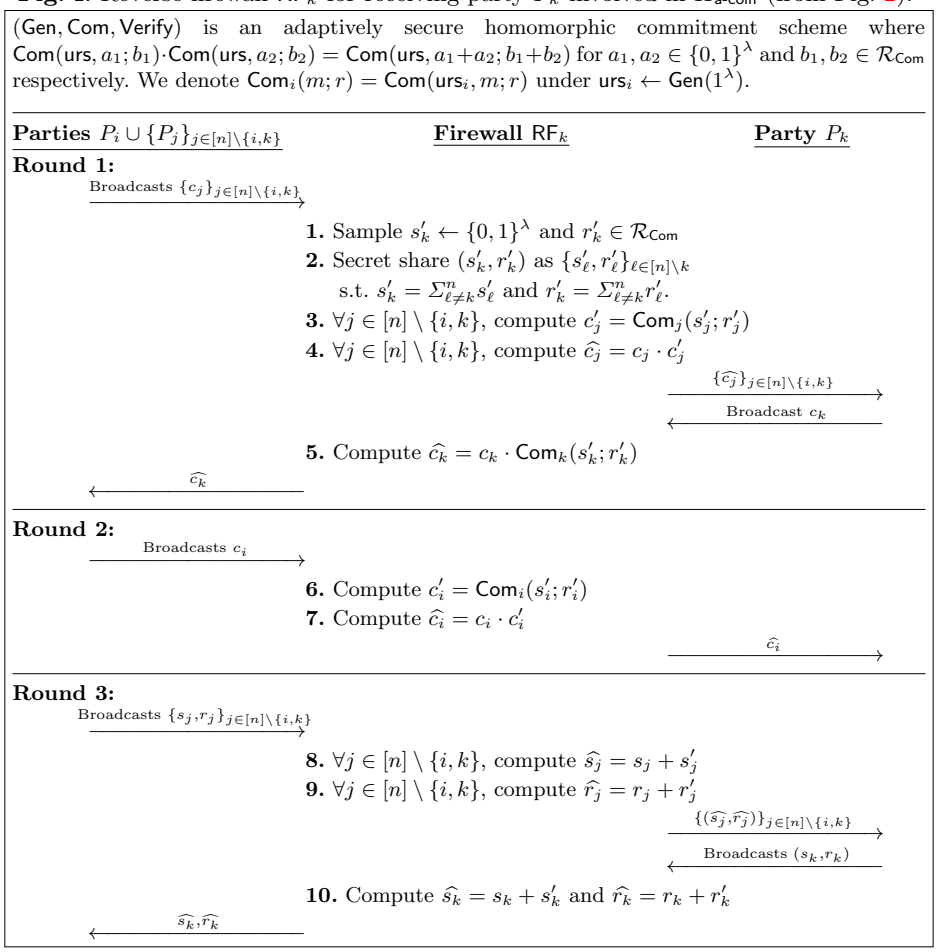
Theorem 7. *$\Pi^{\mathsf{RF}}_{\mathsf{a\text{-}coin}}$ securely implements the augmented coin-tossing functionality (Def. 4) in the* urs *model against adaptive corruption of parties, and in the presence of functionality-maintaining tampering of honest parties.*

Instantiation. We instantiate the adaptively secure homomorphic commitment in the urs model using the recent construction of [10]. It is additively homomorphic in the message and randomness space and can be instantiated based on DDH assumption in the urs model.

5.2 Adaptively-Secure ZK in the $\mathsf{urs}_{\mathsf{zk}}$ model

We construct our adaptively secure ZK protocol Π_{zk} in the common random string model based on the recent ZK protocol of [11] by incorporating a coin tossing protocol to generate the verifier's challenge. We refer to Sec. 1.2 for a

Fig. 4. Reverse firewall RF_k for receiving party P_k involved in $\Pi_{\mathsf{a\text{-}coin}}$ (from Fig. 2).



high level overview and defer the protocol to the full version. The security of our protocol is summarized below.

Theorem 8. *If* Com *is a non-interactive equivocal commitment scheme in the* urs *model and* PKE *is an IND-CPA public key encryption scheme (where the public key is statistically close to a random string) with oblivious ciphertext sampleability, then* Π_{zk} *realizes* $\mathcal{F}_{\mathsf{ZK}}$ *for all NP relations against adaptive corruptions in the* urs *model.*

We provide a firewall $\mathsf{RF}_{\mathsf{zk}}$ which provides security for a tampered prover (respectively, verifier) against a corrupt verifier (respectively, prover) in the full version. We summarize our result below.

Theorem 9. *Let* $\mathsf{RF}_{\mathsf{zk}}$ *be a reverse firewall for a tampered prover (resp. verifier) against a corrupt verifier (resp. prover) in* Π_{zk}*, and* Π_{zk} *implements* $\mathcal{F}_{\mathsf{ZK}}$ *func-*

tionality against adaptive corruption of parties. Let Com *be a non-interactive equivocal commitment scheme in the* urs *model and* PKE *be an IND-CPA public key encryption scheme (where the public key is statistically close to a random string) with oblivious ciphertext sampleability. If* Com *and* PKE *are homomorphic with respect to the (addition) operation defined over the underlying spaces (i.e, the message space, randomness space) and the message space of* PKE *is same as randomness space of* Com *then the firewall* $\mathsf{RF}_{\mathsf{zk}}$ *is transparent, functionality-maintaining and provides weak exfiltration resistance for a tampered prover (resp. verifier) against a corrupt verifier (resp. prover). The firewall also detects failures for all the parties.*

From Thm. 9, and our implication from Thm. 4, we conclude weak security preservation; we have that $\Pi_{\mathsf{zk}}^{\mathsf{RF}_{\mathsf{zk}}}$ securely implements $\mathcal{F}_{\mathsf{ZK}}$ in the presence of adaptive corruptions as summarized in Thm. 10 below.

Theorem 10. $\Pi_{zk}^{\mathsf{RF}_{\mathsf{zk}}}$ *securely implements the* $\mathcal{F}_{ZK}$ *functionality in the* urs *model against adaptive corruption of parties, and in the presence of functionality-maintaining tampering of honest parties.*

We instantiate the commitment using the LWE-based construction of [27] and we instantiate the PKE using the LWE-based construction of [25].

5.3 Adaptively-Secure MPC in the $\mathsf{urs}_{\mathsf{mpc}}$ model

We present our actively-secure protocol Π_{mpc} which withstands adaptive corruption of parties in Fig. 5, 6. Adaptive security is achieved by the adaptive security of the underlying primitives and subprotocols – ZK, augmented coin-tossing, semi-honest MPC protocol and commitments.

Theorem 11. *Assuming* Com *is an adaptively secure commitment in the* $\mathsf{urs}_{\mathsf{com}}$ *model,* $\Pi_{a\text{-}coin}$ *securely implements the augmented coin-tossing functionality against adaptive corruption of parties in the* $\mathsf{urs}_{\mathsf{a\text{-}coin}}$ *model,* Π_{zk} *securely implements the* $\mathcal{F}_{ZK}$ *functionality against adaptive corruption of parties in the* $\mathsf{urs}_{\mathsf{zk}}$ *model and* $\Pi_{sh\text{-}mpc}$ *is an adaptively-secure semi-honest MPC protocol in the* $\mathsf{urs}_{\mathsf{sh\text{-}mpc}}$ *model,* Π_{mpc} *is an actively secure MPC protocol that withstands adaptive corruption of parties in the* $\mathsf{urs}_{\mathsf{mpc}} = (\mathsf{urs}_{\mathsf{com}}, \mathsf{urs}_{\mathsf{zk}}, \mathsf{urs}_{\mathsf{a\text{-}coin}}, \mathsf{urs}_{\mathsf{sh\text{-}mpc}})$ *model.*

Next, we consider a reverse firewall $\mathsf{RF}^i_{\mathsf{mpc}} = (\mathsf{RF}^i_{\mathsf{zk}}, \mathsf{RF}^i_{\mathsf{a\text{-}coin}})$ to be the firewall for P_i in Π_{mpc}. $\mathsf{RF}^i_{\mathsf{mpc}}$ is obtained by first applying $\mathsf{RF}^i_{\mathsf{zk}}$ to the messages of Π_{zk} phase of Π_{mpc}, followed by application of $\mathsf{RF}^i_{\mathsf{a\text{-}coin}}$ to the messages in the $\Pi_{\mathsf{a\text{-}coin}}$ phase, if $\mathsf{RF}^i_{\mathsf{zk}}$ did not output $\perp$. We show that $\mathsf{RF}^i_{\mathsf{mpc}}$ provides weak exfiltration resistance for party P_i in Π_{mpc}.

Theorem 12. *Let* Com *be an adaptively secure commitment in the* $\mathsf{urs}_{\mathsf{com}}$ *model,* $\Pi_{a\text{-}coin}$ *securely implement the augmented coin tossing functionality against adaptive corruption of parties in the* $\mathsf{urs}_{\mathsf{a\text{-}coin}}$ *model,* Π_{zk} *securely implement the* $\mathcal{F}_{ZK}$ *functionality against adaptive corruption of parties in the* $\mathsf{urs}_{\mathsf{zk}}$ *model, and*

Fig. 5. Adaptively secure Multi-party Protocol Π_{mpc} in the urs model

- **Primitives:** $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Verify})$ is an adaptively secure non-interactive commitment scheme and $\Pi_{\mathsf{zk}} = (\mathsf{Gen}, \mathsf{P}_1, \mathsf{V}_1, \mathsf{P}_2, \mathsf{V}_2)$ is an adaptively secure rerandomizable input-delayed protocol implementing $\mathcal{F}_{\mathsf{ZK}}$ in the $\mathsf{urs}_{\mathsf{zk}}$ model.
- **Subprotocols:** Adaptively secure augmented coin-tossing protocol $\Pi_{\mathsf{a\text{-}coin}}$ in the $\mathsf{urs}_{\mathsf{a\text{-}coin}}$ model and adaptively secure semi-honest k-*round* MPC protocol $\Pi_{\mathsf{sh\text{-}mpc}}$ in the $\mathsf{urs}_{\mathsf{sh\text{-}mpc}}$ model. Each invocation of $\Pi_{\mathsf{a\text{-}coin}}$ generates λ for initiating party.
- **Inputs:** Each party gets $\mathsf{urs}_{\mathsf{mpc}} = (\mathsf{urs}_{\mathsf{com}}, \mathsf{urs}_{\mathsf{zk}}, \mathsf{urs}_{\mathsf{a\text{-}coin}}, \mathsf{urs}_{\mathsf{sh\text{-}mpc}})$ and security parameter 1^λ. Party P_i has private input x_i for $i \in [n]$.
- **Output:** Parties compute the ideal functionality $\mathcal{F}_{\mathsf{mpc}}$ and output $y = \mathcal{F}_{\mathsf{mpc}}(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n)$.
- **Notations:** Let $\mathcal{T}^{<T} = \bigcup_{t\in[T-1]}\{\mathcal{T}_i^t\}_{i\in[n]}$ denote the entire transcript of $\Pi_{\mathsf{sh\text{-}mpc}}$ until the end of round $T-1$ for $0 < T \leq k$. We assume $\mathcal{T}^0 = \perp$. NMF is the next message function of $\Pi_{\mathsf{sh\text{-}mpc}}$ for P_i computing $\mathsf{NMF}(\mathcal{T}^{<t}, x_i, r_i) = \mathcal{T}_i^t$. Let $N\lambda$ = no. of random bits required by P_i to commit $|x_i|$ bits + no. of random bits required by P_i to compute $\Pi_{\mathsf{sh\text{-}mpc}}$. $\gamma^{\ell}_{\mathsf{inp},i,j}$ denotes the ℓ-th ($\ell \in [3]$) round message in the ZK proof for $\mathcal{R}_{\mathsf{inp}}$ where P_i is the prover and P_j is the verifier. Similarly, $\gamma^{\ell}_{t,i,j}$ (where $t \in [k]$) denotes the ℓ-th ($\ell \in [3]$) round message in the ZK proof for $\mathcal{R}_{\mathsf{mpc}}$ where P_i is the prover and P_j is the verifier.
- **Relations:** Let $\mathcal{R}_{\mathsf{inp}}((c, c^{\mathsf{inp}}), (x, r, s)) = 1$ iff $(c = \mathsf{Com}(x; r) \wedge c^{\mathsf{inp}} = \mathsf{Com}(r; s))$. Let $\mathcal{R}_{\mathsf{mpc}}((\mathcal{T}, \mathcal{T}', c, c^{\mathsf{inp}}, c'), (x, r, s, r', s')) = 1$ iff $(c = \mathsf{Com}(x; r) \wedge c^{\mathsf{inp}} = \mathsf{Com}(r; s) \wedge c = \mathsf{Com}(r'; s') \wedge \mathsf{NMF}(\mathcal{T}', x, r') = \mathcal{T})$.

Offline Phase:

The parties run the following protocols in parallel:

- For $i \in [n]$, Party P_i invokes $\Pi_{\mathsf{a\text{-}coin}}$ N times in parallel as the initiating party to obtain randomness for input commitment - $(r_i^{\mathsf{inp}}, s_i^{\mathsf{inp}})$ s.t. $c_i^{\mathsf{inp}} = \mathsf{Com}(r_i^{\mathsf{inp}}; s_i^{\mathsf{inp}})$, and randomness for MPC protocol $(r_i^{\mathsf{mpc}}, s_i^{\mathsf{mpc}})$ s.t. $c_i^{\mathsf{mpc}} = \mathsf{Com}(r_i^{\mathsf{mpc}}; s_i^{\mathsf{mpc}}) = \{r_i^{\mathsf{mpc}}[t], c_i^{\mathsf{mpc}}[t]\}_{t\in[k]}$. Every party obtains c_i^{inp} and c_i^{mpc}.
- For $i \in [n], j \in [n] \setminus i$, Party P_i runs $\Pi_{\mathsf{zk}}.\mathsf{P}_1(1^\lambda, 1^{|\mathcal{R}_{\mathsf{inp}}|})$ as prover with P_j as verifier to obtain $\gamma^1_{\mathsf{inp},i,j}$. Upon obtaining $\gamma^1_{\mathsf{inp},i,j}$, P_j runs $\mathsf{V}_2 = \Pi_{\mathsf{zk}}(1^\lambda)$ on $\gamma^1_{\mathsf{inp},i,j}$ to obtain $\gamma^2_{\mathsf{inp},i,j}$. P_i and P_j obtain $(\gamma^1_{\mathsf{inp},i,j}, \gamma^2_{\mathsf{inp},i,j})$. P_i aborts if $\gamma^2_{\mathsf{inp},i,j}$ is invalid.
- For $i \in [n], j \in [n] \setminus j, t \in [k]$, Party P_i runs $\Pi_{\mathsf{zk}}.\mathsf{P}_1(1^\lambda, 1^{|\mathcal{R}_{\mathsf{mpc}}|})$ as prover with P_j as verifier to obtain $\gamma^1_{t,i,j}$. Upon obtaining $\gamma^1_{t,i,j}$, P_j runs $\mathsf{V}_2 = \Pi_{\mathsf{zk}}(1^\lambda)$ on $\gamma^1_{t,i,j}$ to obtain $\gamma^2_{t,i,j}$. P_i and P_j obtain $(\gamma^1_{t,i,j}, \gamma^2_{t,i,j})$. P_i aborts if $\gamma^2_{t,i,j}$ is invalid.

Online Phase:

Each party P_i (for $i \in [n]$) performs the following :

Input Commitment Phase:

- Each party P_i commits to his input x_i as $c_i = \mathsf{Com}(x_i; r_i^{\mathsf{inp}})$ and broadcasts c_i.
- For each $j \in [n] \setminus i$, party P_i proves honest computation of c_i using the committed randomness. P_i computes proof $\gamma^3_{\mathsf{inp},i,j} = \Pi_{\mathsf{zk}}.\mathsf{P}_2$ for $\mathcal{R}_{\mathsf{inp}}((c_i, c_i^{\mathsf{inp}}), (x_i, r_i^{\mathsf{inp}}, s_i^{\mathsf{inp}}))$ on $(\gamma^1_{\mathsf{inp},i,j}, \gamma^2_{\mathsf{inp},i,j})$. P_i sends $\Gamma_{\mathsf{inp}} = (\gamma^1_{\mathsf{inp},i,j}, \gamma^2_{\mathsf{inp},i,j}, \gamma^3_{\mathsf{inp},i,j})$ to P_j.

Fig. 6. Adaptively-Secure Multi-party Protocol in the urs model(cont.)

Local Computation at the end of Input Commitment Phase:
After receiving the n commitments party P_i aborts if $\exists j \in [n]$ s.t. $\Pi_{\mathsf{zk}}.\mathsf{V}_2(\Gamma_{\mathsf{inp},j,i}) = 0$.

. . . . . .

Round $1 \leq t \leq k$*:*

- If any party aborts at the end of round $t-1$ then abort.
- P_i computes the t-th round message of the MPC protocol as $\mathcal{T}^t = \mathsf{NMF}(\mathcal{T}^{<t}, x_i, r_i^{\mathsf{mpc}}[t])$. P_i broadcasts $\mathcal{T}_i^t$.
- For each $j \in [n] \setminus i$, party P_i proves honest computation of $\mathcal{T}_i^t$ to P_j using the committed input x_i and committed randomness $r_i^{\mathsf{mpc}}[t]$. P_i computes proof $\gamma^3_{t,i,j} = \Pi_{\mathsf{zk}}.\mathsf{P}_2$ for $\mathcal{R}_{\mathsf{mpc}}((\mathcal{T}_i^t, \mathcal{T}^{<t}, c_i, c_i^{\mathsf{inp}}, c_i^{\mathsf{mpc}}[t]), (x_i, r_i^{\mathsf{inp}}, s_i^{\mathsf{inp}}, r_i^{\mathsf{mpc}}[t], s_i^{\mathsf{mpc}}[t]))$ on $(\gamma^1_{t,i,j}, \gamma^2_{t,i,j})$. P_i sends $\Gamma_{t,i,j} = (\gamma^1_{t,i,j}, \gamma^2_{t,i,j}, \gamma^3_{t,i,j})$ to P_j.

Local Computation at the end of Round t:

- Party P_i aborts if $\exists j \in [n] \setminus i$ s.t. $\Pi_{\mathsf{zk}}.\mathsf{V}_2(\Gamma_{t,j,i}) = \mathsf{reject}$.
- For $i \in [n]$, P_i sets $\mathcal{T}^{<t+1} = \mathcal{T}^{<t} \bigcup \{\mathcal{T}_j^t\}_{j\in[n]}$ and continues to next round.

Output Computation for Party $\{P_i\}_{i\in[n]}$ *:*

- If any party aborts at the end of round k then abort.
- P_i sets $\mathcal{T}^{\leq k} = \mathcal{T}^{<k} \bigcup \{\mathcal{T}_j^k\}_{j\in[n]}$ and outputs $y = \mathsf{NMF}(\mathcal{T}^{\leq k}, x_i, r_i^{\mathsf{mpc}}[k])$.

$\Pi_{\mathsf{sh\text{-}mpc}}$ be an adaptively-secure semi-honest MPC protocol in the $\mathsf{urs}_{\mathsf{sh\text{-}mpc}}$ model. Let $\mathsf{RF}^i_{\mathsf{zk}}$ and $\mathsf{RF}^i_{\mathsf{a\text{-}coin}}$ be transparent, functionality-maintaining, and weakly exfiltration resistant for P_i in Π_{zk} and $\Pi_{\mathsf{a\text{-}coin}}$ respectively. Then, RF^i_{mpc} is a transparent, functionality-maintaining, weakly exfiltration RF for P_i in Π_{mpc} .

The proof of Thm. 12 is deferred to the full version.

We have shown in Thm. 11 that Π_{mpc} is adaptively secure. Now, consider $\Pi^{\mathsf{RF}}_{\mathsf{mpc}}$, a firewalled version of the protocol Π_{mpc} where all honest parties P_i have their respective firewalls RF_i attached to them. From Thm. 12, and our implication from Thm. 4, we conclude weak security preservation.

Theorem 13. *Π^{RF}_{mpc} is an actively secure MPC protocol against adaptively corruption of parties, and in the presence of functionality maintaining tampering of honest parties.*

Instantiation. We obtain Π_{zk} and $\Pi_{\mathsf{a\text{-}coin}}$ based on LWE and DDH assumptions respectively. We assume that Π_{zk} setup consists of n setup strings –$\{\mathsf{urs}^i_{\mathsf{zk}}\}_{i\in[n]}$, where $\mathsf{urs}^i_{\mathsf{zk}}$ is used by party P_i to prove statements. Similarly, $\Pi_{\mathsf{a\text{-}coin}}$ consists of n setup strings – $\{\mathsf{urs}^i_{\mathsf{a\text{-}coin}}\}_{i\in[n]}$, where $\mathsf{urs}^i_{\mathsf{a\text{-}coin}}$ is used in a session where party P_i is the initiating party. The commitment scheme can be instantiated from the adaptively secure commitment of [10] based on DDH in the $\mathsf{urs}_{\mathsf{com}}$ model. We assume that Com consists of n setup strings – $\{\mathsf{urs}^i_{\mathsf{com}}\}_{i\in[n]}$, where $\mathsf{urs}^i_{\mathsf{com}}$ is used by party P_i to commit. We obtain an adaptively secure semi-honest MPC protocol in the urs model as follows. The work of [9] obtain a two-round semi-honest adaptively secure MPC protocol based on adaptively secure two-round

OT protocol and augmented NCE. The work of [10] construct an adaptively secure two-round OT protocol from DDH in urs model and instantiate [15] the augmented NCE from DDH; thus yielding a two-round adaptively secure MPC protocol in the urs model from DDH. We instantiate $\Pi_{\mathsf{sh\text{-}mpc}}$ using the DDH-based MPC protocol of [10].

Round complexity. In Π_{mpc} the subprotocols $\Pi_{\mathsf{a\text{-}coin}}$ and the first two rounds of Π_{zk} can be run in parallel during the offline phase. Thus, the offline phase requires 3 rounds in total. The input commitment phase requires 1 round and $\Pi_{\mathsf{sh\text{-}mpc}}$ requires 2 rounds when instantiated using the protocol of [10]. We get a 6 round MPC protocol from DDH and LWE.

6 Adaptively-Secure Multi-party Coin-Tossing Protocol with Reverse Firewalls in the Plain Model

In this section we give a protocol in the plain model with reverse firewalls to generate the setup string $\mathsf{urs}_{\mathsf{mpc}}$ required for Π_{mpc}. A high level overview of our construction can be found in Sec. 1.2 and our protocol is presented in Fig. 7. Our protocol satisfies security against adaptive corruptions in the plain model and its security is summarized in Thm. 14 (proven in the full version).

Theorem 14. *Let Discrete Log and Knowledge of Exponent Assumptions hold in a bilinear group $\mathbb{G}$ and PKE is a public key encryption with oblivious ciphertext sampling, oblivious public key sampling, satisfying additive homomorphism over key space, message space, randomness space and ciphertext space with public key space being $\mathbb{Z}_q$. Π_{coin}(Fig. 7) securely implements coin-tossing functionality(Def. 3) against adaptive corruptions in the plain model.*

Next, we turn to constructing a reverse firewall for Π_{coin}. We provide a reverse firewall RF_i for the tampered honest party P_i in Fig. 8. Weak ER for P_i is summarized in Thm. 15 and is proved in the full version.

Theorem 15. *Let RF_i be the reverse firewall for party P_i in Π_{coin}. If Discrete Log assumption and Knowledge of Exponent assumption holds in a bilinear group $\mathbb{G}$ and PKE is a public key encryption with oblivious ciphertext sampling and oblivious public keys sampling satisfying additive homomorphism over key space, message space, randomness space and ciphertext space and public key space of PKE be $\mathbb{Z}_q$. Then RF_i is transparent, functionality maintaining and provides weak exfiltration resistance for party P_i against every other party $\{P_j\}_{j\in[n]\setminus i}$ with valid transcripts, and detects failure for P_i.*

In the final protocol every honest party $\{P_j\}_{j\in\mathsf{H}}$ will have a firewall $\{\mathsf{RF}_j\}_{j\in\mathsf{H}}$ and the firewalls will be composed together. By applying the result of Thm. 4 we obtain the following result.

Theorem 16. *If Discrete Log and Knowledge of Exponent assumptions hold in a bilinear group $\mathbb{G}$ and PKE is a public key encryption with oblivious ciphertext*

Fig. 7. Adaptively-Secure Multi-party Coin-Tossing Protocol Π_{coin} using Reverse Firewalls without Setup

– **Public Inputs:** Each party gets as input the group $\mathbb{G}$ where DLP assumption holds. Every element $g' \in \mathbb{G} \setminus 1$ is a generator. Every party also receives a generator g as input and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{H}$. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{oGen}, \mathsf{oEnc})$ is a public key encryption with oblivious ciphertext sampling and oblivious public keys sampling satisfying additive homomorphism over key space, message space, randomness space and ciphertext space. Let the public key space of PKE be $\mathbb{Z}_q$.

Parameter Generation Phase:
The following protocol steps are repeated by every party P_i for $i \in [n]$ with every party P_j for $j \in [n] \setminus i$ to generate the pair-wise commitment parameter h_{ij} where P_i is the committer and P_j is the verifier. We denote it as h and the pairwise communication without any subscript ij to avoid notation overloading.

1. P_j samples an $R \leftarrow \mathbb{G}$. P_j sends R to P_i.
2. P_i aborts if $R \in \{1, g\}$. P_i samples $a_1, u \leftarrow \mathbb{Z}_q$ and computes $A_1 = g^{a_1}$ and commits to a_1 using randomness u as $v = A_1 R^u$. P_i also samples $\mathsf{pk}_1 \leftarrow \mathsf{oGen}(1^\lambda)$ and commits to it as $v_p = g^{\mathsf{pk}_1} R^{u_p}$ for $u_p \leftarrow \mathbb{Z}_q$. P_i sends (v, v_p) to P_j.
3. P_j samples $A_2 \leftarrow \mathbb{G}, \mathsf{pk}_2 \leftarrow \mathsf{oGen}(1^\lambda)$, and sends (A_2, pk_2) to P_i.
4. P_i opens commitment v_p by sending (u_p, pk_1). P_i also sends u to P_j. P_i computes $\mathsf{pk} = \mathsf{pk}_1 + \mathsf{pk}_2$.
5. P_j aborts if (u_p, pk_1) is not a valid opening of v_p. Else, P_j computes $\mathsf{pk} = \mathsf{pk}_1 + \mathsf{pk}_2$. P_j computes $A_1 = \frac{v}{R^u}$ and sets $A = A_1 \cdot A_2$. P_j samples commitment trapdoor $t \leftarrow \mathbb{Z}_q$ and sets commitment parameter as $h = g^t$. P_j sends (h, A^t) to P_i.
6. P_i computes $A = A_1 \cdot A_2$ and aborts if $e(h, A) \neq e(g, Z)$ where P_i received (h, Z) from P_j. Else, P_i proves knowledge of discrete log of A_1 by sending a_1 to P_j. If $v \neq g^{a_1} R^u$ then P_j aborts. Else, set (g, h) as the pair-wise commitment parameter and pk as the pairwise encryption parameter.

Commitment Generation Phase:
Every party P_i chooses a random coin $s_i \leftarrow \{0, 1\}$, and commits to it pairwise. For $i \in [n]$, every party P_i and constructs $c_{ij} = g^{s_i} h_{ij}^{d_{ij}}$ where h_{ij} is the pairwise commitment parameter. P_i also encrypts the commitment randomness as $e_{ij,s_i} = \mathsf{Enc}(\mathsf{pk}_{ij}, d_{ij}; y_{ij})$ using randomness y_{ij} and samples $e_{ij,\overline{s_i}} \leftarrow \mathsf{oEnc}(\mathsf{pk}_{ij})$, where pk_{ij} is the pair-wise encryption parameter. P_i sends $(c_{ij}, e_{ij,0}, e_{ij,1})$ to P_j.
Commitment Opening Phase:
For all $i \in [n]$, party P_i broadcasts s_i. P_i opens c_{ij} pairwise (for all $j \in [n] \setminus i$) by sending (d_{ij}, y_{ij}) and claims that $e_{ij,\overline{s_i}}$ was obliviously sampled.

Output Phase:
For all $i \in [n]$, party P_i verifies the commitments $c_{ji} \stackrel{?}{=} g^{s_j} h_{ji}^{d_{ji}}$ and $e_{ji,s_j} = \mathsf{Enc}(\mathsf{pk}_{ji}, d_{ji}; y_{ji})$. If all verification checks pass then P_i sets $S = (\Sigma_{k \in [n]} s_k) \mod 2$ and outputs S as the final random coin.

Fig. 8. Reverse Firewall RF_i for Party P_i in Π_{coin}

Parameter Generation Phase:
The following protocol steps are repeated for every party P_j for $j \in [n] \setminus i$ where P_i is the committer and P_j is the verifier:

1. When P_j sends R, RF_i samples an $r \leftarrow \mathbb{Z}_q$ and sends $\hat{R} = R^r$ to P_i.
2. When P_i sends $v = A_1 \hat{R}^u$, RF_i forwards $\hat{v} = \hat{A_1} R^{\hat{u}}$ to P_j where $\hat{A_1} = A_1 \cdot \tilde{A}$, $\hat{v} = v \cdot \tilde{A} \cdot R^{\tilde{u}}$ and $\hat{u} = ru + \tilde{u}$ for random values $\tilde{a}, \tilde{u} \leftarrow Zq$ and $\tilde{A} = g^{\tilde{a}}$. When P_i sends $v_p = g^{\mathsf{pk}_1} \hat{R}^{u_p}$, RF_i forwards $\hat{v_p} = g^{\hat{\mathsf{pk}_1}} R^{\hat{u_p}}$ to P_j where $\hat{\mathsf{pk}_1} = \mathsf{pk}_1 + \tilde{\mathsf{pk}}$, $\hat{v} = v \cdot g^{\tilde{\mathsf{pk}}} \cdot R^{\tilde{u_p}}$ and $\hat{u} = ru_p + \tilde{u_p}$ for random values $\tilde{u_p} \leftarrow Zq$ and $\tilde{\mathsf{pk}} = \mathsf{oGen}(1^\lambda)$.
3. When P_j sends (A_2, pk_2), RF_i forwards $\hat{A_2} = A_2 \cdot \tilde{A}$ and $\hat{\mathsf{pk}_2} = \mathsf{pk}_2 + \tilde{\mathsf{pk}}$ to P_i. P_i computes $\hat{\mathsf{pk}} = \mathsf{pk}_1 + \hat{\mathsf{pk}_2} = \mathsf{pk}_1 + \mathsf{pk}_2 + \tilde{\mathsf{pk}}$.
4. When P_i sends commitment randomness (u, u_p, pk_1), RF_i drops the message if $v_p \neq g^{u_p} \hat{R}^{\mathsf{pk}_1}$. Else, RF_i forwards $(\hat{u}, \hat{u_p}, \hat{\mathsf{pk}_1})$ to P_j.
5. P_j computes $\hat{\mathsf{pk}} = \hat{\mathsf{pk}_1} + \mathsf{pk}_2 = \mathsf{pk}_1 + \mathsf{pk}_2 + \tilde{\mathsf{pk}}$. P_j computes $\hat{A} = A_1 \cdot A_2 \cdot \tilde{A}$. When P_j sends $(h, Z) = (h, \hat{A}^t)$ drop the message if $e(h, \hat{A}) \neq e(g, Z)$. Else, sample a $\tilde{t} \leftarrow \mathbb{Z}_q$ and compute $\hat{h} = h^{\tilde{t}}$. Send $(\hat{h}, \hat{Z}) = (\hat{h}, Z^{\tilde{t}})$ to P_i. P_j sets h as the parameter and P_i sets $\hat{h} = h^{\tilde{t}}$ as the parameter. P_i and P_j sets $\hat{\mathsf{pk}}$ as the pairwise public key parameter.
6. When P_i sends a_1, RF_i drops the message if $v \neq g^{a_1} \hat{R}^u$. Else, RF_i forwards $\hat{a_1} = a_1 + \tilde{a}$ to P_j.

The above steps are also repeated when P_i is the verifier and P_j is the committer.
Commitment Generation Phase:
RF_i chooses a random coin $\tilde{s_i}$ and computes $\{\tilde{s_{ji}}\}_{j \in [n] \setminus i}$ randomly such that $\Sigma_{j \in [n] \setminus i} \tilde{s_{ji}} = \tilde{s_i}$. RF_i performs the following :

- *P_i is the committer:* When P_i sends a commitment $(c_{ij}, e_{ij,0}, e_{ij,1})$ compute $\hat{c_{ij}} = g^{\hat{s_i}} h_{ij}^{\hat{d_{ij}}} = c_{ij} \cdot g^{\tilde{s_i}} \cdot h_{ij}^{\tilde{d_{ij}}}$ where $\hat{s_i} = s_i + \tilde{s_i}$ and $\hat{d_{ij}} = d_{ij} \cdot \tilde{t} + \tilde{d_{ij}}$ for $\tilde{d_{ij}} \leftarrow \mathbb{Z}_q$. Set $\hat{e_{ij,0}} = \tilde{t} \cdot e_{ij,0} + \mathsf{Enc}(\hat{pk}, \tilde{d_{ij}}; \tilde{y_{ij,0}})$ and $\hat{e_{ij,1}} = \tilde{t} \cdot e_{ij,1} + \mathsf{Enc}(\hat{pk}, \tilde{d_{ij}}; \tilde{y_{ij,1}})$. The firewall forwards $(\hat{c_{ij}}, \hat{e_{ij,0}}, \hat{e_{ij,1}})$ to P_j. Here $\tilde{t}, h_{ij}$ and $\hat{\mathsf{pk}}$ correspond to the run where P_j is verifier and P_i is committer.
- *P_j is the committer:* When P_j sends a commitment $(c_{ji}, e_{ji,0}, e_{ji,1})$ compute $\hat{c_{ji}} = g^{\hat{s_j}} h_{ji}^{\hat{d_{ji}}} = c_{ji} \cdot g^{\tilde{s_{ji}}} \cdot h_{ji}^{\tilde{d_{ji}}}$ where $\hat{s_j} = s_j + \tilde{s_{ji}}$ and $\hat{d_{ji}} = d_{ji} \cdot \tilde{t} + \tilde{d_{ji}}$ for $\tilde{d_{ji}} \leftarrow \mathbb{Z}_q$. Set $\hat{e_{ji,0}} = \tilde{t} \cdot e_{ji,0} + \mathsf{Enc}(\hat{pk}, \tilde{d_{ji}}; \tilde{y_{ji,0}})$ and $\hat{e_{ji,1}} = \tilde{t} \cdot e_{ji,1} + \mathsf{Enc}(\hat{pk}, \tilde{d_{ji}}; \tilde{y_{ji,1}})$. The firewall forwards $(\hat{c_{ji}}, \hat{e_{ji,0}}, \hat{e_{ji,1}})$ to P_i. Here $\tilde{t}, h_{ji}$ and $\hat{\mathsf{pk}}$ correspond to the run where P_i is verifier and P_j is committer.

Commitment Opening Phase:

- When party P_i broadcasts s_i, RF_i broadcasts $\hat{s_i}$. When P_i opens commitments by sending (d_{ij}, y_{ij}), RF_i drops the message if $c_{ij} \neq g^{s_i} \hat{h}_{ij}^{d_{ij}}$ or $e_{ij,s_i} \neq \mathsf{Enc}(\hat{\mathsf{pk}}, d_{ij}; y_{ij})$. Else, RF_i sends $(\hat{d_{ij}}, \hat{y_{ij}}) = (\tilde{t} \cdot d_{ij} + \tilde{d_{ij}}, \tilde{t} \cdot y_{ij} + \tilde{y_{ij}})$ and claims that $e_{ij,1-\hat{s_i}}$ was obliviously sampled.
- When party P_j broadcasts s_j, RF_i sends $\hat{s_j}$ to P_i. When P_j opens commitments by sending (d_{ji}, y_{ji}), RF_i drops the message if $c_{ji} \neq g^{s_j} \hat{h}_{ji}^{d_{ji}}$ or $e_{ji,s_j} \neq \mathsf{Enc}(\hat{\mathsf{pk}}, d_{ji}; y_{ji})$. Else, RF_i sends $(\hat{d_{ji}}, \hat{y_{ji}}) == (\tilde{t} \cdot d_{ji} + \tilde{d_{ji}}, \tilde{t} \cdot y_{ji} + \tilde{y_{ji}})$ and claims that $e_{ji,1-\hat{s_j}}$ was obliviously sampled.

sampling and oblivious public keys sampling satisfying additive homomorphism over key space, message space, randomness space, ciphertext space and public key space of PKE *be* $\mathbb{Z}_q$*. Then* Π_{coin} *(Fig. 7) securely implements the coin-tossing functionality (Def. 3) against adaptive corruption of parties in the plain model and in the presence of functionality maintaining tampering of honest parties.*

The FHE scheme of [25] based on LWE assumption satisfies all the properties required from the PKE. We consider $q = \max(q_{\mathsf{LWE}}, q_{\mathsf{DL}})$ where LWE holds for $q \geq q_{\mathsf{LWE}}$ and DL holds for $q \geq q_{\mathsf{DL}}$. Thus we get the result from Discrete Log, Knowledge of Exponent and LWE assumptions.

7 The Final Compiler

We now show our final result, i.e., an *adaptively* secure MPC protocol in the *plain* model that admits reverse firewalls. In particular, the reverse firewall for our final MPC protocol is obtained by combining the reverse firewall for our adaptively secure MPC protocol Π_{mpc} in the uniform random string ($\mathsf{urs}_{\mathsf{mpc}}$) model (see Section 5.3) along with the reverse firewall for our adaptively secure multi-party coin-tossing protocol Π_{coin} in the plain model (see Section 6). Let us denote the final MPC protocol (in the plain model) to be Π which is obtained by first running Π_{coin} to obtain $\mathsf{urs}_{\mathsf{mpc}}$ and then running Π_{mpc} using $\mathsf{urs}_{\mathsf{mpc}}$.

Let us consider a reverse firewall $\mathsf{RF}_i = (\mathsf{RF}^i_{\mathsf{coin}}, \mathsf{RF}^i_{\mathsf{mpc}})$ to be the firewall for a party P_i in the protocol Π. RF_i is obtained by first applying $\mathsf{RF}^i_{\mathsf{coin}}$ to the messages of Π_{coin}, followed by application of $\mathsf{RF}^i_{\mathsf{coin}}$ to the messages of Π_{mpc}, if $\mathsf{RF}^i_{\mathsf{coin}}$ did not output $\perp$. We show that RF_i provides weak ER for party P_i in Π.

We defer the proof of the theorem to the full version.

Theorem 17 (Composition Theorem for Π). *Let* Π_{mpc} *be an adaptively secure MPC protocol in the uniform random string (*$\mathsf{urs}_{\mathsf{mpc}}$*) model,* Π_{coin} *securely implement the coin-tossing functionality (see Def. 3) against adaptive corruption of parties in the plain model. Let* RF^i_{mpc} *and* $\mathsf{RF}^i_{\mathsf{coin}}$ *be transparent, functionality-maintaining, and weakly exfiltration-resistant reverse firewalls for some party* P_i *in* Π_{mpc} *and* Π_{coin} *respectively. Then* RF_i *is transparent, functionality-maintaining, and weakly exfiltration-resistant reverse firewall for party* P_i *in the protocol* Π*.*

References

1. M. Abe and S. Fehr. Perfect NIZK with adaptive soundness. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 118–136. Springer, Heidelberg, Feb. 2007.
2. G. Ateniese, B. Magri, and D. Venturi. Subversion-resilient signature schemes. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 364–375. ACM Press, Oct. 2015.
3. B. Auerbach, M. Bellare, and E. Kiltz. Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 348–377. Springer, Heidelberg, Mar. 2018.

4. J. Ball, J. Borger, G. Greenwald, et al. Revealed: how us and uk spy agencies defeat internet privacy and security. *Know Your Neighborhood*, 2013.
5. M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, Dec. 2016.
6. M. Bellare, J. Jaeger, and D. Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 1431–1440. ACM Press, Oct. 2015.
7. M. Bellare, K. G. Paterson, and P. Rogaway. Security of symmetric encryption against mass surveillance. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, Aug. 2014.
8. P. Bemmann, R. Chen, and T. Jager. Subversion-resilient public key encryption with practical watchdogs. In *PKC 2021, Virtual Event, May 10-13, 2021, Proceedings, Part I*, volume 12710 of *LNCS*, pages 627–658. Springer, 2021.
9. F. Benhamouda, H. Lin, A. Polychroniadou, and M. Venkitasubramaniam. Two-round adaptively secure multiparty computation from standard assumptions. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 175–205. Springer, Heidelberg, Nov. 2018.
10. R. Canetti, P. Sarkar, and X. Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. In *Advances in Cryptology - ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 277–308. Springer, 2020.
11. R. Canetti, P. Sarkar, and X. Wang. Triply adaptive uc nizk. Cryptology ePrint Archive, Report 2020/1212, 2020. https://eprint.iacr.org/2020/1212.
12. S. Chakraborty, S. Dziembowski, and J. B. Nielsen. Reverse firewalls for actively secure MPCs. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 732–762. Springer, Heidelberg, Aug. 2020.
13. R. Chen, X. Huang, and M. Yung. Subvert KEM to break DEM: practical algorithm-substitution attacks on public-key encryption. In *ASIACRYPT 2020, Proceedings, Part II*, volume 12492 of *LNCS*, pages 98–128. Springer, 2020.
14. R. Chen, Y. Mu, G. Yang, W. Susilo, F. Guo, and M. Zhang. Cryptographic reverse firewall via malleable smooth projective hash functions. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 844–876. Springer, Heidelberg, Dec. 2016.
15. S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Improved non-committing encryption with applications to adaptively secure protocols. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 287–302. Springer, Heidelberg, Dec. 2009.
16. E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo. True2F: Backdoor-resistant authentication tokens. In *2019 IEEE Symposium on Security and Privacy*, pages 398–416. IEEE Computer Society Press, May 2019.
17. J. P. Degabriele, P. Farshim, and B. Poettering. A more cautious approach to security against mass surveillance. In *International Workshop on Fast Software Encryption*, pages 579–598. Springer, 2015.
18. J. P. Degabriele, K. G. Paterson, J. C. N. Schuldt, and J. Woodage. Backdoors in pseudorandom number generators: Possibility and impossibility results. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 403–432. Springer, Heidelberg, Aug. 2016.
19. Y. Dodis, C. Ganesh, A. Golovnev, A. Juels, and T. Ristenpart. A formal treatment of backdoored pseudorandom generators. In E. Oswald and M. Fischlin, editors,

EUROCRYPT 2015, Part I, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, Apr. 2015.
20. Y. Dodis, I. Mironov, and N. Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 341–372. Springer, Heidelberg, Aug. 2016.
21. U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999.
22. M. Fischlin and S. Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 76–90. IEEE, 2018.
23. C. Ganesh, B. Magri, and D. Venturi. Cryptographic reverse firewalls for interactive proof systems. In A. Czumaj, A. Dawar, and E. Merelli, editors, *ICALP 2020*, volume 168 of *LIPIcs*, pages 55:1–55:16. Schloss Dagstuhl, July 2020.
24. S. Garg and A. Sahai. Adaptively secure multi-party computation with dishonest majority. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 105–123. Springer, Heidelberg, Aug. 2012.
25. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, Aug. 2013.
26. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In A. Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
27. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In R. A. Servedio and R. Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.
28. I. Mironov and N. Stephens-Davidowitz. Cryptographic reverse firewalls. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Heidelberg, Apr. 2015.
29. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Cliptography: Clipping the power of kleptographic attacks. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Heidelberg, Dec. 2016.
30. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Generic semantic security against a kleptographic adversary. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 907–922. ACM Press, Oct. / Nov. 2017.
31. D. Shumow and N. Ferguson. On the possibility of a back door in the nist sp800-90 dual ec prng. In *Proc. Crypto*, volume 7, 2007.
32. G. J. Simmons. Authentication theory/coding theory. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 411–431. Springer, Heidelberg, Aug. 1984.
33. A. Young and M. Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, Aug. 1996.