Secure Computation Against Adaptive Auxiliary Information

Elette Boyle[1⋆], Sanjam Garg[2], Abhishek Jain[3⋆], Yael Tauman Kalai[4], and Amit Sahai[2⋆⋆]

[1] MIT, eboyle@mit.edu
[2] UCLA, {sanjamg,sahai}@cs.ucla.edu
[3] MIT and Boston University, abhishek@csail.mit.edu
[4] Microsoft Research, New England, yael@microsoft.com

Abstract. We study the problem of secure two-party and multiparty computation (MPC) in a setting where a cheating polynomial-time adversary can corrupt an arbitrary subset of parties and, in addition, learn arbitrary auxiliary information on the *entire states* of all honest parties (including their inputs and random coins), in an *adaptive* manner, *throughout the protocol execution*. We formalize a definition of multiparty computation secure against adaptive auxiliary information (AAI-MPC), that intuitively guarantees that such an adversary *learns no more than the function output and the adaptive auxiliary information*. In particular, if the auxiliary information contains only partial, "noisy," or computationally invertible information on secret inputs, then *only* such information should be revealed.

We construct a universally composable AAI two-party and multiparty computation protocol that realizes any (efficiently computable) functionality against malicious adversaries in the common reference string model, based on the linear assumption over bilinear groups and the n-th residuosity assumption. Apart from theoretical interest, our result has interesting applications to the regime of leakage-resilient cryptography.

At the heart of our construction is a new two-round oblivious transfer protocol secure against *malicious* adversaries who may receive adaptive auxiliary information. This may be of independent interest.

1 Introduction

Historically, when formalizing security definitions for cryptographic protocols, it was noted that adversarial parties may enter a protocol with relevant knowledge *from the past*. Meaningful security definitions should thus embed the important

⋆ Supported by NSF Contract CCF-1018064 and DARPA Contract Number: FA8750-11-2-0225.

⋆⋆ Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

requirement that, even armed with such side information, an adversary still must not be able to break the security of the protocol. For example, in a zero-knowledge proof system [22], it should not be the case that an adversarial verifier with partial information about a witness, can now suddenly recover the entire witness from the protocol. This natural property was formalized by requiring that, for any adversary with any potential auxiliary input z on the inputs of the honest parties *prior* to the execution of the protocol, this adversary still learns *nothing* beyond the inputs and prescribed outputs of corrupted parties (and, of course, the auxiliary input z it learned prior).

In the last two decades, as cryptographic protocols have become increasingly prevalent (often within everyday online activities, run in parallel), and as new classes of strong attacks have emerged, it has become increasingly evident that adversaries may also acquire auxiliary information on the internal state of honest parties *during the protocol execution*. This may take place, e.g., by performing *physical attacks* on an implementation of a processor (say, a smart card or a hardware token) [31, 2, 26], or when the randomness used by an honest party in a protocol is correlated with randomness used in other applications. Unfortunately, this case is no longer covered under historical definitions: the moment an adversary is able to learn any side information, say, about the randomness of an honest party in the protocol, the security guarantees break down.

In this work, we seek to extend the standard notion of security with (static) auxiliary inputs to the setting of general *adaptive* auxiliary information. We study secure general two-party and multiparty computation [34, 21] in the setting where an adversary, who corrupts an arbitrary subset of parties in the protocol, is able to learn arbitrary (efficiently computable) auxiliary information on the *entire states* of all the honest parties (including their inputs and random coins), in an *adaptive* manner, *throughout the protocol execution*. We formalize a meaningful definition of security within this setting, and construct two-party and multiparty computation protocols satisfying our definition.

How to Define Security Against Adaptive Auxiliary Information? Security of MPC protocols is formalized by comparing the real-world protocol execution to an ideal-world experiment where the parties interact directly with a trusted party who receives all parties' inputs and responds only with the correct function output. Formally, an MPC protocol is said to be secure under the classical definition if for every real-world adversary with some auxiliary input (say) z, there exists an ideal-world adversary (a.k.a. simulator) with the same auxiliary input z, who simulates the output of the real-world experiment.

Our goal is to generalize this definition to the setting where side information can be learned during the protocol. We model *adaptive* auxiliary information by allowing the adversary to specify (efficiently computable) functions f_i, adaptively throughout the protocol. For each such query, the selected function is evaluated on the entire secret states of the honest parties, and the result is given to the adversary as auxiliary information. Intuitively, we wish to guarantee that an adversary who participates in the protocol and receives adaptive auxiliary

information throughout the protocol's lifetime still learns nothing beyond the inputs and outputs of the corrupted parties, and the auxiliary information.

Note that it is not immediately apparent how to formalize this notion. Whereas in the static setting both the adversary and the simulator receive the exact same auxiliary input z, in the adaptive setting, it doesn't make sense even syntactically to say that the *same* auxiliary input functions are applied both in the real world and in the ideal world: this is because the real-world auxiliary input function will expect to take as input the secret states of parties executing a protocol, whereas in the ideal world no protocol is ongoing.

A natural attempt to formalize security in this setting may be to require that if the real adversary learns ℓ bits of auxiliary information, then the simulator can also learn at most ℓ bits of auxiliary information. While this is a natural requirement (and our definition will achieve at least this requirement), unfortunately, it may be too weak. For example, the auxiliary information learnt by a real-world adversary (say, via physical processes) may be large but "noisy," giving very little information about the honest parties' inputs. Or, the honest parties' inputs may be information-theoretically determined but *computationally unpredictable* given the real-world auxiliary information. In these cases, the above definition may not provide a meaningful security guarantee since the simulator may be able to simulate the honest parties "trivially" by first learning a large portion (or all!) of their inputs as auxiliary information. Ideally, we would like to formalize the intuitive requirement that the auxiliary information in the ideal world is "no more" than that in the real world.

Our Security Definition. We capture the desired security notion by placing additional restrictions on the ideal-world simulator. In particular, for each auxiliary input function f that the real-world adversary generates, we require that the simulator generates a "translation function" T that takes as input only the secret inputs of the honest parties, and generates "simulated states" for the honest parties at each point in the protocol. These simulated states should be computationally indistinguishable from the real states, and should be consistent with the simulated transcript. Then, for each auxiliary input function f that is requested in the real world, the *same* auxiliary information function is applied in the ideal world, but it is applied to the simulated states. In other words, the ideal world auxiliary function will be the composed function $f \circ T$. This prevents the ideal-world adversary from "learning too much" via the ideal-world auxiliary information: for example, if the requested function f has short output, reveals only useless unused information, or leaves its inputs unpredictable, then the same restrictions will also hold for the ideal-world auxiliary information.

We say that an MPC protocol is *secure against adaptive auxiliary information* if for every PPT real-world adversary who makes arbitrary adaptive (efficiently computable) auxiliary information queries, there exists a PPT ideal-world simulator who, given the corresponding auxiliary information (as described above), is able to simulate the output of the real-world experiment. Intuitively, this definition guarantees that *the security of the honest parties "gracefully degrades" with the amount of auxiliary information that the real adversary is able to obtain.*

We remark that our definition is similar to that of [3, 20], occurring within the setting of zero-knowledge and protocols against passive adversaries.

1.1 Our Results

We construct *two-party* and *multiparty* computation protocols (for any efficiently computable function) secure against adaptive auxiliary information in the universal composability (UC) framework [7] in the common reference string (CRS) model. Namely, we prove the following theorem:

Main Theorem (Informal). *For any $n \geq 2$, there exists a UC-secure n-party protocol in the CRS model for evaluating any efficiently computable function, such that for any malicious PPT adversary who (statically) corrupts any subset of parties and learns any amount of (efficiently computable) adaptive auxiliary information Z, this adversary learns nothing beyond the inputs and outputs of corrupted parties, and the same auxiliary information Z (formalized as discussed above). This holds based on the linear assumption over bilinear groups and the n-th residuosity assumption.*[5]

No bound on the auxiliary information. We emphasize that, as in the classical (static auxiliary input) setting, our result does *not* require any a priori bound on the amount of the auxiliary information that the adversary may be able to learn. Instead, our protocol guarantees that for *any* amount of information the real-world adversary is able to (adaptively) acquire throughout the protocol, this "same amount" of auxiliary information is given to the ideal-world simulator, thus providing *graceful degradation* of security. This advantageous property is in contrast with nearly all existing results in leakage-resilient cryptography, which require the user to specify *at design time* an amount of information leakage he wishes to protect against (growing the system parameters accordingly); if an adversary is able to garner more leakage at runtime than the preset bound, then security of these schemes no longer hold.

Application to Leakage-Resilient Protocols. There has been an extensive amount of work on leakage-resilient cryptography in recent years, primarily focused on the setting of *non-interactive* primitives (see e.g., [29, 16, 1, 15, 32, 30, 6, 14]). Our result can be used to achieve leakage-resilient *interactive protocols*, an area that has received comparatively little attention (see Section 1.3).

As alluded to above, our MPC protocol directly provides meaningful security guarantees in the setting of leakage: where such a "leaky" adversary learns no more than the inputs and outputs of the corrupted parties, and the leakage information. This can be seen by viewing the adaptive auxiliary information as joint leakage on the secret states of the honest parties during the protocol execution.[6]

[5] The n-th residuosity assumption can be replaced with any lossy trapdoor function (LTDF) with some specific properties. Roughly speaking, we require LTDFs that are bijective and "sufficiently lossy".

[6] We note, however, this differs from the security model considered in [3], where leakage on the state of each party is "disjoint" in both the real- and ideal-world experiments. Indeed, achieving security in such a model is an interesting open problem.

Further, when combined with previous work on leakage-resilient cryptography, our result yields applications where *"standard"* security is guaranteed in the face of *bounded* leakage. Below, we discuss two such applications:

- *Leakage-Resilient MPC in the leak-free preprocessing model.* A recent work of Boyle et al. [4] builds upon our results to construct multi-party secure computation protocols to achieve *standard ideal-world security* (where no leakage is allowed in the ideal world) against real-world adversaries that may leak continuously from the secret state of each honest player *separately*, assuming a one-time leak-free preprocessing phase and a large number of parties. At a very high level, they achieve their result by applying our *multi-party* secure computation protocol to the leakage-resilient computation compiler of Goldwasser and Rothblum [23].
- *Leakage-resilient threshold cryptosystems.* In a threshold cryptosystem [13], parties hold shares of a single secret key, and only a quorum of parties can jointly execute the corresponding secret functionality (e.g., decryption). Our MPC protocol, when combined with an underlying leakage-resilient cryptographic primitive (e.g., [1] for public-key encryption), yields a corresponding leakage-resilient threshold cryptosystem. The security guarantee of our protocol implies that any information learned by an adversary who controls any strict subset of a quorum, and obtains leakage on the joint secret states of all honest parties during the collective decryption protocol, reduces to simply the output value and corresponding leakage on the underlying secret key.

1.2 Technical Overview

Our starting point is the GMW paradigm for building MPC protocols [21].

The First Approach. Recall the GMW paradigm begins by designing an MPC protocol secure against semi-honest (i.e., passive) adversaries, and then compiles the protocol into one secure against malicious adversaries by "enforcing" semi-honest behavior via use of zero-knowledge proofs, a commitment scheme, and a coin-tossing protocol.

We begin by mirroring this approach in the setting of adaptive auxiliary input. We directly achieve MPC secure against adaptive auxiliary information in the semi-honest setting by instantiating the basic GMW protocol with the oblivious transfer protocol of [3] (that has analogous semi-honest security properties). Further, continuing onto the GMW compiler, we see that zero-knowledge proofs secure against adaptive auxiliary information were already constructed in [20, 3], and *equivocal* commitment schemes [18] were also shown to have the required security properties [20, 3]. We are thus almost within reach of our final goal.

Unfortunately, in the remaining step, one runs into serious problems. Note that in order to reduce the malicious security of the compiled protocol to the semi-honest security of the original protocol, the coin-tossing protocol to be used in the compiler must be *fully simulatable*, in that the simulator must be able to choose the output of the coin toss, and simulate the protocol to force this output, for both honest and corrupted parties. It is not clear how to construct

a *coin-tossing* protocol secure against adaptive auxiliary information.[7] Indeed, as shown recently by Chung et al. [11], constructing such a protocol in the two-party setting is in fact *impossible*. We refer the reader to the full version for further discussion on this issue.

A New Stepping Stone: "Semi-Malicious" Adversaries. We thus abandon the approach of mimicking the GMW paradigm "out of the box." We instead consider a new intermediate step, lying closer to security against malicious adversaries, with the goal of eliminating the necessity for fully simulatable coin-tossing in the final compiler. This amounts to constructing protocols that remain secure even if an adversary potentially uses "bad" randomness in the protocol execution. To formalize this requirement, we consider the notion of a *semi-malicious* adversary that follows the protocol execution (similar to a semi-honest adversary), but can choose its random coins (and inputs) in any arbitrary manner.[8]

Once we construct a protocol for semi-malicious adversaries (that can learn arbitrary auxiliary information), we can easily compile it into a secure protocol for *malicious* adversaries by standard techniques. We do so using a modified version of the GMW compiler adapted to our setting, implemented with equivocal commitments [18, 9] and the UC-NIZKs of [24] that were shown to be secure against adaptive auxiliary information by Garg et al. [20]. (We refer the reader to the technical sections for more details.) The task then remains to construct an MPC protocol that is secure against adaptive auxiliary information in the presence of semi-malicious adversaries.

A close look at the basic GMW construction reveals that constructing semi-malicious MPC reduces to constructing a semi-malicious oblivious transfer (OT) protocol. (We note that this observation is also implicit in [28].) Since our goal is to protect against adversaries who may learn adaptive auxiliary information, we aim to construct OT protocols with similar security guarantees against semi-malicious adversaries. We discuss this next.

Semi-Malicious OT. Our starting point is the adaptively secure semi-honest OT protocol of Canetti et al. [9]. The [9] construction follows the "standard template" of [17] for semi-honest OT, but replaces the underlying encryption scheme with a non-committing encryption (NCE) scheme [8]. Namely, (1) The receiver R generates and sends two public keys pk_0, pk_1 for the (non-committing) encryption scheme—one for which he knows the secret key, and one "obliviously" sampled; and (2) the sender S sends an encryption of each of his messages m_i, under the corresponding public key pk_i. The [9] scheme was shown to be secure against adaptive auxiliary information in the semi-honest model by [3]. However, the protocol fails in the *semi-malicious* model. Indeed, a semi-malicious receiver can simply choose bad randomness to "obliviously" sample public keys for which

[7] The leakage-resilient coin tossing result of [5] is not relevant to this setting. Their construction requires an honest majority of parties (to attain information theoretic guarantees), whereas our model allows an arbitrary number of corruptions.

[8] The notion of semi-malicious adversaries is somewhat similar in spirit to the notion of defensible adversaries considered by [25]. We refer the reader to Section 3 for a comparison of the two notions.

he can decrypt (and thus learn both messages of the sender). Further, a semi-malicious sender may be able to create "malformed" ciphertexts that cause the honest receiver to abort depending on his secret input. Circumventing these fatal roadblocks demands a new set of techniques. We solve these problems as follows:

- First, we construct an underlying NCE scheme with strong security properties, which will guarantee security in the OT protocol *as long as the adversary's randomness does not fall within a very small "bad" set.* We achieve this by building an NCE scheme where the public keys generated via the oblivious key generation algorithm are almost always *lossy* (except if they belong to some exponentially small set, such as the set of DDH tuples). Now, unless the adversary's randomness falls within this very small set, the encryption of non-requested messages under his obliviously sampled public keys will *information theoretically* hide the messages.
- Second, we develop a new methodology for generating private randomness that *prevents* a malicious party from choosing randomness within this small bad set. The challenge is doing so *in the presence of adaptive auxiliary information*, and while *simultaneously* providing the simulator the necessary flexibility to "force" any randomness of his choice for honest parties.

  A potential idea is to design a modified coin tossing protocol to ensure a malicious party's output randomness still maintains sufficient entropy in the auxiliary information setting. However, approaches of this kind seem to inherently necessitate an *a priori bound* on how much auxiliary information can be handled: if honest parties cannot hold onto *any* secret entropy during the protocol, this path appears hopeless.

  We provide a different approach. We construct a *non-interactive* randomness generation procedure that achieves the desired properties by use of *lossy trapdoor functions* (LTDF), together with a CRS. Namely, each party P_i is assigned an LTDF seed σ_i in the CRS; each time the party must sample randomness in the protocol, he first chooses a random value r in the LTDF domain, and then uses the LTDF evaluation $F(\sigma_i, r)$ as his protocol randomness. Loosely speaking, in the simulation, honest parties will be assigned seeds for *injective* functions, whereas corrupted parties will be assigned (computationally indistinguishable) seeds for *lossy* functions. This allows the simulator to efficiently "explain" any possible output for honest parties, while simultaneously restricting malicious parties to a small set of attainable output values that does not "hit" the small set of *bad* values. We refer the reader to Section 4.3 for more details.

Final Touches. While the above ideas essentially handle the issue of "bad" randomness, we still need to find a way to answer the auxiliary information queries of the adversary correctly. Our starting point for this is the observation of [20, 3] that adaptive security (without erasures) provides simulators that can simulate random tapes for honest parties, which can be used to answer auxiliary information queries. However, this is possible if the simulator is able to decide its random tape *after* viewing the random tape of the adversary. Unfortunately, depending upon the "structure" of the protocol, this may not always be possible.

To address this problem, we somewhat "soften" this asymmetry: instead of generating the entire random tapes of each party a priori, we generate them in an "online" fashion. In part because the GMW protocol provides perfect security in the OT-hybrid model, it turns out that this essentially suffices for simulation.

1.3 Related Work and Organization

The notion of security considered in this paper is somewhat analogous to that considered by Bitansky et al. [3] and Garg et al. [20], in the context of leakage-resilience. Garg *et. al.* [20] consider the case of zero-knowledge proof systems where a malicious verifier can adaptively leak arbitrary information on the state of the honest prover during the execution of the protocol. Bitansky *et. al.* [3] put forth a general definition of leakage tolerance in the setting of two-party interactive protocols, extending the notion of universally composable security. They show how to realize specific tasks (such as oblivious transfer) in such a scenario against semi-honest adversaries as well as zero knowledge proofs in the CRS model. In addition, they prove a composition theorem for leakage-tolerant protocols that we use in our work. Indeed, our work is greatly inspired by theirs.

A concurrent and independent work of Damgård et al. [12] considers the problem of two-party computation protocol in the leakage setting. They formalize a security definition along the lines of entropic leakage as in [32, 27] and show how to realize it for NC^1 functionalities against *semi-honest* adversaries. (In contrast, we consider malicious adversaries, as well as the multiparty setting.) We note that our results, cast in the leakage context, also satisfy their definition (for the case of 2-party protocols).

Guide to the Paper. Section 2 contains partial preliminaries. In Section 3, we present our model and security definition. Section 4 contains the technical core of our work: an oblivious transfer protocol secure against adaptive auxiliary information in the semi-malicious model. Due to space limitations, we defer remaining preliminaries, proofs of security, and formal analysis to the full version.

2 Preliminaries

Non-committing Encryption. Informally, a non-committing (bit) encryption scheme [8] is a semantically secure, possibly interactive encryption scheme, with the additional property that a simulator can generate special ciphertexts that can be "opened" to (i.e. demonstrated to be the encryption of) both 0 and 1.

Definition 1. *[8, 10] A non-committing (bit) encryption scheme consists of a tuple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{NCSim})$*, where* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a semantically secure encryption scheme, and* NCSim *is a* PPT *simulation algorithm that on input* 1^k *outputs a tuple* $(e, c, r_G^0, r_E^0, r_G^1, r_E^1)$ *such that for every* $b \in \{0,1\}$ *the following distributions are computationally indistinguishable:*

1. *The joint view of an honest sender and an honest receiver in a normal encryption of* b*:* $\{(e, c, r_G, r_E) : (e, d) \leftarrow \mathsf{Gen}(1^k; r_G), c \leftarrow \mathsf{Enc}_e(b; r_E)\}$.

2. *A simulated view of an encryption of b:* $\{(e, c, r_G^b, r_E^b) : (e, c, r_G^0, r_E^0, r_G^1, r_E^1) \leftarrow \mathsf{NCSim}(1^k)\}$.

Augmented NCE. In our MPC protocol, we will use an "augmented" NCE scheme, satisfying three additional properties.

Oblivious key generation: It should be possible to sample an encryption key without "knowing" a corresponding secret key via a procedure OGen.

Invertible samplability: The key generation and the oblivious key generation algorithms Gen and OGen should be "invertible." That is, given an output that lies in the range of Gen (resp., OGen) that was potentially generated via a *different* algorithm (e.g., NCSim), we can efficiently generate randomness that "explains" the output as being generated via Gen (resp., OGen).

Alternative simulation: In the standard NCE definition, NCSim generates a simulated ciphertext (and randomness values) *together* with an encryption key e. For our purposes, we require a stronger simulation property, where we can generate a simulated ciphertext for a *fixed* encryption key – namely, one that is obliviously sampled by another party.

In this work, we build upon the NCE construction of Choi *et. al* [10] to construct an *augmented* NCE scheme with additional desired properties. See Section 4 for more details.

Lossy Trapdoor Functions (LTDF). A lossy trapdoor function (LTDF) family [33] consists of two computationally indistinguishable families of functions. Functions in one family are injective and can be efficiently inverted using a trapdoor; functions in the other family are "lossy," in that the size of their image is significantly smaller than the size of their domain. We refer the reader to [33, 19] for a complete definition of an (m, ℓ)-LTDF family[9] $(G_{\mathsf{Loss}}, G_{\mathsf{Inj}}, S, F, F^{-1})$, with function sampling algorithms $G_{\mathsf{Loss}}, G_{\mathsf{Inj}}$, domain sampling algorithm S, evaluation algorithm F, and inversion algorithm F^{-1}.

For our purposes, we require a strengthened LTDF family, in which the injective functions are *bijections* (i.e., surjective onto their target space), the lossy branches are sufficiently lossy, and the size parameters satisfy certain relations.

Definition 2. *A collection of (m, ℓ)-lossy trapdoor functions $(G_{\mathsf{Loss}}, G_{\mathsf{Inj}}, S, F, F^{-1})$ is* bijective *and* (D, α)-admissible *if it satisfies the following properties:*

- **Bijective:** *For every seed σ produced by G_{Inj}, the algorithm $F(\sigma, \cdot)$ computes a* bijective *function $f_\sigma : D_\sigma \to R_\sigma$ (where R_σ is the output space of f_σ).*
- (D, α)-**Admissible:** *The following hold, corresponding to target output size D, where α fraction of the D values are "bad":*
  1. *Efficiently invertible domain sampling: There exists a PPT algorithm S^{-1} such that for each $x \in D_\sigma$, $\{S^{-1}(x)\} \overset{s}{\approx} \{r : S(r) = x\}$.*
  2. *Sufficiently large output space: For each seed σ produced by either $G_{\mathsf{Loss}}(1^k)$ or $G_{\mathsf{Inj}}(1^k)$, the size of the output space R_σ satisfies $|R_\sigma| \geq D$.*

[9] The parameters (m, ℓ): Each function f_σ in the family has domain size $|D_\sigma| \geq 2^{m-1}$, and each lossy function f_σ has image size at most $|D_\sigma| \cdot 2^{-\ell}$.

3. *Sufficiently small image of lossy functions: There exists a negligible function $\mu(k)$ such that for every seed σ produced by G_{Loss}, $\alpha \cdot \left(|D_\sigma| \cdot 2^{-\ell(k)}\right) < \mu(k)$. Recall that $|D_\sigma| \cdot 2^{-\ell(k)}$ is an upper bound for the image size of a lossy function with seed σ.*

In the full version, we show that the composite residuosity-based LTDF family construction (with seed-dependent domains) of Freeman et. al. [19] satisfies our required properties (with parameters tailored to our NCE construction):

Theorem 1. *[19] Under the decisional composite residuosity assumption, there exists a* bijective, (D, α)-admissible *collection of LTDFs* $(G_{\mathsf{Loss}}, G_{\mathsf{Inj}}, S, F, F^{-1})$ *with seed-dependent domains, for the following two choices of* (D, α)*:*
(1) $D = \binom{4k}{k}$. $\alpha = \binom{3k}{k} / \binom{4k}{k}$. *(2)* $D = 2^k$. $\alpha = 2^{-k/3}$.

3 Our Model

To define security against adaptive auxiliary information, we turn to the real/ideal paradigm. We consider a real-world execution where an adversary, in addition to corrupting a number of parties, can adaptively learn arbitrary auxiliary information on the *joint* secret states of the honest parties, throughout the protocol execution. Following the works on leakage-resilient cryptography, we model this by allowing the adversary to make auxiliary information queries of the form L, where L is the circuit representation of an efficiently computable function. On making such a query, the adversary learns $L(\mathsf{state})$, where state represents the joint secret state of the honest parties. In the ideal world experiment, the ideal world adversary, i.e., the simulator is allowed to request auxiliary information on the inputs of all the parties from the trusted party. Note that this is similar to those considered in [20, 3], although these works focused only on the two-party case, whereas we deal with both two-party case and multi-party case. Below, we describe a standalone security definition. (See full version for the UC setting.)

Ideal World. We first define the ideal world experiment, where n parties $P_1, \ldots, P_n$ interact with a trusted party for computing a function f.

- **Inputs:** Each party P_i obtains an input x_i. The adversary is given (initial) auxiliary input z, selects a subset of parties $M \subset \mathcal{P}$ to corrupt, and receives x_i for every $P_i \in M$.
- **Sending inputs to trusted party:** Each honest party P_i sends its input x_i to the trusted party. For each corrupted party $P_i \in M$, the adversary may select any value x'_i and send it to the trusted party.
- **Trusted party computes output:** Let $x'_1, \ldots, x'_n$ be the inputs that were sent to the trusted party. The trusted party computes $f(x'_1, \ldots, x'_n)$.
- **Adversary learns output:** The trusted party first sends $f(x'_1, ..., x'_n)$ to the adversary. The adversary replies with either $\mathsf{continue}$ or abort.

- **Honest parties learn output:** If the message is abort, the trusted party sends $\perp$ to all honest parties. If the adversary's message was continue, then the trusted party sends the evaluation $f(x'_1, \ldots, x'_n)$ to all honest parties.[10]
- **Auxiliary information queries on inputs:** The adversary may send (adaptively chosen) auxiliary information queries in the form of efficiently computable functions L_j (described as a circuit). On receiving such a query, the trusted party computes $L_j(x'_1, \ldots, x'_n)$ and returns the output to the adversary. (We in fact, place further restriction on the communication between the adversary and the trusted party w.r.t. the auxiliary information queries; we discuss this in more detail below.)
- **Outputs:** Honest parties each output the message they obtained from the trusted party. Malicious parties may output an arbitrary PPT function of their initial inputs, auxiliary input, and the interaction with trusted party.

Real World. The real-world execution begins by an adversary $\mathcal{A}$ selecting any arbitrary subset $M \subset \mathcal{P}$ of the parties to corrupt. On being corrupted, each party $P_i \in M$ hands over its input to $\mathcal{A}$. The parties $P_1, \ldots, P_n$ now engage in an execution of a real n-party protocol Π (without any trusted third party). The adversary $\mathcal{A}$ sends all messages on behalf of the corrupted parties, and may follow an arbitrary PPT strategy. In contrast, the honest parties follow the instructions of Π. Furthermore, at any point during the protocol execution, the adversary may make auxiliary information queries of the form L and learn $L(\mathsf{state}_{\mathcal{P}\setminus M})$, where $\mathsf{state}_{\mathcal{P}\setminus M}$ denotes the concatenation of the protocol states state_i of each honest party P_i. We allow the adversary to choose the auxiliary information queries adaptively based on all the information that $\mathcal{A}$ received up to that point (including responses to previous such queries). Honest parties have the ability to toss fresh coins at any point in the protocol; these coins are added to the state of that party at the time they are generated. At the conclusion of the protocol execution, each honest party P_i generates its output according to Π. Malicious parties may output an arbitrary PPT function of the view of $\mathcal{A}$.

Security Definition. We now give our formal definition of MPC secure against adaptive auxiliary information. Our definition crucially relies on the notion of *leakage-oblivious simulation* as defined in [20, 3]. We recall it below.

Leakage-Oblivious Simulation. Loosely speaking, an ideal world adversary, i.e., a simulator $\mathcal{S}$, is said to be leakage-oblivious if the auxiliary information obtained by the simulator is used *only* for the purposes of simulating answers to the auxiliary information queries of the real adversary. More formally, we require that the simulator $\mathcal{S}$ has a special subroutine $\widetilde{\mathcal{S}}$ for handling auxiliary information queries. Whenever $\mathcal{S}$ receives an auxiliary information query L from the real

[10] We can also define a more general case, where f may output a *different* value $f_i(x'_1, ..., x'_n)$ to each party P_i. In this setting, the adversary first learns the set of outputs $\{f_i(x'_1, ..., x'_n)\}_{i \in M}$ corresponding to corrupted parties, and then decides whether to abort or to allow the honest parties to receive their respective outputs. We do not dwell on this detail for simplicity of exposition; however, our construction also handles this more general case.

world adversary, $\widetilde{\mathcal{S}}$ is invoked to produce a "state translation circuit" T that takes as input the inputs of the honest parties and produces their joint states. Once T is produced, the ideal functionality is queried on the composed circuit $L \circ T$. When the auxiliary information is returned, it is forwarded directly to the real adversary and $\mathcal{S}$ returns to its state prior to the event. Such a simulator is referred to as a leakage-oblivious simulator.

We now define security w.r.t. the real and ideal world experiments as discussed above, except that we consider leakage-oblivious simulators in the ideal world experiment. The output of the ideal-world experiment consists of the inputs and outputs of all parties, and the answers of all the auxiliary information queries. It is denoted by $\mathsf{IDEAL}^{f}_{\mathcal{S},M}(1^k, \boldsymbol{x}, z)$. The overall output of the real-world experiment consists of the inputs and outputs of all parties at the conclusion of the protocol, and all the auxiliary information learnt by the adversary (including the protocol transcript). It is denoted by $\mathsf{REAL}^{\Pi}_{\mathcal{A},M}(1^k, \boldsymbol{x}, z)$.

Definition 3 (MPC Secure Against Adaptive Auxiliary Information). *A protocol Π evaluating a functionality f is said to be* secure against adaptive auxiliary information *if for every* PPT *real adversary $\mathcal{A}$, there exists a* PPT *leakage-oblivious simulator $\mathcal{S}$ such that for every input vector $\boldsymbol{x}$, $z \in \{0,1\}^*$, and $M \subset \mathcal{P}$, it holds that,*

$$\left\{\mathsf{IDEAL}^{f}_{\mathcal{S},M}(1^k, \boldsymbol{x}, z)\right\}_{k\in N} \approx_c \left\{\mathsf{REAL}^{\Pi}_{\mathcal{A},M}(1^k, \boldsymbol{x}, z)\right\}_{k\in N}.$$

3.1 Security Against Semi-Malicious Adversaries

As a stepping stone toward realizing our definition of MPC secure against adaptive auxiliary information in the presence of malicious adversaries, we define the notion of a *semi-malicious* adversary. Intuitively, a semi-malicious adversary is similar to a "standard" (real-world) semi-honest adversary, in that it follows the protocol specification. However, it differs from semi-honest adversaries in that it may choose its input and its "random" coins for any protocol step in an online fashion, adaptively, following any arbitrary PPT strategy. Once it has chosen these values, however, it must follow the protocol as specified, given the chosen input, and using the chosen coins in place of the random coins.[11] Furthermore, in our setting, a semi-malicious adversary is allowed to learn arbitrary auxiliary information on the (joint) secret states of the honest parties.

More formally, a *semi-malicious adversary* $\mathcal{A}$ is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special auxiliary tape. At the start of the protocol, $\mathcal{A}$ selects for each corrupted party P_k an input x_k (which may depend on the original inputs of corrupted parties), and writes x_k to its special input auxiliary tape. Then in each round of the protocol, whenever $\mathcal{A}$ produces a new protocol message m on behalf of some party P_k, it must also append to its special auxiliary tape *some* randomness

[11] This is reminiscent of the notion of defensible adversaries, introduced by Haitner *et. al.* [25]. We refer the reader to the full version for a detailed comparison.

that *explains its behavior*. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x_k and randomness r_k written in the special auxiliary tape. We allow $\mathcal{A}$ to make auxiliary information queries on the joint states of the honest parties in the same manner as discussed earlier. We further assume that the adversary is rushing and hence may choose the randomness r in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds), as well as all the auxiliary information that it may have obtained so far. Lastly, the adversary may choose to abort the execution on behalf of P_k in any step of the interaction.

Definition 4 (MPC Secure Against Adaptive Auxiliary Information in the Semi-Malicious Model). *We say that protocol Π evaluating a function f is* secure against adaptive auxiliary information in the semi-malicious model *if it satisfies Def. 3 when we only quantify over all semi-malicious adversaries $\mathcal{A}$.*

4 Semi-Malicious OT

In this section, we construct an oblivious transfer (OT) protocol that is secure against adaptive auxiliary information in the presence of semi-malicious adversaries. This protocol is the technical heart of our MPC protocol.

Recall that our OT protocol construction builds on the adaptively secure protocol of [9] (which is resilient to adaptive auxiliary information in the semi-*honest* model [3]), with two primary new components: (1) A new non-committing encryption scheme with strong security properties (guaranteeing security in the OT of [9] as long as the adversary's randomness does not fall in a very small "bad" set), and (2) A new randomness generation procedure that prevents the adversary from selecting randomness within this small "bad" set, in light of adaptive auxiliary information. We construct these new tools in Section 4.1 and 4.2, and present our OT construction itself in Section 4.3.

4.1 Non-Committing Encryption with Lossy Keys

We construct an augmented NCE scheme (see Section 2) with the property that public keys generated using the oblivious key generation algorithm are almost always *lossy*. The specific NCE we use is a slight variant of the one due to Choi *et. al.* [10], which makes use of an underlying encryption scheme with oblivious sampling and inverting algorithms (both for generating keys and ciphertexts). For our construction, we use an underlying encryption scheme that is also *lossy*.

Our Underlying Lossy Encryption Scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{OGen}, \mathsf{OEnc}, \mathsf{IGen}, \mathsf{IEnc})$. Let $\mathbb{G}$ be a group of prime order p. The algorithm Gen samples $g_1, g_2 \leftarrow \mathbb{G}$, $u \leftarrow \mathbb{Z}_p$, and outputs $\mathsf{pk} = (g_1, g_2, g_1^u, g_2^u)$ and $\mathsf{sk} = u$. To encrypt a message $m \in \mathbb{G}$ under $\mathsf{pk} = (g_1, g_2, g_3, g_4)$, Enc samples $\beta_1, \beta_2 \leftarrow \mathbb{Z}_p$ and outputs $(g_1^{\beta_1}, g_2^{\beta_2}, g_3^{\beta_1}, g_4^{\beta_2} \cdot m)$. To decrypt a ciphertext (c_1, c_2) with $\mathsf{sk} = u$, Dec outputs

$m = \frac{c_1}{c_2^x}$. The oblivious sampling algorithms $\mathsf{OGen}, OEnc$ simply output random values in the appropriate spaces, and the inversion algorithms $\mathsf{IGen}, \mathsf{IEnc}$ to "explain" a given key pair / ciphertext as being obliviously sampled are trivial.

The Augmented NCE scheme $\mathcal{E}_{\mathsf{NCE}}$.

- $\mathsf{NCGen}(1^k)$**:** Generate $4k$ public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_{4k}$ for the underlying scheme, sampling k normally and $3k$ obliviously. Explicitly, choose a random subset $I \subset [4k]$ of size k. For every $i \in I$, sample $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^k)$, and for every $i \in [4k] \setminus I$, sample $\mathsf{pk}_i \leftarrow \mathsf{OGen}(1^k)$. Also choose two random messages $M_0, M_1 \leftarrow \mathbb{G}$ from the message space of the underlying scheme. Output $\mathsf{pk}_{\mathsf{NCE}} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{4k}, M_0, M_1)$, and $\mathsf{sk}_{\mathsf{NCE}} = (I, \{\mathsf{sk}_i\}_{i \in I})$.
- $\mathsf{NCEnc}_{\mathsf{pk}_{\mathsf{NCE}}}(b)$**:** Generate $4k$ ciphertexts $c_1, \ldots, c_{4k}$, where k are encryptions of M_b and $3k$ are obliviously sampled. Explicitly, choose a random subset $J \subset [4k]$ of size k. For every $j \in [J]$ compute $c_j \leftarrow \mathsf{Enc}_{\mathsf{pk}_j}(M_b)$, and for every $j \in [4k] \setminus [J]$ compute $c_j \leftarrow \mathsf{OEnc}(1^k)$. Output $c = (c_1, \ldots, c_{4k})$.
- $\mathsf{NCDec}_{\mathsf{sk}_{\mathsf{NCE}}}(c_1, \ldots, c_{4k})$**:** Decrypt the k ciphertexts for which it knows the secret key. Namely, decrypt $\{c_i\}_{i \in I}$. If $\exists c_i$ decrypting to M_b, and no another c_j decrypts to M_{1-b} then output M_b. Otherwise, output $\perp$.
- We refer the reader to [10] for the simulator algorithm NCSim.
- Augmented NCE algorithms: A straightforward modification to NCSim yields the required alternative simulator algorithm NCSim'. Oblivious key generation ONCGen is achieved by running the oblivious key sampler OGen for the underlying scheme $4k$ times and sampling two random messages M_0, M_1. The corresponding inversion algorithm is immediate.

We remark that for our OT application, the choice of M_0, M_1 will be made once overall for all encryptions, and will be contained in the CRS. In the full version, we prove that $\mathcal{E}_{\mathsf{NCE}}$ is an augmented NCE scheme with two additional properties: first, correctness of decryption holds for all but a tiny fraction of encryption randomness; and second, the NCE scheme inherits certain lossy properties of the underlying encryption scheme. These properties are used in the security proof of our MPC protocol.

4.2 Randomness Generation Procedure

In this section, we present a non-interactive procedure for generating private randomness in the CRS model. Intuitively, we need the following two properties:

Semi-malicious P_j cannot force "bad" output: If party P_j is semi-malicious, then he cannot force the output randomness to lie inside a "special" exponentially small subset of the total space $\{0,1\}^t$.
Simulator can retroactively force any output for honest P_j: Given a trapdoor to the CRS, the simulator can retroactively "explain" any desired outcome within the "special" subset of $\{0,1\}^t$ on behalf of an honest P_j.

The "bad" sets arise as follows. To guarantee security of the OT, we must ensure: (1) a corrupted receiver must sample *lossy* public keys in the oblivious

sampling procedure, to ensure an honest sender's second message is information-theoretically hidden; (2) a corrupted sender cannot generate malformed ciphertexts, which could cause an honest receiver to abort depending on his secret input bit; and (3) a corrupted sender-receiver pair cannot jointly select key generation and encryption randomness yielding an incorrect message decryption, which would compromise the correctness of the OT output.

We now incorporate the specifics of our NCE construction (from the previous section), and describe the corresponding randomness generation procedures.

Forcing Obliviously Sampled Public Keys to be Lossy. A corrupted receiver must be restricted to generate only public keys $(g_1, g_2, g_3, g_4) \in \mathbb{G}^4$ in the underlying encryption scheme that are *non-DDH* tuples (yielding lossy keys), whereas the simulator should be able to choose DDH tuples on behalf of honest parties (allowing him to "obliviously sample" keys for which he can decrypt)

To achieve this, we append an additional 4-tuple of random elements $\boldsymbol{g} = (g_1, g_2, g_3, g_4) \in \mathbb{G}^4$ to the CRS of each party. Each time a party P_j must obliviously sample public keys, he does so by rerandomizing his tuple $\boldsymbol{g}_j$: namely, he samples random exponents $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_p^*$ and outputs the tuple $(g_1^\alpha, g_2^\beta, g_3^{\alpha\cdot\gamma}, g_4^{\beta\cdot\gamma})$ as the desired randomness. Denote this procedure by $\mathsf{DDH\text{-}Rerand}(g_1, g_2, g_3, g_4)$.

Note that if the original tuple (g_1, g_2, g_3, g_4) is a DDH tuple (as will be the case for honest parties in the simulation), then the output of this procedure will be a random DDH tuple; however, if the original tuple is *not* a DDH tuple (as will be the case for corrupted parties), then for *no value* of $\alpha, \beta, \gamma \neq 0$ will the resulting tuple become DDH.

Preventing Malformed Ciphertexts. When encrypting a bit m under the NCE scheme, the sender must select a random subset of positions $J \subset [4k]$ of size k in which to embed encryptions of the appropriate M_m in the underlying scheme. If J is disjoint from an honest receiver's set $I \subset [4k]$ of positions for which he knows the secret keys, then the resulting ciphertext will fail to decrypt. A corrupted sender must then be restricted so that with overwhelming probability over (honestly) chosen random $I \subset [4k]$ with $|I| = k$, it holds that $I \cap J \neq \emptyset$, *even* if he knows I completely. However, on behalf of an honest sender, the simulator should be able to retroactively "explain" arbitrary J values.

To achieve this, our second procedure makes use of a *bijective*, (D, α)-*admissible* lossy trapdoor function (LTDF) family $(G_{\mathsf{Loss}}, G_{\mathsf{Inj}}, S, F, F^{-1})$,[12] for $D = \binom{4k}{k}$ and $\alpha = \binom{3k}{k}/\binom{4k}{k}$, as in Theorem 1(1). Suppose within the CRS each party P_j is associated with a LTDF seed σ_j. At each point when P_j must generate randomness for sampling the set $J \subset [4k]$, he samples a random value $r \leftarrow S(\sigma_j)$ from the domain D_{σ_j} of the LTDF function corresponding to seed σ_j and outputs the evaluation $F(\sigma_j, r)$ as the desired randomness. Denote this procedure by $\mathsf{LTDF\text{-}Sample\text{-}}J(\sigma_j)$.

[12] Such an LTDF family has the additional properties that injective branches are each *bijective*, the output space has size approximately D, and the lossy branches are *very lossy* (to avoid a "bad" set of fractional size α). See Definition 2.

Loosely speaking, the simulation works as follows. For each honest party, the simulator will sample an *injective* seed σ_j together with an inversion trapdoor, allowing him to "explain" any desired randomness output value. In contrast, each corrupted party will be given a *lossy* seed σ_j, whose corresponding function image will be sufficiently small that with high probability it will not hit the small "special" subset of J values which do not intersect the (random) set I.

Preventing Ciphertexts that Decrypt to *Wrong* Messages. When both sender and receiver are corrupted, we must ensure they cannot collaboratively sample a secret key $u \in \mathbb{Z}_p$ and "obliviously" sampled ciphertext $(x, y) \in \mathbb{G}^2$ for the underlying scheme for which $\frac{y}{x^u} = M_{1-m}$. Such a pair may allow an honest encryption of message bit m to (honestly) decrypt to the *wrong* message $(1-m)$.

To prevent this event, the secret key $u \in \mathbb{Z}_p$ and each element of the obliviously generated ciphertext $(x, y) \in \mathbb{G}^2$ will be sampled using the same LTDF-based procedure as above (but with a different choice of parameters (D, α)). Here, the target output space size D will be set to $p = |\mathbb{G}| = 2^k$, and α will be selected so that the *total* collection of possible values of $\frac{y}{x^u}$, over all possible combinations of u, x, y chosen from the range of the the sender and receiver's (lossy) LTDF functions, form a negligible fraction of the entire space of possible values (i.e., the full message space of the underlying encryption scheme). Thus, when the "special" messages M_0, M_1 are chosen randomly as part of the initial CRS, the probability that they will fall into this small set of "bad" attainable messages is negligible. This holds when α is set to $2^{-k/3}$.

More formally, let $(G_{\mathsf{Loss}}, G_{\mathsf{Inj}}, S, F, F^{-1})$ be a bijective, (D, α)-admissible LTDF family for $D = 2^k$ and $\alpha = 2^{-k/3}$, as given in Theorem 1(2). Suppose each party P_j is associated with a LTDF seed σ_j contained in the CRS. Each time party P_j must generate randomness for (1) sampling a secret key u (when acting as receiver in the OT), or (2) selecting each of the two components in an obliviously sampled ciphertext (x, y), he executes the following procedure $\mathsf{LTDF\text{-}Sample\text{-}}\mathbb{G}(\sigma_j)$: first, sample a random value $r \leftarrow S(\sigma_j)$ from the domain D_{σ_j} of the LTDF function corresponding to seed σ_j, output the evaluation $F(\sigma_j, r)$ as the desired randomness.

Combining the Pieces. In Figure 1 we explicitly define the procedures for generating randomness for each relevant application in the OT protocol. The CRS contains a value of $\sigma^{(1)}, \sigma^{(2)}$, and (g_1, g_2, g_3, g_4) for each party, corresponding to LTDF seeds and group elements as described above. Denote by $\mathbf{A}_k$ the set of all subsets of $[4k]$ of size k. We refer the reader to the full version for a formal treatment and analysis of these procedures.

4.3 Our OT Protocol

We now use the augmented NCE scheme from Section 4.1 and the randomness generation procedures from Section 4.2 to construct a 1-out-of-4 OT protocol secure against adaptive auxiliary information in the semi-malicious setting.

$\mathsf{Rand\text{-}KeyGen}(\sigma^{(2)}, (g_1, g_2, g_3, g_4))$**:** Sample $I \leftarrow \mathbf{A}_k$. For each $i \in I$, sample randomness to be used for Gen, by executing $u_i \leftarrow \mathsf{LTDF\text{-}Sample\text{-}}\mathbb{G}(\sigma^{(2)})$. For each $j \in [4k] \setminus I$, sample randomness to be used for *obliviously* sampling a public key, by executing $r_j \leftarrow \mathsf{DDH\text{-}Rerand}(g_1, g_2, g_3, g_4)$. Output $(I, \{u_i\}_{i\in I}, \{r_j\}_{j\in[4k]\setminus I})$.

$\mathsf{Rand\text{-}OblivKeyGen}(g_1, g_2, g_3, g_4)$**:** For each $i \in [4k]$, sample randomness to be used for obliviously sampling a public key, by executing $r_i \leftarrow \mathsf{DDH\text{-}Rerand}(g_1, g_2, g_3, g_4)$. Output $\{r_i\}_{i\in[4k]}$.

$\mathsf{Rand\text{-}Enc}(\sigma^{(1)}, \sigma^{(2)})$**:** Select $J \in \mathbf{A}_k$, by executing $J \leftarrow \mathsf{LTDF\text{-}Sample\text{-}}J(\sigma^{(1)})$. For each $j \in J$, sample encryption randomness $\mathsf{rand}_j \leftarrow \{0,1\}^*$. For each $i \notin J$, sample randomness for obliviously sampling a ciphertext, by running two executions $x_i, y_i \leftarrow \mathsf{LTDF\text{-}Sample\text{-}}\mathbb{G}(\sigma^{(2)})$. Output $(J, \{\mathsf{rand}_j\}_{j\in J}, \{(x_i, y_i)\}_{i\in[4k]\setminus J})$.

Fig. 1: Randomness generation procedures for the semi-malicious OT protocol.

Our 1-out-of-4 OT protocol for semi-malicious parties.
CRS: Uniformly random message $M_0, M_1 \leftarrow \mathbb{G}$. For each party P_i: (injective) LTDF seeds $\sigma_i^{(1)}, \sigma_i^{(2)}$, and (non-DDH) tuple $\boldsymbol{g}^i = (g_1^i, g_2^i, g_3^i, g_4^i)$.

1. The receiver P_i, on input $b \in [4]$, does the following:
   1. For $b \in [4]$, sample a standard NCE key pair (e_b, d_b) via $\mathsf{Rand\text{-}KeyGen}(\sigma_i^{(2)}, \boldsymbol{g}^i)$. For each $b' \in [4] \setminus \{b\}$, obliviously sample an NCE public key $e_{b'}$ via the algorithm $\mathsf{Rand\text{-}OblivKeyGen}(\boldsymbol{g}^i)$.
   2. The receiver sends $(e_1, \ldots, e_4)$ to the sender.
3. The sender P_j, on input $m_1, \ldots, m_4$, where each $m_i \in \mathbb{G}$, and upon receiving the message $(e_1, \ldots, e_4)$ from the receiver, does the following:
   1. For each $b' \in [4]$, encrypt the message $m_{b'}$ under public key $e_{b'}$. This is done by executing the encryption algorithm NCEnc with message $m_{b'}$, key $e_{b'}$, and randomness $r_{b'} \leftarrow \mathsf{Rand\text{-}Enc}(\sigma_j^{(1)}, \sigma_j^{(2)})$.
   2. Send $(c_1, \ldots, c_4)$.
3. The receiver decrypts c_b using the secret key d_b.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC. pp. 474–495 (2009)
2. Anderson, R., Kuhn, M.: Tamper resistance: a cautionary note. In: WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce. pp. 1–11 (1996)
3. Bitansky, N., Canetti, R., Halevi, S.: Leakage tolerant interactive protocols. In: TCC (2012)
4. Boyle, E., Goldwasser, S., Jain, A., Kalai, Y.: Multiparty computation secure against continual memory leakage. In: STOC (2012)
5. Boyle, E., Goldwasser, S., Kalai, Y.: Leakage-resilient coin tossing. In: DISC (2011)
6. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS. pp. 501–510 (2010)
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2005)

8. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC. pp. 639–648 (1996)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC. pp. 494–503 (2002)
10. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: ASIACRYPT (2009)
11. Chung, K.M., Lin, H., Liu, F.H., Pass, R., Zhou, H.S.: Physically-aware composability. Manuscript (2013)
12. Damgard, I., Hazay, C., Patra, A.: Leakage resilient two-party computation. Cryptology ePrint Archive, Report 2011/256 (2011)
13. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: CRYPTO (1989)
14. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: ASIACRYPT. pp. 613–631 (2010)
15. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC. pp. 621–630 (2009)
16. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS (2008)
17. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6) (1985)
18. Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: CRYPTO. pp. 526–544 (1989)
19. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. In: PKC (2010)
20. Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: CRYPTO (2011)
21. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: ACM (ed.) Proc. 19th STOC. pp. 218–229 (1987)
22. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proc. 17th STOC. pp. 291–304 (1985)
23. Goldwasser, S., Rothblum, G.: How to compute in the presence of leakage. In: FOCS (2012)
24. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for nizk. In: CRYPTO. pp. 97–111 (2006)
25. Haitner, I., Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions of protocols for secure computation. SIAM J. Comput. 40(2) (2011)
26. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium. pp. 45–60 (2008)
27. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: TCC (2011)
28. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: CRYPTO. pp. 572–591 (2008)
29. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: CRYPTO. pp. 463–481 (2003)
30. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: ASIACRYPT. pp. 703–720 (2009)
31. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: CRYPTO. pp. 104–113 (1996)
32. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: CRYPTO. pp. 18–35 (2009)
33. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. SIAM J. Comput. 40(6) (2011)
34. Yao, A.C.: How to generate and exchange secrets. In: Proc. 27th FOCS (1986)