Adaptively Secure Computation with Partial Erasures

Carmit Hazay* Yehuda Lindell† Arpita Patra‡

Abstract

Adaptive security is a strong corruption model that captures "hacking" attacks where an external attacker breaks into parties' machines in the midst of a protocol execution. There are two types of adaptively-secure protocols: *adaptive with erasures* and *adaptive without erasures*. Achieving adaptivity without erasures is preferable, since secure erasures are not always trivial. However, it seems far harder.

We introduce a new model of adaptive security called *adaptive security with partial erasures* that allows erasures, but only assumes them in a minimal sense. Specifically, if all parties are corrupted then security holds as long as *any single party* successfully erases. In addition, security holds if any proper subset of the parties is corrupted without erasures. We initiate a *theoretical study* of this new notion and demonstrate that secure computation in this setting is as efficient as static secure computation. In addition, we study the relations between semi-adaptive security [GWZ09], adaptive security with partial erasures, and adaptive security without any erasures. We prove that the existence of semi-adaptive OT implies secure computation in all these settings.

Keywords: Secure Two-Party Computation, Adaptive Security, Erasure, Non-committing Encryption, Oblivious Transfer

*Faculty of Engineering, Bar-Ilan University, Israel. Email: `carmit.hazay@biu.ac.il`. Work partially supported by a grant from the Israel Ministry of Science and Technology (grant No. 3-10883).

†Department of Computer Science, Bar-Ilan University, Israel. Email: `yehuda.lindell@biu.ac.il`. Work supported by the Israel Science Foundation (grant No. 189/11).

‡Department of Computer Science & Automation, Indian Institute of Science, India. Email: `arpita@csa.iisc.ernet.in`. Work partially supported by INSPIRE Faculty Award (DST/INSPIRE/04/2014/015727) offered by Department of Science & Technology of India.

1 Introduction

1.1 Background

Secure multi-party computation (MPC). In the setting of secure MPC, a set of parties with private inputs wish to jointly compute some function of their inputs while preserving certain security properties. Two of these properties are *privacy*, meaning that the output is learned but nothing else, and *correctness*, meaning that no corrupted party or parties can cause the output to deviate from the specified function. Security is formally defined by saying that no adversary attacking a real protocol can do more harm than an adversary in an ideal world where an incorruptible trusted third party computes the function for the parties and provides them their output. The adversary may be *semi-honest* (meaning that it follows the protocol specification but tries to learn more than allowed) or *malicious* (meaning that it can run any arbitrary polynomial-time attack strategy). Despite the stringent security requirements on such protocols, it is known that *any* two-party and multi-party function can be securely computed in the presence of semi-honest and malicious adversaries [Yao82, GMW87].

Adaptive security. The initial model considered for secure computation was one of a *static adversary* where the adversary controls a subset of the parties (who are called *corrupted*) before the protocol begins, and this subset cannot change. A stronger corruption model that allows the adversary to choose which parties to corrupt throughout the protocol execution, and as a function of its view; such an adversary is called *adaptive*. Adaptive corruptions model "hacking" attacks where an external attacker breaks into parties' machines in the midst of a protocol execution. In the case where protocols run over a long period of time (e.g., consider a "secure database search protocol" where a party can ask queries over time without revealing the query to the database), such attacks are very realistic. We remark that there are two types of adaptively-secure protocols: *adaptive with erasures*, where the honest parties may erase intermediate data as part of the protocol specification, and *adaptive without erasures* where no such erasures are assumed. It is clear that achieving adaptivity without erasures is preferable, since secure erasures are not always trivial (e.g., parts of memory can find their way to the swap file of a machine; if memory is not zeroed then "erased data" can remain in memory for a long time until garbage collection takes place). However, achieving adaptive security without erasures seems far harder than with erasures. First, protocols that achieve adaptivity without erasures are more complex and the computational hardness assumptions needed seem stronger; see [CLOS02, KO04, CDD$^+$04, IPS08]. In contrast, protocols that assume erasures are simpler and require seemingly weaker assumptions [BH92, Lin09, IPS09]. Furthermore, achieving efficiency seems also to be much harder. In particular, constant-round two-party computation that is adaptively secure with erasures is known [Lin09], but no analogous result is known for the case of no erasures. In addition, highly efficient protocols exist with erasures [BDOZ11, DPSZ12, NNOB12], but not without.[1]

We conclude that there is a high price of working with a model where no erasures are assumed. However, assuming that all parties successfully erase all data, as specified by the protocol, is also not desirable. This dilemma is the starting point of our work.

[1]Observe that the protocols of [BDOZ11, DPSZ12, NNOB12] all have a preprocessing phase followed by an online phase. The online phase is adaptively secure if all of the secrets used to generate the results of the preprocessing phase are erased. Since the preprocessing phase is independent of the inputs, it is also adaptively secure if corruptions take place during this phase.

1.2 Our Results

1.2.1 Adaptively Secure Computation with Partial Erasures – A New Model

In light of the above dilemma, we introduce a new model of adaptive security that allows erasures, but only assumes them in a minimal sense. Specifically, all parties may be given instructions to erase data (as in the model with erasures). However, security holds as long as *any single party* successfully erases. We stress that the identity of the party that successfully erases is not known, and this means that security is maintained as long as one of the parties' erase mechanism works, and even if all other parties' do not. We also remark that if any proper subset of the parties is corrupted (and so at least one of the parties remains uncorrupted), then all of the corruptions may be without erasures. We formalize this by having two corruption commands that can be issued in the real world: "corrupt-with-erase" (where the party is corrupted and has erased all data as specified by the protocol) and "corrupt-without-erase" (where the party is corrupted and erases nothing). Then, the requirement is that any adversary that corrupts *all* the parties must issue at least one "corrupt-with-erase" command. This elegantly captures the intuitive notion discussed above; no other changes to the standard definition of adaptive security are needed.

We initiate a *theoretical study* of this new notion of adaptivity, with the following results. On the one hand, the cryptographic hardness assumptions needed to achieve this notion of adaptivity *in general* are the same as needed for achieving full adaptivity without erasures. Thus, our goal of reducing the hardness assumptions is not achieved, at least for the general case. On the other hand, we do show that secure channels that are adaptively secure with partial erasures (via non-committing encryption) can be achieved with assumptions that are seemingly weaker than those used in all previous constructions. In addition, we show that adaptivity with partial erasures can yield more efficient and much simpler protocols. We demonstrate this for non-committing encryption, oblivious transfer and secure two-party computation protocol.

Modular composition. One important goal with respect to notions of security is that of composition. Specifically, it is highly desirable that protocols can be composed together in different ways in order to modularly construct more complex protocols. Our new model enables such composition, as long as the number of parties is preserved. Specifically, it is possible to combine a number of two-party (resp., m-party) protocols in order to obtain a more complex two-party (resp., m-party) protocol (this follows from [Can00] who shows that protocols that are adaptively secure with and without erasures compose sequentially). However, it is *not* possible to combine two-party protocols in order to obtain an m-party protocol with $m > 2$. This is because in the setting with m parties, security is guaranteed as long as one party successfully erases, even if the rest do not. Now, if each pair runs two-party protocols between them, then in many pairs neither party may successfully erase. Thus, if the two-party protocols are only secure as long as one party erases, then they may not maintain security. This is certainly a drawback of our model. However, we believe that the advantages (regarding assumptions and efficiency) outweigh this disadvantage. In particular, this is not of concern in the two-party setting (which is in many real-world cases the most interesting). Also, multiparty protocols can be designed from scratch for the desired number of parties in order to bypass this issue.

1.2.2 Relations Amongst the Different Security Notions

Recently, [GWZ09] introduced the notion of *two-party semi-adaptive security* in which one of the parties is statically corrupted (i.e., corrupt from the onset) and the other can be adaptively corrupted. This is a strictly weaker notion than adaptive security with partial erasures since the statically-corrupted party can always be viewed as the party that was corrupted "with erasures" (since at the onset there is nothing to erase anyway). Thus, any protocol that is adaptively secure with partial erasures is semi-adaptively secure.

We study the relation between semi-adaptive security, adaptive security with partial erasures, and adaptive security without any erasures.

We prove that any semi-adaptive oblivious transfer (OT) can be transformed into an adaptively secure OT with partial erasures by sending the messages of the semi-adaptive OT protocol using a non-committing encryption scheme (NCE) that is adaptively secure with partial erasures. (A similar transform was used by [GWZ09] who showed that adaptively secure OT without erasures can be achieved by sending the messages of a semi-adaptive OT protocol with an NCE scheme that is adaptively secure without any erasures.) In addition, we show that NCE that is adaptively secure without any erasures (and thus with partial erasures as well) can be constructed from any semi-adaptive OT. We therefore have the following theorem:

Theorem 1.1 *The following statements are equivalent:*

1. *There exists an OT protocol that is semi-adaptively secure;*
2. *There exists an OT protocol that is adaptively secure with partial erasures;*
3. *There exists an OT protocol that is adaptively secure without any erasures.*

The above holds for semi-honest and malicious adversaries.

This shows that our weaker notion of adaptive security (and the even weaker notion of semi-adaptivity) does not allow the construction of secure protocols that rely on weaker cryptographic hardness assumptions. This deepens our understanding of adaptive security. An important corollary of this result implies that MPC with adaptive security is reduced to semi-adaptive oblivious transfer. This follows by combining our result with the [GWZ09] transformation specified above, and the fact that the [GMW87] protocol is adaptively secure when instantiated with (fully) adaptively secure OT. Namely:

Corollary 1.2 (Informal.) *Assume the existence of semi-adaptive OT. Then, there exists a multi-party protocol with semi-honest and adaptive security.*

We summarize this discussion in Figure 1.

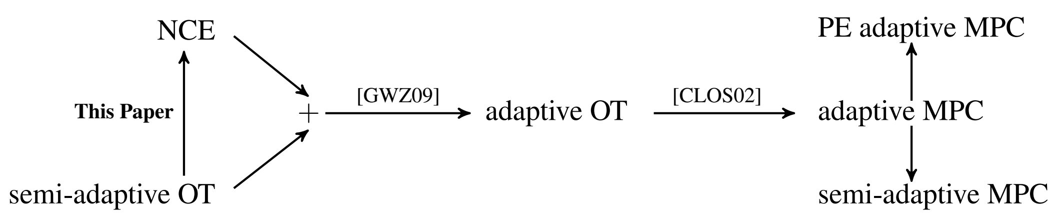

Figure 1: Notation $a \to b$ implies that b exists relying on the assumption that a exists. Notation $+$ implies that both primitives are needed to build the third primitive. In addition, PE denotes partial erasures.

Our next result studies NCE. Informally, NCE implements secure channels in the presence of adaptive corruptions. This is achieved by having an additional property where "dummy" ciphertexts can be generated and later decrypted into any plaintext. This is a strong security requirement and as such NCE schemes are relatively complicated, inefficient, and rely on seemingly stronger cryptographic hardness assumptions. We show that NCE in the partial erasures model can be achieved with a seemingly weaker assumption of public-key encryption with ciphertext samplability (NCE without any erasures is known to be achieved with ciphertext and public-key samplability; here we remove the latter assumption, and rely on the same assumption required for an NCE where *at most one party* is adaptively corrupted [DN00]). Specifically, we show the following in semi-honest and malicious settings:

Theorem 1.3 (Informal.) *Assume the existence of public-key encryption with oblivious and invertible sampling of ciphertexts. Then, there exists an NCE that is secure in the partial erasures model.*

1.2.3 Efficient Constructions with Partial Erasures

We further study the efficiency of basic primitives with partial erasures security and design strictly better constructions than in the adaptive setting, where the first two results hold in the *malicious* setting.

Non-committing encryption. We first construct NCE with partial erasures that is far more efficient than standard NCE without erasures. In particular, known constructions induce overhead of $\mathcal{O}(1)$ public-key operations for every transmitted bit [CFGN96, DN00, CDSMW09, GWZ09], while our protocol implies a constant number of such operations per polynomial-length message. Informally:

Theorem 1.4 (Informal.) *Under the standard assumptions for achieving adaptive security, there exists an NCE that is secure in the partial erasures model, where the sender and receiver compute $\mathcal{O}(1)$ public-key operations to transmit a message of length n.*

Our construction is a slightly modified version of the NCE construction that appears in [HP14].

Oblivious transfer. Oblivious transfer is one of the most fundamental and important primitives used for secure computation. Prior work on adaptively secure OT includes [Bea97, CLOS02, Lin09, GWZ09]. The most efficient protocol achieving adaptively-secure OT without any erasures is due to [GWZ09], who transform a semi-adaptive OT to a fully-adaptive OT without any erasures using NCE, as described above. Their construction transfers ℓ-bit strings using $\mathcal{O}(\ell)$ public-key operations and is built on an extension of the OT of [PVW08] that requires only a constant number of public-key operations, but is only statically secure.

We construct OT that is adaptively secure with partial erasures, and requires only a *a constant number* of public-key operations to transfer a string, like the static protocol of [PVW08]. We achieve this by constructing a semi-adaptive OT with a constant number of public-key operations, and then apply the constant-overhead NCE that is secure with partial erasures mentioned above. This transformation was already mentioned above and yields OT that is adaptively secure with partial erasures. We therefore prove:

Theorem 1.5 (Informal.) *Under the standard assumptions for achieving adaptive security, there exists an OT protocol that is adaptively secure with partial erasures, where the sender and receiver compute $\mathcal{O}(1)$ public-key operations in order to obliviously transfer a message of length n.*

Secure two-party computation. Finally, we show that the [GMW87] protocol is adaptively secure with partial erasures in the semi-honest setting when using oblivious transfer with partial erasures. This implies that when plugging in our oblivious transfer from above, the overall time complexity of [GMW87] is $O(|C|)$ public-key operations. This overhead matches its overhead in the static setting but with stronger security.

2 Preliminaries

We denote the security parameter by n. A function $\mu(\cdot)$ is *negligible* if for every polynomial $p(\cdot)$ there exists a value N such that for all $n > N$ it holds that $\mu(n) < \frac{1}{p(n)}$. We write PPT for (non-uniform) probabilistic polynomial-time and $a \leftarrow A$ to denote the uniform random sampling of a from a set A. We now specify the definitions of computationally indistinguishability and statistical distance.

Definition 2.1 (Computational indistinguishability by circuits) *Let $X = \{X_n(a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ and $Y = \{Y_n(a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ be distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \approx_c Y$, if for every family $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ of polynomial-size circuits, there exists a negligible* *function* $\mu(\cdot)$ *such that for all* $a \in \{0,1\}^*$,

$$|\Pr[\mathcal{C}_n(X_n(a)) = 1] - \Pr[\mathcal{C}_n(Y_n(a)) = 1]| < \mu(n).$$

Definition 2.2 (Statistical distance) *Let* X_n *and* Y_n *be random variables accepting values taken from a finite domain* $\Omega \subseteq \{0,1\}^n$. *The* statistical distance *between* X_n *and* Y_n *is*

$$SD(X_n, Y_n) = \frac{1}{2}\sum_{\omega \in \Omega} |\Pr[X_n = \omega] - \Pr[Y_n = \omega]|.$$

We say that X_n *and* Y_n *are* ϵ-close *if their statistical distance is at most* $SD(X_n, Y_n) \leq \epsilon(n)$. *We say that* X_n *and* Y_n *are* statistically close*, denoted* $X_n \approx_s Y_n$, *if* $\epsilon(n)$ *is negligible in* n.

2.1 Simulatable Public-Key Encryption

Informally, a simulatable public-key encryption scheme is IND-CPA secure PKE with four additional algorithms. An oblivious public-key generator (also denoted by oblivious sampler) $\widetilde{\mathsf{Gen}}$ and corresponding key faking algorithm (also denoted by invertible sampler) $\widetilde{\mathsf{Gen}}^{-1}$, an oblivious ciphertext generator $\widetilde{\mathsf{Enc}}$ and a corresponding ciphertext faking algorithm $\widetilde{\mathsf{Enc}}^{-1}$. Intuitively, the key faking algorithm is used to explain a legitimately generated public-key as an obliviously generated public-key. Similarly, the ciphertext faking algorithm is used to explain a legitimately generated ciphertext as an obliviously generated ciphertext.

Definition 2.3 (Secure simulatable PKE [DN00]) *A secure Simulatable PKE consists of a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Gen}}^{-1}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ *specified as follows:*

- **IND-CPA Security.** $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA secure (cf. Definition A.2).*
- **Oblivious public-key generation.** *Consider the experiment* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$, $r \leftarrow \widetilde{\mathsf{Gen}}^{-1}(\mathrm{PK})$ *and* $\mathrm{PK}' \leftarrow \widetilde{\mathsf{Gen}}(r')$. *Then,* $(r, \mathrm{PK}) \approx_c (r', \mathrm{PK}')$.
- **Oblivious ciphertext generation.** *For any message* m *in the appropriate domain, consider the experiment* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$, $c_1 \leftarrow \widetilde{\mathsf{Enc}}_{\mathrm{PK}}(r_1)$, $c_2 \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r_2)$, $r_1' \leftarrow \widetilde{\mathsf{Enc}}^{-1}(c_2)$. *Then,* $(\mathrm{PK}, r_1, c_1) \approx_c (\mathrm{PK}, r_1', c_2)$.

A simulatable PKE can be instantiated, for instance, by the El Gamal PKE [Gam85].

3 Security Definitions

In this section we introduce a new model of adaptive security that allows erasures, but only assumes them in a minimal sense. Specifically, all parties may be given instructions to erase data (as in the model with erasures). However, security holds as long as *any single party* successfully erases. We stress that the identity of the party that successfully erases is not known, and this means that security is maintained as long as one of the parties erase mechanism works, and even if all other parties' do not. We also remark that if any proper subset of the parties is corrupted (and so at least one of the parties remains uncorrupted), then all of the corruptions may be without erasures. In this section, we also recall the existing notion of semi-adaptive [GWZ09] and adaptive security. We introduce our definitions in the universal composability framework in the two-party setting [Can01], which we briefly recall below.

3.1 The Universal Composability Framework

The universal composability (UC) [Can01] framework was proposed by Canetti for defining security and composition of protocols. In this framework, one first defines an "ideal functionality" of a protocol and then proves that a particular implementation of this protocol operating in a given computational environment securely realizes the ideal functionality. The basic entities involved are two parties P_0 and P_1, a PPT adversary ADV and a PPT environment ENV. The real execution of a protocol Π, run by the parties in the presence of ADV and ENV, with input z, is modeled by a sequence of activations of the entities. The environment ENV is activated first, generating the inputs to the other parties. Then the protocol proceeds by having ADV exchange messages with the parties and ENV. Finally, the environment outputs one bit, which is the output of the protocol. The security of the protocols is defined by comparing the real execution of the protocol to an ideal process in which an additional entity, the ideal functionality $\mathcal{F}$ is introduced. $\mathcal{F}$ is an incorruptible trusted party that is programmed to produce the desired functionality of the given task. The parties are replaced by dummy parties who do not communicate with each other; whenever a dummy party is activated, it forwards its input to $\mathcal{F}$. Let SIM denote the PPT adversary in this idealized execution. As in the real-life execution, the output of the protocol execution is the one bit output of ENV. Now a protocol Π securely realizes an ideal functionality $\mathcal{F}$ if for any real-life adversary ADV there exists an ideal execution adversary SIM such that no ENV, on any input, can tell with non-negligible probability whether it is interacting with adversary ADV and parties running protocol Π in the real execution or with SIM and the ideal functionality $\mathcal{F}$ in the ideal execution. More precisely, a protocol Π securely realizes $\mathcal{F}$ if the two binary distribution ensembles, $\{\mathbf{REAL}_{\Pi,\textsc{Adv},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$ and $\{\mathbf{IDEAL}_{\mathcal{F},\textsc{Sim},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$ are computationally indistinguishable. The first ensemble describes ENV's output after interacting with ADV and the parties P_0, P_1 running protocol Π with inputs x_0, x_1 respectively. The second ensemble describes ENV's output after interacting with adversary SIM, ideal functionality $\mathcal{F}$ and dummy parties P_0, P_1 interacting with $\mathcal{F}$ with inputs x_0, x_1 respectively. Namely,

$$\{\mathbf{IDEAL}_{\mathcal{F},\textsc{Sim},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \approx_c \{\mathbf{REAL}_{\Pi,\textsc{Adv},\textsc{Env}}(n, x_0, x_1, z,)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

The $\mathcal{F}$-hybrid model. In order to construct some of our protocols, we will use secure two-party protocols as subprotocols. The standard way of doing this is to work in a "*hybrid model*" where both the parties interact with each other (as in the real model) in the outer protocol and use ideal functionality calls (as in the ideal model) for the subprotocols. Specifically, when constructing a protocol Π that uses a subprotocol for securely computing some functionality $\mathcal{F}$, the parties run Π and use "ideal calls" to $\mathcal{F}$ (instead of running the subprotocols implementing $\mathcal{F}$). The execution of Π that invokes $\mathcal{F}$ every time it requires to execute the subprotocol implementing $\mathcal{F}$ is called the *$\mathcal{F}$-hybrid execution of* Π and is denoted as $\Pi^{\mathcal{F}}$. The hybrid ensemble $\mathbf{HYBRID}_{\Pi^{\mathcal{F}},\textsc{Adv},\textsc{Env}}(n, x_0, x_1, z)$ describes ENV's output after interacting with ADV and the parties P_0, P_1 running protocol $\Pi^{\mathcal{F}}$ with inputs x_0, x_1 respectively. By UC definition, the hybrid ensemble should be indistinguishable from the real ensemble with respect to protocol Π where the calls to $\mathcal{F}$ are instantiated with a realization of $\mathcal{F}$.

3.2 Defining Semi-Adaptive, Partial Erasures and Adaptive Security

We begin with the formal definition of semi-adaptive security as stated in [GWZ09]. Loosely speaking, a protocol is semi-adaptively secure if it is secure with respect to *second-corruption adaptive adversarial strategy* as defined below.

Definition 3.1 *An adversarial strategy is second-corruption adaptive if either at least one of the parties is corrupted prior to the protocol execution or no party is ever corrupted. In the former case, the other party* *can be adaptively corrupted at any point during or after protocol execution. I.e, the first corruption (if at all it occurs) must be static and the second corruption can be adaptive.*

Intuitively, a semi-adaptive simulator should be able to equivocate the internal state of the party that is adaptively corrupted. There are two subtleties regarding the construction of such a simulator. First, since most functionalities require a trusted setup in order to be realized in UC settings, it must be ensured that this setup is generated independently of the identity of the corrupted party (which is not the case for all statically secure protocols). [GWZ09] denote this property by *setup-adaptive simulation*. Formally stated,

Definition 3.2 *A simulator* $\textsc{Sim} = (\textsc{Sim}_s, \textsc{Sim}_p)$ *is setup-adaptive if it first runs* $\textsc{Sim}_s$ *to simulate all the trusted setup and then runs* $\textsc{Sim}_p$ *(which is given any output generated by* $\textsc{Sim}_s$*) to simulate the protocol execution. While* $\textsc{Sim}_s$ *does not get to see which party is corrupted,* $\textsc{Sim}_p$ *gets to see it.*

Another subtlety that is formalized by *input-preserving* simulation is defined below.

Definition 3.3 *An adversary is protocol-honest if it corrupts one of the parties* P *prior to protocol execution and then follows the honest protocol specification using some input* x *on behalf of the corrupted party. A simulator* $\textsc{Sim}$ *is input-preserving if during the simulation of a protocol-honest adversary that corrupts party* P *and runs the honest protocol with input* x*,* $\textsc{Sim}$ *submits the same input* x *to the ideal functionality* $\mathcal{F}_f$ *on behalf of* P*.*

We are now ready to define semi-adaptive security.

Definition 3.4 *A protocol* Π *semi-adaptively realizes functionality* $\mathcal{F}$ *if for every* PPT *semi-honest/malicious adversary* $\textsc{Adv}$*, and for every* PPT *environment* $\textsc{Env}$ *that follow a second-corruption adaptive adversarial strategy, there exists a non-uniform setup-adaptive and input-preserving* PPT *ideal adversary* $\textsc{Sim}$ *such that for* $|x_0| = |x_1|$*:*

$$\{\mathbf{IDEAL}_{\mathcal{F},\textsc{Sim},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \approx_c \{\mathbf{REAL}_{\Pi,\textsc{Adv},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

Next, we recall adaptive security. A protocol is adaptively secure if it is secure with respect to an adversary who non-restrictively corrupts any party any time during or after the protocol execution.

Definition 3.5 *A protocol* Π *adaptively realizes the functionality* $\mathcal{F}$ *if for every* PPT *semi-honest/malicious adversary* $\textsc{Adv}$ *that can corrupt the parties adaptively during or after the protocol execution, and for every* PPT *environment* $\textsc{Env}$*, there exists a* PPT *ideal adversary* $\textsc{Sim}$ *such that for* $|x_0| = |x_1|$*:*

$$\{\mathbf{IDEAL}_{\mathcal{F},\textsc{Sim},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \approx_c \{\mathbf{REAL}_{\Pi,\textsc{Adv},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

We refer to adaptive security as *fully-adaptive* security in order to distinguish this notion from semi-adaptive security, or as *adaptive security without erasures* whenever we wish to distinguish this notion from adaptive security with partial erasures.

Finally, we describe the new notion of security introduced in this work, *adaptive security with partial erasures*, starting with the definition of adaptive with partial erasures adversarial strategy.

Definition 3.6 *An adaptive with partial erasures adversarial strategy implies that an adversary adaptively corrupts a party by either issuing a "corrupt-with-erase" or a "corrupt-without-erase" command. If it corrupts* P_i *issuing a "corrupt-with-erase" command in* i*th step of the protocol, then upon corruption it does not see the random inputs of* P_i *used up and until* $(i-1)$*th step of the protocol. On the other hand, if it corrupts* P_i *issuing a "corrupt-without-erase" command, then it sees the entire memory of* P_i *that includes the random inputs used in the protocol execution. If the adversary corrupts both the parties, then it must issue at least one "corrupt-with-erase" command.*

Definition 3.7 *A protocol* Π *adaptively realizes the functionality* $\mathcal{F}$ *with partial erasures if for every* PPT *semi-honest/malicious adversary* ADV *that follows the adaptive with partial erasures adversarial strategy, and for every* PPT *environment* ENV*, there exists a* PPT *ideal adversary* SIM *such that for* $|x_0| = |x_1|$*:*

$$\{\mathbf{IDEAL}_{\mathcal{F}_f,\textsc{Sim},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \approx_c \{\mathbf{REAL}_{\Pi,\textsc{Adv},\textsc{Env}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

3.3 Concrete Functionalities

Secure communication (SC). We define the functionality $\mathcal{F}_{\mathrm{SC}}$ for securely communicating a message m from SEN to REC, following the notations from [GWZ09]. To handle the appropriate leakage to the adversary in the ideal setting, the functionality is parameterized using a non-information oracle $\mathcal{O}$ which gets the values of the exchanged messages m and outputs some side information to the adversary. The security of this functionality depends on the security properties required for the oracle and thus can capture several notions such as NCE and ℓ-equivocal NCE. Specifically, for NCE the oracle only leaks the length of the message, whereas for ℓ-equivocal NCE with equivocality parameter ℓ the oracle leaks an ℓ-length vector such that the ith element in the vector depends on m, for some $i \in \{1, \dots, \ell\}$. In Figure 2 we define the message communication functionality with respect to oracle $\mathcal{O}$. Next, we define the oracles for the cases of NCE and ℓ-equivocal NCE, starting with the former.

Functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{O}}$

Functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{O}}$ communicates with sender SEN and receiver REC, and adversary SIM. The functionality starts with a channel-setup phase after which the two parties can send arbitrary many messages from one to another. The functionality is parameterized by a non-information oracle $\mathcal{O}$.

1. **Channel Setup.** Upon receiving input $(\mathsf{ChSetup}, sid, \textsc{Sen})$ from SEN, initialize the machine $\mathcal{O}$ and record the tuple $(sid, \mathcal{O})$. Pass the message $(\mathsf{ChSetup}, \textsc{Sen})$ to REC. In addition pass this message to $\mathcal{O}$ and forward its output to SIM.
2. **Message Transfer.** Upon receiving an input $(\mathsf{send}, sid, \textsc{Sen}, m)$ from party SEN, find a tuple $(sid, \mathcal{O})$, and, if none exits, ignore the message. Otherwise, send the message $(\mathsf{send}, sid, \textsc{Sen}, m)$ to REC. In addition invoke $\mathcal{O}$ with $(\mathsf{send}, sid, \textsc{Sen}, m)$ and forward its output to SIM.
3. **Corruption.** Upon receiving message $(\mathsf{corrupt}, sid, P)$ from SIM where $P \in \{\textsc{Sen}, \textsc{Rec}\}$, send $(\mathsf{corrupt}, sid, P)$ to $\mathcal{O}$ and forward its output to the adversary. After the first corruption, stop the execution of $\mathcal{O}$ and give SIM complete control over the functionality to let it learn all inputs and specify any outputs.

Figure 2: The message communication functionality.

Definition 3.8 $\mathcal{O}$*, on input* $(\mathsf{send}, sid, \textsc{Sen}, m)$*, produces the output* $(\mathsf{send}, sid, \textsc{Sen}, |m|)$*, and on any inputs corresponding to the* ChSetup*,* Corrupt *commands produces no output. We call the functionality* $\mathcal{F}_{\mathrm{SC}}^{\mathcal{O}}$ *or just* $\mathcal{F}_{\mathrm{SC}}$ *for brevity, an NCE functionality. A real world protocol which realizes* $\mathcal{F}_{\mathrm{SC}}$ *is called an NCE scheme.*

In order to define $\mathcal{O}$ for ℓ-equivocal NCE, we present the following definition first.

Definition 3.9 *An oracle* $\mathcal{I}$ *is called message-ignoring oracle if, on any input* $(\mathsf{send}, sid, \textsc{Sen}, m)$*, it ignores the message value* m *and processes only the input* $(\mathsf{send}, sid, \textsc{Sen}, |m|)$*. An oracle* $\mathcal{M}$ *is called message-processing oracle if it has no such restrictions. We call a pair of oracles* $(\mathcal{M}, \mathcal{I})$ *well-matched if no* PPT *distinguisher $\mathcal{D}$ (with oracle access to either $\mathcal{M}$ or $\mathcal{I}$) can distinguish the message-processing oracle $\mathcal{M}$ from the message-ignoring oracle $\mathcal{I}$.*

Definition 3.10 *Let $(\mathcal{M}, \mathcal{I})$ be a well-matched pair which consists of a message-processing and a message-ignoring oracle respectively. Then we define $\mathcal{O}$ for ℓ-equivocal NCE as $\mathcal{O}^\ell$ with the following structure. Note that $\mathcal{O}^\ell$ is a (stateful) oracle.*

- *Upon initialization, $\mathcal{O}^\ell$ chooses a uniformly random index $i \leftarrow \{1, \ldots, \ell\}$. In addition it initializes a tuple of ℓ independent oracles: $(\mathcal{O}_1, \ldots, \mathcal{O}_\ell)$, where $\mathcal{O}_i = \mathcal{M}$ and for $j \neq i$, the oracles $\mathcal{O}_j$ are independent copies of $\mathcal{I}$.*
- *Whenever $\mathcal{O}^\ell$ receives inputs of the form $(\mathsf{ChSetup}, sid, \textsc{Sen})$ or $(\mathsf{send}, sid, \textsc{Sen}, m)$, it passes the input to each $\mathcal{O}_i$ receiving an output y_i. It then outputs the vector $(y_1, \ldots, y_\ell)$.*
- *Upon receiving an input $(\mathsf{corrupt}, sid, P)$, the oracle reveals the internal state of the message-processing oracle $\mathcal{O}_i$ only.*

For any such oracle $\mathcal{O}^\ell$, we call $\mathcal{F}_{\mathrm{SC}}^{\mathcal{O}^\ell}$ an ℓ-equivocal NCE functionality. For brevity, we will also use the notation $\mathcal{F}_{\mathrm{SC}}^\ell$ to denote $\mathcal{F}_{\mathrm{SC}}^{\mathcal{O}^\ell}$. Lastly, a real world protocol which realizes $\mathcal{F}_{\mathrm{SC}}^\ell$ is called an ℓ-equivocal NCE scheme.

As before, no information about message m is revealed during the 'message transfer' stage. However, the internal state of the message-processing oracle $\mathcal{O}_i$, which is revealed upon corruption, might be "committing". Nevertheless, a simulator can simulate the communication between two honest parties over a secure channel, as modeled by $\mathcal{F}_{\mathrm{SC}}^\ell$, in a way that allows it to explain later this communication as any one of ℓ possibilities.

On semi-honest and malicious realizations of $\mathcal{F}_{\mathrm{SC}}$. Notably, a semi-honest secure realization of $\mathcal{F}_{\mathrm{SC}}$ (or $\mathcal{F}_{\mathrm{SC}}^\ell$) implies malicious security. This is because when the sender is statically corrupted (or before the message has been transferred), the simulator can extract the message by playing the role of the honest receiver (which does not introduce any input to the protocol), and then forward to $\mathcal{F}_{\mathrm{SC}}$ whatever the emulated receiver outputs. On the other hand, in case the receiver is statically corrupted the simulator obtains m from $\mathcal{F}_{\mathrm{SC}}$ and perfectly emulates the communication with the corrupted receiver. Moreover, adaptive corruptions that occur after the message has been transferred are easily simulated as in the semi-honest case. Consequently, we only realize $\mathcal{F}_{\mathrm{SC}}$ in the presence of semi-honest adversaries, regardless of the corruption strategy.

Oblivious transfer (OT). The 1-out-of-2 OT functionality is defined in Figure 3. In a bit OT $x_0, x_1 \in \{0,1\}$, whereas in a string OT $x_0, x_1 \in \{0,1\}^n$.

Functionality $\mathcal{F}_{\mathrm{OT}}$

Functionality $\mathcal{F}_{\mathrm{OT}}$ communicates with with sender $\textsc{Sen}$ and receiver $\textsc{Rec}$, and adversary $\textsc{Sim}$.

1. Upon receiving input $(\mathsf{sender}, sid, x_0, x_1)$ from $\textsc{Sen}$ where $x_0, x_1 \in \{0,1\}^n$, record (sid, x_0, x_1).
2. Upon receiving $(\mathsf{receiver}, sid, \sigma)$ from $\textsc{Rec}$, where a tuple (sid, x_0, x_1) is recorded and $\sigma \in \{0,1\}$, send (sid, x_σ) to $\textsc{Rec}$ and sid to $\textsc{Sim}$. Otherwise, abort.

Figure 3: The oblivious transfer functionality.

Secure computation. In Figure 4, we define $\mathcal{F}$ that computes a general function with two inputs and two outputs $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_0, f_1)$ maps pairs of inputs to pairs of outputs. Specifically, the first party with input x_0 wishes to receive $f_0(x_0, x_1)$, while the second party with input x_1 wishes to obtain $f_1(x_0, x_1)$.

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ communicates with two parties P_0 and P_1, and adversary $\textsc{Sim}$.

1. Upon receiving input $(\mathsf{Input}_0, sid, x_0)$ from P_0, record the value (P_0, x_0) and send the message (Input_0, sid) to $\textsc{Sim}$. Ignore future $(\mathsf{Input}_0, ...)$ inputs. Similarly, upon receiving input $(\mathsf{Input}_1, sid, x_1)$ from P_1, record the value (P_1, x_1) and send the message (Input_1, sid) to $\textsc{Sim}$. Ignore future $(\mathsf{Input}_1, ...)$ inputs.
2. Upon receiving (Output_0, sid) from $\textsc{Sim}$, if either (P_0, x_0) and (P_1, x_1) are not recorded, ignore the message. Else if (y_0, y_1) is not recorded, then compute $(y_0, y_1) \leftarrow f(x_0, x_1)$ and record (y_0, y_1); send $(\mathsf{Output}_0, sid, y_0)$ to P_0. Ignore future $(\mathsf{Output}_0, ...)$ inputs from $\textsc{Sim}$. Similarly, upon receipt of (Output_1, sid) from $\textsc{Sim}$, send $(\mathsf{Output}_1, sid, y_1)$ to P_1. Ignore future $(\mathsf{Output}_1, ...)$ inputs from $\textsc{Sim}$.

Figure 4: The two-party computation functionality for function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$.

Setup generation. As noted in the literature, most functionalities cannot be realized in the UC framework without a trusted setup. One common form of setup is the common reference string (CRS) which is modeled by a functionality $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$, defined in Figure 5.

Functionality $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$

$\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$ is parameterized by a PPT sampling algorithm $\mathcal{D}$.

On input CRS, sid from P, if there is no value crs recorded then choose $crs \leftarrow \mathcal{D}()$ and record it. Send $(\text{CRS}, sid, crs, P)$ to $\textsc{Sim}$ and P.

Figure 5: The common reference string ideal functionality.

4 Relations Amongst Semi-Adaptive, Adaptive with Partial Erasures and Adaptive Security

In this section, we study the relations between semi-adaptive security, adaptive security with partial erasures and adaptive security with no erasures (denoted also by adaptive or fully-adaptive). Our first transformation shows that semi-adaptive OT implies NCE (Section 4.1). This further implies that the security of adaptively secure NCE without any erasures can be reduced to the security of semi-adaptive OT. Combining this result with the [GWZ09] compiler that transforms semi-adaptive OT to fully-adaptive OT using NCE, and the fact that the [GMW87] is adaptively secure when instantiated with (fully) adaptively secure OT, the following important theorem is concluded:

Theorem 4.1 *Assume the existence of semi-adaptive semi-honest OT. Then, there exists a multi-party protocol that adaptively (respectively, with partial erasures or semi-adaptively) realizes any well-formed multi-party efficient functionality f in the presence of semi-honest adversaries.*

We refer to [CLOS02] regarding the formal definition of well-formed functionalities. In the second part of this section we demonstrate the feasibility of adaptively secure NCE with partial erasures under *strictly weaker* hardness assumptions than simulatable PKE (Section 4.2), where all known NCE constructions are based on (a variant) of this assumption. Finally, we prove that any semi-adaptive protocol can be transformed into an adaptively secure protocol with partial erasures by encrypting the messages of the semi-adaptive protocol using NCE that is adaptively secure with partial erasures. This result is given in Section 4.3.

4.1 NCE from Semi-Adaptive Oblivious Transfer

Security in the semi-adaptive setting requires that one of the parties is statically corrupted, which implies a weaker notion of security than adaptive. In the following, we show that NCE, that is adaptively secure *without any erasures* (and thus secure with partial erasures as well), can be constructed from any semi-adaptive OT. Namely, recalling the OT ideal functionality from Figure 3, we exploit the fact that a semi-adaptive OT implies a simulator that either equivocates the receiver's input bit σ (in case the sender is statically corrupted) or $x_{1-\sigma}$ (in case the receiver is statically corrupted), but never both. For simplicity we describe our protocol in the $\mathcal{F}_{\rm OT}$-hybrid setting.

Protocol 1 (NCE from semi-adaptive OT)

- **Inputs:** *Sender* SEN *is given input message* $m \in \{0,1\}$.
- **The Protocol:**
  1. **Executing semi-adaptive OT on random inputs.** *The parties invoke $\mathcal{F}_{\rm OT}$ on random inputs, where* SEN *plays the receiver on a random bit σ and* REC *plays the sender on random bits r_0, r_1. Let r denote the output of* SEN.
  2. **Message from the sender.** SEN *sends c to* REC *where $c = \sigma \oplus m$ if $r = \sigma$. Otherwise, $c \leftarrow_R \{0,1\}$.*
  3. **Output.** *Upon receiving c,* REC *recovers the message by computing $m = c \oplus r_0$.*

Theorem 4.2 *Assume that $\mathcal{F}_{\rm OT}$ is implemented in the presence of semi-adaptive semi-honest adversaries. Then, Protocol 1 adaptively realizes $\mathcal{F}_{\rm SC}$ in the presence of semi-honest adversaries, where correctness holds with probability* $5/8$.

The complete proof follows. We note that by repeating Protocol 1 sufficiently many times, one obtains a protocol $\Pi_{\rm NCE}$ where the probability of correctness is arbitrarily close to 1. The above theorem implies a simpler NCE protocol assuming the Quadratic Residuosity (QR) assumption (based on the semi-adaptive OT of [GWZ09] that is proven under the QR assumption). In addition, plugging in our semi-adaptive OT from Section 5.2.4 implies the first NCE under the DCR hardness assumption.

Proof: Our proof is split into correctness and security arguments.

Correctness. We show that REC restores the correct value of m with probability $5/8$. To argue that correctness holds we examine all possible combinations of the three variables r_0, r_1, σ (recall that these bits are chosen uniformly at random) and measure the probability of correctness for each such a combination in Figure 6. For example, consider the first row. Then, whenever the sender picks $r_0 = r_1 = 0$ and the receiver picks $\sigma = 0$, the receiver always outputs the correct message since the conditions $r_\sigma = \sigma$ and $r_0 = r_1$ are met.

On the other hand, if the sender picks $r_0 = r_1 = 0$ but the receiver picks $\sigma = 1$, then the correct recovery of the message is guaranteed with probability $1/2$. This is because $\sigma \neq r_\sigma$ and the sender uses a random bit c. One can verify the other rows similarly. Inspecting Figure 6, we conclude that correctness is satisfied with probability $5/8$ (where this probability is obtained by summing up the probabilities listed in the table, divided by the total number of combinations/rows which equals to 8).

$r_0 r_1 \sigma$	Correctness Probability
000	1
001	1/2
010	1
011	0
100	1/2
101	1/2
110	1/2
111	1

Figure 6: Correctness probability by r_0, r_1, σ.

Security. Before presenting our simulator strategy we examine in Figures 7 and 8 the probabilities of each of the 16 combinations for variables r_0, r_1, σ, c when the message equals 0 and 1 in the hybrid execution, respectively. Each row in these figures corresponds to a unique combination of r_0, r_1, σ, c and the probability that this combination occurs. Note that there are overall 12 combinations for each table that occur with a non-zero probability, whereas the remaining 4 combinations do not occur at all. For instance, 0001 cannot occur when $m = 0$ since SEN picks $r_0 = r_1 = 0$ and REC picks $\sigma = 0$ thus $c = \sigma \oplus m = 0$. Due to similar reasons, combinations 0101, 0110 and 1110 cannot occur when $m = 0$ and combinations 0000, 0100, 0111 and 1111 cannot occur when $m = 1$. We note that each such combination occurs with the following probability: first, any joint value for r_0, r_1, σ is determined with probability $1/8$ since these bits are picked randomly. Next, if $r_\sigma = \sigma$ then c is computed deterministically and any value for r_0, r_1, σ, c is determined with probability $1/8$. Whereas, if $r_\sigma \neq \sigma$ then c is picked randomly and so any potential combination for r_0, r_1, σ, c occurs with probability $1/16$.

Our simulator strategy is as follows. The simulator sends $(\mathsf{ChSetup}, sid, \text{SEN})$ to $\mathcal{F}_{\text{SC}}$, receiving back $(\mathsf{send}, sid, \text{SEN}, |m|)$. It them determines r_0, r_1, σ, c uniformly at random and emulates the role of $\mathcal{F}_{\text{OT}}$. Finally, the simulator concludes by sending c of the appropriate length to REC on behalf of SEN. Specifically, the only message transmitted by the protocol and seen by the adversary until no corruption occurs is c. Then, upon corrupting a party the simulator learns the message m and determines r_0, r_1, σ to be consistent with m and c by equivocating σ or $r_{1-\sigma}$ (if required). The ability to equivocate these values follows from the fact that $\mathcal{F}_{\text{OT}}$ is instantiated with a semi-adaptive OT thus the simulator can equivocate either the receiver's input σ or the sender's other input $r_{1-\sigma}$, sent within the semi-adaptive OT. As demonstrated below, this enables the simulator to generate the same probability distribution for r_0, r_1, σ, c as in hybrid execution. We stress that without equivocating these values the adversary can easily distinguish the hybrid and simulated executions, as some combinations do not occur in the hybrid execution while they will occur in the simulated run. More specifically, each of the 16 combinations for r_0, r_1, σ, c occur with the same probability

$r_0 r_1 \sigma c$	Prob.
0000	1/8
0001	0
0010	1/16
0011	1/16
0100	1/8
0101	0
0110	0
0111	1/8
1000	1/16
1001	1/16
1010	1/16
1011	1/16
1100	1/16
1101	1/16
1110	0
1111	1/8

Figure 7: $m = 0$.

$r_0 r_1 \sigma c$	Prob.
0000	0
0001	1/8
0010	1/16
0011	1/16
0100	0
0101	1/8
0110	1/8
0111	0
1000	1/16
1001	1/16
1010	1/16
1011	1/16
1100	1/16
1101	1/16
1110	1/8
1111	0

Figure 8: $m = 1$.

in the simulation, whereas in the hybrid execution their probability varies between $1/16$, $1/8$ or even 0.

We explain our simulation strategy in Figures 9 and 10 separately for $m = 0$ and $m = 1$, and specify for each value whether the simulator equivocates either σ or $r_{1-\sigma}$. We begin with the case where the first corruption takes place after c is transmitted. Then the rightmost column of each table in Figure 11 specifies the values after equivocation takes place, while the leftmost column specifies the initial values as picked by the simulator (where each value is picked with probability $1/16$). Importantly, while the leftmost column in each table lists all 16 potential combinations for r_0, r_1, σ, c only the 12 combinations that occur with non-zero probability in the hybrid execution appear in the rightmost columns, such that some of the combinations are repeated twice. We now show that the combinations that occur with probability $1/8$ in the hybrid execution appear exactly twice in the rightmost columns, the combinations that occur with probability $1/16$ in the hybrid execution appear exactly once in the rightmost columns, and that the combinations that occur with probability zero in the hybrid execution do not appear at all. This proves that the probability distribution on r_0, r_1, σ, c is identical in both hybrid and simulated executions.

$r_0r_1\sigma c$	Equivocate	$r'_0r'_1\sigma' c$
0000	Nothing	0000
0001	σ	0011
0010	σ	0000
0011	$r_{1-\sigma}$	1011
0100	Nothing	0100
0101	σ	0111
0110	σ	0100
0111	Nothing	0111
1000	σ	1010
1001	$r_{1-\sigma}$	1101
1010	$r_{1-\sigma}$	0010
1011	σ	1001
1100	$r_{1-\sigma}$	1000
1101	σ	1111
1110	σ	1100
1111	Nothing	1111

Figure 9: $m = 0$.

$r_0r_1\sigma c$	Equivocate	$r'_0r'_1\sigma' c$
0000	σ	0010
0001	Nothing	0001
0010	$r_{1-\sigma}$	1010
0011	σ	0001
0100	σ	0110
0101	Nothing	0101
0110	Nothing	0110
0111	σ	0101
1000	$r_{1-\sigma}$	1100
1001	σ	1011
1010	σ	1000
1011	$r_{1-\sigma}$	0011
1100	σ	1110
1101	$r_{1-\sigma}$	1001
1110	Nothing	1110
1111	σ	1101

Figure 10: $m = 1$.

Figure 11: The probability distributions in the simulation.

Say $m = 0$ then each of the following combinations $0000, 0100, 0111, 1111$ occurs in the hybrid execution with probability $1/8$ (see Figure 7). It is easy to verify that these combinations occur with the same probability in the simulated execution as well since for each value there are exactly two rows in the rightmost column of Figure 9 that contain this value. Apart from these combinations there are 8 more values that may occur in the hybrid execution with probability $1/16$. Specifically, 0010, 0011, 1000, 1001, 1010, 1011, 1100 and 1101, as shown in Figure 7. These combinations occur with probability $1/16$ in the simulation as well, since there is a single row in Figure 9 that corresponds to each of these combinations.

Next, say $m = 1$ then each of the following combinations $0001, 0101, 0110, 1110$ occur in the hybrid execution with probability $1/8$ (see Figure 8). It is easy to verify that these combinations occur with the same probability in the simulated execution as well since for each value there are exactly two rows in the rightmost column of Figure 10 that contain this value. As in the case of $m = 0$, the remaining eight combinations that occur in the hybrid execution with probability $1/16$ occur in the simulated execution with the same probability, i.e., each of this values appears only once in the rightmost column of Figure 10.

Finally, any combination that does not occur in the hybrid execution does not occur in simulation as well. We showed that the simulated and hybrid executions are identical in the $\mathcal{F}_{\rm OT}$-hybrid model when corruption takes places after c is transferred. If corruption occurs before Step 2 of the protocol is concluded (note that the message m is not used by the protocol before this step), the simulator learns the message before c is sent and simulates the role of the sender (if SEN is not corrupted) as in the hybrid execution, generating an identical distribution as well. This concludes our proof. ■

4.2 Adaptively Secure NCE with Partial Erasures under Weaker Assumptions

In this section we construct an NCE scheme with partial erasures security based on PKE with oblivious ciphertexts generation. This is in contrast to fully adaptive NCE constructions that require simulatable PKE, which implies the oblivious generation of *both* public-keys and ciphertexts (see a formal definition in Section 2.1). Our construction follows the NCE approach of [DN00] with the difference that instead of locally generating the public-keys, they are now being generated via a two-party protocol $\pi_{\rm KeySetup}$ that realizes functionality,

$$\mathcal{F}_{\rm KeySetup} : (\alpha, \beta) \mapsto \big((\mathrm{PK}_0^1, \mathrm{PK}_1^1, \mathrm{PK}_0^2, \mathrm{PK}_1^2, \mathrm{SK}_\alpha^1), (\mathrm{PK}_0^1, \mathrm{PK}_1^1, \mathrm{PK}_0^2, \mathrm{PK}_1^2, \mathrm{SK}_\beta^2)\big).$$

Namely, the parties first agree on two public messages $m_0, m_1 \in \mathcal{M}$ and invoke $\pi_{\rm KeySetup}$. Next, REC picks a random bit δ and encrypts m_δ under PK_δ^1, and obliviously samples the $(1-\delta)$th ciphertext. Similarly, SEN picks a random bit γ and encrypts m_γ under PK_γ^2, and obliviously samples the $(1-\gamma)$th ciphertext. The parties exchange ciphertexts and decrypt them using the secret keys that they possess. Note that if $\mathcal{M}$ is super polynomial then it is unlikely that an obliviously sampled ciphertext is decrypted into $m_{1-\delta}$, and thus SEN can correctly conclude whether $\alpha = \delta$ with very high probability. Similarly, REC correctly concludes whether $\beta = \gamma$. If both equalities hold SEN sends its message blinded with $\alpha \oplus \gamma$. Otherwise, the parties make another attempt, running the protocol again. This implies that an expected number of four attempts yields a successful attempt since the two pairs of bits equal with probability $1/4$. We observe that a single attempt of our protocol is composed of two attempts of the [DN00] protocol, where the local key generation is replaced by a two-party protocol. Specifically, combining two such attempts enables to equivocate either (α, δ) or (β, γ), which implies message equivocation.

Note that it is sufficient to realize $\mathcal{F}_{\rm KeySetup}$ using a *statically* secure protocol that can be implemented under the same assumption of PKE with oblivious ciphertexts generation. This is because any PKE with this property implies oblivious transfer [EGL85] which, in turn, implies general secure computation in the presence of semi-honest adversaries [Yao82]. We stress that the parties cannot simply choose the public and secret keys by themselves since that would require the additional assumption of oblivious public-key generation, that we wish to avoid here. Formally, denoting by $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ a PKE with oblivious ciphertexts generation, we implement functionality $\mathcal{F}_{\rm SC}$ as follows.

Protocol 2 (NCE with partial erasures ($\Pi_{\rm PE-NCE}$))

- **Inputs:** *Sender* SEN *is given input message* $m \in \{0,1\}$.
- **The Protocol:**

  1. **Agreeing on messages** m_0 **and** m_1**.** *The parties publicly agree on messages* m_0 *and* m_1.
  2. **Invoking** $\pi_{\rm KeySetup}$**.** SEN *and* REC *pick random bits* α *and* β*, respectively, and invoke* $\pi_{\rm KeySetup}$ *on these bits. Denote by* $(\mathrm{PK}_0^1, \mathrm{PK}_1^1, \mathrm{PK}_0^2, \mathrm{PK}_1^2, \mathrm{SK}_\alpha^1)$ *the output of* SEN *and by* $(\mathrm{PK}_0^1, \mathrm{PK}_1^1, \mathrm{PK}_0^2, \mathrm{PK}_1^2, \mathrm{SK}_\beta^2)$ *the output of* REC.

3. **Exchanging ciphertexts.** *Next,* SEN *picks a uniform bit* γ *and sends* c_0^2, c_1^2 *where* $c_\gamma^2 \leftarrow \mathsf{Enc}_{\mathrm{PK}_\gamma^2}(m_\gamma; r_\gamma^2)$ *and* $c_{1-\gamma}^2 \leftarrow \widetilde{\mathsf{Enc}}_{\mathrm{PK}_{1-\gamma}^2}(r_{1-\gamma}^2)$. *Moreover,* REC *picks a random bit* δ *and sends* c_0^1, c_1^1 *where* $c_\delta^1 \leftarrow \mathsf{Enc}_{\mathrm{PK}_\delta^1}(m_\delta; r_\delta^1)$ *and* $c_{1-\delta}^1 \leftarrow \widetilde{\mathsf{Enc}}_{\mathrm{PK}_{1-\delta}^1}(r_{1-\delta}^1)$.
4. **Checking for equality of bits.** *Upon receiving* (c_0^1, c_1^1), SEN *checks whether* $m_\alpha = \mathsf{Dec}_{\mathrm{SK}_\alpha^1}(c_\alpha^1)$ *and sends* $s_0 = 1$ *if equality holds, and* $s_0 = 0$ *otherwise. Similarly, upon receiving* (c_0^2, c_1^2), REC *checks whether* $m_\beta = \mathsf{Dec}_{\mathrm{SK}_\beta^2}(c_\beta^2)$ *and sends* $s_1 = 1$ *if equality holds, and* $s_1 = 0$ *otherwise.*
If one of the parties sent the bit 0 *the parties return to Step 2.*
5. **Message from SEN.** *Otherwise,* SEN *computes* $z = m \oplus \alpha \oplus \gamma$ *and sends* z.
6. **Output.** *Finally,* REC *computes* $m = z \oplus \delta \oplus \beta$ *and outputs* m.

The security proof relies on the ability of a party to erase its randomness within π_{KeySetup}. Specifically, if a party is corrupted before the execution of π_{KeySetup} is completed then the simulator completes the execution as in the real setting. Otherwise, the input/output within π_{KeySetup} of the corrupted party that uses erasures can be equivocated. Combining these observations with oblivious ciphertexts generation enables the simulator to equivocate the common bit that masks the message. Furthermore, unsuccessful attempts (Steps 2-4) are perfectly simulated.

Theorem 4.3 *Assume* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ *is IND-CPA secure PKE with oblivious ciphertexts generation and let* π_{KeySetup} *be a protocol that statically realizes* $\mathcal{F}_{\mathrm{KeySetup}}$ *in the presence of semi-honest adversaries. Then, Protocol 2 adaptively realizes* $\mathcal{F}_{\mathrm{SC}}$ *with partial erasures in the presence of semi-honest adversaries.*

Proof: We say that an attempt is successful if $s_0 = s_1 = 1$ and denote other attempts by failed attempts. We argue first that the expected number of failed attempts is constant. Recall that a successful attempt occurs only when the parties sample the same pair of bits. Namely, whenever $\alpha = \delta$ and $\beta = \gamma$ such that α, γ are uniformly chosen by SEN and β, δ are uniformly chosen by REC. This implies that a successful attempt occurs after 4 independent attempts on the average since both equalities hold with probability $1/4$.

We proceed with the proof of security. We recall first that in the partial erasures settings the adversary either corrupts both parties while at most one of them is allowed to use erasures, or only a single party without allowing erasures. We focus our attention on the former corruption case since it is stronger. We distinguish four types of attempts $\mathrm{ATTEMPT}_{s_0,s_1}$ for $s_0, s_1 \in \{0,1\}$ and construct a different simulator $\mathrm{SIMATTEMPT}_{s_0,s_1}$ for each such pair of values. An attempt $\mathrm{ATTEMPT}_{s_0,s_1}$ with $s_0 = s_1 = 1$ is denoted as a successful attempt. For all other combinations of s_0, s_1, $\mathrm{ATTEMPT}_{s_0,s_1}$ is denoted as a failed attempt. A single simulator SIM can be built based on the four types of simulators $\mathrm{SIMATTEMPT}_{s_0,s_1}$ for $s_0, s_1 \in \{0,1\}$. Specifically, SIM first picks s_0 and s_1 independently and uniformly at random from $\{0,1\}$ and then invokes $\mathrm{SIMATTEMPT}_{s_0,s_1}$. In case of a failed attempt SIM repeats the above for fresh s_0, s_1. In case of a successful attempt SIM sends a random bit z on behalf of SEN and concludes the simulation.

We now proceed with the descriptions of our simulators for the different types of attempts.

The descriptions of $\mathrm{SIMATTEMPT}_{0,0}$, $\mathrm{SIMATTEMPT}_{0,1}$ and $\mathrm{SIMATTEMPT}_{1,0}$. In a failed attempt, (**1**) $\mathrm{SIMATTEMPT}_{0,0}$ randomly picks α and sets $\delta = 1 - \alpha$. It further picks β at random and sets $\gamma = 1 - \beta$. (**2**) $\mathrm{SIMATTEMPT}_{0,1}$ randomly picks α and sets $\delta = 1 - \alpha$. It further picks β at random and sets $\gamma = \beta$. (**3**) $\mathrm{SIMATTEMPT}_{1,0}$ randomly picks α and sets $\delta = \alpha$. It further picks at random β and sets $\gamma = 1 - \beta$. The simulators then play the role of the honest parties using these fixed values for $\alpha, \beta, \gamma, \delta$. We claim that the real and the simulated executions are identically distributed for any combination (even without relying on erasures and oblivious generation). This is because the distribution on α and β is identical as these values are picked at random in both executions. Finally, note that δ and γ are picked with probability

$1/2$ each, since these probabilities depend on the probability we sample s_0 and s_1. We demonstrate our argument for simulator $\textsc{SimAttempt}_{0,0}$ where $s_0 = s_1 = 0$ implying that $\alpha \neq \delta$ and $\beta \neq \gamma$. Specifically, $\Pr[\delta = 1 - \alpha] = \Pr[s_0 = 0] = 1/2$ and $\Pr[\gamma = 1 - \beta] = \Pr[s_1 = 0] = 1/2$. Notably, the distributions are identical irrespective of the time of corruption or the identity of the corrupted party with erasures, since the simulator knows in advance that the attempt fails and thus it will never reach Step 5. A similar argument can be made with respect to the other two simulators.

The description of $\textsc{SimAttempt}_{1,1}$. In case of a successful attempt, simulator $\textsc{SimAttempt}_{1,1}$ fixes α and β by choosing two bits uniformly at random and sends $(\mathsf{ChSetup}, sid, \textsc{Sen})$ to $\mathcal{F}_{\mathrm{SC}}$, receiving back $(\mathsf{send}, sid, \textsc{Sen}, |m|)$. It then honestly emulates $\textsc{Sen}$ and $\textsc{Rec}$ in π_{KeySetup} with the appropriate randomness. Denote by $(\mathrm{PK}_0^1, \mathrm{SK}_0^1), (\mathrm{PK}_1^1, \mathrm{SK}_1^1)$ and $(\mathrm{PK}_0^2, \mathrm{SK}_0^2), (\mathrm{PK}_1^2, \mathrm{SK}_1^2)$ the two pairs of public/secret keys generated within π_{KeySetup}. It then generates two pairs of ciphertexts (on behalf of both $\textsc{Sen}$ and $\textsc{Rec}$) using algorithm Enc instead of obliviously sampling one of the ciphertexts from each pair. Namely, it generates $c_0^1 \leftarrow \mathsf{Enc}_{\mathrm{PK}_0^1}(m_0, r_0^1)$ and $c_1^1 \leftarrow \mathsf{Enc}_{\mathrm{PK}_1^1}(m_1; r_1^1)$ and sends c_0^1, c_1^1 to $\textsc{Sen}$ on behalf of $\textsc{Rec}$. It then generates $c_0^2 \leftarrow \mathsf{Enc}_{\mathrm{PK}_0^2}(m_0, r_0^2)$ and $c_1^2 \leftarrow \mathsf{Enc}_{\mathrm{PK}_1^2}(m_1; r_1^2)$ and sends c_0^2, c_1^2 to $\textsc{Rec}$ on behalf of $\textsc{Sen}$. Finally, $\textsc{SimAttempt}_{1,1}$ sends on behalf of both parties the bits $s_0 = s_1 = 1$.

Consider first the corruption case where the parties are corrupted after z is delivered. Upon corrupting party P, $\textsc{SimAttempt}_{1,1}$ sends $\mathcal{F}_{\mathrm{SC}}$ the message $(\mathsf{corrupt}, sid, P)$ and gets complete control over the functionality $\mathcal{F}_{\mathrm{SC}}$. It also gets to know the message m. We now prove that $\textsc{SimAttempt}_{1,1}$ generates a distribution that is computationally indistinguishable from the distribution generated in the real setting. Furthermore, we show that $\textsc{SimAttempt}_{1,1}$ is able to equivocate the bits $\alpha \oplus \gamma$ and $\beta \oplus \delta$ into any bit (where both bits are equivocated into the same value), which allows $\textsc{SimAttempt}_{1,1}$ to equivocate the message m. We observe that the cases where the parties are corrupted before the sender sends its last message are simple to prove and do not require the equivocation of $\alpha \oplus \gamma$ and $\beta \oplus \delta$ since the simulator learns m before it is required to use it. We show that security in these cases is only based on the oblivious ciphertexts generation (and does not need to rely on erasures). We discuss these cases below.

Recall that the adversary's view is comprised of messages m_0, m_1, two pairs of ciphertexts $c_0^1, c_1^1, c_0^2, c_1^2$ and the bits s_0, s_1, z. We now demonstrate how can $\textsc{SimAttempt}_{1,1}$ explain the state of $\textsc{Sen}$ and $\textsc{Rec}$ with respect to such a view and equivocate the bits $\alpha \oplus \gamma, \beta \oplus \delta$. Consider the case that $\textsc{Sen}$ is corrupted without erasures. This implies that α is fixed and thus cannot be equivocated (this further implies that δ is fixed as well since $\delta = \alpha$). Simulator $\textsc{SimAttempt}_{1,1}$ operates as follows:

1. In order to prove that $\textsc{Rec}$ indeed picked $\delta = \alpha$ simulator $\textsc{SimAttempt}_{1,1}$ presents randomness r_α^1 it used to encrypt m_α within c_α^1 (on behalf of $\textsc{Rec}$). Moreover, in order to claim that $c_{1-\alpha}^1$ was obliviously picked, $\textsc{SimAttempt}_{1,1}$ invokes $r'^1_{1-\alpha} \leftarrow \widetilde{\mathsf{Enc}}^{-1}(c_{1-\alpha}^1)$ and presents $r'^1_{1-\alpha}$.

2. Next, it equivocates β into β' such that $m = z \oplus \beta' \oplus \delta$, conditioned on $\delta = \alpha$. This equivocation relies on erasures in the real execution. Namely, the simulator simply claims that β' is $\textsc{Rec}$'s input to protocol π_{KeySetup} without producing the randomness used by $\textsc{Rec}$ in π_{KeySetup}. As $\textsc{Rec}$'s randomness is erased when it is corrupted with erasures.

3. Finally, in order to prove that $\textsc{Sen}$ indeed picked $\gamma = \beta'$ the simulator presents randomness $r_{\beta'}^2$ it used to encrypt $m_{\beta'}$ within $c_{\beta'}^2$ (on behalf of $\textsc{Sen}$). Moreover, in order to claim that $c_{1-\beta'}^2$ was obliviously picked, it invokes $r'^2_{1-\beta'} \leftarrow \widetilde{\mathsf{Enc}}^{-1}(c_{1-\beta'}^2)$ and presents the randomness $r'^2_{1-\beta'}$ to the adversary.

We now prove indistinguishability of the real and the simulated views. We define the joint view of $\textsc{Sen}$ and $\textsc{Rec}$, generated on respective randomness $r_{\textsc{Sen}}$ and $r_{\textsc{Rec}}$, within a successful attempt in the real setting by: (1) the common view, (2) the private view of SEN and (3) the private view of REC. Formally,

$$\textbf{Attempt}_{1,1} = \Big\{ \left(\mathrm{PK}_0^1, \mathrm{PK}_1^1, \mathrm{PK}_0^2, \mathrm{PK}_1^2, c_0^1, c_1^1, c_0^2, c_1^2, s_0, s_1\right), \left(\alpha, \gamma, \mathrm{SK}_\alpha^1, r_0^2, r_1^2\right), \left(\beta, \delta, \mathrm{SK}_\beta^2, r_0^1, r_1^1\right) \Big\}.$$

We define by $\textbf{SimAttempt}_{1,1}$ the distribution generated by simulator SIM in a successful attempt.

$$\textbf{SimAttempt}_{1,1} = \Big\{ \left(\mathrm{PK}_0^1, \mathrm{PK}_1^1, \mathrm{PK}_0^2, \mathrm{PK}_1^2, c_0^1, c_1^1, c_0^2, c_1^2, s_0, s_1\right), \left(\alpha, \gamma = \beta', \mathrm{SK}_\alpha^1, r_\gamma^2, r_{1-\gamma}'^2\right), \left(\beta', \delta = \alpha, \mathrm{SK}_{\beta'}^2, r_\delta^1, r_{1-\delta}'^1\right) \Big\}.$$

We claim that $\textbf{SimAttempt}_{1,1} \approx_c \textbf{Attempt}_{1,1}$, specifying the differences between the distributions first. $\textsc{SimAttempt}_{1,1}$ does not obliviously sample ciphertexts $c_{1-\delta}^1$ and $c_{1-\gamma}^2$ as required in Protocol 2 but rather encrypts m_0 and m_1 twice. In addition, it invokes algorithm $\widetilde{\mathsf{Enc}}^{-1}$ for generating consisting randomness for $c_{1-\alpha}^1$ and $c_{1-\beta'}^2$. By the oblivious ciphertext generation property these two sets of ciphertexts are computationally indistinguishable even in the presence of the randomness returned by $\widetilde{\mathsf{Enc}}$ and $\widetilde{\mathsf{Enc}}^{-1}$. The second difference is the way the value β' is fixed, where $\textsc{SimAttempt}_{1,1}$ first fixes β and then equivocates it to β'. This is achieved by relying on the erasure of the randomness used for π_{KeySetup}. Specifically, the security of π_{KeySetup} implies that it is infeasible to tell what was the receiver's input without revealing the receiver's random tape. A similar argument can be made if REC is the party that is corrupted without erasures.

We now discuss the remaining two corruption cases: (a) before Step 3 is concluded; (b) between Steps 3 and 5. Note that in case the adversary corrupts a party before exchanging the ciphertexts the simulated view is identical to the real view since the simulation is perfect until this step, while the rest of the simulation is concluded using m that is given upon corruption. Finally, if corruption occurs after Step 3 is concluded but before Step 5 is carried out, then the simulator simulates Steps 2- 4 as $\textsc{SimAttempt}_{1,1}$ would do (i.e. it generates all ciphertexts using Enc, exchanges them on behalf of the parties and sends s_0, s_1). Fixing α and β also fixes γ and δ since $\alpha = \delta$ and $\beta = \gamma$. This requires to exploit the oblivious generation of the ciphertexts in order to generate a view that is consistent with these bits (as shown above). Furthermore, since m was not sent yet the simulator does not need to equivocate α or β. Therefore, the proof does not rely on erasures. Finally, the simulator uses m to simulate the last message of the protocol (if the sender is not yet corrupted).

This concludes our proof. ■

4.3 From Semi-Adaptive to Adaptive Security with Partial Erasures

We show how to transform any semi-adaptive protocol into an adaptively secure protocol with partial erasures. This transformation essentially encrypts all the messages of the semi-adaptive protocol using NCE with partial erasures (similarly to the [GWZ09] transformation from semi-adaptive to fully adaptive that uses fully adaptive NCE; see Theorem 2.10). Recall first that semi-adaptive security assumes that the first corruption takes place statically. Therefore, the corruption case that is not addressed here is when the first party is adaptively corrupted. Security against this corruption case is obtained (with or without erasures) by relying on the security of the NCE with partial erasures. Namely, upon adaptively corrupting the first party, the simulator explains the communication between the parties as if that party was statically corrupted.

Theorem 4.4 *Let Π semi-adaptively realizes the functionality $\mathcal{F}$. Then protocol Π', in which the parties send the messages of Π using NCE with partial erasures adaptively realizes $\mathcal{F}$ with partial erasures.*

Proof: Let $\textsc{Adv}$ be a malicious PPT adversary attacking Protocol Π. We construct a simulator $\textsc{Sim}'$ such that no PPT $\textsc{Env}$ distinguishes the real and the simulated views, i.e., the following computational indistinguishability holds

$$\{\mathbf{IDEAL}_{f,\textsc{Sim}',\textsc{Env}}(n,x_0,x_1,z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \approx_c \{\mathbf{REAL}_{\Pi',\textsc{Adv},\textsc{Env}}(n,x_0,x_1,z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}.$$

Simulator $\textsc{Sim}'$ internally invokes a copy of the real adversary $\textsc{Adv}$ and externally interacts with the ideal functionality f and environment $\textsc{Env}$. We neglect static corruptions (since these are covered by the security of protocol Π) and describe the strategy of $\textsc{Sim}'$ for the following three corruption cases: **(1)** Simulation with no corruptions. **(2)** Simulation when the first adaptive corruption takes place. **(3)** Simulation when the second adaptive corruption takes place. Our proof follows in the $\mathcal{F}_{\text{SC}}$-hybrid model. As in the [GWZ09] proof, we assume that Π is a well-structured protocol. Namely, a protocol execution should have the same number of messages, message lengths and order of communication independent of the inputs or random tape of the participating parties. This technicality is needed for simulating the communication of Π. As pointed out in [GWZ09] almost all known constructed protocols for cryptographic tasks are well-structured and any protocol can be easily converted into a well-structured protocol.

Simulation with no corruptions. In case both parties are honest, first simulator $\textsc{Sim}$ initializes a copy of the semi-adaptive simulator $\textsc{Sim} = (\textsc{Sim}_s, \textsc{Sim}_p)$ and simulates the setup phase of protocol Π. Next, relying on the fact that Π is well-structured, $\textsc{Sim}'$ simulates the communication between the parties by simply forwarding publicly known data. Specifically, for wash round i, the simulator forwards the message $(\mathsf{send}, sid, P_{b_i}, k_i)$ to $\textsc{Adv}$ on behalf of $\mathcal{F}_{\text{SC}}$, where b_i is the identity of the party that sends the message in round i and k_i is its length.

Simulation for the first adaptive corruption. Denote the identity of the first corrupted party by $P_{\mathbf{1}}$ and the other identity by $P_{\mathbf{2}}$. Then $\textsc{Sim}'$ receives from f the corrupted party's input $x_{\mathbf{1}}$ and (possible) output $y_{\mathbf{1}}$. Let $\textsc{Adv}_{\mathbf{1}}$ denote a PPT adversary that corrupts party $P_{\mathbf{1}}$ right after the setup phase, yet uses the real input of $P_{\mathbf{1}}$ in the execution, and let $\textsc{Sim}_{\mathbf{1}}$ denote the simulator for that adversary. We consider two cases here, depend on whether $P_{\mathbf{1}}$ is corrupted with or without erasures.

$P_{\mathbf{1}}$ **is adaptively corrupted without erasures.** $\textsc{Sim}'$ invokes $\textsc{Sim}_{\mathbf{1}}$ until party $P_{\mathbf{1}}$ is corrupted while playing the role of functionality f. Namely, by the input-preserving property $\textsc{Sim}_{\mathbf{1}}$ can only submits the input $x_{\mathbf{1}}$. If it expects to receive an output, $\textsc{Sim}'$ forwards $\textsc{Sim}_{\mathbf{1}}$ the value $y_{\mathbf{1}}$. Once the simulation reaches the point where $P_{\mathbf{1}}$ is corrupted, $\textsc{Sim}'$ takes the internal state of $\textsc{Sim}_{\mathbf{1}}$ and sets it as the internal state of $P_{\mathbf{1}}$.

$P_{\mathbf{1}}$ **is corrupted with erasures.** Whenever $\textsc{Adv}$ corrupts party $P_{\mathbf{1}}$ without erasures, then the difference from the prior simulation strategy is that now $\textsc{Sim}'$ relies on erasures in order to explain the messages sent within Π'.

Simulation for the second adaptive corruption. Upon adaptively corrupting $P_{\mathbf{2}}$, $\textsc{Sim}'$ informs $\textsc{Sim}$ of the second corruption and receives back the internal state of $P_{\mathbf{2}}$. Note that security follows here due to the security of the underlying semi-adaptive protocol Π that enables to generate the internal state of the party that is adaptively corrupted second. This part follows identically to the proof from [GWZ09]. ■

5 Efficient Adaptively Secure Computation with Partial Erasures

In this section we study the efficiency of adaptively secure two-party computation with partial erasures in the malicious setting. Towards defining a generic protocol we study the efficiency of NCE and OT primitives with partial erasures security, and demonstrate that they can be designed with constant overhead. Concretely, we first build NCE with partial erasures for exponentially large plaintext spaces using only *a constant number* of public-key encryption (PKE) operations (Section 5.1). Note that this is significantly better than the overhead of fully adaptive NCE constructions that require $O(1)$ PKE operations per bit. In Section 5.2 we demonstrate the feasibility of string OT, again with constant overhead, which is significantly better than prior work in the fully adaptive setting. These two results are demonstrated in the malicious setting. We conclude this section with the feasibility of generic secure two-party computation with partial erasures and semi-honest security. Our results imply that the [GMW87] protocol achieves $O(|C|)$ time complexity when the oblivious transfer functionality is implemented with our protocol from the previous section, such that C is the boolean circuit that computes the specified functionality. This result is demonstrated in Section 5.3.

5.1 Adaptively Secure String NCE with Partial Erasures

In this section we prove that a slightly modified version of the NCE construction from [HP14] is NCE with partial erasures, which only requires a constant number of public-key operations per polynomial-length message. This construction is built on two IND-CPA secure primitives with an additional equivocation property. (1) NCE for the receiver (NCER) that implies that one can *efficiently* generate a secret key that decrypts a simulated ciphertext into any plaintext. (2) NCE for the sender (NCES) that implies that one can *efficiently* generate randomness for proving that a ciphertext, encrypted under a fake key, encrypts an arbitrary plaintext. We review the formal definitions of these primitives in Appendix B.

The idea of this protocol is to have the receiver create two public/secret key pairs where the first pair is for NCES and the second pair is for NCER. The receiver sends the public-keys and the sender encrypts two shares of its message m, each share with a different key. Upon receiving the ciphertexts the receiver recovers the message by decrypting the ciphertexts. We stress that this idea only works if the simulator knows the identity of the corrupted party prior to the protocol execution, since the simulator must decide in advance whether the keys or the ciphertexts should be explained as valid upon corruption (as we cannot have both generated in a fake mode). Nevertheless, in the one-sided setting we cannot tell which party will be adaptively corrupted prior to the execution. We thus resolve this issue using ℓ-equivocal NCE in order to commit to *the choice* of having fake/valid keys and ciphertexts (so the simulator can postpone its decision regarding having either fake keys or ciphertexts to the point where corruption occurs). The fact that this choice induces a binary equivocation space enables to design a protocol with a constant overhead.

The proof from [HP14] is shown in the one-sided setting where either the sender or the receiver are corrupted, but *not both*. In case both parties are corrupted it seems infeasible to prove security since the parties are required to use fake keys/ciphertexts for equivocation, but at the same time explain the keys and ciphertexts as valid. It turns out that the assumption where one of the parties erases its randomness plays a key role in resolving this technicality. That is, if the randomness used for creating the fake key/ciphertext is erased, then it is computationally hard to distinguish a fake key/ciphertext from a valid one. We are now ready to introduce the protocol in the $\mathcal{F}_{\rm SC}^{\ell}$-hybrid model (for $\ell = 2$). Formally, let $\Pi_{\rm NCES} = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ and $\Pi_{\rm NCER} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^*, \mathsf{Dec}, \mathsf{Equivocate})$ respectively denote NCES and NCER for a plaintext space $\{0,1\}^n$. Then, consider the following protocol for realizing $\mathcal{F}_{\rm SC}$.

Protocol 3 (NCE with partial erasures for exponential message spaces ($\Pi_{\rm PE-NCE-STR}$))

- **Inputs:** *Sender* SEN *is given input message* $m \in \{0,1\}^n$. *Both parties are given security parameter* 1^n.

- **The Protocol:**

  1. **Message from the receiver.** REC *invokes* $\mathsf{Gen}(1^n)$ *of* Π_{NCES} *and* Π_{NCER} *and obtains two public/secret key pairs* $(\mathrm{PK}_S, \mathrm{SK}_S)$ *and* $(\mathrm{PK}_R, \mathrm{SK}_R)$*, respectively.* REC *then forwards* $(\mathsf{ChSetup}, sid, \text{REC})$ *and* $(\mathsf{send}, sid, \text{REC}, \mathrm{PK}_S)$ *to* $\mathcal{F}^{\ell}_{\text{SC}}$*, and the message* PK_R *to* SEN.
  2. **Sending the first share.** *Upon receiving* $(\mathsf{send}, sid, \text{REC}, \mathrm{PK}_S)$ *from* $\mathcal{F}^{\ell}_{\text{SC}}$ *and* PK_R *from* REC*,* SEN *picks* m_S *randomly and encrypts it using* PK_S*, creating ciphertext* c_S*.* SEN *forwards* $(\mathsf{ChSetup}, sid', \text{SEN})$*,* $(\mathsf{send}, sid', \text{SEN}, c_S)$ *to* $\mathcal{F}^{\ell}_{\text{SC}}$*.*
  3. **Receiving the first share.** *Upon receiving* $(\mathsf{send}, sid', \text{SEN}, c_S)$ *from* $\mathcal{F}^{\ell}_{\text{SC}}$*,* REC *decrypts* c_S *and records* m_S*.*
  4. **Sending the second share.** SEN *fixes* m_R *such that* $m = m_S \oplus m_R$ *and encrypts it using* PK_R*, creating ciphertext* c_R*.* SEN *forwards* $(\mathsf{ChSetup}, sid'', \text{SEN})$*,* $(\mathsf{send}, sid'', \text{SEN}, c_S)$ *to* $\mathcal{F}^{\ell}_{\text{SC}}$*.*
  5. **Receiving the second share and output computation.** *Upon receiving* $(\mathsf{send}, sid'', \text{SEN}, c_S)$ *from* $\mathcal{F}^{\ell}_{\text{SC}}$*,* REC *decrypts* c_S *and records* m_S*. It then outputs* $m = m_S \oplus m_R$*.*

Note that the plaintext space of our NCE is equivalent to the plaintext spaces of the NCES/NCER schemes. Therefore, our protocol transmits n-bits messages using a *constant number of PKE operations*, as opposed to fully adaptive NCE that require $O(n)$ such operations.

Theorem 5.1 *Assume the existence of NCER and NCES and* ℓ*-equivocal NCE. Then Protocol 3 adaptively realizes* $\mathcal{F}_{\text{SC}}$ *with partial erasures in the* $\mathcal{F}^{\ell}_{\text{SC}}$*-hybrid model (for* $\ell = 2$*) in the presence of semi-honest adversaries.*

Proof: Let ADV be a semi-honest probabilistic polynomial-time adversary attacking Protocol 3. We construct an adversary SIM such that no PPT distinguisher distinguishes the real and the simulated views. We recall that SIM interacts with the ideal functionality $\mathcal{F}_{\text{SC}}$ such that upon corrupting a party, SIM receives its input and (possibly also) its output. SIM then must produce random coins for this party such that the simulated transcript generated so far is consistent with the values it received. In what follows, we explain the simulation strategies for all corruption cases. As demonstrated below, the corruption cases where at least one of the parties is corrupted before or during Step 4 are easier to simulate since the simulator uses the real message, and thus we ignore the corrupt-with-erase command. The more challenging corruption cases are those where corruption takes place after the conclusion of Step 4. In more details:

SEN is corrupted first at the onset (with or without erasures). Upon corrupting SEN, SIM sends the message $(\mathsf{corrupt}, sid, \text{SEN})$ to $\mathcal{F}_{\text{SC}}$, receiving back the message $(\mathsf{send}, sid, \text{SEN}, |m|)$ and the sender's input m. The adversary ADV now runs on $(1^n, m)$ and its randomness. SIM then picks two public/secret key pairs $(\mathrm{PK}_S, \mathrm{SK}_S)$ and $(\mathrm{PK}_R, \mathrm{SK}_R)$. It then emulates functionality $\mathcal{F}^{\ell}_{\text{SC}}$ and the honest receiver by forwarding ADV the message $(\mathsf{send}, sid, \text{REC}, \mathrm{PK}_S)$ from $\mathcal{F}^{\ell}_{\text{SC}}$ and PK_R from REC. The simulation concludes upon receiving $(\mathsf{send}, sid', \text{SEN}, c_S)$ and $(\mathsf{send}, sid'', \text{SEN}, c_R)$ from ADV. Note that c_S and c_R distribute as in the hybrid execution since the simulator perfectly emulates the adversary's view, and that the real receiver outputs in the protocol $\mathsf{Dec}_{\mathsf{SK}_S}(c_S) \oplus \mathsf{Dec}_{\mathsf{SK}_R}(c_R)$ which equals m due to correctness of the public keys.

REC is corrupted first at the onset (with or without erasures). Upon corrupting REC, SIM sends the message $(\mathsf{corrupt}, sid, \text{SEN})$ to $\mathcal{F}_{\text{SC}}$, receiving back the message $(\mathsf{send}, sid, \text{SEN}, |m|)$ and the receiver's output m. SIM invokes the adversary on 1^n and randomness and receives $(\mathsf{send}, sid, \text{SEN}, \mathrm{PK}_S)$ (which is the message sent to $\mathcal{F}^{\ell}_{\text{SC}}$), and PK_R. Next, SIM completes the execution playing the role of the honest sender on input m. Note that it does not make a difference whether REC generates valid or invalid public-keys since SIM knows m and thus perfectly emulates the receiver's view.

Otherwise, upon receiving $(\mathsf{send}, sid, \text{SEN}, |m|)$ from $\mathcal{F}_{\text{SC}}$, SIM emulates the receiver's message as follows. It creates public/secret key pair $(\text{PK}_R, \text{SK}_R)$ for Π_{NCER}, a valid public/secret key pair $(\text{PK}_S, \text{SK}_S)$ and a fake public-key with a trapdoor $(\text{PK}_S^*, t_{\text{PK}_S^*})$ for Π_{NCES} (using Gen and Gen^*, respectively). SIM emulates the honest receiver by sending PK_R to the sender (recall that the other public-key is sent via $\mathcal{F}_{\text{SC}}^{\ell}$).

SEN is corrupted first between Steps 1 and 2 (with or without erasures). Upon receiving the sender's input m, SIM completes the simulation exactly as in the previous case when SEN was corrupted at the outset of the protocol execution, as no message was sent yet on behalf of the sender.

REC is corrupted first between Steps 1 and 2 (with or without erasures). Upon receiving the receiver's output message m, SIM explains the receiver's internal state which is independent of m. Specifically, it reveals the randomness for generating PK_S, SK_S and PK_R, SK_R, and explains PK_S as the message that was sent to $\mathcal{F}_{\text{SC}}^{\ell}$. SIM then completes the simulation while playing the role of the honest sender with input message m.

Note that in the $\mathcal{F}_{\text{SC}}^{\ell}$-hybrid model the simulation and the hybrid executions are identically distributed since the transcript only includes PK_R. Moreover, we do not need to make use of erasures.

If none of the above corruption cases occurs, SIM emulates the sender's message in Step 2 as follows. It picks a random share m'_S and prepares a pair of ciphertexts (c_S, c_S^*) for Π_{NCES} that respectively encrypt m'_S using PK_S and PK_S^*. (Recall that the sender's message is sent via $\mathcal{F}_{\text{SC}}^{\ell}$ thus it is not part of the transcript seen by the adversary).

SEN is corrupted first between Steps 2 and 4 (with or without erasures). Upon receiving the sender's input m from $\mathcal{F}_{\text{SC}}$, the simulator explains the sender's internal state as follows. It first explains PK_S^* for being the public-key received from $\mathcal{F}_{\text{SC}}^{\ell}$. Furthermore, it explains c_S^* as the ciphertext sent to $\mathcal{F}_{\text{SC}}^{\ell}$. It then plays the role of the honest REC for the rest of the protocol execution as above (when the sender is statically corrupted).

REC is corrupted first between Steps 2 and 4 (with or without erasures). Upon receiving the receiver's output m from $\mathcal{F}_{\text{SC}}$, SIM explains the receiver's internal state as follows. Specifically, it reveals the randomness for generating PK_S, SK_S and PK_R, SK_R, and explains PK_S as the message that was sent to $\mathcal{F}_{\text{SC}}^{\ell}$ and c_S as the message sent via $\mathcal{F}_{\text{SC}}^{\ell}$. SIM then completes the execution while playing the role of the honest sender with input message m.

Note that in the $\mathcal{F}_{\text{SC}}^{\ell}$-hybrid model the simulation and the hybrid executions are identically distributed since the transcript only includes PK_R.

If none of the above corruption occurs, SIM emulates the message of the sender in Step 4 as follows. It picks a random share m'_R and generates a pair of ciphertexts (c_R, c_R^*) for Π_{NCER} such that c_R is a valid encryption of m'_R using the public-key PK_R, and c_R^* is a simulated ciphertext (generated by $(c_R^*, t_{c_R^*}) \leftarrow \mathsf{Enc}^*(\text{PK}_R)$). (Recall that the sender's message is sent via $\mathcal{F}_{\text{SC}}^{\ell}$ thus it is not part of the transcript seen by the adversary).

SEN is corrupted with erasures and REC without erasures between Steps 4 and 5. Upon corrupting SEN and REC in an arbitrary order, SIM obtains the sender's input m. It then explains the sender's and the receiver's internal states as follows. Regarding the sender's internal state, it first explains PK_S as the message received from $\mathcal{F}_{\text{SC}}^{\ell}$. It further explains c_S and c_R^* as the messages sent via $\mathcal{F}_{\text{SC}}^{\ell}$. Finally, SIM sets m''_R such that $m = m'_S \oplus m''_R$ and explains m'_S and m''_R as the shares of m that were encrypted within c_S and c^*_R. Note that SIM does not need to present the randomness used for these ciphertexts since SEN is corrupted with erasures, and that ADV cannot detect that c^*_R is a fake ciphertext without observing its randomness.

Next, SIM explains the receiver's internal state as follows. It explains PK_S as the input to $\mathcal{F}^\ell_{\mathrm{SC}}$ and presents $\mathrm{PK}_S, \mathrm{SK}_S$. It then explains that c_S and c^*_R as the ciphertexts sent by $\mathcal{F}^\ell_{\mathrm{SC}}$. Finally, it reveals a secret key $\mathrm{SK}^*_R \leftarrow \mathsf{Equivocate}(t_{c^*_R}, \mathrm{SK}_R, m''_R)$ so that $m''_R \leftarrow \mathsf{Dec}_{\mathrm{SK}^*_R}(c^*_R)$ and $m''_R \oplus m'_S = m$. That is, it explains $(\mathrm{PK}_R, \mathrm{SK}^*_R)$ as the NCER pair of keys that were generated in the first step.

REC is corrupted with erasures and SEN without erasures between Steps 4 and 5. Upon corrupting SEN and REC in an arbitrary order, SIM obtains the sender's input m. It then explains the sender's and the receiver's internal states as follows. Regarding the sender's internal's state, it first explains PK^*_S message received from $\mathcal{F}^\ell_{\mathrm{SC}}$. It further explains c^*_S and c_R messages sent via $\mathcal{F}^\ell_{\mathrm{SC}}$. It then computes $r'' \leftarrow \mathsf{Equivocate}_{\mathrm{PK}^*_S}(t_{\mathrm{PK}^*_S}, m'_S, r, m''_S)$ for m''_S such that $m = m''_S \oplus m'_R$ and explains r'' as the randomness used for computing the ciphertext c^*_S that encrypts m''_S. The randomness used for computing c_R is revealed unchanged.

SIM next explains the receiver's internal state as follows. It first explains PK^*_S as the input to $\mathcal{F}^\ell_{\mathrm{SC}}$ and presents $\mathrm{PK}_R, \mathrm{SK}_R$. It then explains that c^*_S and c_R as message sent via $\mathcal{F}^\ell_{\mathrm{SC}}$. Finally, it explains m''_S as the message decrypted by the first ciphertext. Note that SIM does not need to reveal the secret key for PK^*_S since REC is corrupted with erasures (and the secret key for PK^*_S will not be used again). Also, ADV cannot detect that PK^*_S is fake without observing the randomness used for generating it.

The security argument follows due to the security of NCER, NCES. Specifically, consider first the view of ADV for the case that SEN is corrupted with erasures. Namely, upon corrupting SEN and REC the adversary sees m'_S, m''_R, $(\mathrm{PK}_S, \mathrm{SK}_S)$, $(\mathrm{PK}_R, \mathrm{SK}_R)$ and c_S, c_R, where the randomness of the ciphertexts has been erased and is not part of the view. Then, the simulated ciphertext c_R is fake, where secret key SK_R is computed using algorithm $\mathsf{Equivocate}$ and m''_R. This is in contrast to the real execution where the equivocation algorithm is not used. Specifically, c_R and SK_R are replaced in the simulation with c^*_R and SK^*_R, respectively. Nevertheless, by the ciphertext indistinguishability property of the NCER it holds that,

$$(\mathrm{PK}_R, \mathrm{SK}_R, c_R, m''_R) \approx_c (\mathrm{PK}_R, \mathrm{SK}^*_R, c^*_R, m''_R).$$

Next, consider the view of ADV for the case that REC is corrupted with erasures. Namely, upon corrupting SEN and REC the adversary obtains m''_S, m'_R, $\mathrm{PK}_S, \mathrm{PK}_R$ and $(c_S, r_S), (c_R, r_R)$, where the secret keys have been erased and are not part of the view. Then the simulated public-key that is computed for NCES is fake, where ciphertext c_S is computed based on the fake public-key whereas randomness r_S is generated using algorithm $\mathsf{Equivocate}$ and m''_S. Specifically, PK_S, c_S and r_S are replaced in the simulation with $\mathrm{PK}^*_S, c^*_S, r^*_S$, respectively. Nevertheless, by the key indistinguishability property of the NCES it holds that,

$$(\mathrm{PK}_S, r_S, c_S, m_S) \approx_c (\mathrm{PK}^*_S, r^*_S, c^*_S, m''_S).$$

Finally, we consider the corruption cases that take place after Step 5 is concluded. The simulation for the case that SEN is corrupted with erasures and REC without erasures is similar to the above description. The other corruption case is also similar to the above, except that the adversary does not see SK_R either since it is erased. The security for these last two corruption cases follows as above.

This concludes our proof. ■

5.2 Adaptively Secure String OT with Partial Erasures

In this section we prove the feasibility of string OT with partial erasures using only a constant number of public-key operations. We recall first that OT with partial erasures can be obtained by encrypting the communication of a semi-adaptive OT using NCE with partial erasures (as formally proven in Section 4.3). Therefore, in order to construct a string OT with partial erasures we need to design semi-adaptive *string* OT and *string* NCE with partial erasures, both with a constant overhead. The later is already demonstrated in Section 5.1. Thus, in this section we focus on designing semi-adaptive *string* OT with constant overhead. Combining the two results we obtain the following theorem.

Theorem 5.2 *There exists a protocol that realizes $\mathcal{F}_{\mathrm{OT}}$ with partial erasures in the presence of malicious adversaries that requires $\mathcal{O}(1)$ public-key operations to transmit a message of length n.*

We recall that [GWZ09] presented a generic construction for semi-adaptive OT based on a coin tossing protocol and an enhanced dual-mode PKE for a *binary plaintext space*. In particular, the dual-mode plaintext space size determines the size of the sender's message space in the OT. [GWZ09] left the feasibility of string semi-adaptive OT with constant overhead open. In what follows, we resolve this problem. Namely, we first present a slightly different security definition for enhanced dual-mode PKE. Next, we build a semi-adaptive OT from our modified enhanced dual-mode PKE (which is similar to the construction of [GWZ09]). We further instantiate our modified enhanced dual-mode PKE with a special type of NCES that requires an extra property. Finally, we build such an NCES based on a construction taken from [HP14] under the hardness of the DCR assumption. Combined together, these results imply semi-adaptive string OT using a constant number of public-key operations for exponential size message domains. We continue with our modified definition of enhanced dual-mode cryptosystem.

5.2.1 A New Enhanced Dual-Mode PKE

Loosely speaking, a dual-mode PKE is a PKE that is initialized with system parameters that can be defined in two modes. For each mode it is possible to generate public and secret keys that are associated with a branch $\sigma \in \{0,1\}$. Similarly, the encryption algorithm generates ciphertexts with respect to a branch $\beta \in \{0,1\}$. Moreover, if the key branch matches the ciphertext branch (that is, $\sigma = \beta$), then the ciphertext can be correctly decrypted. The security of dual-mode PKE relies on the indistinguishability of the two system parameters modes, which are denoted by *messy* and *decryption*. In messy mode the system parameters are generated together with a messy trapdoor, which imply that any public-key (even malformed keys) can be associated with any branch. Moreover, when the key branch does not match the ciphertext branch then the ciphertext becomes statistically independent of the plaintext. On the other hand, in decryption mode the system parameters allow to generate two secret keys that correctly decrypt the two ciphertexts.

In order to construct a semi-adaptive OT based in the [PVW08] OT, [GWZ09] extended the dual-mode notion and define an *enhanced* dual-mode primitive that captures a number of additional requirements. Specifically, it should be possible to equivocate either the receiver's input or the sender's input (depends on which party is statically corrupted). To achieve that, [GWZ09] split the system parameters into two parts; system and temporal. The system part is the same in both modes and is generated prior to the protocol execution. The temporal part defines the mode and is generated during the protocol execution using a coin tossing protocol. The idea is to fix the mode during the simulation depending on which party is statically corrupted.

In this work, we propose a slightly modified definition of enhanced dual-mode encryption. Nevertheless, a semi-adaptive OT in the spirit of [GWZ09] can be constructed from our cryptosystem as well. Our definition is different due to the following reasons. First, we consider a primitive where the system parameters (CRS) are identical in both modes, implying that the temporal part is empty. Moreover, the mode is not determined by the temporal part of the CRS, but by the subkey that is used to encrypt $x_{1-\sigma}$. Namely, a public-key consists of two subkeys where the left subkey is used to compute a left ciphertext and the right subkey is used for the right ciphertext, such that each subkey is generated by a different algorithm. This separation is similar to the partition of the system parameters in [GWZ09] into two parts. In addition, the trapdoor that is associated with the CRS is only useful to distinguish the left and right public-keys in the messy mode. The decryption mode has no trapdoor. Finally, in contrast to the [GWZ09] definition, we do not define a fake ciphertext generation algorithm. More concretely,

Definition 5.3 (Enhanced dual-mode PKE) Enhanced dual-mode PKE *for plaintext space* $\{0,1\}^n$ *consists of a tuple of probabilistic algorithms* $(\mathsf{dSetupGen}, \mathsf{dKeyGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{dFindBranch}, \mathsf{dEquivocate})$ *specified as follows:*

– $\mathsf{dSetupGen}$*, given a security parameter* n*, output* $(\mathbb{G}, \mathrm{CRS}, \tau)$*, where* CRS *is a common reference string,* τ *is the corresponding trapdoor information and* $\mathbb{G}$ *is a group description.*

– $\mathsf{dKeyGen}$*, consists of the following sub-algorithms such that the former is mode-independent and the latter takes the mode as an input:*

- $\mathsf{dKeyGenMI}$*: Given* CRS *and a key type* $\alpha \in \{0,1\}$*, output* $(\mathrm{PK}_\alpha, \mathrm{SK}_\alpha)$.
- $\mathsf{dKeyGenMD}$*: Given* CRS*,* α *and* $\mu = \mathrm{dec}$*, output* $(\mathrm{PK}_{1-\alpha}, \mathrm{SK}_{1-\alpha})$*. Otherwise, given* CRS*,* α *and* $\mu = \mathrm{mes}$*, output* $\mathrm{PK}_{1-\alpha}$.

Output $\mathrm{PK} = (\mathrm{PK}_0, \mathrm{PK}_1)$ *and* $\mathrm{SK} = \mathrm{SK}_\alpha$*. In a decryption mode,* SK_0 *and* SK_1 *decrypt left and right ciphertexts, respectively.*

– dEnc*, given* $\mathrm{CRS}, \mathrm{PK} = (\mathrm{PK}_0, \mathrm{PK}_1)$*, an encryption type* $\beta \in \{0,1\}$ *and a plaintext* m*, output* (c,t)*, where* c *is the encryption of* m *under key* PK_β *and* t *is the random coins used for encryption.*

– dDec*, given* $\mathrm{CRS}, \mathrm{PK}, \mathrm{SK}$ *and a ciphertext* c*, output* m.

– $\mathsf{dFindBranch}$*, given* $\mathrm{CRS}, \tau, \mathrm{PK}$*, output the key type* ρ *of* PK.

– $\mathsf{dEquivocate}$*, given* $\mathrm{CRS}, \tau, \mathrm{PK}, \alpha, c, m', t', m$ *for a public-key* $\mathrm{PK} = (\mathrm{PK}_0, \mathrm{PK}_1)$ *of type* α *in a messy mode, ciphertext and randomness* c, t' *such that* $(c,t') \leftarrow \mathsf{dEnc}(\mathrm{CRS}, \mathrm{PK}, 1-\alpha, m')$ *and a plaintext* m*, output random coins* t *such that the first output of* $\mathsf{dEnc}(\mathrm{CRS}, \mathrm{PK}, 1-\alpha, m; t)$ *is c.*

Definition 5.4 (Secure enhanced dual-mode PKE) *A dual-mode PKE* $\Pi_{\mathrm{DUAL}} = (\mathsf{dSetupGen}, \mathsf{dKeyGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{dFindBranch}, \mathsf{dEquivocate})$ *is* secure *if it satisfies the following properties.*

- **Completeness.** *For every* $\mu \in \{\mathrm{mes}, \mathrm{dec}\}$*,* $(\mathbb{G}, \mathrm{CRS}, \tau) \leftarrow \mathsf{dSetupGen}(1^n)$*,* $m \in \{0,1\}^n$*,* $\alpha \in \{0,1\}$ *and* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{dKeyGen}(\mathrm{CRS}, \mu, \alpha)$*, the decryption on a branch* α *is correct except with negligible probability. Namely, for* $(c,t) \leftarrow \mathsf{dEnc}(\mathrm{CRS}, \mathrm{PK}, \alpha, m)$*,* $m = \mathsf{dDec}(\mathrm{CRS}, \mathrm{PK}, \mathrm{SK}, c)$.
- **Enhanced mode indistinguishability.** *The subkeys generated by* $\mathsf{dKeyGenMD}$ *in a messy mode and in a decryption modes are computationally indistinguishable for* $\alpha \in \{0,1\}$*. Furthermore, they are computationally indistinguishable from a random element in* $\mathbb{G}$*. Formally, for* $\alpha \in \{0,1\}$,

$$\begin{aligned} \{\mathrm{PK}_{1-\alpha}\}_{(\mathrm{PK}_{1-\alpha}, t_{1-\alpha}) \leftarrow \mathsf{dKeyGenMD}(\mathrm{CRS}, \mathrm{mes}, \alpha)} &\approx_c \{\mathrm{PK}_{1-\alpha}\}_{(\mathrm{PK}_{1-\alpha}, \mathrm{SK}_{1-\alpha}) \leftarrow \mathsf{dKeyGenMD}(\mathrm{CRS}, \mathrm{dec}, \alpha)} \\ &\approx_c \{R\}_{R \leftarrow \mathbb{G}}. \end{aligned}$$

- **Messy branch identification and ciphertext equivocation.** *For every* $(\mathbb{G}, \text{CRS}, \tau) \leftarrow \mathsf{dSetupGen}(1^n)$ *and every* $(\text{PK}, \text{SK}) \leftarrow \mathsf{dKeyGen}(\text{CRS}, \text{mes}, \cdot)$*,* $\mathsf{dFindBranch}(\text{CRS}, \tau, \text{PK})$ *outputs a branch value* ρ *such that for every* $m \in \{0,1\}^n$*,* $\mathsf{dEnc}(\text{CRS}, \text{PK}, 1-\rho, \cdot)$ *is simulatable. Namely,*

$$\{c, t\}_{(c,t)\leftarrow \mathsf{dEnc}(\text{CRS},\text{PK},1-\rho,m)} \approx_s \{c, t\}_{(c,t')\leftarrow \mathsf{dEnc}(\text{CRS},\text{PK},1-\rho,m'),t\leftarrow \mathsf{dEquivocate}(\text{CRS},\tau,\text{PK},\rho,c,m',t',m)}.$$

- **Decryption mode key indistinguishability.** *For every* $(\mathbb{G}, \text{CRS}, \tau) \leftarrow \mathsf{dSetupGen}(1^n)$*, a left subkey is statistically indistinguishable from a right subkey in a decryption mode. Namely, for any* $\alpha \in \{0,1\}$*,* $\{\text{PK}_0\}_{(\text{PK}=(\text{PK}_0,\text{PK}_1),\text{SK})\leftarrow \mathsf{dKeyGen}(\text{CRS},\text{dec},\alpha)} v \approx_s \{\text{PK}_1\}_{(\text{PK}=(\text{PK}_0,\text{PK}_1),\text{SK})\leftarrow \mathsf{dKeyGen}(\text{CRS},\text{dec},\alpha)}.$

5.2.2 Semi-Adaptive String OT from Enhanced Dual-Mode PKE

Given an enhanced dual-mode cryptosystem $\Pi_{\text{DUAL}} = (\mathsf{dSetupGen}, \mathsf{dKeyGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{dFindBranch}, \mathsf{dEquivocate})$ defined as above for a plaintext space $\{0,1\}^n$, we construct a semi-adaptive OT following a similar approach to the semi-adaptive construction of [GWZ09]. Namely, the receiver runs $\mathsf{dKeyGenMI}$ locally with σ and CRS, and obtains $(\text{PK}_\sigma, \text{SK}_\sigma)$. The parties then run a coin-tossing protocol in order to mutually generate the output of $\mathsf{dKeyGenMD}$, denoted by $\text{PK}_{1-\sigma}$ (in some unknown mode), where only the receiver learns the outcome. REC then sets $\text{PK} = (\text{PK}_0, \text{PK}_1)$ and $\text{SK} = \text{SK}_\sigma$. The coin tossing protocol ensures that the receiver does not learn both left and right secret keys. In order to prevent a corrupted receiver from cheating, we require that it proves that either PK_0 or PK_1 was generated via the coin-tossing protocol. In Appendix C we explain about the special property we need from the ZK proof used by the receiver (denoted by witness equivocal ZK). Informally, in this type of proofs for compound statements, the simulator knows both witnesses but not which witness will be used by the prover. This notion allows to build weaker, yet meaningful ZK proofs that are secure in the presence of adaptive prover corruption. Formally,

Protocol 4 (Semi-adaptive OT for exponential message spaces ($\Pi_{\text{SA-OT-STR}}$))

- **CRS:** *A group description* $\mathbb{G}$ *and* CRS *that are the output of* $\mathsf{dSetupGen}(1^n)$.
- **Inputs:** *Sender* SEN *is given input messages* x_0, x_1 *and Receiver* REC *is given a bit* σ.
- **The Protocol:**
  1. REC *runs* $\mathsf{dKeyGenMI}$ *locally and obtains* $(\text{PK}_\sigma, \text{SK}_\sigma)$*. The parties also run a coin tossing protocol in order to mutually generate* $\text{PK}_{1-\sigma}$*. Specifically,* REC *selects a random element* r *from* $\mathbb{G}$ *and sends* SEN *a commitment to* r.
  2. SEN *selects a random element* s *from* $\mathbb{G}$ *and sends* s *to* REC.
  3. REC *sets* $\text{PK}_{1-\sigma} = r \cdot s$*,* $\text{PK} = (\text{PK}_0, \text{PK}_1)$ *and* $\text{SK} = \text{SK}_\sigma$ *and sends* PK *to* SEN. REC *proves in ZK that either* PK_0/s *or* PK_1/s *was committed by* REC.
  4. *If* SEN *verifies the proof correctly, it computes* $(c_i, t_i) \leftarrow \mathsf{dEnc}(\text{CRS}, \text{PK}, i, x_i)$ *for all* $i \in \{0,1\}$*, and sends* (c_0, c_1) *to* REC.
  5. REC *decrypts* c_σ *using* $\mathsf{dDec}_{\text{SK}}$ *and outputs the result.*

Theorem 5.5 *Assume the existence of adaptively secure UC commitment schemes, witness equivocal zero-knowledge and that* Π_{DUAL} *meets Definition 5.3. Then, Protocol 4 realizes* $\mathcal{F}_{\text{OT}}$ *with malicious semi-adaptive security in the* $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$*-hybrid model, where the parties compute* $\mathcal{O}(1)$ *public-key operations to transmit a message of length* n.

We note that adaptive UC commitment schemes [DN02] are sufficient for implementing a coin-tossing protocol in the UC setting (and can be realized under the DCR hardness assumption that is used for our concrete instantiation from Section 5.2.4). Intuitively, the security of $\Pi_{\text{SA-OT-STR}}$ when the sender is statically corrupted follows by ensuring that the simulator knows both secret keys in a decryption mode, implying that it can equivocate σ and decrypt the $(1-\sigma)$th ciphertext. On the other hand, when the receiver is statically corrupted, the simulator enforces a messy mode and thus is able to equivocate $x_{1-\sigma}$.

Proof: Let ADV be a malicious probabilistic polynomial-time adversary attacking Protocol 4 by statically corrupting one of the parties and adaptively corrupting the other. We construct a simulator SIM for the ideal functionality $\mathcal{F}_{\rm OT}$ such that no PPT distinguisher distinguishes with a non-negligible probability whether it is interacting with ADV in the real setting or with SIM in the ideal setting. We explain the strategy of the simulation for the following corruption cases: (**1**) SEN is statically corrupted and REC is corrupted after the protocol terminates. (**2**) REC is statically corrupted and SEN is corrupted after the protocol terminates. The rest of the corruption cases follow easily. We note that $\Pi_{\rm SA-OT-STR}$ is statically secure similarly to the security of the [PVW08] protocol, where security is based on the security of the dual-mode PKE. Intuitively, in case SEN is statically corrupted then REC's bit is statistically hidden due to the indistinguishability of the left and right subkeys in a decryption mode. Moreover, when REC is statically corrupted then the privacy of $x_{1-\sigma}$ is guaranteed due to the ciphertext equivocation property of the enhanced dual-mode encryption.

SEN is statically corrupted and REC is corrupted after the protocol terminates. SIM emulates the role of $\mathcal{F}_{\rm CRS}^{\mathcal{D}}$ and generates $(\mathbb{G}, {\rm CRS}, \tau) \leftarrow \mathbf{d}\mathsf{SetupGen}$. It hands the adversary $(\mathbb{G}, {\rm CRS})$ and records τ. Next, SIM invokes $\mathbf{d}\mathsf{KeyGen}$ with CRS, $\mu = {\rm dec}$ and some $\alpha \in \{0,1\}$, and stores $({\rm PK} = ({\rm PK}_0, {\rm PK}_1), {\rm SK}_0, {\rm SK}_1)$. It then commits to an arbitrary share using the UC commitment. Upon receiving ADV's share x, SIM sends SEN the public-keys $({\rm PK}_0, {\rm PK}_1)$. Next, SIM proves that either ${\rm PK}_0$ or ${\rm PK}_1$ were generated via the coin tossing protocol by invoking the simulator for the ZK proof using witnesses ${\rm PK}_0/x$ and ${\rm PK}_1/x$. Finally, upon receiving c_0, c_1 from ADV, SIM decrypts them into (x_0, x_1) using ${\rm SK}_0$ and ${\rm SK}_1$, respectively, and sends $(\mathsf{sender}, sid, x_0, x_1)$ to $\mathcal{F}_{\rm OT}$.

When REC is corrupted after the protocol terminates, SIM receives its input σ and provides to ADV the randomness within the coin tossing protocol that is consistent with ${\rm PK}_{1-\sigma}$ (by providing the randomness of the commitment relative to the share ${\rm PK}_{1-\sigma}/x$ while exploiting the equivocality of the commitment scheme). Next, SIM explains the ZK proof relative to ${\rm PK}_{1-\sigma}$ (by providing the randomness of the proof while exploiting the witness equivocality property). It further explains the secret key SK using ${\rm SK}_\alpha$. This completes the simulation. Note that the simulated and real views are computationally indistinguishable due to the difference between a simulated and a real ZK proof.

REC is statically corrupted and SEN is corrupted after the protocol terminates. SIM emulates the role of $\mathcal{F}_{\rm CRS}^{\mathcal{D}}$ and generates $(\mathbb{G}, {\rm CRS}, \tau) \leftarrow \mathbf{d}\mathsf{SetupGen}$. It hands the adversary $(\mathbb{G}, {\rm CRS})$ and records τ. Upon receiving the receiver's message within the coin tossing protocol, SIM extracts the receiver's share x. It then generates a public-key ${\rm PK}'$ in a messy mode and completes this protocol using the share ${\rm PK}'/x$. Upon completing the coin tossing protocol and receiving public-key ${\rm PK} = ({\rm PK}_0, {\rm PK}_1)$, the simulator invokes $\mathbf{d}\mathsf{FindBranch}$ on PK and extracts σ. SIM then plays the role of the honest verifier in the ZK proof. If the proof is not verified correctly the simulator aborts, sending $(\mathsf{receiver}, sid, \perp)$ to $\mathcal{F}_{\rm OT}$. Otherwise, SIM hands $(\mathsf{receiver}, sid, \sigma)$ to $\mathcal{F}_{\rm OT}$ and receives back x_σ. It then selects an arbitrary value $x'_{1-\sigma}$ and returns ciphertexts (c_0, c_1) encrypting $(x_\sigma, x'_{1-\sigma})$. Let t'_0, t'_1 be the randomness used for computing c_0, c_1, respectively.

When SEN is corrupted after the protocol terminates, SIM receives its input $x_{1-\sigma}$. It then invokes $\mathbf{d}\mathsf{Equivocate}$ with $({\rm CRS}, \tau, {\rm PK}, \sigma, c_{1-\sigma}, x'_{1-\sigma}, t'_{1-\sigma})$ and $x_{1-\sigma}$ and obtains a matching randomness $t_{1-\sigma}$ so that $c_{1-\sigma}$ is a valid encryption of $x_{1-\sigma}$ using randomness $t_{1-\sigma}$. SIM presents $t'_\sigma, t_{1-\sigma}$ to the adversary. Note that the simulated and real views are statistically close.

■

5.2.3 A New Enhanced Dual-Mode PKE from Trapdoor-NCES

We now present our new enhanced dual-mode PKE building on any trapdoor-NCES defined by NCES with the additional ability to distinguish between valid and fake keys given a trapdoor. Informally, NCES, defined by the set of algorithms $(\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$, is a *trapdoor*-NCES if there exist two additional algorithms $\mathsf{SetupGen}$ and $\mathsf{FindKeyType}$ so that $\mathsf{SetupGen}$ generates a global trapdoor τ (along with some parameters) and $\mathsf{FindKeyType}$ efficiently distinguishes a key generated by Gen from a key generated by Gen^* using τ. Furthermore, algorithms Gen^* and $\mathsf{Equivocate}$ are slightly different. Namely, algorithm Gen^* no longer generates a trapdoor and $\mathsf{Equivocate}$ equivocates a ciphertext using the global trapdoor τ (whereas in NCES it uses the trapdoor generated by Gen^*). The security of trapdoor-NCES is defined next.

Definition 5.6 (Trapdoor-NCES) *A secure NCES* $\Pi_{\text{NCES}} = (\mathsf{SetupGen}, \mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate}, \mathsf{FindKeyType})$ *is* trapdoor *NCES if the algorithms are as specified below:*

- $\mathsf{SetupGen}$*, given a security parameter* n*, output* $(\mathbb{G}, \textsc{Params}, \tau)$*, where* $\mathbb{G}$ *is a group description.*
- $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are as specified in Definition A.1. All algorithms take* $\textsc{Params}$ *as an additional input.*
- Gen^**, given* $\textsc{Params}$*, outputs a public-key* PK^*.
- $\mathsf{Equivocate}$*, given* $\textsc{Params}, \tau, \text{PK}^*$*, a tuple* (m', t', c^*) *such that* $c^* \leftarrow \mathsf{Enc}_{\text{PK}^*}(m'; r')$ *and a message* m*, output* r *such that* $c^* = \mathsf{Enc}_{\text{PK}^*}(m; r)$.
- $\mathsf{FindKeyType}$*, given* $\textsc{Params}, \tau$ *and* PK*, output* 1 *if* $\text{PK} \leftarrow \mathsf{Gen}(1^n, \textsc{Params})$*, and* 0 *otherwise.*

Definition 5.7 (Secure Trapdoor-NCES) *Trapdoor-NCES* $\Pi_{\text{NCES}} = (\mathsf{SetupGen}, \mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate}, \mathsf{FindKeyType})$ *is* secure *if*

- **Completeness.** *For every* $(\mathbb{G}, \textsc{Params}, \tau) \leftarrow \mathsf{SetupGen}(1^n)$*,* $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n, \textsc{Params})$*,* $m \in \{0,1\}^n$ *and* $c \leftarrow \mathsf{Enc}(\textsc{Params}, \text{PK}, m)$*,* $m = \mathsf{Dec}(\textsc{Params}, \text{PK}, \text{SK}, c)$.
- **Key indistinguishability.** *The keys generated by* Gen *and* Gen^* *are computationally indistinguishable. Furthermore, they are computationally indistinguishable from a random element in* $\mathbb{G}$*. Formally,*

$$\{\text{PK}\}_{(\text{PK},\text{SK})\leftarrow\mathsf{Gen}(\textsc{Params},1^n)} \approx_c \{\text{PK}^*\}_{\text{PK}^*\leftarrow\mathsf{Gen}^*(\textsc{Params},1^n)} \approx_c \{R\}_{R\leftarrow\mathbb{G}}.$$

- **Key type identification and ciphertext equivocation.** *For every* $(\mathbb{G}, \textsc{Params}, \tau) \leftarrow \mathsf{SetupGen}(1^n)$ *and any* PK *generated by either* Gen *or* Gen^**,* $\mathsf{FindKeyType}(\textsc{Params}, \tau, \text{PK})$ *outputs a key type* ρ*. If* $\rho = 0$*, then for every* $m \in \{0,1\}^n$*,* $\mathsf{Enc}(\textsc{Params}, \text{PK}, \cdot)$ *is simulatable. Namely,*

$$\{c,t\}_{c\leftarrow\mathsf{Enc}(\textsc{Params},\text{PK},m;t)} \approx_s \{c,t\}_{c\leftarrow\mathsf{Enc}(\textsc{Params},\text{PK},m';t'),t\leftarrow\mathsf{Equivocate}(\textsc{Params},\tau,\text{PK},c,m',t',m)}.$$

We next build an enhanced dual-mode PKE based on a trapdoor-NCES.

- **Common reference string ($\mathsf{dSetupGen}$).** Invoke $(\mathbb{G}, \textsc{Params}, \tau) \leftarrow \mathsf{SetupGen}(1^n)$ and define $\text{CRS} = \textsc{Params}$ and the trapdoor for the CRS as τ.
- **Key Generation ($\mathsf{dKeyGen}$).** Given CRS, a key type α and a mode $\mu \in \{\text{mes}, \text{dec}\}$, output (PK, SK) such that $\text{PK} = (\text{PK}_0, \text{PK}_1)$ and $\text{SK} = \text{SK}_\alpha$. Namely, algorithm $\mathsf{dKeyGen}$ is defined by algorithm $\mathsf{dKeyGenMI}$ that computes $(\text{PK}_\alpha, \text{SK}_\alpha) \leftarrow \mathsf{Gen}(1^n, \textsc{Params})$ and algorithm $\mathsf{dKeyGenMD}$ that computes the following.

– If $\mu = \text{dec}$ compute $(\text{PK}_{1-\alpha}, \text{SK}_{1-\alpha}) \leftarrow \mathsf{Gen}(1^n, \text{PARAMS})$.
– If $\mu = \text{mes}$ compute $\text{PK}_{1-\alpha} \leftarrow \mathsf{Gen}^*(1^n, \text{PARAMS})$.

- **Encryption** (dEnc). Given CRS, a public-key $(\text{PK}_0, \text{PK}_1)$, a plaintext m from the message space of underlying trapdoor-NCES and an encryption type $\beta \in \{0,1\}$, pick randomness t, invoke $c \leftarrow \mathsf{Enc}_{\text{PK}_\beta}(m; t)$ and output (c, t).

- **Decryption** (dDec). Given a secret key SK and a ciphertext c, output $m = \mathsf{Dec}_{\text{SK}}(c)$.

- **Messy Branch Identification** ($\mathsf{dFindBranch}$). Given CRS, a messy trapdoor τ and a public-key $\text{PK} = (\text{PK}_0, \text{PK}_1)$, invoke $\mathsf{FindKeyType}$ with τ and PK_1. If $\mathsf{FindKeyType}$ returns 0 implying that PK_1 is a fake NCES key, output 0. Otherwise output 1.

- **Equivocation** ($\mathsf{dEquivocate}$). Given $\text{CRS}, \tau, \text{PK} = (\text{PK}_0, \text{PK}_1), \alpha, c, m', t', m$, invoke $t = \mathsf{Equivocate}(\text{CRS}, \tau, \text{PK}_{1-\alpha}, c, m', t', m)$ such that the first output of $\mathsf{dEnc}(\text{CRS}, \text{PK}, 1-\alpha, m; t)$ is c.

Theorem 5.8 *Assume that Π_{NCES} is a trapdoor-NCES. Then, Π_{DUAL} is a secure enhanced dual-mode PKE.*

Proof:

- **Completeness.** In any given mode and $\alpha \in \{0,1\}$, PK_α is a valid key relative to the underlying trapdoor-NCES. Furthermore, the secret key SK corresponding to PK is SK_α which is the secret key of PK_α. If the encryption type β matches the key type α, a ciphertext that is encrypted under PK_α will be correctly decrypted using the decryption algorithm of trapdoor-NCES.

- **Enhanced mode indistinguishability.** We claim that the subkey generated by $\mathsf{dKeyGenMD}$ in a messy mode is computationally indistinguishable from the subkey generated in a decryption mode. This follows from the security of trapdoor-NCES that ensures that the keys generated by Gen and Gen^* are computationally indistinguishable from an random element from $\mathbb{G}$.

- **Messy branch identification and ciphertext equivocation.** Messy branch identification follows from the trapdoor security of NCES. Namely, in a messy mode, $\text{PK}_{1-\alpha}$ is a fake key relative to the underlying trapdoor-NCES. Thus, if the key type α does not match β then ciphertext equivocation with respect to $\text{PK}_{1-\alpha}$ is ensured via the equivocality of the underlying trapdoor-NCES.

- **Decryption Mode Key Indistinguishability.** In a decryption mode, PK contains two valid keys relative to the underlying NCES for any value of α. Thus, the left subkey is statistically indistinguishable from right subkey for any public-key in a decryption mode.

5.2.4 Trapdoor-NCES under the DCR Assumption

We briefly overview the NCES from [HP14], proving that this construction is a trapdoor-NCES. Instantiating our semi-adaptive OT from Section 5.2.2 using an enhanced dual-mode PKE based on our DCR trapdoor-NCES from below, implies that the receiver's public-key is defined by the elements $\text{PK} = ((g, h_0, \bar{g}_0, \bar{h}_0), (g, h_1, \bar{g}_1, \bar{h}_1))$, where $(g, h_{1-\sigma}, \bar{g}_{1-\sigma}, \bar{h}_{1-\sigma})$ is the public-key part that is generated using a coin tossing in order to encrypt $x_{1-\sigma}$. We note that it is sufficient to mutually generate $h_{1-\sigma}$ in order to prevent the receiver from learning the secret key. Nevertheless, if the receiver locally generates $\bar{h}_{1-\sigma}$, then the simulator would not be able to enforce a messy mode when the receiver is statically corrupted. We thus use the coin tossing protocol to generate all the elements from $\text{PK}_{1-\alpha}$ except for g, by picking random elements from $\mathbb{Z}^*_{N^2}$. Namely, for each mutually generated element from $\text{PK}_{1-\alpha}$, the receiver commits first to a random element from $\mathbb{Z}^*_{N^2}$. Next, the sender sends a random element from this group, say x. The receiver then multiplies the two elements. Note that the receiver's commitment can be implemented using the adaptively secure commitment schemes from [DN02] based on the DCR assumption (in an equivocal mode), and the ZK proof boils down to a witness equivocal proof for a compound statement of an Nth root. That is, the receiver proves that it committed to either h_0/x or h_1/x. (The receiver further proves similar statements with respect to $(\bar{g}_0, \bar{g}_1)$ and $(\bar{h}_0, \bar{h}_1)$.) See Appendix C for more details about the proof. Formally,

- $\mathsf{SetupGen}$, given a security parameter n, generate an RSA modulus $N = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ and primes p, q, p', q'. Pick $g' \leftarrow \mathbb{Z}^*_{N^2}$ and set $g = g'^{2N} \bmod N^2$. Set $\textsc{Params} = (N, g)$, $\tau = \phi(N)$ and $\mathbb{G} = (\mathbb{Z}^*_{N^2})^4$.
- Gen, given a security parameter n and $\textsc{Params}$ (N, g), pick $s \leftarrow \mathbb{Z}_{N^2/4}$ and set $h = g^s \bmod N^2$. Choose a random $r \leftarrow \mathbb{Z}_{N/4}$ and compute $(\bar{g} = g^r \bmod N^2, \bar{h} = [(1 + N) \cdot h^r] \bmod N^2)$. Output $\mathrm{PK} = (g, h, \bar{g}, \bar{h})$ and $\mathrm{SK} = s$.
- Gen^*, given a security parameter n and $\textsc{Params}$, set $h = [(1 + N) \cdot g^s] \bmod N^2$. Choose a random $r \leftarrow \mathbb{Z}_{N/4}$ and compute $(\bar{g} = g^r \bmod N^2, \bar{h} = h^r \bmod N^2)$. Output $\mathrm{PK}^* = (g, h, \bar{g}, \bar{h})$.
- Enc, given a public-key $\mathrm{PK} = (N, g, h, \bar{g}, \bar{h})$ and a message $m \in \mathbb{Z}_N$, choose a random $t \leftarrow \mathbb{Z}_{N/4}$ and output the ciphertext $\mathsf{Enc}(m; t) = \left((\bar{g}^m g^t) \bmod N^2, (\bar{h}^m h^t) \bmod N^2\right)$.
- Dec, given a public-key $\mathrm{PK} = (g, h, \bar{g}, \bar{h})$, a secret key $\mathrm{SK} = s$ and a ciphertext $c = (c_1, c_2)$, compute $\hat{m}$ as follows, and output $m \in \mathbb{Z}_N$ such that $\hat{m} = 1 + mN$.

$$\hat{m} = (c_2/c_1^s)^{N+1} = [(1 + N)^m]^{N+1} = (1 + N)^m.$$

- $\mathsf{Equivocate}$, given $\phi(N)$, a fake key $\mathrm{PK}^* = (g, h, \bar{g}, \bar{h})$, a tuple (m', t', c^*) such that $c^* \leftarrow \mathsf{Enc}_{\mathrm{PK}^*}(m'; t')$ and a message m, extract r from PK^* using $\phi(N)$ and output $t = (rm' + t' - rm) \bmod \phi(N)/4$. It is simple to verify that

$$\begin{aligned}\mathsf{Enc}_{\mathrm{PK}^*}(m; t) &= \left((\bar{g}^m g^t), \bar{h}^m h^t)\right) = \left((g^{rm} g^{rm' + t' - rm}), (h^{rm} h^{rm' + t' - rm})\right) \\ &= \left((\bar{g}^{m'} g^{t'}), \bar{h}^{m'} h^{t'})\right) = c^*.\end{aligned}$$

- $\mathsf{FindKeyType}$, given $\phi(N)$ and a public-key PK, check if the second element in PK is an Nth power. If yes output 1. Otherwise output 0.

■

5.3 Secure Two-Party Computation with Partial Erasures

In this section we show a general result, demonstrating that efficient secure two-party computation in the presence of semi-honest adversaries can be achieved using our oblivious transfer protocol from Section 5.2. Concretely, the efficiency of this protocol is as in the static setting and implies $O(|C|)$ time complexity, for C the boolean circuit that computes the specified functionality. That is, we consider the [GMW87] protocol and plug-in our efficient OT protocol with partial erasures. Relying on the UC composition theorem from [Can01] we conclude that the combined protocol is adaptively secure with partial erasures. We note that the theorem from [Can01] is stated in the adaptive setting and holds even in the presence of erasures, assuming that the same party in all sub-protocols is always corrupted with erasures.

Next, denote the [GMW87] protocol when combined with the oblivious transfer protocol from Section 5.2 by $\Pi^{\rm OT}$. We claim that $\Pi^{\rm OT}$ adaptively realizes any efficient two-party well-formed functionality f in the presence of semi-honest adversaries. More formally,

Theorem 5.9 *Let f be a well-formed two-party functionality. Then, Protocol $\Pi^{\rm OT}$ adaptively realizes f with partial erasures in the presence of semi-honest adversaries.*

Intuitively, the proof follows directly from the composition theorem and is shown in two steps. First, that the [GMW87] protocol is information theoretic secure in the $\mathcal{F}_{\rm OT}$-hybrid model. Next, when replacing the ideal calls of $\mathcal{F}_{\rm OT}$ with a protocol that is adaptively secure with partial erasures, the security of $\Pi^{\rm OT}$ is implied by the composition theorem. The overall time complexity of $\Pi^{\rm OT}$ is reduced to the time complexity of the OT protocol. Now, since each such invocation requires a constant overhead, the total overhead grows linearly with the size of C. This overhead matches the [GMW87] overhead in the static setting.

6 Conclusions

We introduce the notion of adaptive security with partial erasures and show that it has the potential to yield simpler and more efficient protocols. We believe that it is a natural security guarantee that provides a good tradeoff between paying the price of achieving adaptive security without any erasures and trusting that all honest parties erase securely. Our work leaves a number of interesting questions open. Most notably, the question whether there exists a constant round generic two-party protocol with partial erasures.

References

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.

[Bea97] Donald Beaver. Plug and play encryption. In *CRYPTO*, pages 75–89, 1997.

[BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *EUROCRYPT*, pages 307–323, 1992.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CDD+04] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *J. Cryptology*, 17(3):153–207, 2004.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *ASIACRYPT*, pages 287–302, 2009.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *TCC*, pages 150–168, 2005.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, 2002.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO*, pages 581–596, 2002.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO*, pages 505–523, 2009.

[HP14] Carmit Hazay and Arpita Patra. One-sided adaptively secure two-party computation. In *TCC*, pages 368–393, 2014.

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

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

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *EUROCRYPT*, pages 221–242, 2000.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.

[Lin09] Yehuda Lindell. Adaptively secure two-party computation with erasures. In *CT-RSA*, pages 117–132, 2009.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, pages 681–700, 2012.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

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

A Preliminaries - Appendix

A.1 Public-Key Encryption Schemes

We specify the definitions of public-key encryption, IND-CPA and simulatable public-key encryption.

Definition A.1 (PKE) *A* public-key encryption scheme *consists of a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *specified as follows:*

- Gen*, given a security parameter n (in unary), outputs keys $(\mathrm{PK}, \mathrm{SK})$, where PK is a public-key and SK is a secret key. We denote this by $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.*

- Enc*, given the public-key PK and a plaintext message m, outputs a ciphertext c encrypting m. We denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)$; and when emphasizing the randomness r used for encryption, we denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r)$.*

- Dec*, given the public-key PK, secret key SK and a ciphertext c, outputs a plaintext message m s.t. there exists randomness r for which $c = \mathsf{Enc}_{\mathrm{PK}}(m; r)$ (or $\perp$ if no such message exists). We denote this by $m \leftarrow \mathsf{Dec}_{\mathrm{PK},\mathrm{SK}}(c)$.*

For a public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-uniform adversary $\textsc{Adv} = (\textsc{Adv}_1, \textsc{Adv}_2)$, we consider the following *indistinguishability game*:

$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.
$(m_0, m_1, history) \leftarrow \textsc{Adv}_1(\mathrm{PK})$, s.t. $|m_0| = |m_1|$.
$c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m_b)$, where $b \in_R \{0, 1\}$.
$b' \leftarrow \textsc{Adv}_2(c, history)$.
$\textsc{Adv}$ wins if $b' = b$.

Denote by $\textsc{Adv}_{\Pi,\textsc{Adv}}(n)$ the probability that $\textsc{Adv}$ wins the IND-CPA game.

Definition A.2 (IND-CPA) *A public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is* IND-CPA secure*, if for every non-uniform adversary $\textsc{Adv} = (\textsc{Adv}_1, \textsc{Adv}_2)$ there exists a negligible function* negl *such that* $\textsc{Adv}_{\Pi,\textsc{Adv}}(n) \leq \frac{1}{2} + \mathsf{negl}(n)$.

A.2 Hardness Assumptions

Definition A.3 (DCR [Pai99]) *We say that* the Decisional Composite Residuosity (DCR) problem is hard *if for all polynomial-sized circuits $\mathcal{C} = \{\mathcal{C}_n\}$ there exists a negligible function* negl *such that*

$$\Big| \Pr\left[\mathcal{C}_n(N, z) = 1 |\ z = y^N \bmod N^2\right] - \Pr\left[\mathcal{C}_n(N, z) = 1 |\ z = (1+N)^r \cdot y^N \bmod N^2\right] \Big| \leq \mathsf{negl}(n),$$

where N is a random n-bit RSA composite, r is chosen at random in $\mathbb{Z}_N$, and the probabilities are taken over the choices of N, y and r.

Definition A.4 (QR) *We say that* the Quadratic Residuosity (QR) problem is hard relative to $\mathbb{G}$ *if for all polynomial-sized circuits $\mathcal{C} = \{\mathcal{C}_n\}$ there exists a negligible function* negl *such that*

$$\Big| \Pr\left[\mathcal{C}_n(N, z) = 1 |\ z \leftarrow \mathbb{QR}_N\right] - \Pr\left[\mathcal{C}_n(N, z) = 1 |\ z \leftarrow \mathbb{J}_N \setminus \mathbb{QR}_N\right] \Big| \leq \mathsf{negl}(n),$$

where $N \leftarrow \mathbb{G}(1^n)$, N is a random n-bit RSA composite, $\mathbb{J}_N$ denote the group of Jacobi symbol (+1) elements of $\mathbb{Z}_N^*$, $\mathbb{QR}_N = \{x^2 \ : \ x \in \mathbb{Z}_N^*\}$ denote $\mathbb{J}_N$'s subgroup of quadratic residues and the probabilities are taken over the choices of N, z.

A.3 Zero-knowledge Proofs and Proofs of Knowledge

Our protocols employ zero-knowledge proofs (of knowledge) for assuring correct behavior. We formally define zero-knowledge and knowledge extraction as stated in [Gol01]. We then conclude with a definition of a Σ-protocol which constitutes a zero-knowledge proof of a special type.

Definition A.5 (Interactive proof system) *A pair of* PPT *interactive machines* (P, V) *is called an* interactive proof system for a language L *if there exists a negligible function* negl *such that the following two conditions hold:*

1. COMPLETENESS: *For every $x \in L$,*

$$\Pr[\langle \mathsf{P}, \mathsf{V} \rangle(x) = 1] \geq 1 - \mathsf{negl}(|x|).$$

2. SOUNDNESS: *For every $x \notin L$ and every interactive* PPT *machine B,*

$$\Pr[\langle B, \mathsf{V} \rangle(x) = 1] \leq \mathsf{negl}(|x|).$$

Definition A.6 (Zero-knowledge) *Let* (P, V) *be an interactive proof system for some language L. We say that* (P, V) *is* computational zero-knowledge *if for every* PPT *interactive machine* V^* *there exists a* PPT *algorithm M^* such that*

$$\{\langle \mathsf{P}, \mathsf{V}^* \rangle(x)\}_{x \in L} \stackrel{\text{c}}{\approx} \{\langle M^* \rangle(x)\}_{x \in L}$$

where the left term denote the output of V^* *after it interacts with* P *on common input x whereas, the right term denote the output of M^* on x.*

Definition A.7 (Knowledge extraction) *Let R be a binary relation and $\kappa \rightarrow [0, 1]$. We say that an interactive function* V *is* a knowledge verifier for the relation R with knowledge error κ *if the following two conditions holds:*

NON-TRIVIALITY: *There exists an interactive machine* P *such that for every $(x, y) \in \mathcal{R}$, (implying that $x \in L_{\mathcal{R}}$), all possible interactions of* V *with* P *on common input x and auxiliary input y are accepting.*

VALIDITY (WITH ERROR κ): *There exists a polynomial $q(\cdot)$ and a probabilistic oracle machine K such that for every interactive function* P*, every $x \in L_{\mathcal{R}}$, and every machine K satisfies the following condition:*

> *Denote by $p(x, y, r)$ the probability that the interactive machine* V *accepts, on input x, when interacting with the prover specified by* $\mathsf{P}_{x,y,r}$ *that uses randomness r (where the probability is taken over the coins of* V*). If $p(x, y, r) > \kappa(|x|)$, then, on input x and with access to oracle* $\mathsf{P}_{x,y,r}$*, machine K outputs a solution $s \in \mathcal{R}(x)$ within an expected number of steps bounded by*
>
> $$\frac{q(|x|)}{p(x, y, r) - \kappa(|x|)}$$
>
> *The oracle machine K is called a universal knowledge extractor.*

B A Review of the Different NCE Security Notions

B.1 NCE for the Receiver

NCE for the receiver is a secure PKE with an additional property that enables generating a secret key that decrypts a fake ciphertext into any plaintext. Specifically, the scheme operates in two modes. The real mode enables to encrypt and decrypt as in the standard definition of PKE. Whereas the fake mode enables to generate fake ciphertexts that are computationally indistinguishable from real ciphertexts, such that using a special trapdoor one can produce a secret key that decrypts a fake ciphertext into any plaintext. More formally, an NCE for the receiver encryption scheme with message space $m \in \{0,1\}^n$ consists of a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^*, \mathsf{Dec}, \mathsf{Equivocate})$ specified as follows:

- $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are as specified in Definition A.1.
- Enc^*, given the public-key PK output a ciphertext c^* and a trapdoor t_{c^*}.
- $\mathsf{Equivocate}$, given the secret key SK, trapdoor t_{c^*} and a plaintext $m \in \{0,1\}^n$, output SK^* such that $m = \mathsf{Dec}_{\mathrm{SK}^*}(c^*)$.

Definition B.1 (NCER) *NCE for the receiver is a tuple of algorithms defined above that satisfy the following properties:*

1. $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *imply an IND-CPA secure encryption scheme as in Definition A.2.*
2. **Ciphertext indistinguishability.** *For any $m \in \{0,1\}^n$ the following distributions are computationally indistinguishable:*

$$\{(\mathrm{PK}, \mathrm{SK}, c, m) \mid (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n), c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)\}$$

and

$$\{(\mathrm{PK}, \mathrm{SK}^*, c^*, m) \mid (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n), (c^*, t_{c^*}) \leftarrow \mathsf{Enc}^*(\mathrm{PK}), \mathrm{SK}^* \leftarrow \mathsf{Equivocate}(\mathrm{SK}, c^*, t_{c^*}, m)\}.$$

NCER can be realized under the DDH assumption [JL00, CHK05] for polynomial-size message spaces and under the DCR assumption for exponential-size message spaces [CHK05].

B.2 NCE for the Sender

NCE for the sender is a secure PKE with an additional property that enables generating a fake public-key, such that any ciphertext encrypted under this key can be viewed as the encryption of any message together with the matched randomness. Specifically, the scheme operates in two modes. The real mode enables to encrypt and decrypt as in standard definition of PKE. Whereas the fake mode enables to generate fake public-keys and an additional trapdoor, such that the two modes keys are computationally indistinguishable. In addition, given this trapdoor and a ciphertext generated using a fake public-key, one can produce randomness that is consistent with any plaintext. More formally, an NCE for the sender encryption scheme with message space $m \in \{0,1\}^n$ consists of a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ specified as follows:

- $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are as specified in Definition A.1.
- Gen^* generates public-key PK^* and a trapdoor t_{PK^*}.

- $\mathsf{Equivocate}$, given a ciphertext c^* computed using PK^*, a trapdoor t_{PK^*} and a plaintext $m \in \{0,1\}^n$, output r such that $c^* = \mathsf{Enc}(m;r)$.

Definition B.2 (NCES) *An NCE for the sender is a tuple of algorithms defined above that satisfy the following properties:*

1. $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *imply an IND-CPA secure encryption scheme as in Definition A.2.*
2. **Public key indistinguishability.** *For any $m \in \{0,1\}^n$ the following distributions are computationally indistinguishable:*

$$\{(\mathrm{PK}, r, m, c) \mid (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n), c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r)\}$$

and

$$\{(\mathrm{PK}^*, r^*, m, c^*) \mid (\mathrm{PK}^*, t_{\mathrm{PK}^*}) \leftarrow \mathsf{Gen}^*(1^n), c^* \leftarrow \mathsf{Enc}_{\mathrm{PK}^*}(m'; r'), r^* \leftarrow \mathsf{Equivocate}(c^*, t_{\mathrm{PK}^*}, m)\}.$$

NCES can be realized under the DDH assumption [BHY09] for polynomial-size message spaces and under the DCR assumption for exponential-size message spaces [HP14].

B.3 ℓ-Equivocal Non-Committing Encryption [GWZ09]

The idea of ℓ-Equivocal NCE is to exploit the fact that it is often unnecessary for the simulator to explain a fake ciphertext with respect to *any* potential plaintext. Rather, in many scenarios the potential number of plaintexts is a smaller set of size ℓ (where ℓ might be as small as 2). Specifically, two parameters are considered here: a plaintext of bit length l and an equivocality parameter ℓ which denotes the potential number of plaintexts (namely, the non-committed domain size). The parameter ℓ further dominates the overhead of the ℓ-Equivocal NCE construction from [GWZ09], and thus improves over fully NCE whenever ℓ is very small but the plaintext length is still large. In this paper, we use this primitive to encrypt small domains (i.e., binary) plaintexts of length n with constant overhead. Somewhat NCE is realized in [GWZ09] under the same hardness assumptions that imply NCE.

C Witness Equivocal ZK Proofs for Compound Statements

In [HP14], the authors introduced a new technique for zero-knowledge (proofs of knowledge) for compound statements, where the statement is comprised of sub-statements for which the prover only knows a subset of the witnesses. The security of these proofs relies on the fact that the simulator knows the witnesses for *all* sub-statements but not which subset is given to the real prover. Yet, the simulator is still able to convince an adaptively corrupted prover that it does *not know* a different subset of witnesses than what should be known to the real prover. This notion, denoted by *witness equivocal*, is weaker than the typical adaptive security notion (that requires simulation without the knowledge of any witness), but is still useful here.

In compound statements for Σ-protocols the prover separates the challenge c that is given by the verifier into two values; c_1 and c_2 such that $c = c_1 \oplus c_2$. Assume w.l.o.g. that the prover does not have a witness for the first statement, then it always chooses c_1 in which it knows how to complete the proof (similarly to what the simulator does), and uses its witness for the other statement to complete the second proof on a given challenge c_2. Note that the verifier cannot distinguish whether the prover knows the first or the second witness (or both); see [CDS94] for more details. In our simulation, the simulator uses both witnesses to answer the challenge. Then, when the prover is adaptively corrupted, the simulator gets the real witness from the trusted party and hands it to the adversary. It further claims that the transcript of the other challenge is generated obliviously of the other witness (as should have been computed by a real prover). The concrete proof that we consider in Section 5.2.2 is a proof of an Nth root in group $\mathbb{Z}_{N^2}^*$, where N is an RSA composite that is defined by the following relation,

$$\mathcal{R}_{\text{NR}} = \left\{((u, N), v) |\ u = v^N \bmod N^2\right\}.$$

This proof is formally given in [HP14].