J Cryptol (2017) 30:321–371
DOI: 10.1007/s00145-015-9222-4

Journal of
CRYPTOLOGY

Efficient One-Sided Adaptively Secure Computation*

Carmit Hazay
Department of Computer Engineering, Bar-Ilan University, Ramat Gan, Israel
carmit.hazay@biu.ac.il

Arpita Patra
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
arpitapatra10@gmail.com

Communicated by Jonathan Katz.

Received 14 April 2014 / Revised 11 September 2015
Online publication 30 December 2015

Abstract. Adaptive security is a strong security notion that captures additional security threats that are not addressed by static corruptions. For instance, it captures real-world scenarios where "hackers" actively break into computers, possibly while they are executing secure protocols. Studying this setting is interesting from both theoretical and practical points of view. A primary building block in designing adaptively secure protocols is a non-committing encryption (NCE) that implements secure communication channels in the presence of adaptive corruptions. Current constructions require a number of public key operations that grow linearly with the length of the message. Furthermore, general two-party protocols require a number of NCE calls that dependent both on the circuit size and on the security parameter. In this paper, we study the two-party setting in which at most one of the parties is adaptively corrupted, and demonstrate the feasibility of (**1**) NCE with constant number of public key operations for large message spaces, (**2**) oblivious transfer with constant number of public key operations for large sender's input spaces, and (**3**) constant round secure computation protocols with an overall number of public key operations that is linear in the circuit size. Our study demonstrates that such primitives indeed exist in the presence of single corruptions without erasures, while this is not known for fully adaptive security under standard assumptions (where both parties may get corrupted). Our results are shown in the UC setting with a CRS setup.

Keywords. Secure two-party computation, Adaptive security, Non-committing encryption, Oblivious transfer.

* A preliminary version of this paper appeared in [34]

1. Introduction

1.1. *Background*

1.1.1. *Secure Two-Party Computation*

In the setting of secure two-party computation, two parties with private inputs wish to jointly compute some function of their inputs while preserving certain security properties like privacy, correctness and more. In this setting, security is formalized by comparing a protocol execution to a protocol executed in an ideal setting where the parties send inputs to a trusted party that performs the computation and returns its result (also known by simulation-based security). Starting with the works of [27,51], it is by now well known that any polynomial-time function can be compiled into a secure function evaluation protocol with practical complexity; see [1,25,42,45] for a few recent works. The security proofs of these constructions assume that parties are statically corrupted. Meaning, corruptions take place at the outset of the protocol execution and the identities of the corrupted parties are fixed throughout the computation. A stronger notion is *adaptive security* where corruptions take place *at any point* during the course of the protocol execution. That is, upon corruption the adversary learns the internal state of the corrupted party which includes its input, randomness and the incoming messages. This notion is much stronger than static security since the adversary may choose at any point which party to corrupt, even after the protocol is completed! It therefore models real-world threats more accurately.

Typically, when dealing with adaptive corruptions we distinguish between corruptions with erasures and without erasures. In the former case, honest parties are trusted to erase data if they are instructed to do so by the protocol, whereas in the latter case no such assumption is made. This assumption is often problematic since it relies on the willingness of the honest parties to carry out this instruction, even though they know that no other party will be able to verify whether they have carried out the instruction or not. In settings where the parties are distrustful it may not be a good idea to base security on such an assumption. In addition, it is generally unrealistic to trust parties to fully erase data since this may depend on the operating system. Nevertheless, assuming that there are no erasures come with a price since the complexity of adaptively secure protocols without erasures is much higher than the analogue complexity of protocols that rely on erasures. In this paper, we do *not* rely on erasures.

1.1.2. *Adaptive Security*

It is known by now that security against adaptive attacks capture important real-world concerns that are not addressed by static corruptions. For instance, such attacks capture scenarios where "hackers" actively break into computers, possibly while they are running secure protocols, or when the adversary learns from the communication which parties are worth to corrupt more than others. This later issue can be demonstrated by the following example. Consider a protocol where some party (denoted by the dealer) shares a secret among a public set of $\sqrt{n}$ parties, picked at random from a larger set of n parties. This scheme is insecure in the adaptive model if the adversary corrupts $\sqrt{n}$ parties since it can always corrupt the particular set of parties that share the secret. On the other hand,

in the static setting the adversary can only corrupt the exact same set of parties that share the secret with a negligible probability in n.

Further difficulties arise when proving security. For instance, consider the following protocol for transferring a message: A receiver picks a public key and sends it to a sender that uses it to encrypt its message. Then, security in the static model is simple and relies on the semantic security of the underlying encryption scheme. Nevertheless, this protocol is insecure in the adaptive model since standard semantically secure encryption binds the receiver to a single message (meaning, given the public key, a ciphertext can only be decrypted into a single plaintext). Thus, upon corrupting the receiver after simulating its communication, it would not be possible to "explain" the simulated ciphertext with respect to the real message. This implies that adaptive security is much harder to achieve.

In the two-party setting, there may be scenarios where the system is comprised of only two parties that do not communicate with any external device. In this case, it makes more sense to study the system's security in the presence of single corruptions (namely, when at most one party is under attack). This is because we have no security guarantee once both parties are under attack. In this paper, we study secure two-party computation with single adaptive corruptions in the non-erasure model. To distinguish this notion from fully adaptive security, where both parties may get corrupted, we denote it by *one-sided* adaptive security. Our goal in this work is to make progress in the study of the asymptotic efficiency of secure two-party protocols with one-sided security.

Our measure of efficiency is associated with the number of public key encryption (PKE) operations, where our underlying primitives are parameterized by a PKE scheme for which we count the number of key generation/encryption/decryption operations. These operations are captured by the number of exponentiations in several important groups, i.e. groups where the decisional Diffie–Hellman (DDH) assumption is hard, as well as composite-order groups where the decisional composite residuosity (DCR) and quadratic residuosity (QR) hardness assumptions are believed to hold. Finally, our proofs are given with universal composable (UC) security proofs [6] in the common reference string (CRS) setting. We note that the reductions of our non-committing encryption (NCE) and oblivious transfer (OT) with one-sided security are tight, whereas the reductions of our general two-party protocols are tighter than proofs in prior works; see more details below. All our theorems *are not* known to hold in the fully adaptive setting.

1.2. *Our Results*

1.2.1. *New One-Sided NCE Constructions*

A non-committing encryption (NCE) scheme [12] implements a secure channel between two parties in the presence of adaptive corruptions and is an important building block in designing adaptively secure protocols. One-sided NCE (resp. NCE) implies a secure channel where a single (resp. both) parties are adaptively corrupted. Theoretically speaking, one-sided NCE was demonstrated in [22] under a strictly weaker hardness assumption than the assumption needed for NCE [9,22], where the later assumption is simulatable PKE scheme. Nevertheless, all known schemes, in both security settings, require a number of PKE operations that grow linearly with the bit representation of the transmit-

ted message. It was unknown whether this bound is tight for one-sided NCE or whether the overhead can be made closer to the overhead of PKE.

We suggest a new approach for designing NCE with security against one-sided adaptive attacks. Our protocols are built based on two public key building blocks that are non-committing with respect to a single party. We denote these primitives by NCE for the sender and NCE for the receiver. *Non-committing for the receiver* (NCER) implies that one can efficiently generate a secret key that decrypts a fake ciphertext into any plaintext, whereas *non-committing for the sender* (NCES) implies that one can efficiently generate randomness for any plaintext for proving that a ciphertext, encrypted under a fake key, encrypts this plaintext. A core building block in our one-sided construction is (a variant) of the following protocol, in which the receiver generates two sets of public/secret keys, one pair of keys for each public key system, and sends these public keys to the sender. Next, the sender partitions its message into two shares and encrypts the distinct shares under the distinct public keys. Finally, the receiver decrypts the ciphertexts and reconstructs the message. Our construction is slightly more technical since it must allow the simulator to decide whether to send fake keys/ciphertexts only after corruption takes place. To ensure that, we use an additional tool, denoted by ℓ-equivocal NCE, which is discussed in details below (informally, this primitive improves NCE constructions for small equivocation space) . We note that our protocol is secure as long as either the sender or the receiver are adaptively corrupted, but not both. Informally, we prove that

Theorem 1.1. *Assume the existence of NCER, NCES and ℓ-equivocal NCE, and then there exists one-sided NCE with a constant number of invocations of these primitives and $\ell = 2$.*

Secure realizations of NCER and NCES exist under several concrete assumptions. Specifically, NCER implementations were shown in [14,37] under the respective DDH and DCR hardness assumptions, whereas NCES was realized under the DDH assumption in [4]. In this paper, we further show how to realize NCES under the DCR assumption. Note that when viewing these primitives as two-round protocols, where the receiver forwards the sender its public key that is followed by a ciphertext sent by the sender, these primitives are equivalent in the sense that it is possible to convert one primitive to another at the expense of one additional round. Specifically, given a two-round NCER protocol, a three-round NCES can be constructed as follows: the sender creates a public key/secret key pair for the NCER and forwards the public key to the receiver. The receiver then encrypts a one-time pad masking under the public key. Finally, the sender uses this pad to mask its message in the third round. An NCER protocol can be constructed from an NCES protocol in a similar way.

Our theorem is also interesting in the sense that it implies one-sided NCE with a number of PKE operations that is independent of the message length. Concretely, if the underlying NCER and NCES are implemented using a constant number of PKE operations, that is independent of the message length, then this also holds for our one-sided NCE. In this paper, we consider implementations for NCER/NCES that are efficient in that sense which implies efficient DCR-based one-sided NCE with a constant overhead. We further consider DDH-based constructions that achieve the same overhead, but for polynomial-size domains. We provide comparison with prior work in Table 1.

Table 1. Comparisons with prior NCE constructions for message space $\{0,1\}^n$ and security parameter n.

References	Security	Hardness assumption	Overhead (number of exp.)
[12]	One-sided NCE	Simulatable common-domain trapdoor	$O(n)$
[22]	One-sided NCE\ NCE	Enhanced trapdoor permutation \ simulatable PKE	$O(n)\backslash O(n)$
[9]	NCE	Trapdoor simulatable	$O(n)$
[33]	NCE	Φ-hiding	$O(n)$
This work	One-sided NCE	NCER+NCES+ℓ- equivocal NCE	$O(1)$

1.2.2. *Witness Equivocal UC ZK PoK for Compound Statements*

A basic tool in constructing maliciously secure protocols that we exploit in this paper is zero-knowledge (ZK) proofs. More specifically, in this work we focus on compound statements (where the statement is comprised of sub-statements for which the prover only knows a subset of the witnesses). We consider a new notion of *witness equivocal* UC ZK proofs of knowledge (PoK) where the simulator knows the witnesses for *all* sub-statements but not which subset is known to the real prover, and show how to build adaptively secure witness equivocal proofs for a large class of Σ-protocols. In particular, the security proof for this notion implies that the simulator convinces the adaptive adversary that it knows the exact same subset of witnesses known also to the real prover. We demonstrate that even though this notion is weaker than one-sided security (which requires simulation without any knowledge of witnesses), it is still meaningful in designing one-sided secure protocols.

As a side result, we demonstrate a technique for efficiently generating statically secure UC ZK PoK for the same class of Σ-protocols. Our protocols use a new approach where the prover commits to an additional transcript which enables to extract the witness in the CRS setting with a constant overhead where previously, Σ-protocols were complied into the UC setting using UC commitments [11].

1.2.3. *One-Sided Oblivious Transfer*

In the next step, we combine our one-sided NCE and witness equivocal proofs in order to implement one-sided 1-out-of-2 OT. We build our protocol based on the generic framework of [48] with the following modifications. (**1**) First, we require that the sender sends its ciphertexts via a one-sided non-committing channel (based on our previous result, this only inflates the overhead by a constant). (**2**) We fix the common parameters in a single mode (whereas the [31] and [48] proofs need to alternate between the two modes). To ensure correctness with respect to the receiver's message, we employ a witness equivocal ZK PoK. Informally, we prove that

Theorem 1.2. *Assume the existence of one-sided NCE, dual-mode PKE* (*cf. Definition 4.1; see also* [48]) *and witness equivocal ZK PoK, and then there exists one-sided OT with $O(1)$ PKE operations.*

Table 2. Comparisons with prior malicious UC OT constructions for sender's message space $\{0,1\}^n$ and security parameter n.

References	Security	Hardness assumption	Overhead (number of exp.)
[48]	Static	Dual-mode PKE	$O(1)$
[15]	Adaptive	Augmented NCE	$O(n)$
[10]	Adaptive	Adaptive semi-honest OT + UC commitments	$O(n)$
[31]	Adaptive	Enhanced dual-mode PKE + UC commitments	$O(n)$
This work	One-sided adaptive	One-sided NCE+dual-mode PKE+ Witness equivocal ZK PoK	$O(1)$

Considering efficiency, our construction requires a number of PKE operations that is independent of the sender's input space. This is significantly better than all prior work on fully adaptively UC secure OT that require $O(1)$ such operations for implementing bit OT; see Table 2 for comparison with prior work.

We note that this protocol further serves as the basis for the cut-and-choose OT protocol we design next. In addition, a semi-honest variant of our OT protocol can be considered if the parties mutually generate the CRS using a coin tossing protocol that is UC secure. By plugging-in this one-sided semi-honest OT protocol into the [27] static semi-honest protocol, we obtain a one-sided semi-honest adaptively secure protocol, with round complexity that depends on the computed circuit's depth. This implies the following theorem,

Theorem 1.3. *Assume the existence of one-sided semi-honest OT and statically secure UC commitment scheme, and then there exists a semi-honest one-sided adaptively secure two-party protocol that requires* $(O(|C|))$ *public key operations.*

1.2.4. *Constant Round One-Sided Secure Computation*

Notably, in the plain model any statically secure protocol can be transformed into a protocol with one-sided adaptive security by encrypting the communication of the static protocol using NCE. This approach, taken by [38], implies that the number of public key operations grows linearly with the communication complexity of the static protocol. Moreover, currently the overhead of generic protocols using this approach depends on the circuit's size times the security parameter.[1]

In this work, we revisit the general compiler of [38] and design improved generic one-sided constant round protocols tolerating semi-honest and malicious behaviour. This improvement is based on the observation that it is sufficient to employ one-sided NCE rather than NCE (that is fully secure) in order to secure the communication. Moreover, by plugging-in our one-sided NCE we obtain a better transformation from static semi-honest/malicious security into the corresponding attack model with one-sided security. It is important to note that this transformation does not hold in the UC setting due to the

[1] We note that this statement is valid regarding protocols that do not employ fully homomorphic encryptions (FHE). To this end, we only consider protocols that do not take the FHE approach. As a side note, it was recently observed in [39] that adaptive security is impossible for FHE satisfying compactness.

Table 3. Comparisons with prior generic semi-honest secure protocols for n the security parameter.

References	Security	Hardness assumption	Overhead (number of exp.)
[41]	Static, plain model	Semi-honest OT	$O(\|input\|)$
[38]	One-sided, plain model	Simulatable PKE	$O(n^2\|C\|)$
[15]	Adaptive, UC	Augmented NCE	$O(\|C\|)$
This work	One-sided, UC	Static semi-honest secure Computation + one-sided NCE	$O(\|C\|)$

additional setup. Specifically, the security proof may crucially rely on the fact that the CRS is chosen dependently on the identity of the corrupted party, as for instance in [48]. Nevertheless, since this identity is not known in advance in the adaptive setting, fixing the CRS in one particular mode breaks down the static security proof.

To conclude, in the semi-honest setting (when no trusted setup is required) we prove that our transformation applies even in the UC setting. Informally, we first prove that

Theorem 1.4. *Assume the existence of one-side NCE. Then:* (1) *statically secure semi-honest secure computation implies one-sided semi-honest UC computation, and* (2) *statically secure malicious secure computation implies one-sided malicious secure computation.*

As a corollary, we obtain that constant round semi-honest secure computation can be achieved based on the assumptions needed in [41] and one-sided NCE. Informally stating,

Corollary 1.5. *Under the assumptions of achieving security in* [41] *and one-sided NCE, there exists a constant round one-sided semi-honest UC secure two-party protocol that requires $O(|C|)$) public key operations, where C is the computed circuit.*

We provide a detailed comparison with prior work for the semi-honest setting in Table 3. Next, in order to obtain one-sided UC security against malicious attacks we adapt the cut-and-choose-based protocol from [42], which relies heavily on [48] DDH-based OT protocol. The idea of the cut-and-choose technique is to ask one party to send s garbled circuits and later open half of them by the choice of the other party. This ensures that with very high probability the majority of the unopened circuits are valid. Proving security in the one-sided setting requires dealing with new subtleties and designing a modified cut-and-choose building blocks, since [42] defines the public parameters for these building blocks in a way that precludes equivocation of the parties' inputs. Informally,

Theorem 1.6. *Under the assumptions of achieving static malicious UC security in* [42]*, one-sided cut-and-choose OT and simulatable PKE, there exists a constant round one-sided malicious UC secure two-party protocol that requires $O(s|C|)$ public key operations where s is a statistical parameter that determines the cut-and-choose soundness error.*

Table 4. Comparisons with prior generic maliciously secure protocols for n the security parameter.

References	Security	Hardness assumption	Overhead (number of Exp.)
[42]	Static, plain model	DDH	$O(\|input\|)$
[38]	One-sided, plain model	Simulatable PKE	$O(n^2\|C\|)$
[15]	Adaptive, UC	Augmented NCE	$O(n^2\|C\|)$
[35]	Adaptive, UC	Adaptive malicious OT	$O(\|n^2C\|)$
This work	One-sided, UC	DDH+DCR	$O(\|sC\|)$

We provide a details comparison with prior work for the malicious setting in Table 4.

To conclude, our results imply that one-sided security is easier to achieve than fully adaptive security. We leave open the efficiency of constant round one-sided secure protocols in the multi-party setting. Currently, it is not clear how to extend our techniques beyond the two-party setting (such as for the [5] protocol). Another open problem is whether it is feasible to achieve secure constructions with a number of PKE operations that are strictly less than what we achieve here.

1.3. *Prior Work*

We describe prior work on NCE, adaptively secure OT and adaptively secure two-party computation.

1.3.1. *Non-Committing Encryption*

(One-sided) NCE was introduced in [12] which demonstrated its feasibility under the RSA assumption. Next, NCE was studied in [9,22]. More concretely, the construction of [22] requires constant rounds on the average and is based on simulatable PKE, whereas [9] presented an improved expected two rounds NCE based on a weaker primitive. [22] further presented one-sided NCE based on a weakened simulatable PKE notion. The computational overhead of all these constructions depends on the message length. Following that, the relatively new ℓ-equivocal NCE notion was introduced in [31]. This primitive enables to send arbitrarily long messages at the cost of $\log \ell$ PKE operations, where ℓ is the equivocality parameter, and improves over NCE for sufficiently small ℓ's. In [44], Nielsen proved that NCE must have a decryption key that is at least as long as the transmitted message. Finally, Hemenway et al. [33] presented the first two-round NCE construction under the Φ-hiding hardness assumption where the communication complexity of the second message is $O(m \log m + n)$ where m is the message length and n is the security parameter, which improves over all prior constructions.

1.3.2. *Adaptively Secure Oblivious Transfer*

In [2,15], semi-honest adaptively secure OT protocols were shown that were then compiled into the malicious setting using generic ZK proofs. More recently, in a weaker model that assumes erasures, Lindell [40] used the method of [50] to design an efficient transformation from static OT to semi-honest composable adaptive OT. Another recent work by Garay *et al.* [31] presented a UC adaptively secure OT, building on the static OT

of [48] and ℓ-equivocal NCE. This paper introduced an OT protocol with security under a weaker semi-adaptive notion that is then compiled into an adaptive OT by encrypting the transcript of the protocol using ℓ-equivocal NCE.[2] Finally, [10] presented a compiler for UC adaptive OT in the malicious setting based on semi-honest adaptive OT and UC commitment schemes.

1.3.3. *Adaptively Secure Two-Party Computation*

The work by Katz and Ostrovsky [38] was the first to study the round complexity of one-sided secure protocols. Their round efficient protocol takes a naive approach of encrypting the entire communication using NCE. Next, the work of [35] provided all-but-one adaptively secure protocols based on honest majority adaptively secure protocols (where their particular instantiation uses the constant rounds protocol from [18]). Finally, a recent work by Garg and Sahai [30] presents adaptively secure constant round protocols tolerating all-but-one corrupted parties using a non-black box simulation approach. Their approach uses the OT hybrid compiler of [35].

Adaptive secure computation has been extensively studied as well. In the non-erasure model, the work of [15] demonstrated the feasibility of adaptive UC security of any well-formed functionality. The followup work of [24] showed how to use a threshold encryption to achieve UC adaptive security but required honest majority. A generic compiler from static to adaptive security was shown in [7] (yet without considering post-execution corruptions). Assuming erasures, which significantly simplify the problem, one of the earliest works by Beaver and Haber [3] showed an efficient generic transformation from adaptively secure protocols with ideally secure communication channels, to adaptively secure protocols with standard (authenticated) communication channels. A more recent work by Lindell [40] presented an efficient semi-honest constant round two-party protocol with adaptive security.

A recent line of works [13,21,29] studies constant rounds adaptively secure computation using obfuscation techniques. This approach is different than all prior work on adaptive security since it obfuscates the circuit that computes the next message in the protocol and places the result in the CRS.

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 probabilistic polynomial-time. We denote the message spaces of our NCE schemes and the message space of the sender in our OT protocols by $\{0,1\}^q$ for $q = n$.

We specify the definitions of computational 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

[2] We stress that the semi-adaptive notion is incomparable to the one-sided notion since the former assumes that either one party is statically corrupted or none of the parties get corrupted.

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

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

We say that X_n and Y_n are ϵ-*close* if their statistical distance is at most $\mathrm{SD}(X_n, Y_n) \leq \epsilon(n)$. 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. *Security Definitions*

In the following, we formalize the notion of UC one-sided adaptive security [6]. Formally, a two-party computation protocol is cast by specifying the participating parties P_0 and P_1 and a function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_0, f_1)$ mapping pairs of inputs to pairs of outputs (one for each party). That is, for every pair of inputs $x_0, x_1 \in \{0,1\}^n$ the output pair is a random variable $(f_0(x_0, x_1), f_1(x_0, x_1))$ ranging over pair of strings. 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 $f_1(x_0, x_1)$.

2.1.1. *One-Sided Adaptive Security*

In the two-party setting, a real execution of some protocol Π_f that implements f is run between two parties P_0 and P_1 in the presence of an adversary Adv and an environment Env (that is given an input z, a random tape r_{Env} and a security parameter n) and is modelled as a sequence of activations of the entities. Env is activated first and generates the inputs for the other entities. Then the protocol proceeds by having the parties communicate with each other and Adv exchange messages with Env. Upon completing the real execution Env outputs a bit. In the ideal model, the computation involves an incorruptible trusted third party $\mathcal{F}_f$ which receives the parties' inputs, computes the function f on these inputs and returns to each party its respective output. The parties are replaced by dummy parties that do not communicate with each other, such that whenever a dummy party is activated it sends its input to the ideal functionality. Upon completing the ideal execution Env outputs a bit. We say that a protocol Π_f UC realizes functionality $\mathcal{F}_f$ if for any real-world adversary Adv there is a ideal-world adversary Sim such that no Env can tell with non-negligible probability whether it is interacting with Adv and the parties running Π_f in a real execution or with Sim and the dummy parties in an ideal execution; details follow.

Execution in the real model. We now proceed with a real-world execution, where a real two-party protocol is executed. Whenever ENV is activated, it first fixes input $x_i \in \{0,1\}^*$ for party P_i. Each party P_i then starts the execution with an input $x_i \in \{0,1\}^*$, a random tape r_i and a security parameter n. A *one-sided* adversary ADV is a probabilistic polynomial-time interactive Turing machine that is given a random tape r_{ADV} and a security parameter n and is allowed to corrupt at most one party. At the outset of the protocol, ADV receives some initial information from ENV. Then the computation proceeds in rounds such that in each round ADV sees all the messages sent between the parties. At the beginning of each round, ADV may choose to corrupt P_{i^*} for $i^* \in \{0,1\}$. Upon corrupting P_{i^*}, ADV learns its input and the random tape, and notifies ENV, obtaining back some auxiliary information. In case ADV is malicious P_{i^*} follows ADV's instructions from the time it is corrupted. At the end of the protocol execution, the honest parties locally compute their outputs and output the value specified by the protocol, whereas the corrupted party outputs a special symbol $\perp$. The adversary ADV outputs an arbitrary function of its internal state that includes, r_{ADV}, the messages received from ENV and the corrupted party's view. Next, a post-execution corruption process begins where ENV first learns the outputs. Then, ADV and ENV interact in at most one additional round. If none of the parties is corrupted yet, ENV can ask ADV to corrupt P_{i^*} for $i^* \in \{0,1\}$, receiving back the state of this party. At the end, ENV outputs a bit.

Let f be as specified above and Π_f be a two-party protocol that computes f. We denote by the variable $\mathbf{OREAL}_{\Pi_f,\text{ADV},\text{ENV}}(n, x_0, x_1, z, \mathbf{r})$ the output of ENV on input z, random tape r_{ENV} and a security parameter n upon interacting with ADV and parties P_0, P_1 that engage in protocol Π_f on inputs r_{ADV} and (x_0, r_0), (x_1, r_1), respectively, where $\mathbf{r} = (r_{\text{ENV}}, r_{\text{ADV}}, r_0, r_1)$. Let $\mathbf{OREAL}_{\Pi_f,\text{ADV},\text{ENV}}(n, x_0, x_1, z)$ denote a random variable describing $\mathbf{OREAL}_{\Pi_f,\text{ADV},\text{ENV}}(n, x_0, x_1, z, \mathbf{r})$ where the random tapes are chosen uniformly. Let $\mathbf{OREAL}_{\Pi_f,\text{ADV},\text{ENV}}$ denote the distribution ensemble:

$$\{\mathbf{OREAL}_{\Pi_f,\text{ADV},\text{ENV}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}.$$

Execution in the ideal model. A one-sided ideal-world adversary SIM is a probabilistic polynomial-time interactive Turing machine that is given a random tape r_{SIM} and a security parameter n and is allowed to corrupt at most one party. The ideal process is defined with respect to a trusted party that implements functionality $\mathcal{F}_f$ as follows:

First corruption phase: SIM receives some auxiliary information from ENV. Next, SIM may decide whether to corrupt party P_{i^*} for $i^* \in \{0,1\}$. Upon corrupting party P_{i^*}, SIM notifies to ENV and learns its input x_{i^*}. In addition, ENV hands some auxiliary information to SIM.

Computation phase: In the semi-honest setting, each party forwards its input to the trusted party. In the malicious settings, the corrupted party hands $\mathcal{F}_f$ the values handed to it by SIM. The trusted party computes $(y_0, y_1) = f(x_0, x_1)$ and hands each P_i the value y_i. SIM receives the output of the corrupted party.

Second corruption phase: SIM continues to another corruption phase, where it might choose to corrupt P_{i^*} for $i^* \in \{0,1\}$ (in case it did not corrupt any party in the first corruption phase), where this choice is made based on SIM's random tape and

all the information gathered so far. Upon corrupting P_{i^*}, SIM notifies to ENV and learns the party's input x_{i^*}. ENV hands SIM some auxiliary information.

Output: The uncorrupted party P_{1-i^*} outputs y_{1-i^*} and the corrupted party outputs $\perp$. SIM outputs an arbitrary efficient function of its view. ENV learns all the outputs.

Post-execution corruption phase: After the outputs are generated, SIM proceeds with ENV in at most one round of interaction, where ENV can instruct SIM to corrupt P_{i^*} for $i^* \in \{0, 1\}$ (if none of the parties are corrupted yet). SIM generates some arbitrary answer and might choose to corrupt P_{i^*}. The interaction continues until ENV halts with an output.

We denote by $\mathbf{OIDEAL}_{\mathcal{F}_f,\text{SIM},\text{ENV}}(n, x_0, x_1, z, \mathbf{r})$ the output of ENV on input z, random tape r_{ENV} and security parameter n upon interacting with SIM and parties P_0, P_1, running an ideal process with inputs r_{SIM} and x_0, x_1, respectively, where $\mathbf{r} = (r_{\text{ENV}}, r_{\text{SIM}})$. Let $\mathbf{OIDEAL}_{\mathcal{F}_f,\text{SIM},\text{ENV}}(n, x_0, x_1, z)$ denote a random variable describing $\mathbf{OIDEAL}_{\mathcal{F}_f,\text{SIM},\text{ENV}}(n, x_0, x_1, z, \mathbf{r})$ when the random tapes r_{ENV} and r_{SIM} are chosen uniformly. Let $\mathbf{OIDEAL}_{\mathcal{F}_f,\text{SIM},\text{ENV}}$ denote the distribution ensemble:

$$\{\mathbf{OIDEAL}_{\mathcal{F}_f,\text{SIM},\text{ENV}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

Then we define security as follows.

Definition 2.3. Let $\mathcal{F}_f$ and Π_f be as defined above. Protocol Π_f UC realizes $\mathcal{F}_f$ in the presence of one-sided semi-honest/malicious adversaries if for every non-uniform probabilistic polynomial-time one-sided semi-honest/malicious adversary ADV, there exists a one-sided non-uniform probabilistic polynomial-time ideal adversary SIM such that:

$$\begin{aligned}&\{\mathbf{OIDEAL}_{\mathcal{F}_f,\text{SIM},\text{ENV}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}\\ &\quad\approx_c \{\mathbf{OREAL}_{\Pi_f,\text{ADV},\text{ENV}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}\end{aligned}$$

where $|x_0| = |x_1|$.

Composition. In order to simplify our security proofs we consider a hybrid setting where the parties implement some functionalities using ideals calls. We rely on the composition theorem introduced by Canetti [6] in the adaptive setting. (Note that we are only interested in cases where the same party is corrupted with respect to all composed protocols.)

2.1.2. *Concrete Functionalities*

We specify the definition of three important functionalities for this work.

Secure communication (SC). We define the functionality $\mathcal{F}_{\text{SC}}$ for securely communicating a message m from SEN to REC, following the notations from [31]. 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

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

Functionality $\mathcal{F}_{\text{SC}}^{\mathcal{O}}$ 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 $(\mathsf{ChSetup}, sid, \text{SEN})$ from SEN, initialize the machine $\mathcal{O}$ and record the tuple $(sid, \mathcal{O})$. Pass the message $(\mathsf{ChSetup}, \text{SEN})$ to REC. In addition pass this message to $\mathcal{O}$ and forward its output to SIM.
2. **Message Transfer.** Upon receiving an input $(\mathsf{send}, sid, \text{SEN}, m)$ from party SEN, find a tuple $(sid, \mathcal{O})$, and, if none exits, ignore the message. Otherwise, send the message $(\mathsf{send}, sid, \text{SEN}, m)$ to REC. In addition invoke $\mathcal{O}$ with $(\mathsf{send}, sid, \text{SEN}, m)$ and forward its output to SIM.
3. **Corruption.** Upon receiving message $(\mathsf{corrupt}, sid, P)$ from SIM where $P \in \{\text{SEN}, \text{REC}\}$, 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.

Fig. 1. The message transfer functionality.

depends on the security properties required for the oracle and thus can capture several notions such as NCE and ℓ-equivocal NCE (see Sect. 3.1). 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 $i \in \{1, \ldots, \ell\}$. In Fig. 1 we define the message transfer 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.

Definition 2.4. Let $\mathcal{O}$ be an oracle, which on input $(\mathsf{send}, sid, \text{SEN}, m)$, produces the output $(\mathsf{send}, sid, \text{SEN}, |m|)$, and on any inputs corresponding to the $\mathsf{ChSetup}$, $\mathsf{Corrupt}$ commands produce no output. We call the functionality $\mathcal{F}_{\text{SC}}^{\mathcal{O}}$ or just $\mathcal{F}_{\text{SC}}$ for brevity, an NCE ideal functionality. A real-world protocol which realizes $\mathcal{F}_{\text{SC}}$ is called an NCE scheme.

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

Definition 2.5. An oracle $\mathcal{I}$ is called message-ignoring oracle if, on any input $(\mathsf{send}, sid, \text{SEN}, m)$, it ignores the message value m and processes only the input $(\mathsf{send}, sid, \text{SEN}, |m|)$. 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 2.6. Let $(\mathcal{M}, \mathcal{I})$ be a well-matched pair which consists of a message-processing and a message-ignoring oracle, respectively. Let $\mathcal{O}^{\ell}$ be a (stateful) oracle with the following structure.

– 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 $\mathcal{O}_i =$

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

Functionality $\mathcal{F}_{\text{OT}}$ communicates with with sender SEN and receiver REC, and adversary SIM.

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

Fig. 2. The oblivious transfer functionality.

$\mathcal{M}$ and for $j \neq i$, the oracles $\mathcal{O}_j$ are independent copies of the message-ignoring oracle $\mathcal{I}$.

– Whenever $\mathcal{O}^\ell$ receives inputs of the form (ChSetup, sid, SEN) or (send, sid, SEN, m), it passes the input to each oracle $\mathcal{O}_i$ receiving an output y_i. It then outputs the vector $(y_1, \ldots, y_\ell)$.

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

As before, no information about messages m is revealed during the "send" 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 modelled by $\mathcal{F}_{\text{SC}}^{\ell}$, in a way that allows it to later explain this communication as any one of ℓ possibilities.

Oblivious transfer (OT). The 1-out-of-2 OT functionality $\mathcal{F}_{\text{OT}}$ is defined in Fig. 2.

3. One-Sided Adaptively Secure NCE

In this section, we discuss our first result regarding one-sided NCE, that is building on the cryptographic primitives NCE for the receiver (NCER), NCE for the sender (NCES) and ℓ-equivocal NCE. Before presenting our construction we review the security definitions of these primitives.

3.1. *NCER, NCES and ℓ-Equivocal NCE*

3.1.1. *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 trap-

door 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 algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^*, \mathsf{Dec}, \mathsf{Equivocate})$ specified as follows:

- Gen, Enc, Dec are as specified in Definition 7.4.
- 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 3.1.]rm(*NCER*) NCE for the receiver is a tuple of algorithms defined above that satisfy the following properties:

1. Gen, Enc, Dec imply an IND-CPA secure encryption scheme as in Definition 7.5.
2. **Ciphertext indistinguishability.** For any $m \in \{0,1\}^n$ the following distributions are computationally indistinguishable:

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

NCER can be realized under the DDH assumption [14,37] for polynomial-size message spaces and under the DCR assumption for exponential-size message spaces [14].

3.1.2. *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 algorithms $(\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ specified as follows:

- Gen, Enc, Dec are as specified in Definition 7.4.
- 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 3.2. (*NCES*) An NCE for the sender is a tuple of algorithms defined above that satisfy the following properties:

1. Gen, Enc, Dec imply an IND-CPA secure encryption scheme as in Definition 7.5.
2. **Public key indistinguishability.** For any $m \in \{0,1\}^n$ the following distributions are computationally indistinguishable:

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

NCES can be realized under the DDH assumption [4] for polynomial-size message spaces and under the DCR assumption for exponential-size message spaces. The later construction is presented in Section 3.1.4.

3.1.3. *ℓ-Equivocal NCE* [31].

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. Instead, the simulator is given a set of ℓ messages during the generation of the fake ciphertext and must later be able to plausibly explain the ciphertext as the encryption of any one of those ℓ messages (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 [31] and thus improves over NCE whenever ℓ is very small, but the plaintext length is large. Specifically, the [31] construction requires $O(\log \ell)$ PKE operations. In this paper, we use this primitive to encrypt plaintexts domains of length n with constant overhead. ℓ-equivocal NCE is realized in [31] under the same hardness assumptions that imply NCE.

3.1.4. *NCES for Exponential-Size Message Spaces*

In what follows, we introduce a new NCE for the sender based on the security of the DCR assumption. Our scheme is based on the PKE from [14], building on earlier work by Cramer and Shoup [16]. Let $N = pq$ be an RSA modulus, then define $\Pi^{\mathrm{DCR}}_{\mathrm{NCES}} = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ an NCES as follows.

- Gen, given the security parameter n, generate an RSA modulus $N = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ where p, q, p', q' are primes. Pick $g' \leftarrow \mathbb{Z}^*_{N^2}$ and $\alpha \leftarrow \mathbb{Z}_{N^2/4}$ and set $g_0 = g'^{2N} \bmod N^2$ and $h_0 = g_0^{\alpha} \bmod N^2$. Choose a random $r \leftarrow \mathbb{Z}_{N/4}$ and compute $g_1 = g_0^r \bmod N^2$, $h_1 = ((1+N) \cdot h_0^r) \bmod N^2$. Output $\mathrm{PK} = (N, g_0, h_0, g_1, h_1)$ and secret key $\mathrm{SK} = \alpha$.
- Gen^*, given the security parameter n, generate N, g_0, h_0 as in Gen. Choose a random $r \leftarrow \mathbb{Z}_{N/4}$ and compute $g_1 = g_0^r \bmod N^2$, $h_1 = h_0^r \bmod N^2$. Output $\mathrm{PK}^* = (N, g_0, h_0, g_1, h_1)$ and trapdoor $t_{\mathrm{PK}^*} = r$.
- Enc, given the public key $\mathrm{PK} = (N, g_0, h_0, g_1, h_1)$ (or PK^*) and a message $m \in \mathbb{Z}_N$, choose a random $t \leftarrow \mathbb{Z}_{N/4}$ and output the ciphertext

$$c \leftarrow \mathsf{Enc}(m; t) = \left((g_1^m g_0^t) \bmod N^2, (h_1^m h_0^t) \bmod N^2\right).$$

- Dec, given the public key $\mathrm{PK} = (N, g_0, h_0, g_1, h_1)$, secret key $\mathrm{SK} = \alpha$ and ciphertext $c = (g_c, h_c)$, compute $\hat{m}$ as follows and output $m \in \mathbb{Z}_N$ such that $\hat{m} = 1 + mN$.

$$\hat{m} = \left(h_c / g_c^{\alpha}\right)^{N+1} = [(1+N)^m]^{N+1} = (1+N)^m.$$

– Equivocate, given $\Phi(N)$, the fake key $\mathrm{PK}^* = (N, g_0, h_0, g_1, h_1)$, trapdoor $t_{\mathrm{PK}^*} = r$, a ciphertext $c^* \leftarrow \mathsf{Enc}_{\mathrm{PK}^*}(m; t) = (g_c, h_c)$ and a message m', output $t' = (rm + t - rm') \bmod \Phi(N)/4$. It is easy to see that

$$\begin{aligned}\mathsf{Enc}_{\mathrm{PK}^*}(m'; t') &= \left((g_1^{m'} g_0^{t'}), h_1^{m'} h_0^{t'})\right) \\ &= \left((g_0^{rm'} g_0^{(rm+t-rm')}), (h_0^{rm'} h_0^{(rm+t-rm')})\right) = c.\end{aligned}$$

Next, we show that this scheme meets Definitions 3.2.

Proposition 3.3. *Assume that the DCR assumption is hard in $\mathbb{Z}^*_{N^2}$, then $\Pi^{\mathrm{DCR}}_{\mathrm{NCES}}$ is a NCES.*

Proof. Given the public key $\mathrm{PK} = (N, g_0, h_0, g_1, h_1)$ and two messages $m, m' \in \mathbb{Z}_N$, it holds that encryptions of any two messages m and m', namely the tuples (1) $\left((g_1^m g_0^t) \bmod N^2, (h_1^m h_0^t) \bmod N^2\right)$ and (2) $\left((g_1^{m'} g_0^t) \bmod N^2, (h_1^{m'} h_0^t) \bmod N^2\right)$, are computationally close. This follows immediately from the IND-CPA security proof for the modified scheme in [19] (cf. Theorem 2 of [19]). The fact that fake and valid public keys are computationally indistinguishable follows from the IND-CPA security of [14] and [16], as the former key is an encryption of zero, whereas the latter key is an encryption of one. □

3.2. *Our Construction*

The idea of our 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. We are now ready to introduce our protocol in the $\mathcal{F}^{\ell}_{\mathrm{SC}}$-hybrid model (for $\ell = 2$). Formally, let $\Pi_{\mathrm{NCES}} = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ and $\Pi_{\mathrm{NCER}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^*, \mathsf{Dec}, \mathsf{Equivocate})$, respectively, denote NCES and NCER for message space $\{0,1\}^n$. Then, consider the following protocol that realizes $\mathcal{F}_{\mathrm{SC}}$ with one-sided security.

Protocol 1. (**One-sided NCE** (Π_{OSC}))

- **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* $(\text{PK}_S, \text{SK}_S)$ *and* $(\text{PK}_R, \text{SK}_R)$*, respectively.* REC *then forwards* (ChSetup, sid, REC) *and* (send, sid, REC, PK_S) *to* $\mathcal{F}_{\text{SC}}^{\ell}$*, and the message* PK_R *to* SEN.
  2. **Message from the sender.** *Upon receiving* (send, sid, REC, PK_S) *from* $\mathcal{F}_{\text{SC}}^{\ell}$ *and* PK_R *from* REC, SEN *creates two shares of* m, m_S *and* m_R, *such that* $m = m_S \oplus m_R$. *It then encrypts* m_S (resp. m_R) *using* PK_S (resp. PK_R)*, creating ciphertext* c_S (resp. c_R)*, and forwards* (ChSetup, sid′, SEN), (send, sid′, SEN, c_S) *and* (ChSetup, sid″, SEN), (send, sid'', SEN, c_R) *to* $\mathcal{F}_{\text{SC}}^{\ell}$
  3. **Output.** *Upon receiving* (send, sid′, SEN, c_S) *and* (send, sid″, SEN, c_R) *from* $\mathcal{F}_{\text{SC}}^{\ell}$, REC *decrypts* c_S (resp. c_R) *using* SK_S *(resp.* SK_R*) and outputs the bitwise XOR of the decrypted plaintexts.*

Our protocol can be instantiated using DDH-based NCES and NCER for polynomial-size message spaces, and using DCR-based NCES and NCER for exponential-size message spaces. Note that the message space of our one-sided NCE is implied by the message spaces of the underlying NCES/NCER schemes. Therefore, when instantiated with the above concrete constructions, our protocol can transmit n-bits messages using a *constant number of PKE operations*, as opposed to NCE that requires $O(n)$ such operations. We conclude with the following theorem and the complete proof.

Theorem 3.4. *Assume the existence of NCER and NCES. Then Protocol 1 UC realizes* $\mathcal{F}_{\text{SC}}$ *in the* $\mathcal{F}_{\text{SC}}^{\ell}$*-hybrid model* (for $\ell = 2$) *in the presence of one-sided adaptive malicious adversaries.*

Intuitively, security follows due to the fact that the simulator is not committed to either valid keys or valid ciphertexts as it plays the role of functionality $\mathcal{F}_{\text{SC}}^{\ell}$. Thus, upon corrupting one of the parties it is able to explain that party's internal state while equivocating the communication with respect to message m.

Proof. Let ADV be a malicious probabilistic polynomial-time adversary attacking Protocol 1 by adaptively corrupting one of the parties. We construct an adversary SIM for the ideal functionality $\mathcal{F}_{\text{SC}}$ such that no environment ENV distinguishes with a non-negligible probability whether it is interacting with ADV in the hybrid setting or with SIM in the ideal setting. We recall that SIM interacts with the ideal functionality $\mathcal{F}_{\text{SC}}$ and the environment ENV. We refer to the interaction of SIM with $\mathcal{F}_{\text{SC}}$ and ENV as the external interaction. The interaction of SIM with the simulated ADV is the internal interaction. We explain the strategy of the simulation for all corruption cases.

Simulating the communication with ENV. Every input value received by the simulator from ENV is written on ADV's input tape. Likewise, every output value written

by ADV on its output tape is copied to the simulator's output tape (to be read by the environment ENV).

SEN is corrupted at the outset of the protocol. SIM begins by activating ADV and 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. It then picks two public/secret key pairs $(\text{PK}_S, \text{SK}_S)$ and $(\text{PK}_R, \text{SK}_R)$. It then emulates functionality $\mathcal{F}_{\text{SC}}^{\ell}$ and the honest receiver by forwarding ADV the message $(\mathsf{send}, sid, \text{REC}, \text{PK}_S)$ from $\mathcal{F}_{\text{SC}}^{\ell}$ and PK_R from REC. Upon receiving $(\mathsf{send}, sid', \text{SEN}, c_S)$ and $(\mathsf{send}, sid'', \text{SEN}, c_R)$ from ADV, SIM extracts m (that may be different than the message received from $\mathcal{F}_{\text{SC}}$) by computing $m = \mathsf{Dec}_{\text{SK}_S}(c_S) \oplus \mathsf{Dec}_{\text{SK}_R}(c_R)$. In case decryption does not follow correctly, SIM fixes m to be a default value. It then externally forwards $(\mathsf{ChSetup}, sid, \text{SEN})$ and $(\mathsf{send}, sid, \text{SEN}, m)$ to the ideal functionality $\mathcal{F}_{\text{SC}}$. Note that corruption takes place at the outset of the protocol execution and thus the simulator is able to simulate the transcript exactly as in the hybrid setting.

REC is corrupted at the outset of the protocol. SIM begins by activating ADV and 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 ADV and receives $(\mathsf{send}, sid, \text{SEN}, \text{PK}_S)$ (which is the message sent to $\mathcal{F}_{\text{SC}}^{\ell}$), 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 between Steps 1 and 2. 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 between Steps 1 and 2. Upon receiving the receiver's output message m, 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 $\mathcal{F}_{\text{SC}}^{\ell}$. 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.

If none of the above corruption cases occur, SIM emulates the sender's message as follows. It first chooses two random shares m'_S, m'_R and generates a pair of ciphertexts (c'_S, c^*_S) for Π_{NCES} where both encrypt m'_S, respectively, using PK_S and PK_S^*. It then generates a pair of ciphertexts (c'_R, c^*_R) for Π_{NCER} such that c'_R is a valid encryption of m'_R under public key PK_R, and c^*_R is a fake ciphertext generated using Enc^* and 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 after Step 2 is concluded. 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^* and c'_R as the ciphertexts sent to $\mathcal{F}_{\text{SC}}^{\ell}$. Finally, it computes $r'' \leftarrow \mathsf{Equivocate}(c_S^*, t_{\text{PK}_S^*}, m''_S)$ for m''_S such that $m = m''_S \oplus m'_R$, and presents r'' as the randomness used to generate c_S^* for encrypting m''_S. The randomness used for generating c'_R is revealed honestly.

Consider the adversary's view which is comprised from three invocations of $\mathcal{F}_{\text{SC}}^{\ell}$ (one played with the role of the receiver and two with the role of the sender) and a public key PK_R that was honestly generated. Note that the message received via the first $\mathcal{F}_{\text{SC}}^{\ell}$ invocation is a fake public key PK_S^*. Moreover, the remaining two invocations are for sending the honestly generated ciphertexts relative to keys PK_S^* and PK_R. Now, since the only different between the simulated and hybrid views is with respect to the key for NCES, we reduce the security of this case to the security of Π_{NCES}.

Specifically, here the difference boils down to the key indistinguishability property of Π_{NCES}. Given m and a tuple in one of these forms $(\text{PK}^*, r^*, \tilde{m}, c^*)$ or $(\text{PK}, r, \tilde{m}, c)$ for an arbitrary $\tilde{m}$, as specified in Definition 3.2, a distinguisher $\mathbf{Adv}_{\text{NCES}}$ continues as follows. Say it was given $(\text{PK}', r', \tilde{m}, c')$, then $\mathbf{Adv}_{\text{NCES}}$ emulates the role of the honest receiver and forwards the key PK' to $\mathcal{F}_{\text{SC}}^{\ell}$ and a valid key PK_R to SEN. It then completes the simulation as the simulator would do while plugging-in c' as the ciphertext that encrypts $\tilde{m}$ under PK'. The distinguisher then explains the sender's internal state as in the simulation. Clearly, any advantage in distinguishing the hybrid and simulated views can be reduced to breaking the indistinguishability of Π_{NCES} since $\mathbf{Adv}_{\text{NCES}}$ generates a view that is distributed according to one of these executions.

REC is corrupted after Step 2 is concluded. Upon receiving the receiver's message m from $\mathcal{F}_{\text{SC}}$, the simulator explains the receiver's internal state as follows. It explains PK_S for being the public key sent to $\mathcal{F}_{\text{SC}}^{\ell}$ and presents the randomness for generating $(\text{PK}_S, \text{SK}_S)$ and $(\text{PK}_R, \text{SK}_R^*)$ where SK_R^* is as defined below. It then explains c_S and c_R^* for being the ciphertexts received from $\mathcal{F}_{\text{SC}}^{\ell}$. Specifically, SK_R^* is defined such that $m''_R \leftarrow \mathsf{Dec}_{\text{SK}_R^*}(c_R^*)$ and $m''_R \oplus m'_S = m$.

Security is proven similarly to the case the sender is corrupted at the end. Namely, the receiver's view is comprised from three invocations of $\mathcal{F}_{\text{SC}}^{\ell}$ and ciphertexts c_S, c_R^*. Security follows due to the security of Π_{NCER}, where a distinguisher $\mathbf{Adv}_{\text{NCER}}$ that wishes to distinguish between a fake and a real ciphertext receives either $(\text{PK}, \text{SK}_R^*, c_R^*, m''_R)$ or $(\text{PK}, \text{SK}, c_R, m''_R)$. □

4. One-Sided Adaptively Secure OT

$\mathcal{F}_{\text{OT}}$, formally defined in Figure 2, is one of the fundamental functionalities in secure computation. In this section we design one-sided OT based on the [48] construction which is designed based on dual-mode PKE. We recall this definition first.

4.1. *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, 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 ciphertexts generated for either branch. Formally, a dual-mode PKE Π_{DUAL} with message space $\{0, 1\}^n$ consists of a tuple of probabilistic algorithms (Setup, dGen, dEnc, dDec, FindBranch, TrapKeyGen) specified as follows:

- $\mathsf{Setup}(1^n, \mu)$, given security parameter n and mode $\mu \in \{0, 1\}$, outputs (CRS, t). The CRS is a common string for the remaining algorithms, and t is a trapdoor value that is given to either FindBranch or TrapKeyGen, depending on the mode.
  The setup algorithms for messy and decryption modes are denoted by SetupMessy and SetupDecryption, respectively; namely $\mathsf{SetupMessy}(1^n) := \mathsf{Setup}(1^n, 0)$ and $\mathsf{SetupDecryption}(1^n) := \mathsf{Setup}(1^n, 1)$.
  All the remaining algorithms take CRS as their first input. For clarity, we usually omit it from their lists of arguments.
- $\mathsf{dGen}(\sigma)$, given a desired decryptable branch value σ, outputs (PK, SK) where PK is a public encryption key and SK is a corresponding secret decryption key for messages encrypted in branch σ. For our purpose, the secret decryption key denotes the randomness used by dGen which further induces the secret key.
- $\mathsf{dEnc}(\text{PK}, \beta, m)$, given a public key PK, a branch value $\beta \in \{0, 1\}$ and a message $m \in \{0, 1\}^n$, outputs a ciphertext c encrypted on branch β.
- $\mathsf{dDec}(\text{SK}, c)$, given a secret key SK and a ciphertext c, outputs a message $m \in \{0, 1\}^n$.
- $\mathsf{FindBranch}(t, \text{PK})$, given a trapdoor t and some (possibly even malformed) public key PK, outputs a branch value $\beta \in \{0, 1\}$ corresponding to a messy branch of PK.
- $\mathsf{TrapKeyGen}(t)$, given a trapdoor t, outputs $(\text{PK}, \text{SK}_0, \text{SK}_1)$, where PK is a public encryption key and SK_0, SK_1 are corresponding secret decryption keys for branches 0 and 1, respectively.

Definition 4.1. (*Dual-mode* PKE) A dual-mode PKE is a tuple of algorithms described above that satisfy the following properties:

1. **Completeness for decryptable branch.** For every $\sigma \in \{0, 1\}$, every $(\text{CRS}, t) \leftarrow \mathsf{Setup}(1^n, \sigma)$, every $(\text{PK}, \text{SK}) \leftarrow \mathsf{dGen}(\sigma)$, and every $m \in \{0, 1\}^n$, decryption is correct on branch σ, i.e. $\mathsf{dDec}_{\text{SK}}(\mathsf{dEnc}_{\text{PK}}(m, \sigma)) = m$.

2. **Indistinguishability of modes.** The CRS generated by $\mathsf{SetupMessy}$ and $\mathsf{SetupDecryption}$ is computationally indistinguishable, i.e. $\mathsf{SetupMessy}(1^n) \approx_c \mathsf{SetupDecryption}(1^n)$.
3. **Trapdoor extraction of a messy branch (messy mode).** For every $(\mathrm{CRS}, t) \leftarrow \mathsf{SetupMessy}(1^n)$ and every (possibly malformed) PK, $\mathsf{FindBranch}(t, \mathrm{PK})$ outputs a branch value $\beta \in \{0,1\}$ such that $\mathsf{dEnc}(\mathrm{PK}, \beta, \cdot)$ is messy. Namely, for every $m_0, m_1 \in \{0,1\}^n$, $\mathsf{dEnc}_{\mathrm{PK}}(m_0, \beta) \approx_s \mathsf{dEnc}_{\mathrm{PK}}(m_1, \beta)$.
4. **Trapdoor generation of keys decryptable on both branches (decryption mode).** For every $(\mathrm{CRS}, t) \leftarrow \mathsf{SetupDecryption}(1^n)$, $\mathsf{TrapKeyGen}(t)$ outputs $(\mathrm{PK}, \mathrm{SK}_0, \mathrm{SK}_1)$ such that for every $\sigma \in \{0,1\}$, $(\mathrm{PK}, \mathrm{SK}_\sigma) \approx_c \mathsf{dGen}(\sigma)$.

This notion was introduced by Peikert et al. in [48], which further showed how to instantiate it based on the DDH, QR and Learning with Errors hardness assumptions. We note that our definition is slightly different than the original [48] definition in the sense that algorithm dGen returns the randomness it uses in order to generate the secret key and the public key. This is required for equivocating the internal state of the receiver upon being adaptively corrupted (and is not required in the static setting). We note that this modification does not effect the former two instantiations in [48] and further arises in [31].

4.2. *Our Construction*

We now concentrate on designing our one-sided OT based on the [48] construction. Recall that in PVW protocol [48] the receiver first generates public and secret keys for branch σ. In response, the sender returns the encryptions of x_0 with respect to branch 0 and x_1 with respect to branch 1. Finally, the receiver decrypts the ciphertext for branch σ. Our one-sided OT requires the following modifications with respect to [48]. First, we fix the system parameters of the dual-mode PKE in decryption mode (where originally the system parameters are defined in either messy mode or decryption mode). Specifically, fixing the system parameters in decryption mode as we suggest here implies that in the security proof the simulator can extract the sender's input and equivocate the receiver's input. In addition, in order to be able to equivocate the sender's input in the security proof we encrypt its message using our one-sided NCE (see Section 3.1). Finally, in order to extract the receiver's bit, we ask the receiver to prove its behaviour using a special ZK PoK type for compound statements. In particular, the receiver proves that it knows a secret key for either branch 0 or 1, where the extracted branch value implies the extraction of σ. We design this proof so that it exploits the fact that the simulator knows the secret keys for both branches. Specifically, the proof allows the simulator to use both secret keys when emulating the receiver's role and later convince the adversary that it only knows the σth secret key. We denote these types of proofs by *witness equivocal* and explain them in more details in Section 6.2. More formally, the compound relation $\mathcal{R}_{\mathrm{LR}}$ we use in our OT protocol is combined of the following two relations:

$$\mathcal{R}_{\mathrm{ZERO}} = \big\{(\mathrm{PK}, \mathrm{SK}) \mid (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{dGen}(\mathrm{CRS}, 0)\big\},$$

where CRS are the system parameters and dGen is the key generation algorithm for the underlying dual-mode system. Similarly, we define $\mathcal{R}_{\text{ONE}}$ for public keys generated with respect to branch 1. Then, the compound relation $\mathcal{R}_{\text{LR}}$ is defined by

$$\mathcal{R}_{\text{LR}} = \big\{((\text{PK}_0, \text{PK}_1), (\text{SK}, \sigma)) \mid (\text{PK}_\sigma, \text{SK}) \leftarrow \mathsf{dGen}(\text{CRS}, \sigma)\big\}.$$

In Section 6.2 we consider two instantiations of $\mathcal{R}_{\text{LR}}$ based on the DDH and QR assumptions.

We are now ready to describe our protocol in details. Formally, we denote the dual-mode PKE of [48] by $\Pi_{\text{DUAL}} = (\mathsf{SetupMessy}, \mathsf{SetupDecryption}, \mathsf{dGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{FindBranch}, \mathsf{TrapKeyGen})$ and describe our construction in the $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{LR}})$-hybrid model, where $\mathcal{F}_{\text{LR}}$ is the ideal functionality that verifies the relation $\mathcal{R}_{\text{LR}}$ and returns `Accept` to the verifier only if the witness provided by the prover meets the definition of $\mathcal{R}_{\text{LR}}$. More formally,

Protocol 2. (**One-sided OT** (Π_{OT}))

- **Inputs:** *Sender* SEN *has* $x_0, x_1 \in \{0,1\}^n$ *and receiver* REC *has* $\sigma \in \{0,1\}$.
- **CRS:** CRS *such that* $(\text{CRS}, t) \leftarrow \mathsf{SetupDecryption}$.
- **The Protocol:**
  1. REC *generates* $(\text{PK}, \text{SK}) \leftarrow \mathsf{dGen}(\text{CRS}, \sigma)$ *and sends* PK. REC *further calls* $\mathcal{F}_{\text{LR}}$ *with* $(\text{PK}, (\text{SK}, \sigma))$.
  2. *Upon receiving* `Accept` *from* $\mathcal{F}_{\text{LR}}$ *and* PK *from* REC, SEN *generates* $c_0 \leftarrow \mathsf{dEnc}_{\text{PK}}(x_0, 0)$ *and* $c_1 \leftarrow \mathsf{dEnc}_{\text{PK}}(x_1, 1)$. SEN *sends* $\mathcal{F}_{\text{SC}}$ *the messages* $(\mathsf{send}, sid, \text{SEN}, c_0)$ *and* $(\mathsf{send}, sid', \text{SEN}, c_1)$.
  3. *Upon receiving* $(\mathsf{send}, sid, \text{SEN}, c_0)$ *and* $(\mathsf{send}, sid', \text{SEN}, c_1)$ *from* $\mathcal{F}_{\text{SC}}$, REC *outputs* $\mathsf{dDec}_{\text{SK}}(c_\sigma)$.

Note that our protocol implies OT with constant number of PKE operations for sender's input space $\{0,1\}^n$ if the underlying one-sided NCE requires a constant overhead for message space $\{0,1\}^n$, as holds for our construction from Section 3.1.

Theorem 4.2. *Assume the existence of one-sided NCE and dual-mode PKE. Then Protocol 2 UC realizes* $\mathcal{F}_{\text{OT}}$ *in the* $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{LR}})$*-hybrid model in the presence of one-sided adaptive malicious adversaries.*

Proof. Let ADV be a probabilistic polynomial-time malicious adversary attacking Protocol 2 by adaptively corrupting one of the parties. We construct an adversary SIM for the ideal functionality $\mathcal{F}_{\text{OT}}$ such that no environment ENV distinguishes with a non-negligible probability whether it is interacting with ADV in the hybrid setting or with SIM in the ideal setting. We recall that SIM interacts with the ideal functionality $\mathcal{F}_{\text{OT}}$ and the environment ENV. We refer to the interaction of SIM with $\mathcal{F}_{\text{OT}}$ and ENV as the external interaction. The interaction of SIM with the simulated ADV is the internal interaction. Upon computing $(\text{CRS}, t) \leftarrow \mathsf{SetupDecryption}(1^n)$, SIM proceeds as follows:

Simulating the communication with ENV. Every input value received by the simulator from ENV is written on ADV's input tape. Likewise, every output value written

by ADV on its output tape is copied to the simulator's output tape (to be read by its environment ENV).

SEN is corrupted at the outset of the protocol. SIM begins by activating ADV and emulates the receiver by running $(\mathrm{PK}, \mathrm{SK}_0, \mathrm{SK}_1) \leftarrow \mathsf{TrapKeyGen}(t)$. It then sends PK to the adversary on behalf of the honest receiver and an `Accept` message on behalf of $\mathcal{F}_{\mathrm{LR}}$. Next, upon receiving from ADV the messages $(\mathsf{send}, sid, \mathrm{SEN}, c_1)$ and $(\mathsf{send}, sid', \mathrm{SEN}, c_1)$, sent to $\mathcal{F}_{\mathrm{SC}}$, SIM computes $x_0 = \mathsf{dDec}_{\mathrm{SK}_0}(c_0)$ and $x_1 = \mathsf{dDec}_{\mathrm{SK}_1}(c_1)$. It then forwards $\mathcal{F}_{\mathrm{OT}}$ the message $(\mathsf{sender}, sid, x_0, x_1)$ and completes the execution playing the role of the receiver using an arbitrary σ.

Note that, in contrast to the hybrid execution where the receiver uses its real input σ as input to dGen in order to create public/secret keys pair, the simulator does not know σ and thus creates the keys using $\mathsf{TrapKeyGen}$. Nevertheless, when the CRS is set in a decryption mode the left public key is statistically indistinguishable from the right public key. Furthermore, the keys $(\mathrm{PK}, \mathrm{SK}_i)$ that are generated by $\mathsf{TrapKeyGen}$ are statistically close to the keys generated by dGen with input bit i. This implies that the hybrid and simulated executions are statistically close.

REC is corrupted at the outset of the protocol. SIM begins by activating ADV and receives a public key PK and a witness SK_σ as the input to $\mathcal{F}_{\mathrm{LR}}$. Given SK_σ, SIM checks if PK is the left or the right key and uses it to extract the receiver's input σ. If the adversary's message is invalid then SIM aborts, sending $\perp$ to $\mathcal{F}_{\mathrm{OT}}$. Otherwise, it forwards the message $(\mathsf{receiver}, sid, \sigma)$, receiving back x_σ. Finally, SIM computes the sender's message using x_σ and an arbitrary $x'_{1-\sigma}$.

Note that unlike in the hybrid execution, the simulator uses an arbitrary $x'_{1-\sigma}$ instead of the real $x_{1-\sigma}$. However, a decryption mode implies computational privacy of $x_{1-\sigma}$. This follows from the same proof in [48]. Therefore, the hybrid view is also computationally indistinguishable from the simulated view as in the static setting.

If none of the above corruption cases occur, SIM invokes $(\mathrm{PK}, \mathrm{SK}_0, \mathrm{SK}_1) \leftarrow \mathsf{TrapKeyGen}(t)$ and emulates the message PK sent to the sender.

SEN is corrupted between Steps 1 and 2. SIM trivially explains the sender's internal state since SEN did not compute any message so far. The simulator completes the simulation by playing the role of REC using an arbitrary bit σ as in the case when the sender is corrupted at the outset of the execution.

Indistinguishability for this case follows similarly to the prior corruption case when SEN is corrupted at the outset of the execution since the simulator uses the same simulation strategy as above. Namely, the adversary's simulated view is identically distributed in both simulation cases. This is because this view only contains the public key which is statically independent of σ in a decryption mode.

REC corrupted between Steps 1 and 2. Upon corrupting the receiver SIM obtains (sid, σ, x_σ) from $\mathcal{F}_{\mathrm{OT}}$ and explains the receiver's internal state as follows. It first explains SK_σ as the witness given to $\mathcal{F}_{\mathrm{LR}}$ and PK as the outcome of $\mathsf{dGen}(\mathrm{CRS}, \sigma)$. The simulator then completes the simulation playing the role of the honest sender with x_σ and an arbitrary $x'_{1-\sigma}$.

Indistinguishability for this case follows similarly to the prior corruption case since the simulator did not emulate the sender's message yet.

If none of the above corruption cases occur then SIM chooses two arbitrary inputs x'_0, x'_1 for the sender and encrypts them using the dual-mode encryption. Denote these ciphertexts by c'_0, c'_1. SIM emulates the sender that sends these ciphertexts using $\mathcal{F}_{\text{SC}}$.

SEN is corrupted after Step 2. Upon corrupting the sender, SIM obtains (sid, x_0, x_1) from $\mathcal{F}_{\text{OT}}$. It then explains the sender's internal state as follows. It first computes c_0, c_1 that, respectively, encrypt x_0 and x_1. It then explains c_0 and c_1 as being sent to the receiver using $\mathcal{F}_{\text{SC}}$.
Indistinguishability follows as in the prior corruption case of the sender since the one-sided non-committing channel enables the simulator to "rewind" the simulation back, assuming that the sender is corrupted before simulating its message. Therefore, the same simulation strategy as before, of emulating the sender's incoming message using an arbitrary bit σ, works here as well.

REC is corrupted after Step 2. Upon corrupting the receiver, SIM obtains REC's input and output (sid, σ, x_σ) from $\mathcal{F}_{\text{OT}}$. It then explains the receiver's internal state as follows. It first explains SK_σ as the witness given to $\mathcal{F}_{\text{LR}}$ and PK as the outcome of $\mathsf{dGen}(\text{CRS}, \sigma)$. Finally, it explains the output of $\mathcal{F}_{\text{SC}}$ as c_0, c_1, so that c_σ is a valid encryption of x_σ under PK and $c_{1-\sigma}$ is an encryption of an arbitrary input.
Indistinguishability follows similarly to a static corruption case of the receiver.

□

Concrete instantiations. A DDH-based instantiation implies that the CRS is a Diffie–Hellman tuple (g_0, g_1, h_0, h_1) and the trapdoor is $\log_{g_0} g_1$. Moreover, the witness equivocal ZK PoK is realized with the statement and witness relation $\big(((g_0 h_0, g_\sigma^r h_\sigma^r), (g_1 h_1, g_\sigma^r h_\sigma^r)), r\big)$ such that $\text{PK} = (g_\sigma^r, h_\sigma^r)$, $\text{SK} = r$ and $r \leftarrow \mathbb{Z}_p$. An additional QR-based instantiation implies that the CRS is a quadratic residue y from $\mathbb{Z}_N^*$, whereas the trapdoor is s such that $y = s^2 \bmod N$, where N is an RSA composite. Finally, the ZK PoK is realized with the statement and witness $\big((y \cdot \text{PK}, \text{PK}), r\big)$ such that $\text{PK} = r^2/y^\sigma$, $\text{SK} = r$ and $r \leftarrow \mathbb{Z}_N^*$.

5. One-Sided Adaptively Secure Computation

In the following, we demonstrate the feasibility of one-sided adaptively secure protocols in the plain model with overhead that grows linearly with the static communication complexity (Section 5.1) and in the presence of malicious adversaries in the UC setting (Section 5.2). Our generic compiler requires from the parties to communicate with each other using non-committing secure channels that achieve one-sided security (as formally modelled by $\mathcal{F}_{\text{SC}}$ in Section 3.1). This transformation demonstrates the feasibility of semi-honest (UC) security in the presence of one-sided attacks with the specified overhead. Moreover, in the plain model our compiler implies one-sided malicious security with the same overhead. We recall that it is not clear how to compile static maliciously secure protocols into the one-sided setting while preserving UC security, which is due to the additional setup. We thus build a concrete maliciously one-sided secure construction by modifying the [42] static Yao based protocol. Our construction is constant round UC secure which requires a number of PKE operations that depend linearly on the circuit's size times a statistical parameter.

5.1. *From Static to One-Sided Security with no Setup*

In this section, we present a simple compiler that achieves one-sided adaptive security given a static protocol. Loosely speaking, given a protocol Π our compiler encrypts the communication of Π using one-sided NCE. The intuition behind the security proof follows from the fact that one-sided NCE allows to equivocate the communication until the point corruption takes place. Specifically, whenever corruption occurs the simulator can essentially "rewind" back to the outset of the execution and recompute the entire protocol transcript using the corrupted party's input, while pretending that this party was statically corrupted and that it follows the protocol execution until the corruption point. It then presents the adversary the generated transcript. As explained in [31], a subtlety arises in the malicious setting where it must hold that the simulator submits to the ideal functionality the exact same input used by the corrupted party (whom we assume to follow the protocol). This is because the ideal functionality cannot accept a different input, as it already computed the outputs using the honest inputs. This property is denoted by *input preserving*; see [31] for more details. We are now ready to prove the following theorem, our proof follows similarly to the proof from [31] which demonstrates a compiler from semi-adaptive security into fully adaptive.

Theorem 5.1. *Let f be a deterministic same-output functionality and let Π be a statically secure protocol with input preserving simulation. Then protocol Π', in which the parties run Π but only communicate with each other using one-sided NCE is one-sided adaptively secure.*

Proof. Let ADV be a malicious PPT adversary attacking Protocol Π. We construct a simulator SIM$'$ such that no PPT ENV distinguishes the real and the simulated views, i.e. the following computational indistinguishability holds

$$\{\mathbf{OIDEAL}_{f,\text{SIM}',\text{ENV}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \approx_c \{\mathbf{OREAL}_{\Pi',\text{ADV},\text{ENV}}(n, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,n\in\mathbb{N}}.$$

Simulator SIM$'$ internally invokes a copy of the real adversary ADV and externally interacts with the ideal functionality f and environment ENV. We neglect static corruptions (since these are covered by the statically secure protocol Π) and describe the strategy of SIM$'$ for the following two corruption cases: **(1)** Simulation with no corruptions. **(2)** Simulation when the first adaptive corruption takes place. Our proof follows in the $\mathcal{F}_{\text{SC}}$-hybrid model. As in the [31] 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 [31], 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, simulator SIM$'$ initializes a copy of the static simulator SIM that completes the setup phase (if required). Next,

relying on the fact that Π is well structured, 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 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_1 and the other identity by P_2. Then SIM' receives from f the corrupted party's input x_1 and (possible) output y_1. Let ADV_1 denote a PPT adversary that corrupts party P_1 right after the setup phase, yet uses the real input of P_1 in the execution, and let SIM_1 denote the simulator for that adversary. Then SIM' invokes SIM_1 until party P_1 is corrupted while playing the role of functionality f. Note that by the input preserving property SIM_1 can only submits the input x_1. If it expects to receive an output, SIM' forwards SIM_1 the value y_1. Once the simulation reaches the point where P_1 is corrupted, SIM' takes the internal state of SIM_1 and sets it as the internal state of P_1.

Indistinguishability follows easily since in the hybrid $\mathcal{F}_{\text{SC}}$-model the adversary's views are identical. □

One implication of our compiler is that it implies the feasibility of constant rounds one-sided security with $O(|C|)$ symmetric key operations and $O(|C|)$ public key operations by applying our compiler to the semi-honest two-party protocol from [41,51]. More formally,

Theorem 5.2. (One-sided semi-honest) *Let f be a deterministic same-output functionality and assume the assumptions specified in [41] and one-sided NCE. Then there exists a constant rounds protocol with semi-honest one-sided adaptive security that UC realizes f with $O(|C|)$ public key operations.*

In addition, since our compiler is applied to any protocol in the plain model, even with malicious security, by combining our result with [42] we obtain the following theorem,

Theorem 5.3. (One-sided malicious) *Let f be a deterministic same-output functionality and assume the assumptions specified in [42] and one-sided NCE. Then there exists a constant rounds protocol with malicious one-sided adaptive security that realizes f with $O(s|C|)$ public key operations.*

In the following section, we present an alternative malicious two-party construction with one-sided security that is also based on [42] yet obtains UC security.

5.2. *UC Security against Malicious Adversaries*

As mentioned in the introduction, it is insufficient to simply encrypt the communication using one-sided NCE as in the semi-honest setting, due to the additional CRS setup which might be based on which party is corrupted. Instead, we take a different approach and directly compile the [42] protocol (which is based on the DDH hardness assumption) into a protocol that achieves one-sided malicious security in the UC setting, proving the feasibility of UC constant round with malicious one-sided security. The [42] protocol

relies on a cut-and-choose technique for which P_0 sends s garbled circuits and later open half of them (aka, *check circuits*) by the choice of P_1. This ensures that with very high probability the majority of the unopened circuits (aka, *evaluation circuits*) are valid. The cut-and-choose OT primitive of [42] allows P_1 to choose a secret random subset $\mathcal{J}$ of size $s/2$ for which it learns both keys for each input wire that corresponds to the check circuits, and the individual keys associated with its input with respect to the evaluation circuits.

In order to ensure that P_0 hands P_1 consistent input keys for all the circuits, the [42] protocol ensures that the keys associated with P_0's input are obtained via a Diffie–Hellman pseudorandom synthesizer [46]. Namely, P_0 chooses values $g^{a_1^0}, g^{a_1^1}, \ldots, g^{a_n^0}, g^{a_n^1}$ and $g^{c_1}, \ldots, g^{c_s}$, where n is the input/output length, s is the cut-and-choose parameter and g is a generator of a prime-order group $\mathbb{G}$. So that the pair of keys associated with the ith input of P_0 in the jth circuit is $(g^{a_i^0 c_j}, g^{a_i^1 c_j})$.[3] Given values $\{g^{a_i^0}, g^{a_i^1}, g^{c_j}\}$ and any subset of keys associated with P_0's input, the remaining keys associated with its input are pseudorandom by the DDH assumption. Furthermore, when the keys are prepared this way P_0 can efficiently prove that it used the same input for all circuits. P_1 then evaluates the evaluation circuits and takes the majority value for the final output. In Section 5.2.1 we demonstrate how to adapt the cut-and-choose OT protocol into the one-sided setting using the building blocks introduced in this paper. This requires dealing with new subtleties regarding the system parameters and the ZK proofs.

5.2.1. *One-sided Single Choice Cut-and-Choose OT*

We describe next the single choice cut-and-choose OT functionality $\mathcal{F}_{\text{CCOT}}$ from [42] and present a protocol that implements this functionality with UC one-sided malicious security. In Sect. 5.2.2, we briefly describe our batch single choice cut-and-choose OT construction using a single choice cut-and-choose OT, which is used as a building block in the two-party protocol. Formally, $\mathcal{F}_{\text{CCOT}}$ is defined as follows,

1. **Inputs:**
   (a) SEN inputs a vector of pairs $\{(x_0^j, x_1^j)\}_{j=1}^s$.
   (b) REC inputs a bit σ and a set of indices $\mathcal{J} \subset [s]$ of size exactly $s/2$.
2. **Output:** If $\mathcal{J}$ is not of size $s/2$, then SEN and REC receive $\perp$ as output. Otherwise,
   (a) For all $j \in \mathcal{J}$, REC obtains the pair (x_0^j, x_1^j).
   (b) For all $j \notin \mathcal{J}$, REC obtains x_σ^j.

This functionality is implemented in [42] using s instances of the DDH-based OT from [48], where the receiver generates the system parameters in a decryption mode for the $s/2$ indices that correspond to $\mathcal{J}$, whereas the remaining system parameters are generated in a messy mode. Specifically, the decryption mode trapdoor enables the receiver to learn both of the sender's inputs for the instances within $\mathcal{J}$. This idea is

[3] The actual key pair used in the circuit garbling is derived from $(g^{a_i^0 c_j}, g^{a_i^1 c_j})$ using an extractor. A universal hash function is used in [42] for this purpose, where the seeds for the function are picked by P_0 before it knows $\mathcal{J}$.

coupled with two proofs that ensure the correctness of the receiver's computations. Our first step towards making the [42] construction one-sided adaptively secure is to invoke s instances of our one-sided OT where all system parameters are generated in a decryption mode.

Similarly to [42], our construction includes two phases: a *setup phase* and a *transfer phase*. In the setup phase, the receiver generates the system parameters in a decryption mode for the $s/2$ instances that correspond to indices in $\mathcal{J}$, while the remaining system parameters are generated in the same mode but in a way that does not allow REC to learn the trapdoor. This is obtained by fixing two random generators g_0, g_1 in the CRS, so that the CRS for all one-sided OT instances $j \notin \mathcal{J}$ include g_0 and a power of g_1, whereas the CRS for OT instances $j \in \mathcal{J}$ includes g_0 and an element that is a power of g_0. This ensures that REC does not know $\log_{g_0} g_1$ which is the decryption mode trapdoor for $j \notin \mathcal{J}$. To ensure correctness, REC proves that it knows the discrete logarithm of the second element with respect to g_1 of at least $s/2$ pairs. We denote this relation by $\mathcal{R}_{\mathrm{DL,COMP}(s,s/2)}$ and present discuss more details in Section 6.

In the transfer phase, the receiver uses these system parameters to create a public/secret key pair for the OT instances not in $\mathcal{J}$. For the rest of the OT executions, the receiver invokes the $\mathsf{TrapKeyGen}$ algorithm of the dual-mode PKE and obtains a public key together with a pair of secret keys that enable it to decrypt both of the sender's inputs. The receiver then proves that it used the same input σ for all OT instances using a relation denoted by $\mathcal{R}_{\mathrm{DH,OR}(s)}$. Formally, let the DDH-based dual-mode PKE of [48] be specified by the set of algorithms $\Pi_{\mathrm{DUAL}} = (\mathsf{SetupMessy}, \mathsf{SetupDecryption}, \mathsf{dGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{FindBranch}, \mathsf{TrapKeyGen})$. We denote our one-sided OT by Π_{CCOT} and present it in the $(\mathcal{F}_{\mathrm{SC}}, \mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}, \mathcal{F}_{\mathrm{DH,OR}(s)})$-hybrid model (where $\mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}, \mathcal{F}_{\mathrm{DH,OR}(s)}$ ideally implement the corresponding relations $\mathcal{R}_{\mathrm{DL,COMP}(s,s/2)}, \mathcal{R}_{\mathrm{DH,OR}(s)}$).

Protocol 3. (**One-sided adaptive single choice cut-and-choose OT** ($\boldsymbol{\Pi_{\mathrm{CCOT}}}$))

- **Inputs:** SEN *inputs a vector of pairs* $\{(x_0^i, x_1^i)\}_{i=1}^s$ *and* REC *inputs a bit* σ *and a set of indices* $\mathcal{J} \subset [s]$ *of size exactly* $s/2$.
- **Auxiliary Inputs:** *Both parties hold a security parameter* 1^n *and* $\mathbb{G}$, p, *where* $\mathbb{G}$ *is an efficient representation of a group of order* p *and* p *is of length* n.
- **CRS:** *The CRS consists of a pair of random group elements* g_0, g_1 *from* $\mathbb{G}$.
- **Setup phase:**
  1. REC *chooses a random* $x_j \in \mathbb{Z}_p$ *and sets* $g_1^j = g_0^{x_j}$ *for all* $j \in \mathcal{J}$ *and* $g_1^j = g_1^{x_j}$ *otherwise.*
     For all j, REC *chooses a random* $y_j \in \mathbb{Z}_p$ *and sets* $\mathrm{CRS}_j = \left(g_0, g_1^j, h_0^j = (g_0)^{y_j}, h_1^j = (g_1^j)^{y_j}\right)$.
     Finally, for all $j \in \mathcal{J}$, REC *stores trapdoor* $t_j = x_j$. *It then sends* $\{\mathrm{CRS}_j\}_{j=1}^s$ *to* SEN.
  2. REC *calls* $\mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}$ *with* $(\{g_1, g_1^j\}_{j=1}^s, \{x_j\}_{j\in\mathcal{J}})$ *to prove the knowledge of the discrete logarithms of* $s/2$ *values within the second element in* $\{\mathrm{CRS}_j\}_j$ *and with respect to* g_1.
- **Transfer phase** (repeated in parallel for all j):
  1. *For all* $j \notin \mathcal{J}$, REC *computes* $(\mathrm{PK}_j, \mathrm{SK}_j) = ((g_j, h_j), r_j) \leftarrow \mathsf{dGen}(\mathrm{CRS}_j, \sigma)$.

For all $j \in \mathcal{J}$, REC *computes* $(\mathrm{PK}_j, \mathrm{SK}_j^0, \mathrm{SK}_j^1) = ((g_j, h_j), r_j, r_j/t_j) \leftarrow \mathsf{TrapKeyGen}(\mathrm{CRS}_j, t_j)$.
Finally, REC *sends the set* $\{\mathrm{PK}_j\}_{j=1}^s$ *and stores the secret keys.*

2. REC *calls* $\mathcal{F}_{\mathrm{DH,OR}(s)}$ *with input* $((\{(g_0, h_0^j, g_j, h_j)\}_{j=1}^s, \{(g_1^j, h_1^j, g_j, h_j)\}_{j=1}^s), \{r_j\}_{j=1}^s)$ *to prove that all the tuples in one of the sets* $\{(g_0, h_0^j, g_j, h_j)\}_{j=1}^s$ *or* $\{(g_1^j, h_1^j, g_j, h_j)\}_{j=1}^s$ *are DH tuples.*
3. *For all* j, SEN *generates* $c_0^j \leftarrow \mathsf{dEnc}_{\mathrm{PK}_j}(x_0^j, 0)$ *and* $c_1^j \leftarrow \mathsf{dEnc}_{\mathrm{PK}_j}(x_1^j, 1)$. *Let* $c_0^j = (c_{00}^j, c_{01}^j)$ *and* $c_1^j = (c_{10}^j, c_{11}^j)$. SEN *calls* $\mathcal{F}_{\mathrm{SC}}$ *with* $(\mathsf{send}, sid, \mathrm{SEN}, c_{01}^j)$ *and* $(\mathsf{send}, sid, \mathrm{SEN}, c_{11}^j)$ *and sends* c_{00}^j, c_{10}^j *to* REC.

- **Output:** *Upon receiving* $(\mathsf{send}, sid, \mathrm{SEN}, c_{01}^j)$ *and* $(\mathsf{send}, sid, \mathrm{SEN}, c_{11}^j)$ *from* $\mathcal{F}_{\mathrm{SC}}$, *and* c_{00}^j, c_{10}^j, REC *defines ciphertexts* $c_0^j = (c_{00}^j, c_{01}^j)$ *and* $c_1^j = (c_{10}^j, c_{11}^j)$ *for all* $j \notin \mathcal{J}$.
  1. REC *outputs* $x_\sigma^j \leftarrow \mathsf{dDec}_{\mathrm{SK}_j}(c_\sigma^j)$.
  2. REC *outputs* $(x_0^j, x_1^j) \leftarrow (\mathsf{dDec}_{\mathrm{SK}_j^0}(c_0^j), \mathsf{dDec}_{\mathrm{SK}_j^1}(c_1^j))$.

Theorem 5.4. *Assume that the DDH assumption is hard in* $\mathbb{G}$. *Then Protocol 3 UC realizes* $\mathcal{F}_{\mathrm{CCOT}}$ *in the* $(\mathcal{F}_{\mathrm{SC}}, \mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}, \mathcal{F}_{\mathrm{DH,OR}(s)})$*-hybrid model in the presence of one-sided malicious adversaries.*

Proof. Let ADV be a probabilistic polynomial-time malicious adversary attacking Protocol 3 by adaptively corrupting one of the parties. We construct an adversary SIM for the ideal functionality of a single choice cut-and-choose OT $\mathcal{F}_{\mathrm{CCOT}}$ such that no environment ENV distinguishes with a non-negligible probability whether it is interacting with ADV in the hybrid setting or with SIM in the ideal setting. We recall that SIM interacts with the ideal functionality and the environment ENV. We refer to the interaction of SIM with the ideal functionality $\mathcal{F}_{\mathrm{CCOT}}$ and ENV as the external interaction. The interaction of SIM with the simulated ADV is the internal interaction. We describe the simulator's strategy for all corruption cases. SIM begins by creating a CRS (g_0, g_1) and storing $x = \log_{g_0} g_1$. It then proceeds as follows:

Simulating the communication with ENV. Every input value received by the simulator from ENV is written on ADV's input tape. Likewise, every output value written by ADV on its output tape is copied to the simulator's output tape (to be read by its environment ENV).

The sender is corrupted at the onset of the protocol. SIM begins by activating ADV and emulates the receiver as follows. In the setup phase, it picks s system parameters in a decryption mode in which it knows their trapdoors. Namely for each $j \in [s]$, it creates $\mathrm{CRS}_j = (g_0, g_1^j, h_0^j, h_1^j)$ where $g_1^j = (g_0)^{x_j}$, $h_0^j = (g_0)^{y_j}$ and $h_1^j = (g_1^j)^{y_j} = (g_0^{x_j})^{y_j}$ for random x_j's and y_j's, and records the trapdoor $t_j = x_j$. The simulator further computes $x_j' = \log_{g_1} g_1^j$ for all j using the knowledge of $x = \log_{g_0} g_1$. It then sends the adversary the system parameters, chooses an arbitrary set $\mathcal{J}'$ of size $s/2$ and sends `Accept` to ADV on behalf of $\mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}$

for the statement $\{g_1, g_1^j\}_{j=1}^s$. Note that the simulator knows the discrete logarithms for each pair (g_1, g_1^j) within the statement.

In the transfer phase, the simulator invokes $\mathsf{TrapKeyGen}$ for all $j \in [s]$ and computes $(\mathrm{PK}_j, \mathrm{SK}_j^0, \mathrm{SK}_j^1) = ((g_j, h_j), r_j, r_j/t_j) \leftarrow \mathsf{TrapKeyGen}(\mathrm{CRS}_j, t_j)$ for $j \in [s]$, and sends the public keys to SEN. It further sends `Accept` to ADV on behalf of $\mathcal{F}_{\mathrm{DH,OR}(s)}$. Upon receiving ADV's message, SIM extracts the sender's input (x_0^j, x_1^j) using $\mathrm{SK}_j^0, \mathrm{SK}_j^1$ for every $j \in [s]$ and sends it to the ideal functionality $\mathcal{F}_{\mathrm{CCOT}}$.

Note that the adversary's views differ only with respect to the ZK statements, since in a decryption mode the receiver's bit is perfectly hidden as well as the subset picked by the receiver. Now, since the proofs are run via ideal calls the simulated and hybrid views are statistically close.

The receiver is corrupted at the onset of the protocol. SIM begins by activating ADV and emulates the honest sender as well as the ideal functionalities $\mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}$ and $\mathcal{F}_{\mathrm{DH,OR}(s)}$. It extracts $\mathcal{J}$ and σ from the inputs to these functionalities and sends them to $\mathcal{F}_{\mathrm{CCOT}}$, receiving back (x_0^j, x_1^j) for all $j \in \mathcal{J}$ and x_σ^j for all $j \notin \mathcal{J}$. SIM chooses an arbitrary $x_{1-\sigma}^j$ for all $j \notin \mathcal{J}$ and emulates the role of SEN using inputs (x_0^j, x_1^j) for all $j \in [s]$.

Note that the difference between the simulated and hybrid views is with respect to inputs $x_{1-\sigma}^j$ for all $j \notin \mathcal{J}$ for which the simulator uses arbitrary values. Indistinguishability is implied by the privacy of the dual-mode PKE when a left ciphertext is computed with a right key (or vice versa).

If none of the parties get corrupted at the onset of the protocol execution, SIM plays the role of the honest receiver in the setup phase using an arbitrary subset $\mathcal{J}'$, and $\mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}$. Note that SIM knows all the witnesses for the proof in the setup phase (i.e. the discrete logarithms of $\{g_1^j\}_{j=1}^s$ with respect to g_1 for all j values). It can thus later equivocate the proof with respect to the real set $\mathcal{J}$. SIM further plays the role of the honest receiver in the transfer phase using an arbitrary σ', and $\mathcal{F}_{\mathrm{DH,OR}(s)}$. Specifically the simulator simulates the receiver by invoking $\mathsf{TrapKeyGen}$ for all $j \in [s]$, computing $(\mathrm{PK}_j, \mathrm{SK}_j^0, \mathrm{SK}_j^1) = ((g_j, h_j), r_j, r_j/t_j) \leftarrow \mathsf{TrapKeyGen}(\mathrm{CRS}_j, t_j)$. It then sends the public keys to SEN and an `Accept` message on behalf of $\mathcal{F}_{\mathrm{DH,OR}(s)}$. Note that SIM knows witnesses for both sub-statements $\{(g_0, h_0^j, g_j, h_j)\}_{j=1}^s$ and $\{(g_1^j, h_1^j, g_j, h_j)\}_{j=1}^s$, which equal $\{r_j\}_{j=1}^s$ for the first set and $\{r_j/t_j\}_{j=1}^s$ for the second set.

The sender is corrupted between Steps 2 and 3. Upon corrupting SEN, SIM explains the internal state of the sender by honestly presenting the randomness used so far on the sender's behalf. Finally, SIM completes the execution in the transfer phase by playing the role of the receiver using an arbitrarily chosen σ'. Indistinguishability follows due to the same argument as in the previous corruption case since the simulator follows the same strategy relative to the sender.

The receiver is corrupted between Steps 2 and 3. Upon corrupting REC, SIM receives $\mathcal{J}$ and σ from $\mathcal{F}_{\mathrm{CCOT}}$, as well as (x_0^j, x_1^j) for all $j \in \mathcal{J}$ and x_σ^j for all $j \notin \mathcal{J}$. It then explains the internal state of REC as follows. It first explains the witness for the ZK PoK functionality $\mathcal{F}_{\mathrm{DL,COMP}(s,s/2)}$ as the discrete logarithms of $\{g_1^j\}_{j \notin \mathcal{J}}^s$

with respect to g_1. It also explains the witness for $\mathcal{F}_{\mathrm{DH,OR}(s)}$ as the witness for the σth set. Finally, it plays the role of the sender as in the previous corruption case. Indistinguishability follows similarity to the previous corruption case due to the security of the dual-mode PKE and the fact that the simulator follows the same strategy.

If none of the parties is corrupted until now, Sim plays the role of the sender in the transfer phase using arbitrary $(x_0'^j, x_1'^j)$ for all $j \in [s]$.

The sender is corrupted after Step 3 is concluded. Upon corrupting SEN, Sim receives (x_0^j, x_1^j) for all $j \in [s]$ from $\mathcal{F}_{\mathrm{CCOT}}$. It then explains the internal state of SEN as in the previous corruption case and further explains the inputs to $\mathcal{F}_{\mathrm{SC}}$ as ciphertexts that encrypt the real inputs. Indistinguishability follows from the fact that the receiver's input is statistically hidden given the public keys.

The receiver is corrupted after Step 3 is concluded. Upon corrupting REC, Sim receives $\mathcal{J}, \sigma$, from $\mathcal{F}_{\mathrm{CCOT}}$ as well as (x_0^j, x_1^j) for all $j \in \mathcal{J}$ and x_σ^j for all $j \notin \mathcal{J}$. It then explains the internal state of REC as in the previous corruption case and further explains the messages received from $\mathcal{F}_{\mathrm{SC}}$ as the encryptions of $\{x_0^j, x_1^j\}_{j\in\mathcal{J}}$ and $\{x_\sigma^j\}_{j\notin\mathcal{J}}$. Indistinguishability follows as above. □

5.2.2. *Malicious One-Sided Adaptively Secure Two-Party Computation*

First, we remark that the single choice cut-and-choose protocol from Section 5.2.1 is executed for every input bit of P_1 in the main two-party protocol, but with respect to *the same* set $\mathcal{J}$. In order to ensure that the same $\mathcal{J}$ is indeed used the parties engage in a *batch single choice cut-and-choose OT* where a single setup phase is run first, followed by n parallel invocations of the transfer phase. Note that CRS and the set $\mathcal{J}$ are fixed in the setup phase and remain the same for all n parallel invocations of the transfer phase. The same modifications can be carried out relative to our single choice one-sided OT. We denote the batch functionality by $\mathcal{F}_{\mathrm{CCOT}}^{\mathrm{BATCH}}$. Next, in order to achieve one-sided malicious security we modify the [42] two-party protocol by using our batch single choice cut-and-choose OT with one-sided security as a building block. In addition, we use our one-sided NCE from Section 3.1 in order to encrypt the garbled circuits, as well as the input keys that depend on P_0's inputs. We are now ready to formally describe our protocol Π_f^{MAL}.

Protocol 4. *(*One-sided adaptively secure malicious Yao (Π_f^{MAL}))

- **Inputs:** *P_0 has $x_0 \in \{0,1\}^n$ and P_1 has $x_1 \in \{0,1\}^n$. Let $x_0 = x_0^1, \ldots, x_0^n$ and $x_1 = x_1^1, \ldots, x_1^n$.*
- **Auxiliary Input:** *A boolean circuit C such that for every $x_0, x_1 \in \{0,1\}^n$, $C(x, y) = f(x, y)$ where $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Furthermore, we assume that C is such that if a circuit-output wire leaves some gate, then the gate has no other wires leading from it into other gates (i.e. no circuit-output wire is also a gate-output wire). Likewise, a circuit-input wire that is also a circuit-output wire enters no gates.*
  Convention: *Unless specified differently, $i \in [n]$. We further assume that the gates of circuit C induce a topological sort.*

- **The Protocol:**
  1. **Garbled circuit computation.** *P_0 constructs s independent garbled circuits for C as follows:*
     (a) *P_0 picks n pairs of random values $((a_1^0, a_1^1), \ldots, (a_n^0, a_n^1)) \in \mathbb{Z}_q$ and $c_1, \ldots, c_s \in_R \mathbb{Z}_q$.*
     (b) *Let $w_1, \ldots, w_n$ be the input wires corresponding to P_0's input in C, and denote by $w_{i,j}$ the instance of wire w_i in the jth garbled circuit. Further let $k_{i,j}^b$ denote the key associated with bit b on wire $w_{i,j}$. Then P_0 sets the keys for its input wires to $(k_{i,j}^0 = H(g^{a_i^0 c_j}), k_{i,j}^1 = H(g^{a_i^1 c_j}))$, where H is a randomness extractor such as a universal hash function [17,32,36].*
     (c) *P_0 constructs s independent garbled circuits for C, denoted as $GC_1, \ldots, GC_s$, using random keys except for the wires $w_1, \ldots, w_n$ for which the keys are as above.*
  2. **Oblivious transfers.** *The parties call $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ with their respective inputs and obtain outputs as follows:*
     (a) *P_0 defines vectors $\mathbf{z_1}, \ldots, \mathbf{z_n}$, where $\mathbf{z_i}$ contains the s pairs of random symmetric keys associated with P_1's ith input bit x_1^i in all garbled circuits $GC_1, \ldots, GC_s$.*
     (b) *P_1 inputs a random subset $\mathcal{J} \subset [s]$ of size $s/2$ and the bits $x_1^1, \ldots, x_1^n$.*
     (c) *P_1 receives from $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ all the keys that are associated with its input wires for the circuits GC_i with $i \in \mathcal{J}$ (denoted as the check circuits). In addition, it receives the keys that correspond to its input for the remaining circuits (denoted the evaluation circuits).*
  3. **Sending garbled circuits and commitments.** *P_0 transfers P_1 s encrypted copies of the garbled circuit using $\mathcal{F}_{\text{SC}}$, and the values $((g^{a_1^0}, g^{a_1^1}), \ldots, (g^{a_n^0}, g^{a_n^1}), (g^{c_1}, \ldots, g^{c_s}))$ along with the "seed" of the hash function H which constitutes the commitments to the input keys on the wires associated with P_0's input.*[4]
  4. **Revealing $\mathcal{J}$.** *P_1 reveals $\mathcal{J}$ and proves that it used this subset in the cut-and-choose OT protocol by sending the* pair *of keys associated with P_1's first input bit in each check circuit, i.e. for every GC_i with $i \in \mathcal{J}$. Note that P_1 knows the key pair only for the check circuits. If the values received from P_1 are wrong, then P_0 aborts.*
  5. **Decommitting $\mathbf{P_0}$'s input keys.** *In order to let P_1 know the keys for the input wires of P_0 within the check circuits, P_0 sends c_j for all $j \in \mathcal{J}$. P_1 computes the key pair $(H(g^{a_i^0 c_j}), H(g^{a_i^1 c_j}))$.*
  6. **Verifying the check circuits.** *P_1 verifies the validity of the check circuits using all the keys associated with their input wires. This ensures that the evaluation circuits are correct with high probability.*
  7. **Sending the garbled inputs for the evaluation circuits.** *In order to complete the evaluation, phase P_1 is given the keys for the input wires of P_0. P_0 must be*

[4] At this point P_0 is committed to all the keys associated with the s circuits.

forced to give the keys that are associated with the same input for all circuits. Specifically, the following code is executed for all input bits of P_0:

(a) For every evaluation circuit GC_j, P_0 transfers $\tau_{i,j} = g^{a_i^{x_0^i} c_j}$ using $\mathcal{F}_{\text{SC}}$, where x_0^i is the ith input bit of P_0.

(b) P_0 then proves that $a_i^{x_0^i}$ is in common for all keys associated with the ith input bit, which is reduced to showing that either the set $\{(g, g^{a_i^{x_0^i}}, g^{c_j}, \tau_{i,j})\}_{j=1}^s$ or the set $\{(g, g^{a_i^{1-x_0^i}}, g^{c_j}, \tau_{i,j})\}_{j=1}^s$ is comprised of DH tuples. Notably, it is sufficient to use a single UC ZK proof for the simpler relation $\mathcal{R}_{\text{DH,OR}}$ since the above statement can be compressed into a compound statement of two DH tuples as follows: P_0 first chooses s random values $\gamma_1, \ldots, \gamma_s \in \mathbb{Z}_p$ and sends them to P_1. Both parties compute $\tilde{g} = \prod_{j=1}^s (g^{c_j})^{\gamma_j}$, $\tilde{\tau} = \prod_{j=1}^s (\tau_{i,j})^{\gamma_j}$, of which P_0 proves that either $(g, g^{a_i^{x_0^i}}, \tilde{g}, \tilde{\tau})$ or $(g, g^{a_i^{1-x_0^i}}, \tilde{g}, \tilde{\tau})$ is a DH tuple. Thus, P_0 invokes $\mathcal{F}_{\text{DH,OR}}$ with $\sum_{j=1}^s c_j \gamma_j$ as the witness.

8. **Circuit evaluation.** *Upon receiving* `Accept` *from $\mathcal{F}_{\text{DH,OR}}$, P_1 completes the evaluation of the circuits and sets the majority of these values as the output y.*
9. **Output communication.** *P_1 sends y using $\mathcal{F}_{\text{SC}}$.*

Informally, to ensure the one-sided security of Π_f^{MAL} we realize the functionalities used in the protocol as follows: **(1)** $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ is realized in **Step 2** using our one-sided batch single choice cut-and-choose OT. This implies the equivocation of P_1's input. **(2)** Moreover, recall that the statement for the relation $\mathcal{R}_{\text{DH,OR}}$ is encrypted in **Step 7(a)** using one-sided NCE. Thus, it is sufficient to employ a standard static proof to realize $\mathcal{R}_{\text{DH,OR}}$ where the prover sends the third message of the proof using one-sided NCE. This implies the equivocation of P_0's input. Next, we prove

Theorem 5.5. (One-sided malicious) *Let f be a deterministic same-output functionality and assume that the encryption scheme for garbling has indistinguishable encryptions under chosen plaintext attacks, an elusive and efficiently verifiable range, and that the DDH and DCR assumptions are hard in the respective groups. Then Protocol Π_f^{MAL} UC realizes $\mathcal{F}_f$ in the presence of one-sided malicious adversaries using only $O(s|C|)$ private key operations and $O(s(|C|))$ public key operations where s is a statistical parameter that determines the cut-and-choose soundness error.*

We recall that the DDH and DCR hardness assumptions imply a cut-and-choose OT with constant number of PKE operations for large sender's input spaces (where DCR is required in order to implement the ideal functionalities $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{DL,COMP}(s,s/2)}, \mathcal{F}_{\text{DH,OR}(s)})$). In addition, we recall the [42] protocol is secure under the DDH assumption and IND-CPA symmetric key encryption scheme with the above special properties.

Proof. Our proof is shown in the $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{CCOT}}^{\text{BATCH}}, \mathcal{F}_{\text{DH,OR}})$-hybrid model. Let Adv be a probabilistic polynomial-time malicious adversary attacking Protocol 4 by adaptively corrupting one of the parties. We construct an adversary Sim for the ideal functionality $\mathcal{F}_f$

such that no environment ENV distinguishes with a non-negligible probability whether it is interacting with ADV in the real setting or with SIM in the ideal setting. We recall that SIM interacts with the ideal functionality $\mathcal{F}_f$ and the environment ENV. We refer to the interaction of SIM with $\mathcal{F}_f$ and ENV as the external interaction. The interaction of SIM with the simulated ADV is the internal interaction.

We now explain the actions of the simulation for the following corruption cases: **(1)** no corruption takes place; **(2)** corruption takes place at the outset; **(3)** corruption takes place between Steps 2 and 3; **(4)** corruption takes place between Steps 3 and 7; **(5)** corruption takes place between Steps 7 and 9; **(6)** Corruption takes place at the end. We describe a simulator for all these cases considering the corruption of each party. These cases cover all potential cases of corruption.

No corruption. When no corruption takes place the simulator simulates both P_0 and P_1 as follows:

1. **Garbled circuit construction.** No communication is carried out in this step. SIM internally picks n pairs of random values $(a_1^0, a_1^1), \ldots, (a_n^0, a_n^1) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_p$ and $c_1, \ldots, c_s \leftarrow \mathbb{Z}_q$ (which define the keys for P_0's input). It further picks a pair of random keys that correspond to each input bit of P_1.
2. **Oblivious transfers.** No communication is carried out in this step due to the ideal call to $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$. SIM internally chooses a random subset $\mathcal{J}$ on the behalf of P_1.
3. **Sending garbled circuits and commitments.** No communication is carried out with respect to the garbled circuit due to the ideal call to $\mathcal{F}_{\text{SC}}$. SIM emulates the honest P_0 sending the values $((g^{a_1^0}, g^{a_1^1}), \ldots, (g^{a_n^0}, g^{a_n^1}), (g^{c_1}, \ldots, g^{c_s}))$.
4. **Revealing** $\mathcal{J}$. SIM emulates the honest P_1 and sends $\mathcal{J}$ together with the pair of keys that are associated with P_1's first input bit for each check circuit.
5. **Decommitting $\mathbf{P_0}$'s input keys.** SIM emulates the honest P_0 and sends c_j for all $j \in \mathcal{J}$.
6. **Verifying the check circuits.** No communication is carried out in this step.
7. **Sending the garbled inputs for the evaluation circuits.** No communication is carried out in this step as it involves calling the two ideal functionalities $\mathcal{F}_{\text{SC}}$ and $\mathcal{F}_{\text{DH,OR}}$.
8. **Circuit evaluation.** No communication is carried out in this step.
9. **Output communication.** No communication is carried out in this step due to the ideal call to $\mathcal{F}_{\text{SC}}$.

Note that the simulated the hybrid executions are statistically close. Specifically, communication takes place only in Steps 3, 4, and 5 independently of the parties' inputs and outputs. Therefore, indistinguishability follows trivially for this case.

Corruption at the outset of the protocol execution

- P_0 **is corrupted.** In Step 2, SIM emulates $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$, receiving the input of ADV for this functionality, namely the key pairs that correspond to all the input wires of the circuit. In Step 3, SIM receives s garbled circuits from ADV on the behalf of $\mathcal{F}_{\text{SC}}$ and the commitments to the key pairs that correspond to ADV's input. In Step 4, the simulator sends a random subset $\mathcal{J}$ and the pairs of the input keys that are associated with P_1's first input wire in each check circuit GC_i for which $i \in \mathcal{J}$

(recall that SIM knows all the input keys in Step 2). In Step 5, the simulator receives c_j for all $j \in \mathcal{J}$ from ADV. In Step 6, the simulator verifies the check circuits just as the honest P_1 would do, and aborts if a problem is detected. In Step 7, the simulator receives the keys $\{\tau_{i,j}\}_{i\in\{1,\ldots,n\}, j\notin\mathcal{J}}$ as well as the statement and its corresponding witness for the ideal call of $\mathcal{F}_{\text{DH,OR}}$. If the witness is verified correctly, then SIM extracts ADV's input as follows. For some fixed $j \notin \mathcal{J}$, SIM extracts the ith bit of ADV's input x_0' by applying the hash function H on $\tau_{i,j}$ and then checking whether the result matches either $H(g^{a_i^0 c_j})$ which implies that $x_0'^i = 0$, or $H(g^{a_i^1 c_j})$ which implies that $x_0'^i = 1$. SIM sends x_0' to $\mathcal{F}_f$ and receives the output y'. Next it sends y' to ADV on behalf of $\mathcal{F}_{\text{SC}}$ in Step 9 and concludes the simulation.

- P_1 **is corrupted.** SIM mimics the [42] simulation that is designed for the static case, building fake circuits that always compute $y' = f(x_0, x_1')$ for x_1' the input of ADV. Specifically, in Step 1 SIM emulates P_0 as in the no corruption case, i.e. it picks key pairs that correspond to the input wires of both parties. In Step 2, SIM emulates $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ and receives ADV's input x_1' and the subset $\mathcal{J}$ and returns the adversary's output. It then sends x_1' to $\mathcal{F}_f$ and receives back the output y'. Then in Step 3, SIM constructs $s/2$ correct circuits that correspond to indices from $\mathcal{J}$ and additional $s/2$ fake circuits that always output y' irrespective of the input. It then sends this set of correct and fake garbled circuits to ADV on behalf of $\mathcal{F}_{\text{SC}}$. It further sends the commitments to the correct key pairs that correspond to the inputs wires of P_0. In Step 4, the simulator receives $\mathcal{J}$ and the key pairs that correspond to the first input bit of ADV for all the circuits GC_i such that $i \in \mathcal{J}$. If ADV sends invalid keys, or the set $\mathcal{J}$ does not match the set SIM obtains in Step 2 during the simulation of the ideal cut-and-choose OT, then SIM aborts. In Step 5, SIM correctly decommits c_j such that $j \in \mathcal{J}$. In Step 7, SIM plays the role of the honest P_0 with input 0^n. Following that, SIM emulates the ideal functionality for $\mathcal{F}_{\text{DH,OR}}$ and sends `Accept` to ADV. In Step 9 the simulator receives output y'' from ADV via $\mathcal{F}_{\text{SC}}$. If y'' is not equal y' then SIM aborts.
  We note that in the $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{CCOT}}^{\text{BATCH}}, \mathcal{F}_{\text{DH,OR}})$-hybrid execution, corruption at the onset of the protocol is proven similarly to the static proof provided in [42]. Specifically, the only difference between the proofs is that some parts of the communication of our protocol are transferred via $\mathcal{F}_{\text{SC}}$ (where this communication is sent non-encrypted in [42]). Therefore, when corruption takes place later in the protocol, the simulator can pretend that the computation was carried out correctly and consistently with the corrupted party's input. Specifically, the adaptive security reduces to static security.

Corruption between Steps 2 and 3

- P_0 **is corrupted.** SIM simulates P_0 and P_1 in Steps 1 and 2 as in the no corruption case. Then, whenever P_0 is corrupted SIM constructs s garbled circuits correctly using the keys that were picked in Step 1. It then discloses the circuits and the input keys to the adversary. It further explains the internal state of P_0 so that its input to the OT functionality $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ is consistent with the above input keys. Finally, SIM emulates the honest P_1 exactly as in the case when P_0 is corrupted at the outset.

Note that the adversary's views in case P_0 is corrupted at the outset of the protocol execution and in case P_0 is corrupted between Steps 2 and 3 is identical since P_0 does not use its input yet at this point, and thus the simulation in both corruption cases is essentially the same.

- P_1 **is corrupted.** SIM simulates P_0 and P_1 in Steps 1 and 2 as in the no corruption case. Then, whenever P_1 is corrupted SIM receives its input and output (x_1, y). SIM explains the internal state of P_1 such that the input of P_1 to the $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ functionality is x_1 and a random subset $\mathcal{J}$. It then discloses the keys that P_1 should have received upon entering x_1 and $\mathcal{J}$ to $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ in Step 2. SIM completes the simulation by simulating P_0 exactly as in the case when P_1 is corrupted at the outset.
  Note that in the hybrid setting the adversary's views in case P_1 are corrupted at the outset of the protocol execution and in case P_1 are corrupted between Steps 2 and 3, which is identical since the simulator can equivocate P_1's input to $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$. This implies that the adversary's internal state until corruption takes place is identically distributed to the real internal state of the honest P_1. Moreover, the simulation proceeds exactly as in the static case.

Corruption between Steps 3 and 7

- P_0 **is corrupted.** SIM simulates P_0 and P_1 in Step 1–3 as in the no corruption case. Then, whenever P_0 is corrupted, SIM constructs s garbled circuits correctly using the keys that were picked in Step 1. It then discloses the circuits and the input keys to the adversary. It further explains the internal state of P_0 so that its inputs to $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ in Step 2 and to $\mathcal{F}_{\text{SC}}$ in Step 3 are consistent with garbled circuits constructed above. SIM completes the simulation by simulating P_1 exactly as in the case when P_0 is corrupted at the outset.
  Note that in this corruption case P_0 does not use its input yet and thus the indistinguishability argument is as in the prior corruption case.
- P_1 **is corrupted.** SIM simulates P_0 and P_1 in Steps 1–3 as in the no corruption case. Then, whenever P_1 is corrupted, SIM receives its input and output (x_1, y). SIM explains the internal state of P_1 such that the input of P_1 to the $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ functionality is x_1 and a random subset $\mathcal{J}$. It further discloses the input keys that P_1 should have received upon entering inputs x_1 and $\mathcal{J}$ to $\mathcal{F}_{\text{CCOT}}^{\text{BATCH}}$ in Step 2. Next, SIM constructs $s/2$ correct circuits that correspond to indices from $\mathcal{J}$ and $s/2$ additional fake circuits that always output y. It then explains the internal state of P_1 such that P_1 has received the garbled circuits constructed as above (so that the correct circuits are the check circuits). SIM concludes the simulation by simulating P_0 exactly as in the case when P_1 is corrupted at the outset.
  The adversary's view is distributed as in the static corruption case relying on security of $\mathcal{F}_{\text{SC}}$ that transfers the garbled circuits. In the static corruption case, SIM sends a set of garbled circuits containing $s/2$ fake circuits and $s/2$ good circuits on behalf of $\mathcal{F}_{\text{SC}}$. No equivocation is required since SIM knows P_1 is corrupted at the outset. In the current scenario, SIM equivocates and explains to ADV that such a set is delivered by $\mathcal{F}_{\text{SC}}$. In $\mathcal{F}_{\text{SC}}$-hybrid model, the security reduces to the security of static corruption case.

Corruption between Steps 7 and 9

- P_0 **is corrupted.** SIM simulates P_0 and P_1 in Steps 1–7 as in the no corruption case. Then, whenever P_0 is corrupted SIM receives its input and output (x_0, y) and explains the internal state of P_0 until Step 7 as in the prior corruption case. Next, SIM explains the inputs of P_0 to the ideal call of $\mathcal{F}_{\text{SC}}$ in Step 7 so that they are consistent with the input x_0. Namely, the ith input wire of P_0 in the jth evaluation circuit is explained as $g^{a_i^{x_0^i} c_j}$. It further explains the witness to $\mathcal{F}_{\text{DH,OR}}$ correctly as $\sum_{j=1}^{s} c_j \gamma_j$ (recall that $\gamma_1, \ldots, \gamma_s$ are random elements that enable to combine s proofs into a single proof). The simulator concludes by simulating the honest P_1 exactly as in the case when P_0 is corrupted at the outset.
  In the hybrid setting, indistinguishability follows easily as above since the privacy of P_1 follows from the privacy of the OT protocol. Moreover, all the messages sent by P_0 are emulated as being sent by the honest P_0, and therefore security is reduced to the static corruption case.
- P_1 **is corrupted.** SIM simulates P_0 and P_1 in Steps 1–7 as in the no corruption case. Then, whenever P_1 is corrupted SIM receives its input and output (x_1, y) and explains the internal state of P_1 until Step 7 as in the prior corruption case. Next, SIM explains the internal state of P_1 in Step 7 using the garbled inputs keys of P_0 that are consistent with 0^n as the keys received from $\mathcal{F}_{\text{SC}}$. Namely, the ith input wire of P_0 in the jth evaluation circuit is explained as $g^{a_i^0 c_j}$. Finally, the simulator explains the message `Accept` as being received from the ZK functionality and concludes the simulation as above.
  The proof for this corruption case is identical to the prior case since P_1 already used its input in the cut-and-choose OT phase, and the additional simulated messages only correspond to the garbled inputs for the check circuits transferred from P_0.

Post-execution corruption

- P_0 **is corrupted.** SIM simulates P_0 and P_1 during all the steps as in the no corruption case. The internal state of P_0 is explained as above except that in Step 9 SIM explains y as the output received via the ideal call of $\mathcal{F}_{\text{SC}}$.
- P_1 **is corrupted.** SIM simulates P_0 and P_1 during all the steps as in the no corruption case. The internal state of P_0 is explained as above except that in Step 9 SIM explains y as the input entered to the ideal call of $\mathcal{F}_{\text{SC}}$.

Security in the hybrid model is proven exactly as above since the messages that are sent via $\mathcal{F}_{\text{SC}}$ can be equivocated relative the correct output value. This concludes the proof. □

6. Efficient Statically Secure and Witness Equivocal UC ZK PoKs

We present two results in this section. The first (side) result illustrates a technique for generating efficient statically secure UC ZK PoK for various Σ-protocols. Our protocols take a new approach where the prover commits to an additional transcript which enables witness extraction without using rewinding, and enable to achieve UC ZK PoK

constructions that incur constant overhead with a negligible soundness error. Prior to that, Σ-protocols were complied into the UC setting using UC commitments [11].

Next, we define a new notion denoted by *witness equivocal* UC ZK PoK for compound statements [8] (where the statement is comprised of several sub-statements that are associated using an OR relation). The additional feature that witness equivocal UC ZK PoK offers over static security is that it allows the simulator to equivocate the simulated proof upon adaptively corrupting the prover. Specifically, this notion implies that the simulator knows the witnesses for all sub-statements, but not which subset is used by the real prover. Furthermore, the simulator is able to convince the corrupted prover that it indeed knows the correct subset of witnesses. Note that this notion is weaker than one-sided UC ZK PoK where the simulator is required to simulate the proof obliviously of the witness and later prove consistency with respect to the real witness. In this work, we build efficient witness equivocal UC ZK PoKs for a class of fundamental compound Σ-protocols. Our protocols are constant rounds and overhead, with a negligible soundness error.

6.1. *Statically Secure UC ZK PoK with Constant Overhead*

We briefly describe our technique for generating efficient UC ZK PoK for Σ-protocols. Recall that in order to obtain a UC secure ZK PoK in the CRS model for a Σ-protocol it suffices to build a straight line simulator and witness extractor. A straight line simulator can be obtained by using standard techniques of committing the challenge of the verifier at the onset of the proof using UC commitments [23]. In what follows, we will focus on designing straight line extractors. We begin with a generalization of our UC ZK PoKs for Σ-protocols for relations of the form $\mathcal{R}_\Gamma = \left\{((\tilde{\mathbb{G}}, \tilde{\mathbb{H}}, y), x) |\ y = \Gamma(x)\right\}$ defined with respect to a one-way homomorphic mapping $\Gamma : \tilde{\mathbb{G}} \to \tilde{\mathbb{H}}$ from a source group $(\tilde{\mathbb{G}}, \oplus)$ to a target group $(\tilde{\mathbb{H}}, \odot)$ (where Γ is homomorphic if $\Gamma(x_0 \oplus x_1) = \Gamma(x_0) \odot \Gamma(x_1)$).[5] Loosely speaking, given a Σ-protocol Π_Γ for $\mathcal{R}_\Gamma$ we define a new proof Π'_Γ by instructing the prover to send two responses z, z' to a pair of distinct challenges c, c' queried by the verifier. Specifically, the former response z is sent on clear and publicly verified as specified in Π_Γ, whereas the latter response z' is encrypted using a homomorphic PKE with plaintext space $\tilde{\mathbb{G}}$. Moreover, the validity of z' is carried out by a (Σ-protocol) UC ZK proof Π_Σ for consistency. Observe that an extractor can be easily constructed for Π'_Γ by placing a public key for the homomorphic PKE in the CRS, of which the extractor knows the corresponding secret key. Clearly, the efficiency of our new proof depends heavily on the overhead of Π_Σ. For simplicity, we describe our protocols for a honest verifier; standard techniques can be used to achieve full security. We discuss two concrete implementations below.

6.1.1. *UC Secure ZK PoK for Discrete Logarithm*

We illustrate our technique on the Σ-protocol for proving the knowledge of a discrete logarithm (DL) in a prime-order group $\mathbb{G}$. We instantiate $(\tilde{\mathbb{G}}, \oplus)$ with $(\mathbb{Z}_p, +)$ for operation $+$ denoting addition in $\mathbb{Z}_p$, and $(\tilde{\mathbb{H}}, \odot)$ with $(\mathbb{G}, \cdot)$ for operation $\cdot$ denot-

[5] This notation covers many basic relations such as discrete logarithm and quadratic residuosity.

ing multiplication in $\mathbb{G}$. Furthermore, the one-way group homomorphism is defined by $\Gamma(x) = \mathrm{EXP}(x) = g^x$ where g is a generator of $\mathbb{G}$ and induces the relation,

$$\mathcal{R}_{\mathrm{DL}} = \left\{((\mathbb{G}, g, h), x) | \; h = g^x\right\}.$$

We apply our technique on the Σ-protocol due to [49] and instantiate the additively homomorphic PKE within Π_Σ with Paillier [47], that is defined by $\mathsf{Enc}_{\mathrm{PK}}(x; r) = (1+N)^x \cdot r^N \bmod N^2$ where N is an RSA composite and is IND-CPA secure under the DCR hardness assumption. Formally,

Protocol 5. (UC ZK PoK of DL (Π_{DL}))

- **CRS:** *A public key* PK *for Paillier PKE.*
- **Joint statement:** *The description of a group* $\mathbb{G}$ *of prime order* p *and a generator* g*, and the public statement* h*.*
- **Auxiliary input for the prover:** $x \in \mathbb{Z}_p$ *such that* $h = g^x$*.*
- **The Protocol:**
  1. *Prover* P *picks a random* $r \leftarrow \mathbb{Z}_p$ *and sends the verifier* $a = g^r$*.*
  2. *Verifier* V *returns random challenges* $c, c' \leftarrow \mathbb{Z}_p$*.*
  3. P *sends* $z \leftarrow r + cx \bmod p$ *and encrypts* $z' \leftarrow r + c'x \bmod p$ *using* PK*, generating ciphertext* e*.* P *sends* z *and* e *to* V *and proves in UC ZK that the plaintext of* e *and the discrete log of* $ah^{c'}$ *are the same.*
  4. V *accepts if the ZK proof is verified correctly and* $g^z = ah^c$*.*

The proof used in Step 3 is obtained from a Σ-protocol for the following relation,

$$\mathcal{R}_1 = \left\{((N, \mathrm{PK}, e, \mathbb{G}, g, h), (x, \alpha)) | \; e = (1+N)^x \alpha^N \bmod N^2 \wedge h = g^x\right\}.$$

Namely, the goal is to prove consistency of discrete logarithms with respect to two different group orders with generators $(1+N)$ and g, respectively. This can be achieved by combining the proof of knowledge of discrete logarithms over the integers [20] and the proof of plaintext knowledge for Pailler (we note that [20] shows a proof for consistent exponents, i.e. for Diffie–Hellman tuples, but the same proof technique works here as well). Namely, the prover selects at random y and β, computes $e' = (1+N)^y \beta^N \bmod N^2$ and $h' = g^y$ and sends e', h' to the verifier, who returns a random challenge $c \in \mathbb{Z}_p$. The prover then replies with $z = y + cx$ (over integers) and $\gamma = \alpha\beta^c \bmod N$. However, to ensure the privacy of x within $y + cx$, y must be chosen so that its length is at least $|c| + |x| + \kappa$, where κ is a statistical parameter. The verifier then accepts if $(1+N)^z \gamma^N \bmod N^2 = e'e^c \bmod N^2$ and $g^z = h'h^c$. We further note that the above proof requires a special care since it must ensure that the exact same value x is encrypted under Paillier rather than $x + ip$, for p the order of $\mathbb{G}$ and i some integer. Nevertheless, an extractor that decrypts and learns $x + ip$ can still find x. Thus our extractor first learns z' by decrypting the Paillier ciphertext and then extracts x from z and z'. Finally, the above Σ-protocol and the PoK presented in Protocol 5 are proven in the UC framework using standard techniques of committing the verifier's challenge at the beginning of the proof using UC commitment scheme [26].

Proposition 6.1. *Assume that the DCR and DDH assumptions are hard in the respective groups. Then Protocol 5 UC realizes $\mathcal{R}_{\text{DL}}$ with negligible soundness error and constant overhead.*

Proof Sketch. Informally, the proof follows by having the extractor pick a pair of keys (PK, SK) and place PK in the CRS. Then, whenever receiving ciphertext e from the prover, the extractor decrypts it using SK and extracts the witness from z and z'. From the security of the ZK proof of discrete logarithms consistency, it holds that the prover must encrypt with overwhelming probability the correct value of z'. This implies that the extractor can extracts the witness correctly. Furthermore, the ZK property is implied by the ZK of the original proof for $\mathcal{R}_{\text{DL}}$ and the ZK proof of consistency. Specifically, a simulator for Π_{DL} will compute the first message and z as in the original simulation of [49]. It then obliviously samples a ciphertext e rather than encrypting the real message z', and employs the simulator for the ZK proof of consistency Π_{Σ} (which is a proof for relation $\mathcal{R}_1$ in Protocol 5). Note that the simulator can also encrypt an arbitrary value instead of obliviously sampling the ciphertext. Nevertheless, we stick to the former description since it simplifies the description of our protocol for the adaptive setting. It is simple to verify that the simulated view is computationally indistinguishable from a real view since the only difference is relative to the ciphertext e and the simulated view of Π_{Σ}. Finally, the overhead of the protocol is constant since the overhead of the internal ZK proof is constant. □

Consistency of discrete logarithms. Next, we consider a UC PoK for the following relation,

$$\mathcal{R}_{\text{DH}} = \left\{((\mathbb{G}, g_0, g_1, h_0, h_1), x) | \; h_0 = g_0^x \wedge h_1 = g_1^x \right\}.$$

Here $(\tilde{\mathbb{G}}, \oplus)$ is instantiated with $(\mathbb{Z}_p, +)$ and $(\tilde{\mathbb{H}}, \odot)$ with $(\mathbb{G} \times \mathbb{G}, \cdot)$. We further define by $\Gamma(x) = (g_0^x, g_1^x)$ where g_0, g_1 are two generators in $\mathbb{G}$. As above, we use Paillier PKE to encrypt the second reply of the prover. The proof is an immediate extension of the Protocol 5 and the standard Σ-protocol for $\mathcal{R}_{\text{DH}}$. The underlying ZK proof for proving the correctness of the plaintext encrypted by the prover is an extension of the proof for the relation used in Protocol 5. Specifically, the relation for the underlying ZK proof is,

$$\mathcal{R}_2 = \Big\{((N, \text{PK}, e, \mathbb{G}, g_0, g_1, h_0, h_1), (x, \alpha)) | \; e \\ = (1+N)^x \alpha^N \bmod N^2 \wedge h_0 = g_0^x \wedge h_1 = g_1^x | \Big\}.$$

6.1.2. *UC Secure PoK for Nth Root and Quadratic Residuosity*

Finally, we demonstrate our technique for the proof of knowledge of an Nth root formally defined by,

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

We instantiate $(\tilde{\mathbb{G}}, \oplus)$ with $(\mathbb{Z}_N^*, \cdot)$ and $(\tilde{\mathbb{H}}, \odot)$ with $(\mathbb{Z}_{N^2}^*, \cdot)$, where multiplication is computed in the respective group. Furthermore, $\Gamma(x) = x^N \bmod N^2$. Note that in order to encrypt the message of the prover we need to use a multiplicative PKE, and we therefore consider a variant of El Gamal PKE that operates in $\mathbb{Z}_N^*$ for a message space $\mathbb{QR}_N$ where $N = pq$ is an RSA composite such that $(p-1)/2$ and $(q-1)/2$ are relatively primes. Specifically, encrypting a message $m \in \mathbb{QR}_N$ is computed by $(e_1, e_2) = (g^r \bmod N, m \cdot h^r \bmod N)$ where g is a random element in $\mathbb{QR}_N$, $h = g^x \bmod N$ with a secret key $x \in \mathbb{Z}_{\phi(N)/4}$ and randomness $r \leftarrow \mathbb{Z}_{\phi(N)/4}$. The security of this scheme is based on the composite DDH assumption [19] in $\mathbb{Z}_N^*$ (defined below). In the proof below, the verifier is required to ensure that $z'^N = au^{2c'}$. This is achieved by raising the ciphertext $e = (e_1, e_2)$ encrypting z' to the power of N component-wise modulo N^2, and then have the prover prove that $e_1^N, e_2^N/au^{2c'}$ is a Diffie–Hellman tuple in $\mathbb{Z}_{N^2}^*$. Such a ZK proof is provided in [20]. Namely, we use $2c'$ instead of c' to ensure that z' is in $\mathbb{QR}_N$.

The Composite DDH Assumption. Let $N = pq$ be an RSA modulus and g is an element of $\mathbb{QR}_N$ the group of squares in $\mathbb{Z}_N^*$. Then values a and b are chosen uniformly at random in $\mathbb{Z}_{\phi(N)/4}$ and the value y is either random in $\mathbb{QR}_N$ or satisfies $y = g^{ab} \bmod N$. Finally, the assumption asserts that for any polynomial-time algorithm, the advantage in guessing which way y was sampled when given $(N, g, g^a \bmod N, g^b \bmod N, y)$ is negligibly close to $1/2$.

Protocol 6. (**UC ZK PoK for $\mathcal{R}_{\text{NR}}$ (Π_{NR})**)

- **Joint statement:** $u \in \mathbb{Z}_{N^2}^*$.
- **Auxiliary input for the prover:** $v \in \mathbb{Z}_N^*$ *such that* $u = v^N \bmod N^2$.
- **CRS:** *A composite* N *and a public key* $\text{PK} = (\mathbb{G}, h = g^x)$ *for El Gamal PKE in* $\mathbb{Z}_N^*$.
- **The Protocol:**
  1. *Prover* P *picks a random* $r' \leftarrow \mathbb{Z}_N^*$ *and sends verifier* V *the value* a *where* $a = r^N \bmod N^2$ *where* $r \leftarrow r'^2 \bmod N$.
  2. V *returns random challenges* $c, c' \leftarrow \mathbb{Z}_N^*$.
  3. P *sets* $z \leftarrow rv^c \bmod N$ *and* $z' \leftarrow rv^{2c'} \bmod N$, *and encrypts* z' *using* PK *(note that* $z' \in \mathbb{QR}_N$*). Denote the generated ciphertext by* $e = (e_1, e_2)$. P *sends* V *values* z *and* e *and proves in UC ZK that the decryption of* $e^N \bmod N^2$ *corresponds to* $au^{2c'} \bmod N^2$. *That is,* P *proves that* $(\mathbb{Z}_{N^2}^*, g, h, e_1^N, e_2^N/au^{2c'})$ *is a Diffie–Hellman tuple in* $\mathbb{Z}_{N^2}^*$ *using the proof from [20].*
  4. V *accepts if it accepts the ZK proof and if* $z^N = au^c \bmod N^2$.

Proposition 6.2. *Assume that the DCR and composite DDH assumptions are hard in the respective groups. Then Protocol 6 UC realizes* $\mathcal{R}_{\text{NR}}$ *with negligible soundness error and constant overhead.*

Finally, we consider a proof of knowledge for the square root relation that is formally defined by,

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

We instantiate $(\tilde{\mathbb{G}}, \oplus)$ with $(\mathbb{Z}_N^*, \cdot)$ and $(\tilde{\mathbb{H}}, \odot)$ with $(\mathbb{QR}_N, \cdot)$, where multiplication is computed in the respective groups. Furthermore, $\Gamma(x) = x^2 \bmod N$. Following a similar technique used for the ZK PoK of $\mathcal{R}_{\text{NR}}$ we design a proof for $\mathcal{R}_{\text{QR}}$ based on the QR and composite DDH assumptions. Formally,

Proposition 6.3. *Assume that the QR and composite DDH assumptions are hard in the respective groups. Then there exists a protocol that UC realizes $\mathcal{R}_{\text{QR}}$ with negligible soundness error and constant overhead.*

6.2. *Witness Equivocal UC ZK PoK for Compound Statements*

The proof technique discussed above cannot be used in the adaptive setting since it does not allow witness equivocation when the prover is adaptively corrupted. Fortunately, in this work we only need to consider proofs of consistency for compound statements for which the simulator knows all witnesses but not which one is used by the real prover, since this choice depends on the prover's input. Consider the simple case of compound two statements for Σ-protocols, where the prover separates the verifier's challenge c 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, and 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 [8] for more details. This type of compound statements generalizes to s sub-statements for which the prover proves the knowledge of witnesses of some subset t.

Our next step is to design proofs for compound statements that are secure in the presence of adaptive attacks. Specifically, we design a weaker primitive for which the simulator knows the witnesses for all sub-statements, but not the correct subset. Note first that by simply allowing the simulator to use all potential witnesses is insecure in the adaptive setting, since an adversary that corrupts the prover can detect a simulated execution by simply computing the multiple witnesses. In order to resolve this difficulty we instruct the prover to obliviously sample the ciphertexts for the statements and it does not know the witnesses, which are then used as the statements for the internal proof of consistency Π_Σ. More concretely, assume a binary compound statement (the more general case follows easily). Then the prover encrypts a valid response with respect to the relation for which the witness is known, as discussed in Section 6.1. In addition, it obliviously picks the ciphertext for the statement it does not know the witness. Finally, the prover proves that one of these ciphertexts was computed correctly using an OR relation for Π_Σ. We note that both homomorphic PKEs discussed in Section 6.1 support oblivious ciphertext sampling.

Formally, we describe our protocol for compound statements that are defined relative to relations $\mathcal{R}_0$ and $\mathcal{R}_1$ (following the ideas from [8]). We denote by Π_0 and Π_1 the respective UC ZK PoK Σ-protocols for $\mathcal{R}_0$ and $\mathcal{R}_1$ and by Π_Σ the proof of consistency (we implicitly assume that Π_Σ is associated with a PKE).

Protocol 7. (**UC ZK PoK for $\mathcal{R}_0$ and $\mathcal{R}_1$ ($\Pi_{\mathbf{OR}}$)**)

- **Joint statement:** $x_0 \in L_0$ *and* $x_1 \in L_1$.
- **Auxiliary input for the prover:** ω_i *for* $i \in \{0, 1\}$ *such that* $(x_i, \omega_i) \in \mathcal{R}_i$.
- **CRS:** *A CRS for* Π_Σ.
- **The Protocol:**
  1. *Prover* P *computes the first message as follows.*
     It first invokes the simulator SIM_{1-i} *for* Π_{1-i} *on* x_{1-i} *and arbitrary challenge* $\tilde{c}$*, and obtains message* m_{1-i}.[6]
     It then invokes the real prover P_i *for* Π_i *on* (x_i, ω_i) *and obtains message* m_i.
     P *sends messages* (m_0, m_1) *to the verifier.*
  2. V *returns two random challenges* c, c' *from the appropriate space.*
  3. P *computes its response as follows. It first invokes simulator* SIM_{1-i} *on* x_{1-i} *and arbitrary challenge* $\tilde{c}$*, and receives a prover's response* z_{1-i} *for* Π_{1-i}. P *then computes an obliviously sampled ciphertext* e_{1-i}. *Next,* P *invokes* P_i *on* $(x_i, \omega_i), m_i, c \oplus \tilde{c}$ *and receives a prover's response* z_i *for* Π_i. P *then invokes* P_i *on* $(x_i, \omega_i), m_i, c'$ *and receives a prover's response* z'_i *for* Π_i. *Let* e_i *denote a ciphertext that encrypts* z'_i *Finally,* P *engages with* V *in an execution of an OR relation for the proof* Π_Σ*, proving that either* e_i *encrypts a valid response for* Π_i *or* e_{i-1} *encrypts a valid response for* Π_{i-1} *relative to challenge* $c' \notin \{\tilde{c}, c \oplus \tilde{c}\}$. P *further sends the verifier the messages* z_0, z_1.
  4. V *invokes the verifiers for* Π_0 *and* Π_1 *and accepts if they both accept the messages* z_0, z_1*, if the two challenges received from the prover are valid shares of* c *and if the proof for the OR relation of* Π_Σ *is verified correctly.*

We conclude with the following theorem.

Theorem 6.4. *Assume the existence of homomorphic PKE with respect to a group* $\mathbb{H}$ *and operation* $\odot$ *that supports oblivious and invertible sampling of ciphertexts, and that* $\Gamma : \tilde{\mathbb{G}} \to \tilde{\mathbb{H}}$ *is a one-way group homomorphism. Then, Protocol 7 is a witness equivocal UC ZK* Σ*-protocol for relations* $\mathcal{R}_0$ *and* $\mathcal{R}_1$ *with negligible soundness error and constant overhead.*

Proof Sketch. Proving PoK follows easily using the trapdoor from the CRS and the soundness of Π_Σ, which allows the extractor to decrypt the ciphertext e_i and extract the witness. We next prove that the protocol is ZK. Note that standard simulation follows from the ZK property of each sub-protocol and the [8] proof. We recall next that our protocols only consider simulators that know both witnesses ω_0 and ω_1, but do not know which one is used by the real prover. Simulation in this case is trivial since the simulator simply uses its two witnesses. By the IND-CPA security of the homomorphic encryption scheme and the security of Π_{1-i}, the simulated view (when using two witnesses) and the real view (when using only one witness) are computationally indistinguishable. We now show that the protocol is witness equivocal. Recall that this property implies that the simulator must explain the internal state of the simulated prover with respect to

[6] Note that the simulator returns the entire view for the proof from which we extract the first message.

the real prover's witness. Say the real prover knows ω_i, then the view for Π_i can be easily explained as if the real prover generated it since the simulator used ω_i in its simulation. In addition, the simulated proof for x_{1-i} differs from the real view by (1) honestly encrypting message z'_{1-i} rather than obliviously sampling the ciphertext and (2) running the real prover for Π_{1-i} rather than the simulated one. Witness equivocal follows from the ciphertext simulatability of the PKE and the fact that the real view of Π_{1-i} can be explained as a simulated view. Namely, the simulator for Protocol 7 can claim that the honestly generated ciphertext e_{1-i} was obliviously sampled and that the real view generated for Π_Σ relative to statement e_{1-i} is a simulated view (since Π_Σ is a Σ-protocol with perfect simulation; see Definition 7.9). □

Our constructions for the malicious setting make use of compound relations defined as follows. We denote by $\mathcal{R}_{\Gamma,\mathrm{OR}}$ a compound OR relation of two sub-statements where both statements correspond to the same relation $\mathcal{R}_\Gamma$. We further denote by $\mathcal{R}_{\Gamma,\mathrm{OR}(s)}$ a relation for which the statement is a combination of two sub-statements, each contains a tuple of s elements. Specifically, we consider a proof of knowledge for the relation $\mathcal{R}_{\mathrm{DH},\mathrm{OR}(s)}$ of which only one of the two sets is comprised from Diffie–Hellman tuples. Finally, we consider a relation $\mathcal{R}_{\Gamma,\mathrm{COMP}(s,t)}$ where the statement consists s sub-statements for relation $\mathcal{R}_\Gamma$ for which the prover proves the knowledge of only t sub-statements out of s (for some $t < s$).

7. Appendix 1: Standard Security Notions

7.1. *Hardness Assumptions*

Our constructions rely on the following hardness assumptions.

Definition 7.1. (*DDH*) We say that *the decisional Diffie–Hellman (DDH) problem is hard relative to* $\mathcal{G}$ if for all polynomial-sized circuits $\mathcal{C} = \{\mathcal{C}_n\}$ there exists a negligible function negl such that

$$\left|\Pr\left[\mathcal{C}_n(\mathbb{G}, q, g, g^x, g^y, g^z) = 1\right] - \Pr\left[\mathcal{C}_n(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1\right]\right| \leq \mathsf{negl}(n),$$

where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$ and the probabilities are taken over the choices of g and $x, y, z \in \mathbb{Z}_q$.

We require the DDH assumption to hold for prime-order groups. In a few places we use a different version of the DDH assumption: for random generators $g, h \in \mathbb{G}$ and for distinct but otherwise random $a, b \in \mathbb{Z}_q$, the tuples (g, h, g^a, h^a) and (g, h, g^a, h^b) are computationally indistinguishable. This version of the DDH assumption is equivalent to the common form discussed above.

Definition 7.2. (*DCR*) We say that *the decisional composite residuosity (DCR) problem is hard relative to* $\mathcal{G}$ if for all polynomial-sized circuits $\mathcal{C} = \{\mathcal{C}_n\}$ there exists a negligible function negl such that

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

where $N \leftarrow \mathcal{G}(1^n)$, 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 7.3. (*QR*) We say that *the quadratic residuosity (QR) problem is hard relative to* $\mathcal{G}$ if for all polynomial-sized circuits $\mathcal{C}=\{\mathcal{C}_n\}$ there exists a negligible function negl such that

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

where $N \leftarrow \mathcal{G}(1^n)$, 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.

7.2. *Public Key Encryption Scheme*

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

Definition 7.4. (*PKE*) We say that $\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ is a *public key encryption scheme* if Gen, Enc, Dec are polynomial-time algorithms specified as follows:

- Gen, given a security parameter n (in unary), outputs keys (PK, 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 probabilistic adversary $\textsc{Adv}=(\textsc{Adv}_1,\textsc{Adv}_2)$, we consider the following *IND-CPA 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 $\mathbf{Adv}_{\Pi,\textsc{Adv}}(n)$ the probability that $\textsc{Adv}$ wins the IND-CPA game.

Definition 7.5. IND-(*CPA*) A public key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has *indistinguishable encryptions under chosen plaintext attacks (IND-CPA)*, if for every non-uniform probabilistic adversary $\textsc{Adv} = (\textsc{Adv}_1, \textsc{Adv}_2)$ there exists a negligible function negl such that $\mathbf{Adv}_{\Pi,\textsc{Adv}}(n) \leq \frac{1}{2} + \mathsf{negl}(n)$.

We say that a protocol π *realizes functionality* $\mathcal{F}$ *with t PKE operations* (relative to Π) if the number of calls π makes to either one of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is at most t. Importantly, this definition is not robust in the sense that one might define an encryption algorithm Enc' that consists of calling Enc, n times in parallel. In this work we do not abuse this definition and consider algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for a single basic operation, which are implemented by $O(1)$ group exponentiations in various group descriptions.

7.3. *Zero-knowledge Proofs and Proofs of Knowledge*

Our protocols employ zero-knowledge proofs (of knowledge) for assuring correct behaviour. We formally define zero-knowledge and knowledge extraction as stated in [28]. We then conclude with a definition of a Σ-protocol which constitutes a zero-knowledge proof of a special type.

Definition 7.6. (*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 $x \in L$,

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

2. Soundness: For every $x \notin L$ and every interactive PPTmachine B,

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

Definition 7.7. (*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

$$\{\langle \mathrm{P}, \mathrm{V}^*\rangle(x)\}_{x\in L} \approx_c \{\langle M^*\rangle(x)\}_{x\in L}$$

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

Definition 7.8. (*Knowledge extraction*) Let R be a binary relation and $\kappa \to [0, 1]$. We say that an interactive function V is *a knowledge verifier for the relation R with knowledge error* κ if the following two conditions hold:

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 $\mathrm{P}_{x,y,r}$ 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 $\mathrm{P}_{x,y,r}$, 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.

Definition 7.9. (Σ-*protocol*) A protocol π is a Σ-*protocol* for relation R if it is a 3-round public-coin protocol and the following requirements hold:

- COMPLETENESS: If P and V follow the protocol on input x and private input w to P where $(x, w) \in \mathcal{R}$, then V always accepts.
- SPECIAL SOUNDNESS: There exists a polynomial-time algorithm A given any x and any pair of accepting transcripts (a, e, z), (a, e', z') on input x, where $e \neq e'$, outputs w such that $(x, w) \in \mathcal{R}$.
- SPECIAL HONEST-VERIFIER ZERO KNOWLEDGE: There exists a PPTalgorithm M^* such that

$$\left\{\langle \mathrm{P}(x, w), \mathrm{V}(x, e)\rangle\right\}_{x \in L_{\mathcal{R}}} \approx_c \left\{M(x, e)\right\}_{x \in L_{\mathcal{R}}}$$

  where $M(x, e)$ denotes the output of M upon input x and e, and $\langle \mathrm{P}(x, w), \mathrm{V}(x, e)\rangle$ denotes the output transcript of an execution between P and V, where P has input (x, w), V has input x, and V's random tape (determining its query) equals e.

8. Appendix 2: A High-Level Overview of Yao's Garbling Technique

We briefly describe the garbling technique of Yao as described by Lindell and Pinkas in [41]. In this construction, the desired function f is represented by a boolean circuit C that is computed gate by gate from the input wires to the output wires. In the following, we distinguish four different types of wires used in a given boolean circuit: (**a**) circuit-input wires; (**b**) circuit-output wires; (**c**) gate-input wires (that enter some gate g); and (**d**) gate-output wires (that leave some gate g). The underlying idea is to associate every wire w with two random values, say k_w^0, k_w^1, such that k_w^0 represents the bit 0 and k_w^1 represents the bit 1. The garbled table for each gate maps random input values to random output values, with the property given two input values it is only possible to learn the output value that corresponds to the output bit. This is accomplished by viewing the four potential inputs to the gate k_1^0, k_1^1 (values associated with the first input wire) and k_2^0, k_2^1

(values associated with the second input wire), as encryption keys. So that the output key values k_3^0, k_3^1 are encrypted under the appropriate input keys. For instance, let g be a NAND gate. Then, k_3^1 (that corresponds to bit 1) is encrypted under the pair of keys associated with the values $(0,0)$, $(0,1)$, $(1,0)$, whereas k_3^0 is encrypted under the pair of keys associated with $(1,1)$ which yields the following four ciphertexts

$$\mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^0}(k_3^1)),\ \ \mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^1}(k_3^1)),\ \ \mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^0}(k_3^1))\ \text{ and }\ \mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^1}(k_3^0)),$$

where $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private key encryption scheme that has *chosen double encryption security* and an *elusive efficiently verifiable range*; see [41] for the formal definitions. These ciphertexts are randomly permuted in order to obtain the garbled table for gate g. Then, given the input wire keys k_1^α, k_2^β that correspond to the bits α and β and the garbled table containing the four encryptions, it is possible to obtain the output wire key $k_3^{g(\alpha,\beta)}$. The description of the garbled circuit is concluded with the *output decryption tables*, mapping the random values on the circuit-output wires back to their corresponding boolean values.

A useful lemma. Next, we state a useful lemma regarding garbled circuits taken verbatim from [43] and further stated in [40,42]. The lemma states that it is possible to build a fake garbled circuit that outputs a fixed value $y = f(x_0, x_1)$ which is indistinguishable relative to an adversary who has only a single set of keys that corresponds to the inputs x_0, x_1. We rely on this lemma in our one-sided security proofs when P_1 is corrupted. Formally,

Lemma 8.1. *Given a circuit C and an output value y (of same length as the output of C) it is possible to construct a garbled circuit $\widetilde{GC}$ such that:*

1. *The output of $\widetilde{GC}$ is always y, regardless of the garbled values that are provided for the input wires of P_0 and P_1, and*
2. *If $y = f(x_0, x_1)$, then no non-uniform PPT adversary* Adv *can distinguish between the distribution ensemble of $\widetilde{GC}$ and a single arbitrary garbled value for every input wire, and the distribution ensemble consisting of a real garbled version of C, together with garbled values that correspond to x_0 for P_0's input wires and to x_1 for P_1's input wires.*

References

[1] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation, in *EUROCRYPT* (2011), pp. 169–188.

[2] D. Beaver. Plug and play encryption, in *CRYPTO* (1997), pp. 75–89.

[3] D. Beaver, and S. Haber. Cryptographic protocols provably secure against dynamic adversaries, in *EUROCRYPT* (1992), pp. 307–323.

[4] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening, in *EUROCRYPT* (2009), pp. 1–35.

[5] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract), in *STOC* (1990), pp. 503–513.

[6] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols, in *FOCS* (2001), pp. 136–145.
[7] R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, and T. Malkin. Adaptive versus non-adaptive security of multi-party protocols. *J. Cryptology*, 17(3):153–207, 2004.
[8] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols, in *CRYPTO* (1994), pp. 174–187.
[9] S.G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Improved non-committing encryption with applications to adaptively secure protocols, in *ASIACRYPT* (2009), pp. 287–302.
[10] S.G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Simple, black-box constructions of adaptively secure protocols, in *TCC* (2009), pp. 387–402.
[11] R. Canetti, and M. Fischlin. Universally composable commitments, in *Proceedings of Advances in Cryptology—CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA* (Aug. 19–23, 2001), pp. 19–40.
[12] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation, in *STOC* (1996), pp. 639–648.
[13] R. Canetti, S. Goldwasser, and O. Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation, in *TCC* (2015), pp. 557–585.
[14] R. Canetti, S. Halevi, and J. Katz. Adaptively-secure, non-interactive public-key encryption, in *TCC* (2005), pp. 150–168.
[15] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation, in *STOC* (2002).
[16] R. Cramer, and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in *EUROCRYPT* (2002), pp. 45–64.
[17] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes, in *CRYPTO* (2004), pp. 494–510.
[18] I. Damgård, and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator, in *CRYPTO* (2005), pp. 378–394.
[19] I. Damgård, and M. Jurik. A length-flexible threshold cryptosystem with applications, in *ACISP* (2003), pp. 350–364.
[20] I. Damgård, M. Jurik, and J.B. Nielsen. A generalization of Paillier's public-key system with applications to electronic voting. *Int. J. Inf. Sec.*, 9(6):371–385, 2010.
[21] D. Dachman-Soled, J. Katz, and V. Rao. Adaptively secure, universally composable, multiparty computation in constant rounds, in *TCC* (2015), pp. 586–613.
[22] I. Damgård, and J.B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption, in *CRYPTO* (2000), pp. 432–450.
[23] I. Damgård, and J.B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor, in *CRYPTO* (2002), pp. 581–596.
[24] I. Damgård, and J.B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption, in *CRYPTO* (2003), pp. 247–264.
[25] I. Damgård, V. Pastro, N.P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption, in *CRYPTO* (2012), pp. 643–662.
[26] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.
[27] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority, in *STOC* (1987), pp. 218–229.
[28] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
[29] S. Garg, and A. Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation, in *TCC* (2015), pp. 614–637.
[30] S. Garg, and A. Sahai. Adaptively secure multi-party computation with dishonest majority, in *CRYPTO* (2012), pp. 105–123.
[31] J.A. Garay, D. Wichs, and H.-S. Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer, in *CRYPTO* (2009), pp. 505–523.
[32] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[33] B. Hemenway, R. Ostrovsky, and A. Rosen. Non-committing encryption from Φ-hiding, in *TCC* (2015), pp. 591–608.
[34] C. Hazay, and A. Patra. One-sided adaptively secure two-party computation, in *TCC* (2014), pp. 368–393.
[35] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer—efficiently, in *CRYPTO* (2008), pp. 572–591.
[36] M.N. Wegman J.L. Carter. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.
[37] S. Jarecki, and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures, in *EUROCRYPT* (2000), pp. 221–242.
[38] J. Katz, and R. Ostrovsky. Round-optimal secure two-party computation, in *CRYPTO* (2004), pp. 335–354.
[39] J. Katz, A. Thiruvengadam, and H.-S. Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption, in *Public Key Cryptography* (2013), pp. 14–31.
[40] Y. Lindell. Adaptively secure two-party computation with erasures, in *CT-RSA* (2009), pp. 117–132.
[41] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.
[42] Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, 2012.
[43] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *J. Cryptology*, 28(2):312–350, 2015.
[44] J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case, in *CRYPTO* (2002), pp. 111–126.
[45] J.B. Nielsen, P.S. Nordholt, C. Orlandi, and S.S. Burra. A new approach to practical active-secure two-party computation, in *CRYPTO* (2012), pp. 681–700.
[46] M. Naor, and O. Reingold. Synthesizers and their application to the parallel construction of psuedo-random functions, in *FOCS* (1995), pp. 170–181.
[47] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes, in *EUROCRYPT* (1999), pp. 223–238.
[48] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer, in *CRYPTO* (2008), pp. 554–571.
[49] C.-P. Schnorr. Efficient identification and signatures for smart cards, in *CRYPTO* (1989), pp. 239–252.
[50] S. Wolf, and J. Wullschleger. Oblivious transfer is symmetric, in *EUROCRYPT* (2006), pp. 222–232.
[51] A.C. Yao. Protocols for secure computations (extended abstract), in *FOCS* (1982), pp. 160–164.