One-Sided Adaptively Secure Two-Party Computation

Carmit Hazay[1] and Arpita Patra[2]

[1] Faculty of Engineering, Bar-Ilan University, Israel. carmit.hazay@biu.ac.il
[2] Dept. of Computer Science, University of Bristol, UK. arpita.patra@bristol.ac.uk

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 grows linearly with the length of the message. Furthermore, general two-party protocols require a number of NCE calls that is linear in the circuit size.
In this paper we study the two-party setting in which at most one of the parties is adaptively corrupted, which we believe is the right security notion in the two-party setting. We study 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 input spaces of the sender, and (**3**) constant round secure computation protocols with a number of NCE calls, and an overall number of public key operations, that are independent of the circuit size. Our study demonstrates that such primitives indeed exist in the presence of single corruptions, while this is not known for fully adaptive security.

Keywords: Adaptively Secure Computation, Non-Committing Encryption, Oblivious Transfer

1 Introduction

1.1 Background

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 viewing a protocol execution as if the computation is 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 work of [36, 22], it is by now well known that (in various settings) any polynomial-time function can be compiled into a secure function evaluation protocol with practical complexity; see [30, 16, 33] for a few recent works. The security proofs of these constructions assume that a party is 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. Adaptive security is a stronger notion where corruptions takes place *at any point* during the course of the protocol execution. That is, upon corruption the adversary sees the internal data of the corrupted party which includes its input, randomness and the incoming messages. This notion is much stronger than static security due to the fact that 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 than the static corruption model.

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 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 without the ability to verify its execution. 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 comes 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.

Adaptive security. It is known by now that security against adaptive attacks captures 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. In the static setting the adversary corrupts the exact same set of parties that share the secret with a negligible probability in n.

Other difficulties also arise when proving security. 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. However, 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 value). Thus, upon corrupting the receiver at the end of the protocol execution 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.

Adaptively secure two-party computation. In the two-party setting there are scenarios where the system is comprised from two devices communicating between themselves without being part of a bigger system. For instance, consider a scenario where two devices share an access to an encrypted database that contains highly sensitive data (like passwords). Moreover, the devices communicate via secure computation but do not communicate with other devices due to high risk of breaking into the database. Thus, attacking one of the devices does not disclose any useful information about the content of the database, while attacking both devices is a much harder task. It is reasonable to assume that the devices are not necessarily statically corrupted since they are protected by other means, while attackers may constantly try to break into these devices (even while running secure computation).

In 2011, RSA secureID authentication products were breached by hackers that leveraged the stolen information from RSA in order to attack the U.S. defense contractor Lockheed Martin. The attackers targeted SecurID data as part of a broader scheme to steal defense secrets and related intellectual property. Distributing the SecureID secret keys between two devices potentially enables to defend against such an attack since in order to access these keys the attackers need to *adaptively* corrupt both devices, which is less likely to occur. Many other applications face similar threats when attempt to securely protect their databases.

We therefore focus on a security notion that seems the most appropriate in this context. In this paper, we study secure two-party computation with single adaptive corruptions in the non-erasure model where at most one party is adaptively corrupted. 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 efficiency of two-party protocols with one-sided security. Our measure of efficiency is the number of public key encryption (PKE) operations. Loosely speaking, our primitives are parameterized by a public key encryption scheme for which we count the number of key generation/encryption/decryption operations. More concretely, these operations are captured by the number of exponentiations in several important groups (e.g., groups where the DDH assumption is hard and composite order groups where the assumptions DCR and QR are hard), and further considered in prior works such as [19]. Finally, our proofs are given in the universal composable (UC) setting [6] with a common reference string (CRS) setup. The reductions of our non-committing encryption and oblivious transfer with one-sided security are tight. The reductions of our general two-party protocols are tighter than in prior works since we do not need to encrypt the entire communication using non-committing encryption; see more details below. All our theorems *are not* known to hold in the fully adaptive setting.

1.2 Our Results

One-sided NCE with constant overhead. A non-committing encryption (NCE) scheme [8] implements secure channels in the presence of adaptive corruptions and is an important building block in designing adaptively secure protocols. In [13], Damgård and Nielsen presented a theoretical improvement in the one-sided setting by designing an NCE under strictly weaker assumptions than simulatable public key encryption scheme (the assumption for fully adaptive NCE). Nevertheless, all known one-sided [8, 13] and fully adaptive NCE constructions [13, 11] require $\mathcal{O}(1)$ PKE operations for each transmitted bit. It was unknown whether this bound can be reduced for one-sided NCEs and even matched with the overhead of standard PKEs.

We suggest a new approach for designing NCEs secure against one-sided adaptive attacks. Our protocols are built on two cryptographic building blocks that are non-committing with respect to a single party. We denote these 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 simulated 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. Both NCES and NCER are semantically secure PKEs and they are as efficient as standard PKEs. Informally, we prove that,

Theorem 1 (Informal) *Assume the existence of NCER and NCES with constant number of PKE operations for message space* $\{0,1\}^q$ *and simulatable PKE. Then there exists a one-sided NCE with constant number of PKE operations for message space* $\{0,1\}^q$*, where* $q = O(n)$ *and* n *is the security parameter.*

Importantly, the security of this protocol only works if the simulator knows the identity of the corrupted party since fake public keys and ciphertexts cannot be explained as valid ones. We resolve this issue by slightly modifying this protocol using somewhat NCE [19] in order to encrypt only three bits. Namely, we use somewhat NCE to encrypt the *choice* of having fake/valid keys and ciphertexts (which only requires a single non-committing bit per choice). This enables the simulator to “explain” fake keys/ciphertext as valid and vice versa using only a constant number of asymmetric operations. In this work we consider two implementations of NCER and NCES. For polynomial-size message spaces the implementations are secure under the DDH assumption, whereas for exponential-size message spaces security holds under the DCR assumption. The NCER implementations are taken from [25, 9]. NCES was further discussed in [17] and realized under the DDH assumption in [5] using the closely related notion of lossy encryption.[3] In this paper we realize NCES under the DCR assumption.

One-sided oblivious transfer with constant overhead. We use our one-sided NCEs to implement 1-out-of-2 oblivious transfer (OT) between a sender and a receiver. We consider a generic framework that abstracts the statically secure OT of [34] that is based on a dual-mode PKE primitive, while encrypting only a small portion of the communication using our one-sided NCE. Our construction requires a constant number of PKE operations for an input space $\{0,1\}^q$ of the sender, where $q = O(n)$. This is significantly better than the fully adaptively secure OT of [19] (currently the most efficient fully adaptive construction), that requires $\mathcal{O}(q)$ such operations. We prove that:

Theorem 2 (Informal) *Assume the existence of one-sided NCE with constant number of PKE operations for message space* $\{0,1\}^q$ *and dual-mode PKE. Then there exists a one-sided OT with constant number of PKE operations for sender's input space* $\{0,1\}^q$*, where* $q = O(n)$ *and* n *is the security parameter.*

[3] This notion differs from NCES by not requiring an efficient opening algorithm that enables to equivocate the ciphertext's randomness. We further observe that the notion of NCES is also similar to mixed commitments [14].

We build our one-sided OT based on the PVW protocol as follows. (**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 of the dual-mode PKE in a single mode (instead of alternating between two modes as in the [19] protocol). To ensure correctness, we employ a special type of ZK PoK which uses a novel technique; see below for more details. Finally, we discuss two instantiations based on the DDH and QR assumptions.

Constant round one-sided secure computation. Theoretically, it is well known that any statically secure protocol can be transformed into a one-sided adaptively secure protocol by encrypting the *entire* communication using NCE. This approach, adopted by [26], implies that the number of PKE operations grows linearly with the circuit size times a computational security parameter.[4] A different approach in the OT-hybrid model was taken in [24] and achieved a similar overhead as well.

In this work we demonstrate the feasibility of designing generic constant round protocols based on Yao's garbled circuit technique with one-sided security, tolerating semi-honest and malicious attacks. Our main observation implies that one-sided security can be obtained even if only the keys corresponding to the inputs and output wires are communicated via a one-sided adaptively secure channel. This implies that the bulk of communication is transmitted as in the static setting. Using our one-sided secure primitives we obtain protocols that outperform the constant round one-sided constructions of [26, 24] and all known generic fully adaptively secure two-party protocols. Our proofs take a different simulation approach, circumventing the difficulties arise due to the simulation technique from [28] that builds a fake circuit (which cannot be applied in the adaptive setting). Specifically, we prove that

Theorem 3 (Informal) *Under the assumptions of achieving statically secure two-party computation and one-sided OT with constant number of PKE operations for sender's input space $\{0,1\}^q$, where $q = O(n)$ and n is the security parameter, there exists a* constant round *one-sided semi-honest adaptively secure two-party protocol that requires $\mathcal{O}(|C|)$ private key operations and $\mathcal{O}(|\mathsf{input}| + |\mathsf{output}|)$ public key operations.*

In order to obtain one-sided security against malicious attacks we adapt the cut-and-choose based protocol introduced in [30]. 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 requires a modified cut-and-choose OT protocol, since [30] defines the public parameters of their cut-and-choose OT protocol in a way that precludes the equivocation of the receiver's input. Our result in the malicious setting follows.

Theorem 4 (Informal.) *Under the assumptions of achieving static security in [30], one-sided cut-and-choose OT with constant number of PKE operations for sender's input*

[4] 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 [27] that adaptive security is impossible for FHE satisfying compactness.

space $\{0,1\}^q$, where $q = O(n)$ and n is the security parameter, and simulatable PKE, there exists a constant round *one-sided malicious adaptively secure two-party protocol that requires $\mathcal{O}(s \cdot |C|)$ private key operations and $\mathcal{O}(s \cdot (|\mathsf{input}| + |\mathsf{output}|))$ public key operations where s is a statistical parameter that determines the cut-and-choose soundness error.*

This asymptotic efficiency is significantly better than in prior protocols [26, 24].

Witness equivocal UC ZK PoK for compound statements. As a side result, we demonstrate a technique for efficiently generating statically secure UC ZK PoK for known Σ-protocols. Our protocols use a new approach where the prover commits to an additional transcript which enables to extract the witness with a constant overhead.

We further focus on compound statements (where the statement is comprised of sub-statements for which the prover only knows a subset of the witnesses), and denote a UC ZK PoK by *witness equivocal* if the simulator knows the witnesses for *all* sub-statements but not which subset is given to the real prover. We extend our proofs for this notion to the adaptive setting as well. In particular, the simulator must be able to convince an adaptive adversary that it does *not know* a different subset of witnesses. This notion is weaker than the typical one-sided security notion (that requires simulation without the knowledge of any witness), but is still meaningful in designing one-sided secure protocols. In this work, we build witness equivocal UC ZK PoKs for a class of fundamental compound Σ-protocols, without relying on NCE. Our protocols are round efficient and achieve a negligible soundness error. Finally, they are proven secure in the UC framework [6].

To conclude, our results may imply that one-sided security is strictly easier to achieve than fully adaptive security, and for some applications this is indeed the right notion to consider. We leave open the feasibility 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 within the [4] protocol), and achieve secure constructions with a number of PKE operations that does not depend on the circuit size.

1.3 Prior Work

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

Non-committing encryption. One-sided NCE was introduced in [8] which demonstrated feasibility of the primitive under the RSA assumption. Next, NCE was studied in [13, 11]. The construction of [13] requires constant rounds on the average and is based on simulatable PKE, whereas [11] presents an improved expected two rounds NCE based on a weaker primitive. [13] further presented a one-sided NCE based on a weakened simulatable PKE notion. The computational overhead of these constructions is $\mathcal{O}(1)$ PKE operations for each transmitted bit. An exception is the somewhat NCE introduced in [19] (see Section 2.5 for more details). This primitive enables to send arbitrarily long messages at the cost of $\log \ell$ PKE operations, where ℓ is the equivocality parameter that determines the number of messages the simulator needs to explain. This construction improves over NCEs for sufficiently small ℓ's. Finally, in [32] Nielsen proved that adaptively secure non-interactive encryption scheme must have a decryption key that is at least as long as the transmitted message.

Adaptively secure oblivious transfer. [1, 10] designed semi-honest adaptively secure OT (using NCE) and then compiled it into the malicious setting using generic ZK proofs. More recently, in a weaker model that assumes erasures, Lindell [29] used the method of [35] to design an efficient transformation from any static OT to a semi-honest composable adaptively secure OT. Another recent work by Garay *et al.* [19] presented a UC adaptively secure OT, building on the static OT of [34] and somewhat NCE. This paper introduces an OT protocol with security under a weaker *semi-adaptive* notion, that is then compiled into a fully adaptively secure OT by encrypting the transcript of the protocol using somewhat NCE.[5] Finally, [12] presented an improved compiler for a UC adaptively secure OT in the malicious setting (using NCE as well).

Adaptively secure two-party computation. In the non-erasure model, adaptively secure computation has been extensively studied [10, 15, 7, 26, 24, 29, 11, 12, 20]. Starting with the work of [10], it is known by now how to compute any well-formed two-party functionality in the adaptive settings. The followup work of [15] showed how to use a threshold encryption to achieve UC adaptive security but requires honest majority. A generic compiler from static to adaptive security was shown in [7] (yet without considering post-execution corruptions). Then the work by Katz and Ostrovsky [26] studied the round complexity in the one-sided setting. Their protocol is the first round efficient construction, yet it takes the naive approach of encrypting the entire communication using NCE. Moreover, the work of [24] provided a UC adaptively secure protocol given an adaptively secure OT. Their compiler generates one-sided schemes that either require a number of adaptively secure OTs that is proportional to the circuit's size, or a number of rounds that is proportional to the depth of the circuit. Finally, a recent work by Garg and Sahai [20] shows adaptively secure constant round protocols tolerating $n-1$ out of n corrupted parties using a non-black box simulation approach. Their approach uses the OT hybrid compiler of [24].

In the erasure model, 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 [29] presents an efficient semi-honest constant round two-party protocol with adaptive security.

2 Preliminaries

We denote the security parameter by n. A function $\mu(\cdot)$ is *negligible* if for every polynomial $p(\cdot)$ there exists a value N such that for all $n > N$ it holds that $\mu(n) < \frac{1}{p(n)}$. We denote by $a \leftarrow A$ the random sampling of element a from a set A and write PPT for probabilistic polynomial-time. We denote the message spaces of our schemes and the message space of the sender in our OT protocols by $\{0,1\}^q$ for $q = O(n)$.

[5] 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.

Definition 5 (Computational indistinguishability) *Let* $X = \{X_n(a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ *and* $Y = \{Y_n(a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ *be distribution ensembles. We say that* X *and* Y *are* computationally indistinguishable*, denoted* $X \approx_c Y$*, if for every family* $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *of polynomial-size circuits, there exists a negligible function* $\mu(\cdot)$ *such that for all* $a \in \{0,1\}^*$, $|\Pr[\mathcal{C}_n(X_n(a)) = 1] - \Pr[\mathcal{C}_n(Y_n(a)) = 1]| < \mu(n)$.

We denote a PKE by three algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. 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 encrypting n times in parallel using Enc. In this work we do not abuse this definition and achieve a single basic operation relative to algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, which are implemented by $O(1)$ group exponentiations in various group descriptions.

2.1 Simulatable Public Key Encryption

A simulatable public key encryption scheme is a semantically secure PKE with four additional algorithms. I.e., an oblivious public key generator $\widetilde{\mathsf{Gen}}$ and a corresponding key faking algorithm $\widetilde{\mathsf{Gen}}^{-1}$, and an oblivious ciphertext generator $\widetilde{\mathsf{Enc}}$ and a corresponding ciphertext faking algorithm $\widetilde{\mathsf{Enc}}^{-1}$. Intuitively, the key faking algorithm is used to explain a legitimately generated public key as an obliviously generated public key. Similarly, the ciphertext faking algorithm is used to explain a legitimately generated ciphertext as an obliviously generated one.

Definition 6 (Simulatable PKE [13]) *A Simulatable PKE is a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Gen}}^{-1}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ *that satisfy the following properties:*

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

The El Gamal PKE [18] is one example for simulatable PKE.

2.2 Dual-Mode PKE

A dual-mode PKE Π_{DUAL} is specified by the algorithms $(\mathsf{Setup}, \mathsf{dGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{FindBranch}, \mathsf{TrapKeyGen})$ described below.

- Setup is the system parameters generator algorithm. Given a security parameter n and a mode $\mu \in \{0,1\}$, the algorithm outputs (CRS, t). The CRS is a common string for the remaining algorithms, and t is a trapdoor value that is given to either $\mathsf{FindBranch}$ or $\mathsf{TrapKeyGen}$, depends on the mode. The setup algorithms for messy and decryption modes are denoted by $\mathsf{SetupMessy}$ and $\mathsf{SetupDecryption}$, respectively; namely $\mathsf{SetupMessy} := \mathsf{Setup}(1^n, 0)$ and $\mathsf{SetupDecryption} := \mathsf{Setup}(1^n, 1)$.

– dGen is the key generation algorithm that takes a bit α and the CRS as input. If $\alpha = 0$, then it generates left public and secret key pair. Otherwise, it creates right public and secret key pair.
– dEnc is the encryption algorithm that takes a bit β, a public key PK and a message m as input. If $\beta = 0$, then it creates the left encryption of m, else it creates the right encryption.
– dDec decrypts a message given a ciphertext and a secret key SK.
– $\mathsf{FindBranch}$ finds whether a given public key (in messy mode) is left key or right key given the messy mode trapdoor t.
– $\mathsf{TrapKeyGen}$ generates a public key and two secret keys using the decryption mode trapdoor t such that both left encryption as well as the right encryption using the public key can be decrypted using the secret keys.

Definition 7 (Dual-mode PKE) *A dual-mode PKE is a tuple of algorithms described above that satisfy the following properties:*

1. **Completeness.** *For every mode* $\mu \in \{0,1\}$*, every* $(\mathit{CRS}, t) \leftarrow \mathsf{Setup}(1^n, \mu)$*, every* $\alpha \in \{0,1\}$*, every* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{dGen}(\alpha)$*, and every* $m \in \{0,1\}^\ell$*, decryption is correct when the public key type matches the encryption type, i.e.,* $\mathsf{dDec}_{\mathrm{SK}}(\mathsf{dEnc}_{\mathrm{PK}}(m, \alpha)) = m$*.*
2. **Indistinguishability of modes.** *The CRS generated by* $\mathsf{SetupMessy}$ *and* $\mathsf{SetupDecryption}$ *are computationally indistinguishable, i.e.,* $\mathsf{SetupMessy}(1^n) \approx_c \mathsf{SetupDecryption}(1^n)$*.*
3. **Trapdoor extraction of key type (messy mode).** *For every* $(\mathit{CRS}, t) \leftarrow \mathsf{SetupMessy}(1^n)$ *and every (possibly malformed)* PK*,* $\mathsf{FindBranch}(t, \mathrm{PK})$ *outputs the public key type* $\alpha \in \{0,1\}$*. Encryption at branch* $1-\alpha$ *is then message-lossy; namely, for every* $m_0, m_1 \in \{0,1\}^\ell, \mathsf{dEnc}_{\mathrm{PK}}(m_0, 1-\alpha) \approx_s \mathsf{dEnc}_{\mathrm{PK}}(m_1, 1-\alpha)$*.*
4. **Trapdoor generation of keys decrypt both branches (decryption mode).** *For every* $(\mathit{CRS}, t) \leftarrow \mathsf{SetupDecryption}(1^n)$*,* $\mathsf{TrapKeyGen}(t)$ *outputs* $(\mathrm{PK}, \mathrm{SK}_0, \mathrm{SK}_1)$ *such that for every* α*,* $(\mathrm{PK}, \mathrm{SK}_\alpha) \approx_c \mathsf{dGen}(\alpha)$*.*

2.3 NCE for the Receiver

An NCE for the receiver is a semantically secure PKE with an additional property that enables generating a secret key that decrypts a *simulated* (i.e., 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. The “simulated mode” enables to generate simulated ciphertexts that are computationally indistinguishable from real ciphertexts. Moreover, using a special trapdoor one can produce a secret key that decrypts a fake ciphertext into any plaintext. Intuitively, this implies that simulated ciphertexts are generated in a lossy mode where the plaintext is not well defined given the ciphertext and the public key. This leaves enough entropy for the secret key to be sampled in a way that determines the desired plaintext. Formally,

Definition 8 (NCE for the receiver (NCER)) *An NCE for the receiver encryption scheme is a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^*, \mathsf{Dec}, \mathsf{Equivocate})$ *specified as follows:*

– $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are as specified for public key encryption scheme.*

– 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, *output* SK^* *such that* $m \leftarrow \mathsf{Dec}_{\mathrm{SK}^*}(c^*)$.

Definition 9 (Secure NCER) *An NCE for the receiver* $\Pi_{\mathrm{NCR}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Enc}^*, \mathsf{Equivocate})$ *is* secure *if it satisfies the following properties:*

– $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are as specified in the standard semantically secure encryption scheme.*
– *The following* ciphertext indistinguishability *holds for any plaintext* m*:* $(\mathrm{PK}, \mathrm{SK}^*, c^*, m)$ *and* $(\mathrm{PK}, \mathrm{SK}, c, m)$ *are computationally indistinguishable, for* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$, $(c^*, t_{c^*}) \leftarrow \mathsf{Enc}^*(\mathrm{PK})$, $\mathrm{SK}^* \leftarrow \mathsf{Equivocate}(\mathrm{SK}, c^*, t_{c^*}, m)$ *and* $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)$.

A review of two implementations of NCER under the DDH [25, 9] and DCR [9] assumptions is found in our full version [23].

2.4 NCE for the Sender

NCE for the sender is a semantically 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" that enables to encrypt and decrypt as in standard PKEs and the "simulated mode" that enables to generate simulated public keys and an additional trapdoor, such that the keys in the two modes are computationally indistinguishable. In addition, given this trapdoor and a ciphertext generated using the simulated public key, one can produce randomness that is consistent with any plaintext. We continue with a formal definition.

Definition 10 (NCE for the sender (NCES)) *An NCE for the sender encryption scheme is a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ *specified as follows:*

– $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are as specified for public key encryption scheme.*
– 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, *output* r *such that* $c^* \leftarrow \mathsf{Enc}(m, r)$.

Definition 11 (Secure NCES) *An NCE for the sender* $\Pi_{\mathrm{NCES}} = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ *is* secure *if it satisfies the following properties:*

– $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are as specified in the standard semantically secure encryption scheme.*
– *The following* public key indistinguishability *holds for any plaintext* m*:* $(\mathrm{PK}^*, r^*, m, c^*)$ *and* (PK, r, m, c) *are computationally indistinguishable, for* $(\mathrm{PK}^*, t_{\mathrm{PK}^*}) \leftarrow \mathsf{Gen}^*(1^n)$, $c^* \leftarrow \mathsf{Enc}_{\mathrm{PK}^*}(m', r')$, $r^* \leftarrow \mathsf{Equivocate}(c^*, t_{\mathrm{PK}^*}, m)$ *and* $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m, r)$.

A review of the DDH based implementation from [5] and a new DCR based implementation is found in our full version [23].

2.5 Somewhat Non-Committing Encryption [19]

The idea of somewhat NCE is to exploit the fact that it is often unnecessary for the simulator to explain a fake ciphertext for *any* plaintext. Instead, in many scenarios it suffices to explain a fake ciphertext with respect to a small set of size ℓ determined in advance (where ℓ might be as small as 2). Therefore there are two parameters that are considered here: a plaintext of bit length l and an equivocality parameter ℓ which is the number of plaintexts that the simulator needs to explain a ciphertext for (namely, the non-committed domain size). Note that for NCE $\ell = 2^l$. Somewhat NCE typically improves over fully NCE whenever ℓ is very small but the plaintext length is still large, say $O(n)$ where n is the security parameter.

3 One-sided Adaptively Secure NCE

In this section we design one-sided NCE, building on NCE for the sender (NCES) and NCE for the receiver (NCER). 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. Therefore, equivocality of the sender's input can be achieved if the public key of the NCES is fake, whereas, equivocality of the receiver's input can be achieved if the ciphertext of the NCER is fake. Nevertheless, this idea only works if the simulator is aware of the identity of the corrupted party prior to the protocol execution in order to decide whether the keys or the ciphertexts should be explained as valid upon corruption (since it cannot explain fake keys/ciphertext as valid). We resolve this problem using somewhat NCE in order to commit to *the choice* of having fake/valid keys and ciphertexts. Specifically, it enables the simulator to "explain" fake keys/ciphertext as valid and vice versa using only a constant number of asymmetric operations, as each such non-committing bit requires an equivocation space of size 2.

Formally, denote by $\mathcal{F}_{\mathrm{SC}}\,(m,-) \mapsto (-,m)$ the secure message transfer functionality, and let $\Pi_{\mathrm{NCES}} = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Equivocate})$ and $\Pi_{\mathrm{NCER}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^*, \mathsf{Dec}, \mathsf{Equivocate})$ denote secure NCES and NCER for a message space $\{0,1\}^q$. Consider the following one-sided protocol for $\mathcal{F}_{\mathrm{SC}}$.

Protocol 1 (One-sided NCE (Π_{OSC}))

- **Inputs:** *Sender* SEN *is given input message* $m \in \{0,1\}^q$*. Both parties are given security parameter* 1^n.
- **The Protocol:**
  1. **Message from the receiver.** REC *invokes* $\mathsf{Gen}(1^n)$ *of* Π_{NCES} *and* Π_{NCER} *and obtains two public/secret key pairs* $(\mathrm{PK}_0, \mathrm{SK}_0)$ *and* $(\mathrm{PK}_1, \mathrm{SK}_1)$*, respectively.* REC *sends* PK_1 *on clear and* PK_0 *using somewhat NCE with equivocality parameter* $\ell = 2$.
  2. **Message from the sender.** *Upon receiving* PK_0 *and* PK_1, SEN *creates two shares of* m*,* m_0 *and* m_1*, such that* $m = m_0 \oplus m_1$*. It then encrypts each* m_i *using* PK_i*, creating ciphertext* c_i*, and sends* c_0 *and* c_1 *using two instances of somewhat NCE with equivocality parameter* $\ell = 2$.

3. **Output.** *Upon receiving c_0, c_1,* REC *decrypts c_i using* SK_i *and outputs the bitwise XOR of the decrypted plaintexts.*

Note that the message space of our one-sided NCE is equivalent to the message space of the NCES/NCER schemes, where q can be as large as n. Therefore, our protocol transmits q-bits messages using a *constant number of PKE operations*, as opposed to NCEs that require $O(q)$ such operations. We provide two instantiations for the above protocol. One for polynomial-size message spaces using DDH based NCES and NCER, and another for exponential-size message spaces using DCR based NCES and NCER. We conclude with the following theorem and the complete proof.

Theorem 12 *Assume the existence of NCER and NCES with constant number of PKE operations for message space $\{0,1\}^q$ and simulatable PKE. Then Protocol 1 UC realizes $\mathcal{F}_{\text{SC}}$ in the presence of one-sided adaptive malicious adversaries with constant number of PKE operations for message space $\{0,1\}^q$, where $q = O(n)$ and n is the security parameter.*

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 real 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 its environment ENV).

SEN is corrupted at the onset of the protocol. SIM begins by activating ADV and emulates the honest receiver by sending to ADV, PK_0 using the somewhat NCE and PK_1 in clear. Upon receiving two ciphertexts c_0 and c_1 from ADV, SIM extracts m by computing $\mathsf{Dec}_{\mathrm{SK}_0}(c_0) \oplus \mathsf{Dec}_{\mathrm{SK}_1}(c_1)$. SIM externally forwards m to the ideal functionality $\mathcal{F}_{\text{SC}}$.

REC is corrupted at the onset of the protocol. SIM begins by activating ADV and obtains REC's output m from $\mathcal{F}_{\text{SC}}$. SIM invokes ADV and receives PK_0 from ADV via the somewhat NCE and PK_1 in clear. 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 invalid public keys since SIM knows m and thus perfectly emulates the receiver's view.

If none of the parties is corrupted as above, SIM emulates the receiver's message as follows. It creates public/secret key pair $(\mathrm{PK}_1, \mathrm{SK}_1)$ for Π_{NCER} and sends the public key in clear. It then creates a valid public/secret key pair $(\mathrm{PK}_0, \mathrm{SK}_0)$ and a fake public key with a trapdoor $(\mathrm{PK}_0^*, t_{\mathrm{PK}_0^*})$ for Π_{NCES} (using Gen and Gen^*, respectively). SIM sends $(\mathrm{PK}_0, \mathrm{PK}_0^*)$ using somewhat NCE. Namely, the simulator does not send the valid

PK_0 as the honest receiver would do, rather it encrypts both valid and invalid keys within the somehwat NCE.

SEN is corrupted after Step 1 is concluded. Since no message was sent yet on behalf of the sender, SIM completes the simulation playing the role of the honest sender using m.

REC is corrupted after Step 1 is concluded. Upon receiving m, SIM explains the receiver's internal state which is independent of the message m so far. Specifically, it reveals the randomness for generating $\mathrm{PK}_0, \mathrm{SK}_0$ and $\mathrm{PK}_1, \mathrm{SK}_1$ and presents the randomness for the valid key PK_0 being the message sent by the somewhat NCE. SIM plays the role of the honest sender with input m as the message.

If none of the above corruption cases occur, SIM emulates the sender's message as follows. It first chooses two random shares m'_0, m'_1 and generates a pair of ciphertexts (c_0, c^*_0) for Π_{NCES} that encrypts m'_0 using PK_0 and PK^*_0. It then generates a pair of ciphertexts (c_1, c^*_1) for Π_{NCER} such that c_1 is a valid encryption of m'_1 using the public key PK_1, and c^*_1 is a fake ciphertext generated using Enc^* and PK_1. SIM sends (c_0, c^*_0) and (c^*_1, c_1) via two instances of somewhat NCE.

SEN is corrupted after Step 2 is concluded. Upon receiving a corruption instruction from ENV, SIM corrupts the ideal SEN and obtains SEN's input m. It then explains the sender's internal state as follows. It explains PK^*_0 for being the public key sent by the receiver using the somewhat NCE. Furthermore, it presents the randomness for c^*_0 and c_1 being the ciphertexts sent via the somewhat NCE. Finally, it computes $r'' \leftarrow \mathsf{Equivocate}_{\mathrm{PK}^*_0}(t_{\mathrm{PK}^*_0}, m'_0, r, m''_0)$ for m''_0 such that $m = m''_0 \oplus m'_1$ and r the randomness used to encrypt m'_0, and presents r'' as the randomness used to generate c^*_0 that encrypts m''_0. The randomness used for generating c_1 is revealed honestly.

REC is corrupted after Step 2 is concluded. Upon receiving a corruption instruction from ENV, SIM corrupts the ideal REC and obtains REC's output m. It then explains the receiver's internal state as follows. It presents the randomness for PK_0 for being the public key sent via the somewhat NCE and presents the randomness for generating $(\mathrm{PK}_0, \mathrm{SK}_0)$. It then explains c_0 and c^*_1 for being sent via the somewhat NCE. Finally, it generates a secret key SK^*_1 so that $m''_1 \leftarrow \mathsf{Dec}_{\mathrm{SK}^*_1}(c^*_1)$ and $m''_1 \oplus m'_0 = m$. That is, it explains $(\mathrm{PK}_1, \mathrm{SK}^*_1)$ as the other pair of keys generated by the receiver.

We now show that for every corruption case described above, there is not any polynomial-time ENV that distinguishes with a non-negligible probability the real execution with ADV and the simulated execution with SIM.

SEN/REC is corrupted at the onset of the protocol. In these corruption cases there is no difference between the real execution and the simulated execution and the views are statistically indistinguishable.

SEN/REC is corrupted after Step 1 is concluded. In these cases the only difference between the real and simulated executions is with respect to the somewhat NCE that delivers the public key of NCES. Specifically, in the real execution it always delivers a valid public key while in the simulated execution it delivers a fake key. Indistinguishability follows from the security of the somewhat NCE.

SEN/REC is corrupted after Step 2 is concluded. Here the adversary sees in the simulation either a fake public key or a fake ciphertext. Indistinguishability follows from the security of Π_{NCES} and Π_{NCER} and the security of somewhat NCE.

■

4 One-Sided Adaptively Secure OT

A common approach to design an adaptive OT [2, 10] is by having the receiver generate two public keys $(\text{PK}_0, \text{PK}_1)$ such that it only knows the secret key associated with PK_σ. The sender then encrypts x_0, x_1 under these respective keys so that the receiver decrypts the σth ciphertext. The security of this protocol in the adaptive setting holds if the underlying encryption scheme is an *augmented non-committing encryption scheme* [10]. In this section we follow the approach from [19] and build one-sided OT based on the static OT from [34], which is based on a primitive called *dual-mode PKE*.

The PVW OT. Dual-mode PKE is a semantically secure encryption scheme that is initialized with system parameters of two types. For each type one can generate two types of public/secret key pair, labeled by the left key pair and the right key pair. Similarly, the encryption algorithm generates a left or a right ciphertext. Moreover, if the key label matches the ciphertext label (i.e., a left ciphertext is generated under the left public key), then the ciphertext can be correctly decrypted. (A formal definition of dual-mode PKE is given in Section 2.2.) This primitive was introduced in [34] which demonstrates its usefulness in designing efficient statically secure OTs under various assumptions. First, the receiver generates a left key if $\sigma = 0$, and a right key otherwise. In response, the sender generates a left ciphertext for x_0 and a right ciphertext for x_1. The receiver then decrypts the σth ciphertext.

The security of dual-mode PKE relies on the two indistinguishable modes of generating the system parameters: *messy* and *decryption* mode. In a messy mode the system parameters are generated together with a messy trapdoor. Using this trapdoor, any public key (even malformed ones) can be labeled as a left or as a right key. Moreover, when the key type does not match the ciphertext type, the ciphertext becomes statistically independent of the plaintext. The messy mode is used to ensure security when the receiver is corrupted since it allows to extract the receiver's input bit while hiding the sender's other input. On the other hand, the system parameters in a decryption mode are generated together with a decryption trapdoor that can be used to decrypt both left and right ciphertexts. Moreover, left public keys are statistically indistinguishable from right keys. The decryption mode is used to ensure security when the sender is corrupted since the decryption trapdoor enables to extract the sender's inputs while statistically hiding the receiver's input. [34] instantiated dual-mode PKE and their generic OT construction based on various assumptions, such as DDH, QR and lattice-based assumptions.

Our construction. We build our one-sided OT based on the PVW protocol considering the following modifications. (**1**) First, we require that the sender sends its ciphertexts using one-sided NCE (see Section 3). (**2**) We fix the system parameters in a decryption mode, which immediately implies extractability of the sender's input and equivocality of the receiver's input. We further achieve equivocality of the sender's input using our one-sided NCE. In order to ensure extractability of the receiver's input we employ a special type of ZK PoK. Namely, this proof exploits the fact that the simulator knows both witnesses for the proof yet it does not know which witness will be used by the real receiver, since this choice depends on σ. Specifically, it allows the simulator to use both witnesses and later convince the adversary that it indeed used a particular witness. In addition, it enables to extract σ since the real receiver does not know both witnesses. We denote these proofs for compound statements by witness equivocal and refer to Section 6 for more details.

Our construction is one-sided UC secure in the presence of malicious adversaries, and uses a number of non-committed bits that is *independent* of the sender's input size or the overall communication complexity. We formally denote the dual-mode PKE of [34] by $\Pi_{\text{DUAL}} = (\mathsf{SetupMessy}, \mathsf{SetupDecryption}, \mathsf{dGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{FindBranch}, \mathsf{TrapKeyGen})$ and describe our construction in the $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}})$-hybrid model, where $\mathcal{F}_{\text{SC}}$ is instantiated with one-sided NCE. Furthermore, the latter functionality is required to ensure the correctness of the public key and is defined for a compound statement that is comprised from the following two relations,

$$\mathcal{R}_{\text{LEFT}} = \big\{(\text{PK}, r_0) \mid (\text{PK}, \text{SK}) \leftarrow \mathsf{dGen}(\text{CRS}, 0; r_0)\big\},$$

where CRS are the system parameters. Similarly, we define $\mathcal{R}_{\text{RIGHT}}$ for the right keys. Specifically, $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$ receives a public key PK and randomness r_σ for $\sigma \in \{0,1\}$ and returns `Accept` if either $\sigma = 0$ and $\text{PK} = \mathsf{dGen}(\text{CRS}, 0; r_0)$, or $\sigma = 1$ and $\text{PK} = \mathsf{dGen}(\text{CRS}, 1; r_1)$ holds. Security is proven by implementing this functionality using a witness equivocal ZK PoK that allows the simulator to equivocate the witness during the simulation (i.e., explaining the proof transcript with respect to either r_0 or r_1). We consider two instantiations of dual-mode PKE (based on the DDH and QR assumptions). For each implementation we design a concrete ZK PoK, proving that the prover knows r_σ with respect to $\sigma \in \{0,1\}$; see details below.

We define our OT protocol as follows,

Protocol 2 (One-sided OT (Π_{OT}))

- **Inputs:** *Sender* SEN *has* $x_0, x_1 \in \{0,1\}^q$ *and receiver* REC *has* $\sigma \in \{0,1\}$.
- **CRS:** CRS *such that* $(\text{CRS}, t) \leftarrow \mathsf{SetupDecryption}$.
- **The Protocol:**
  1. REC *sends* SEN PK*, where* $(\text{PK}, \text{SK}) \leftarrow \mathsf{dGen}(\text{CRS}, \sigma; r_\sigma)$. REC *calls* $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$ *with* (PK, r_σ).
  2. *Upon receiving* `Accept` *from* $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$ *and* PK *from* REC*,* SEN *generates* $c_0 \leftarrow \mathsf{dEnc}_{\mathsf{PK}}(x_0, 0)$ *and* $c_1 \leftarrow \mathsf{dEnc}_{\mathsf{PK}}(x_1, 1)$. SEN *calls* $\mathcal{F}_{\text{SC}}$ *twice with inputs* c_0 *and* c_1*, respectively.*
  3. *Upon receiving* (c_0, c_1)*,* REC *outputs* $\mathsf{dDec}_{\mathsf{SK}}(c_\sigma)$.

Theorem 13 *Assume the existence of one-sided NCE with constant number of PKE operations for message space* $\{0,1\}^q$ *and dual-mode PKE. Then Protocol 2 UC realizes* $\mathcal{F}_{\text{OT}}$ *in the* $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}})$*-hybrid model in the presence of one-sided adaptive malicious adversaries with constant number of PKE operations for sender's input space* $\{0,1\}^q$*, where* $q = O(n)$ *and* n *is the security parameter.*

Proof: Let ADV be a probabilistic polynomial-time malicious adversary attacking Protocol2 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 real 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 $(\text{PK}, \text{SK}_0, \text{SK}_1) \leftarrow \mathsf{TrapKeyGen}(t)$. It then sends PK and an `Accept` message to ADV on behalf of $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$. Whenever ADV returns c_0, c_1 via $\mathcal{F}_{\text{SC}}$, SIM extracts SEN's inputs x_0, x_1 by invoking $\mathsf{dDec}_{\text{SK}_0}(c_0)$ and $\mathsf{dDec}_{\text{SK}_1}(c_1)$ as in static case. It then sends x_0, x_1 to $\mathcal{F}_{\text{OT}}$ 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 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 right public key. Furthermore, the keys (PK, 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 its public key PK and a witness r_σ on behalf of $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$. Given r_σ, SIM checks if PK is the left or the right key and use it to extract the receiver's input σ. It then sends σ to $\mathcal{F}_{\text{OT}}$, receiving back x_σ. Finally, SIM computes the sender's message using x_σ and an arbitrary $x_{1-\sigma}$.

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 $x_{1-\sigma}$. Therefore, the hybrid view is also computationally indistinguishable from the simulated view as in the static setting proven in [34].

If none of the above corruption cases occur SIM invokes $(\text{PK}, \text{SK}_0, \text{SK}_1) \leftarrow \mathsf{TrapKeyGen}(t)$ and sends PK to the sender. Note that the simulator knows a witness r_0 such that $\text{PK} = \mathsf{dGen}(\text{CRS}, 0; r_0)$ and a witness r_1 such that $\text{PK} = \mathsf{dGen}(\text{CRS}, 1; r_1)$.

SEN is corrupted between Steps 1 and 2. SIM trivially explains the 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 arbitrary 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.

REC is corrupted between Steps 1 and 2. Upon corrupting the receiver $\textsc{Sim}$ obtains σ, x_σ from $\mathcal{F}_{\text{OT}}$ and explains the receiver's internal state as follows. It first explains r_σ as the witness given to $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$ and PK as the outcome of $\mathsf{dGen}(\text{CRS}, \sigma; r_\sigma)$. The simulator completes the simulation playing the role of the honest sender with x_σ and an arbitrary $x_{1-\sigma}$.
Indistinguishability for this case in the hybrid setting follows similarly to the prior corruption case, since the only difference in the simulation is relative to the witness equivocality proof which only makes a difference in the real execution.

If none of the above corruption cases occur then $\textsc{Sim}$ 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. $\textsc{Sim}$ pretends sending these ciphertexts using $\mathcal{F}_{\text{SC}}$.

SEN is corrupted after Step 2. Upon corrupting the sender, $\textsc{Sim}$ obtains (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 encrypts x_0 and x_1 respectively. It then explains c_0 and c_1 as being sent using $\mathcal{F}_{\text{SC}}$.
In the hybrid setting indistinguishability follows as in the prior corruption case of the sender, since the simulator emulates the sender's message via the one-sided non-committing channel. In the real execution, security is reduced to the security of the one-sided encryption scheme implementation.

REC is corrupted after Step 2. Upon corrupting the receiver, $\textsc{Sim}$ obtains REC's input and output (σ, x_σ) from $\mathcal{F}_{\text{OT}}$. It then explains the receiver's internal state as follows. It first explains r_σ as the witness given to $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{LR}}}$ and PK as the outcome of $\mathsf{dGen}(\text{CRS}, \sigma; r_\sigma)$. Finally, it explains the output of $\mathcal{F}_{\text{SC}}$ as c_σ so that c_σ is indeed a valid encryption of x_σ.
Indistinguishability follows similarly to the prior corruption case of the receiver since the second message is computed by the sender which is not corrupted.

■

Concrete instantiations. In the DDH-based instantiation 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 concrete ZK PoK functionality is $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{DL,OR}}}$ which is invoked with the statement and witness $\big(((g_0 h_0, g_\sigma^r h_\sigma^r), (g_1 h_1, g_\sigma^r h_\sigma^r)), r\big)$, such that $\text{PK} = (g_\sigma^r, h_\sigma^r)$, $\text{SK} = r$ and $r \leftarrow \mathbb{Z}_p$.

In the QR-based instantiation the CRS is a quadratic residue y and the trapdoor is s such that $y = s^2 \bmod N$ and N is an RSA composite. The concrete ZK PoK functionality is $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{QR,OR}}}$ which is invoked 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 Constant Round One-Sided Adaptively Secure Computation

In the following section we demonstrate the feasibility of one-sided adaptively secure two-party protocols in the presence of semi-honest and malicious adversaries. Our constructions are constant round and UC secure and use a number of non-committed bits that is independent of the circuit size, thus reduce the number of PKE operations so that it only depends on the input and output lengths. A high-level overview of Yao's garbled circuit construction $G(C)$ for a circuit C is found in the full version.

5.1 One-Sided Secure Computation for Semi-Honest Adversaries

Our first construction adapts the semi-honest two-party protocol [36, 28] into the one-sided adaptive setting at a cost of $\mathcal{O}(|C|)$ private key operations and $\mathcal{O}(|\mathsf{input}| + |\mathsf{output}|)$ public key operations. Using our one-sided secure primitives we obtain efficient protocols that outperform the constant round one-sided constructions of [26, 24] and all known fully adaptively secure two-party protocols. Namely, we show that one-sided security can be obtained by only communicating the keys corresponding to the input/output wires via a non-committing channel. This implies that the number of PKE operations *does not* depend on the garbled circuit size as in prior work.

Informally, the input keys that correspond to P_0's input are transferred to P_1 using somewhat NCE with equivocation parameter $\ell = 2$, whereas P_1's input keys are sent using one-sided OT. Next, the entire garbled circuit (without the output decryption table) is sent to P_1 using a standard communication channel. P_1 evaluates the garbled circuit and finds the keys for the output wires. The parties then run a one-sided bit OT for each output key where P_1 plays the role of the receiver, and learns the output bit that corresponds to its output wire. Finally, P_1 sends P_0 the output using one-sided NCE. We note that obtaining the output via one-sided OT is crucial to our proof since it enables us to circumvent the difficulties arise when implementing the simulation technique from [28] that uses a fake circuit. To carry out these OTs successfully we require that the keys associated with a output wire have distinct most significant bits that are fixed independently of the bits they correspond to. For simplicity we only consider deterministic and same-output functionalities. This can be further generalized using the reductions specified in [21]. The formal description of our one-sided semi-honest protocol Π_f^{SH} is given below in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model.

Protocol 3 (One-sided adaptively secure semi-honest Yao (Π_f^{SH}))

– **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.*
– **The Protocol:**
  1. **Setup and garbling circuit computation.** *P_0 constructs garbled circuit $G(C)$ subject to the constraint that the keys corresponding to each circuit-output wire have a distinct most significant bit.*
  2. **Transferring the garbled circuit and input keys to $\mathbf{P}_1$.** *Let (k_i^0, k_i^1) be the key pair corresponding to the circuit-input wire that takes the ith bit of x_0 and let (k_{n+i}^0, k_{n+i}^1) be the key pair corresponding to the circuit-input wire that takes the ith bit of x_1. Then,*
     (a) For all $i \in [1, \ldots, n]$, P_0 sends $k_i^{x_0^i}$ using an instance of somewhat NCE with $\ell = 2$.
     (b) For all $i \in [1, \ldots, n]$, P_0 and P_1 call $\mathcal{F}_{\mathrm{OT}}$ with input (k_{n+i}^0, k_{n+i}^1) and x_1^i, respectively. Let $k_{n+i}^{x_1^i}$ denotes P_1's ith output.
     (c) P_0 sends $G(C)$ without the output decryption table to P_1.

3. **Circuit evaluation and interactive output computation.** P_1 *evaluates* $G(C)$ *on the above input keys and obtains the keys that correspond to* $f(x_0, x_1)$ *in the circuit-output wires. Let* (k^0_{2n+i}, k^1_{2n+i}) *be the key pair corresponding to the* i*th circuit-output wire with distinct most significant bits. Also assume* P_1 *obtains key* k^α_{2n+i} *corresponding to the* i*th circuit-output wire of* $G(C)$*. Then,*
   (a) For all $i \in [1, \ldots, n]$*,* P_0 *and* P_1 *call* $\mathcal{F}_{\mathrm{OT}}$ *in which* P_0*'s input equals* $(0, 1)$ *if the most significant bit of* k^0_{2n+i} *is* 0*, and* $(1, 0)$ *otherwise.* P_1*'s input is the most significant bit of* k^α_{2n+i}*.*
   (b) P_1 *computes* $f(x_0, x_1)$ *by concatenating the bits received from the above* n *calls.*
4. **Output communication.** P_1 *sends* y *using an instance of one-sided NCE.*

Theorem 14 (One-sided semi-honest) *Let* f *be a deterministic same-output functionality and assume that the encryption scheme for garbling has indistinguishable encryptions under chosen plaintext attacks, and an elusive and efficiently verifiable range. Furthermore, assume that* $\mathcal{F}_{\mathrm{OT}}$ *is realized in the presence of one-sided semi-honest adversaries with constant number of PKE operations for sender's input space* $\{0, 1\}^q$*, where* $q = O(n)$ *and* n *is the security parameter. Then Protocol 3 UC realizes* $\mathcal{F}_f$ *in the presence of one-sided semi-honest adversaries at a cost of* $\mathcal{O}(|C|)$ *private key operations and* $\mathcal{O}(|\mathsf{input}| + |\mathsf{output}|)$ *public key operations.*

We note that the ideal OT calls in Step 2 can be realized using string one-sided OTs, whereas the OT calls in Step 3 can be replaced with bit one-sided OTs. The complete proof can be found in our full version [23].

5.2 Security against Malicious Adversaries

Next, we modify Π^{SH}_f and adapt the cut-and-choose OT protocol introduced in [30] in order to achieve security against malicious adversaries. The idea of the cut-and-choose technique is to ask P_0 to send 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 [30] 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 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 [30] protocol ensures that the keys associated with P_0's input are obtained via a Diffie-Hellman pseudorandom synthesizer [31]. Namely, P_0 chooses $g^{a^0_1}, g^{a^1_1}, \ldots, g^{a^0_n}, g^{a^1_n}$ 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 $(g^{a^0_i c_j}, g^{a^1_i c_j})$.[6] Given values $\{g^{a^0_i}, g^{a^1_i}, g^{c_j}\}$ 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

[6] The actual key pair used in the circuit garbling is derived from $(g^{a^0_i c_j}, g^{a^1_i c_j})$ using an extractor. A universal hash function is used in [30] for this purpose, where the seeds for the function are picked by P_0 before it knows $\mathcal{J}$.

then evaluates the evaluation circuits and takes the majority value for the final output. In Section 5.2 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. Formally,

Theorem 15 (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 at a cost of $\mathcal{O}(s \cdot |C|)$ private key operations and $\mathcal{O}(s \cdot (|\mathsf{input}| + |\mathsf{output}|))$ public key operations where s is a statistical parameter that determines the cut-and-choose soundness error.*

Specifically, the concrete DCR assumption implies cut-and-choose OT with constant number of PKE operations for sender's input space $\{0,1\}^q$, where $q = O(n)$ and n is the security parameter.

One-sided Single Choice Cut-and-Choose OT We describe next the single choice cut-and-choose OT functionality $\mathcal{F}_{\text{CCOT}}$ from [30] and present a protocol that implements this functionality with UC one-sided malicious security. We then 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 our 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 [30] by invoking the DDH based [34] OT s times, where the receiver generates the system parameters in a decryption mode for $s/2$ indices corresponding to $\mathcal{J}$ and the remaining system parameters are generated in a messy mode. The decryption mode trapdoor enables the receiver to learn both sender's inputs for the instances corresponding to $\mathcal{J}$. This idea is coupled with two proofs that are run by the receiver: (**i**) a ZK PoK for proving that half of the system parameters set is in a messy mode which essentially boils down to a ZK PoK realizing functionality $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{DDH,COMP}(s,s/2)}}$ (namely, the statement is a set of s tuples and the prover proves the knowledge of $s/2$ Diffie-Hellman tuples within this set). (**ii**) A ZK PoK to ensure that the same input bit σ has been used for all s instances which boils down to a ZK proof realizing functionality $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{DDH,OR}(s)}}$ (namely, the statement contains two sets of tuples, each of size s, for which the prover proves that one of the sets contains DH tuples).

Our first step towards making the [30] construction one-sided adaptively secure is to invoke our one-sided OT scheme s times with all system parameters in a decryption mode. Notably, we cannot use the messy mode for the $s/2$ instances not in $\mathcal{J}$ as in the static settings since that would preclude the equivocation of the receiver's bit. Similarly to [30], our constructions have 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 $s/2$ OTs corresponding 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, so that the receiver sets the first component of every CRS from the system parameters to be g_0. Moreover, the second component in positions $j \notin \mathcal{J}$ is a power of g_1, else this element is a power of g_0. Note 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. This is achieved using a witness equivocal proof for functionality $\mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DL,COMP}(s,s/2)}}$.

In the transfer phase, the receiver uses these system parameters to create a public/secret key pair for each OT execution, for keys not in the set $\mathcal{J}$. For the rest of the OT executions the receiver invokes the $\mathsf{TrapKeyGen}$ algorithm of the dual-mode PKE and obtains a public key and two secret keys that enable it to decrypt both of the sender's inputs. In order to ensure that the receiver uses the same input bit σ for all OTs the receiver proves its behavior using a proof for functionality $\mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DDH,OR}(s)}}$. To ensure one-sided security, the proof if further witness equivocal (see Section 6). Finally, we prove the equivocality of the sender's input and the receiver's output based on our one-sided NCE.

Formally, denote by $\Pi_{\mathrm{DUAL}} = (\mathsf{SetupMessy}, \mathsf{SetupDecryption}, \mathsf{dGen}, \mathsf{dEnc}, \mathsf{dDec}, \mathsf{FindBranch}, \mathsf{TrapKeyGen})$ the DDH based dual-mode PKE of [34]. We present our one-sided OT Π_{CCOT} in the $(\mathcal{F}_{\mathrm{SC}}, \mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DL,COMP}(s,s/2)}}, \mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DDH,OR}(s)}})$-hybrid model.

Protocol 4 (One-sided adaptive single choice cut-and-choose OT (Π_{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* $\mathrm{CRS}_j = \big(g_0, g_1^j, h_0^j = (g_0)^{y_j}, h_1^j = (g_1^j)^{y_j}\big)$.
     It then sends $\{\mathrm{CRS}_j\}_{j=1}^s$ *to* SEN.
     Furthermore, for all $j \in \mathcal{J}$, REC *stores the decryption mode trapdoor* $t_j = x_j$.
  2. REC *calls* $\mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DL,COMP}(s,s/2)}}$ *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* $\mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DDH,OR}(s)}}$ *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* $c_0^j \leftarrow \mathsf{dEnc}_{\mathsf{PK}_j}(x_0^j, 0)$ *and* $c_1^j \leftarrow \mathsf{dEnc}_{\mathsf{PK}_j}(x_1^j, 1)$. *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* c_{01}^j *and* c_{11}^j.
– **Output:** *Upon receiving* (c_{01}^j, c_{11}^j) *from* $\mathcal{F}_{\mathrm{SC}}$,
   1. REC *outputs* $x_\sigma^j \leftarrow \mathsf{dDec}_{\mathsf{SK}_j}(c_\sigma^j)$ *for all* $j \notin \mathcal{J}$.
   2. REC *outputs* $(x_0^j, x_1^j) \leftarrow (\mathsf{dDec}_{\mathsf{SK}_j^0}(c_0^j), \mathsf{dDec}_{\mathsf{SK}_j^1}(c_1^j))$ *for all* $j \in \mathcal{J}$.

Theorem 16 *Assume that the DDH assumption is hard in* $\mathbb{G}$. *Then Protocol 4 UC realizes* $\mathcal{F}_{\mathrm{CCOT}}$ *in the* $(\mathcal{F}_{\mathrm{SC}}, \mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DL,COMP}(s,s/2)}}, \mathcal{F}_{\mathrm{ZKPoK}}^{\mathcal{R}_{\mathrm{DDH,OR}(s)}})$*-hybrid model in the presence of one-sided malicious adversaries.*

The complete proof can be found in our full version [23].

Malicious One-Sided Adaptively Secure Two-Party Computation First, we remark that the single choice cut-and-choose protocol is executed for every input bit of P_1 in the main two-party computation 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. We 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. We denote the batch functionality by $\mathcal{F}_{\mathrm{CCOT}}^{\mathrm{BATCH}}$ and the protocol by $\Pi_{\mathrm{CCOT}}^{\mathrm{BATCH}}$.

We are now ready to describe the steps of our generic protocol Π_f^{MAL} computing any functionality f on inputs x_0 and x_1. We continue with a high-level overview of [30] adapted to the one-sided setting.

Step 1. P_0 constructs s copies of Yao's garbled circuit for computing the function f. All wires keys are picked at random. Keys that are associated with P_0's input wires are picked as follows. P_0 picks n pairs of random values $((a_1^0, a_1^1), \ldots, (a_n^0, a_n^1))$ and $(c_1, \ldots, c_s)$ and sets the keys associated with the ith input wire of the jth circuit as the pair $(g^{a_i^0 c_j}, g^{a_i^1 c_j})$. These values constitute commitments to all $2ns$ keys of P_0.[7] This set of keys forms a pseudorandom synthesizer [31], implying that if some subset of the keys is revealed then the remaining keys are still pseudorandom. We also require that each pair of keys that is associated with a circuit output wire differs within the most significant bit.

Step 2. The parties call $\mathcal{F}_{\mathrm{CCOT}}^{\mathrm{BATCH}}$ where P_0 inputs the key pairs associated with P_1's input and P_1 inputs its input x_1 and a random subset $\mathcal{J} \subset [s]$ of size $s/2$. P_1 receives from $\mathcal{F}_{\mathrm{CCOT}}^{\mathrm{BATCH}}$ the keys that are associated with its input wires for the $s/2$ circuits indexed by $\mathcal{J}$ (denoted the check circuits). In addition, it receives the keys corresponding to its input for the remaining circuits (denoted the evaluation circuits).

Step 3. P_0 sends P_1 s copies of the garbled circuit (except for the output tables) and the values $((g^{a_1^0}, g^{a_1^1}), \ldots, (g^{a_n^0}, g^{a_n^1}), (g^{c_1}, \ldots, g^{c_s}))$ which are the commitments to the input keys on the wires associated with P_0's input. At this point P_0 is committed to all the keys associated with the s circuits.

[7] Recall that the actual symmetric keys of the ith input within the jth circuit are derived from $(g^{a_i^0 c_j}, g^{a_i^1 c_j})$ using randomness extractor such as a universal hash function.

Step 4. P_1 reveals $\mathcal{J}$ and proves that it used this subset in the cut-and-choose batch OT protocol by sending the keys that are associated with P_1's first input bit in each check circuit. Note that P_1 knows the keys corresponding to both bits only for the check circuits.

Step 5. 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 $j \in \mathcal{J}$. P_1 computes the key pair $(g^{a_i^0 c_j}, g^{a_i^1 c_j})$.

Step 6. 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.

Step 7. To complete the evaluation phase P_1 is given the keys for the input wires of P_0. P_0 must be 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:

1. For every evaluation circuit C_j, P_0 sends $y_{i,j} = g^{a_i^{x_0^i} c_j}$ using an instance of somewhat NCE with $\ell = 2$, where x_0^i is the ith input bit of P_0.
2. 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 $\{(g, g^{a_i^{x_0^i}}, g^{c_j}, y_{i,j})\}_{j=1}^s$ or the set $\{(g, g^{a_i^{1-x_0^i}}, g^{c_j}, y_{i,j})\}_{j=1}^s$ is comprised of DH tuples. Notably, it is sufficient to use a single UC ZK proof for the simpler relation $\mathcal{R}_{\text{DDH,OR}}$ 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 $\tilde{g} = \prod_{j=1}^s (g^{c_j})^{\gamma_j}$, $\tilde{y} = \prod_{j=1}^s (y_{i,j})^{\gamma_j}$, of which P_0 proves that either $(g, g^{a_i^{x_0^i}}, \tilde{g}, \tilde{y})$ or $(g, g^{a_i^{1-x_0^i}}, \tilde{g}, \tilde{y})$ is a DH tuple.

Step 8. Upon receiving `Accept` from $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{DDH,OR}}}$, P_1 completes the evaluation of the circuits. Namely, for every $i \in [1, \ldots, ns]$ P_0 and P_1 call $\mathcal{F}_{\text{OT}}$ in which P_0's input equals $(0, 1)$ if the most significant bit of the output wire key is associated with 0, and $(1, 0)$ otherwise. Moreover, P_1's input is the most significant bit of its output key. P_1 concatenates the bits obtained from these OTs and sets the majority of these values as the output y.

Step 9. P_1 sends y using an instance of one-sided NCE.

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)** The statement of $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{DDH,OR}}}$ is transferred in **Step 7.1** via a somewhat NCE with $\ell = 2$. To obtain a witness equivocal proof for functionality $\mathcal{F}_{\text{ZKPoK}}^{\mathcal{R}_{\text{DDH,OR}}}$ (invoked in **Step 7.2**), it is sufficient to employ a standard static proof realizing this ZK functionality where the prover sends the third message of the proof using a somewhat NCE with $\ell = 2$ (this is due to the fact that we anyway send the statement using a somewhat NCE). Specifically, a statically secure proof is sufficient whenever both the statement and the third message of the (Σ-protocol) proof can be equivocated. This implies the equivocation of P_0's input. **(3)** Finally, in **Step 8** the $\mathcal{F}_{\text{OT}}$ calls are realized using one-sided bit OT. This implies output equivocation.

6 Efficient Statically Secure and Witness Equivocal UC ZK PoKs

This section includes two results that are given in details in the full version. First, we discuss 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, in turn, enables witness extraction without using rewinding. Our instantiations imply UC ZK PoK constructions that incur constant overhead and achieve negligible soundness error. Briefly, the prover is instructed to send two responses to a pair of distinct challenges. The first response is sent on clear and publicly verified as specified in the protocol, whereas the second response is encrypted using a homomorphic PKE and its validity is carried out by a UC ZK proof of consistency.

Next, we show how to generate efficient witness equivocal UC ZK PoK for various *compound* Σ-protocols. The additional feature that witness equivocal UC ZK PoK offers over statically secure UC ZK PoK is that it allows the simulator to equivocate the simulated proof upon corrupting the prover. Interestingly, we build witness equivocal UC ZK PoKs for a class of fundamental compound Σ-protocols without relying on NCE. Our approach yields witness equivocal UC ZK PoK only for compound statements where the simulator knows the witnesses for all sub-statements (but not the real witness). This notion is weaker than the notion of 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. The rest of the details can be found in [23].

References

1. Donald Beaver. Plug and play encryption. In *CRYPTO*, pages 75–89, 1997.
2. Donald Beaver. Adaptively secure oblivious transfer. In *ASIACRYPT*, pages 300–314, 1998.
3. Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *EUROCRYPT*, pages 307–323, 1992.
4. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.
5. Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.
6. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
7. Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *J. Cryptology*, 17(3):153–207, 2004.
8. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.
9. Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *TCC*, pages 150–168, 2005.
10. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, 2002.
11. Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *ASIACRYPT*, pages 287–302, 2009.
12. Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In *TCC*, pages 387–402, 2009.

13. Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.
14. Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO*, pages 581–596, 2002.
15. Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO*, pages 247–264, 2003.
16. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.
17. Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In *EUROCRYPT*, pages 381–402, 2010.
18. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
19. Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO*, pages 505–523, 2009.
20. Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In *CRYPTO*, pages 105–123, 2012.
21. Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.
22. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
23. Carmit Hazay and Arpita Patra. One-sided adaptively secure two-party computation. *IACR Cryptology ePrint Archive*, 2013:593, 2013.
24. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.
25. Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *EUROCRYPT*, pages 221–242, 2000.
26. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.
27. Jonathan Katz, Aishwarya Thiruvengadam, and Hong-Sheng Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In *Public Key Cryptography*, pages 14–31, 2013.
28. Y. Lindell and B. Pinkas. A proof of security of yaos protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.
29. Yehuda Lindell. Adaptively secure two-party computation with erasures. In *CT-RSA*, pages 117–132, 2009.
30. Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, 2012.
31. Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of psuedo-random functions. In *FOCS*, pages 170–181, 1995.
32. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126, 2002.
33. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, pages 681–700, 2012.
34. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.
35. Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In *EUROCRYPT*, pages 222–232, 2006.
36. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.