Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs

Yevgeniy Dodis[1], Victor Shoup[1], and Shabsi Walfish[2]

[1] New York University
{dodis,shoup}@cs.nyu.edu
[2] Google
walfish@cs.nyu.edu

Abstract. Canetti et al. [7] recently proposed a new framework — termed *Generalized Universal Composability* (GUC) — for properly analyzing concurrent execution of cryptographic protocols in the presence of a global setup, and constructed the first known GUC-secure implementations of commitment (GUCC) and zero-knowledge (GUC ZK), which suffice to implement any two-party or multi-party functionality under several natural and relatively mild setup assumptions. Unfortunately, the feasibility results of [7] used rather inefficient constructions.

In this paper, we dramatically improve the efficiency of (adaptively-secure) GUCC and GUC ZK assuming data erasures are allowed. Namely, using the same minimal setup assumptions as those used by [7], we build

– a direct and efficient constant-round GUC ZK for R from any "dense" Ω-protocol [21] for R. As a corollary, we get a semi-efficient construction from any Σ-protocol for R (*without doing the Cook-Levin reduction*), and a very efficient GUC ZK for proving knowledge of a discrete log representation.
– the first *constant-rate* (and constant-round) GUCC scheme.

Additionally, we show how to properly model a random oracle in the GUC framework without losing *deniability*, which is one of the attractive features of the GUC framework. In particular, by adding the random oracle to the setup assumptions used by [7], we build the first two-round (which we show is optimal), deniable, straight-line extractable and simulatable ZK proof for any NP relation R.

1 Introduction

UC Framework. The *Universal Composability* (UC) framework introduced by Canetti [6] is an increasingly popular framework for analyzing cryptographic protocols that are expected to be concurrently executed with other, possibly malicious protocols. The UC framework has many attractive properties, one of which is a powerful composition theorem enabling the design of complex protocols to be split into simpler sub-protocols. In particular, Canetti, Lindell, Ostrovsky and Sahai [13] showed that, under well established cryptographic assumptions, UC-secure commitments and zero-knowledge (ZK) proofs are sufficient to implement any other functionality, confirming our long-standing intuition that commitments and ZK proofs are fundamental cryptographic primitives.[1]

Unfortunately, a series of sweeping impossibility results [6, 9, 12] showed that most useful cryptographic functionalities, including commitment and ZK, are impossible to realize in the "plain UC" framework. This means that some form of a "trusted setup", such as a common reference string (CRS) or a public-key infrastructure (PKI), is necessary to build UC-secure protocols (unless one is willing to relax some important properties of UC-security). To address this issue, the original UC framework was augmented to allow trusted setup. However, until the recent work of [7], this extension only allowed one to model such setup as a *local setup*. This means that the setup cannot be seen by the environment or other protocols, and, as a consequence, it only exists meaningfully in the real model. In particular, the simulator had complete control over the setup in the ideal model. For example, in the CRS model the simulator had the freedom to choose its own CRS and embed some trapdoor information into it. As was argued in a series of papers [3, 7, 9, 14], this modeling creates several serious problems not present in the "plain UC" framework. Two of the most significant such problems are *lack of deniability* and *restrictive composition*. For example, an ideal ZK proof is "deniable", since the verifier only learns that the statement is true, but cannot reliably prove it to a third party. Unfortunately, it was argued in [7] that any UC-secure realization of ZK in the CRS model is *never deniable*. The composition problem is a bit more subtle to explain. In essence, one can only compose several instances of *specially-designed protocols*. In particular, it is not safe to use protocols which can depend on the setup information (*e.g.*, the CRS), even if these protocols are perfectly secure in the ideal model. We refer the reader to [7, 18, 27] for more discussion of this issue.

GUC Framework. Motivated by solving the problems caused by modeling the setup as a local subroutine, Canetti et al. [7] introduced a new extension of the UC framework — termed *Generalized Universal Composability* (GUC) — for properly analyzing concurrent execution of cryptographic protocols in the presence of a *global setup*. We stress that GUC is a general *framework* strictly more powerful than UC. Namely, one can still model local setup as before. However, the GUC framework also allows one to model *global setup* which is directly accessible to the environment. More precisely, the GUC framework allows one to design protocols that share state via *shared functionalities* (such as a *global CRS* or *global* PKI). Since the same shared functionality will exist in multiple sessions, the environment effectively has direct access to the functionality, meaning that the simulator cannot "tamper" with the setup in the ideal model. In fact, the same setup exists both in the real *and in the ideal models*. As the result, modeling the global setup in this manner regains the attractive properties of the "plain UC", including deniability and general composition. This was formally

[1] Although [13] presented their results in the common reference string (CRS) model using the JUC theorem [14], one can extract a general implication which is independent of the CRS and does not use JUC. See page 131 of Walfish's thesis [27] for details.

shown by [7] for the case of composition, and informally argued for deniability (since the simulator no longer has any "unfair" advantage over the real-model attacker, so the real-model attacker can run the simulator "in its head" to make up transcripts of conversation which never happened in real life). To put this (convincing but) informal argument on firmer ground, in the full version [18] we give a very strong definition of deniable zero-knowledge (much stronger than previous notions appearing in the literature), and show that GUC-security implies this notion, as long as the setup is modeled as a shared functionality.

Of course, having introduced GUC, a natural question is whether one can actually build GUC-secure protocols under *natural* setup assumptions. On the positive side, one can always artificially model local setup as "global setup" by ensuring that a fresh instance of a setup is run for every protocol instance, and, more importantly, that only the participants of a given protocol have reliable access to this setup information. For example, the CRS setup of UC could be equivalently modeled in GUC as a kind of one-time "*session* reference string" (SRS) functionality: the SRS will pick a fresh reference string for each protocol instance, and will make this string available precisely to the parties running this session (and the adversary), but nobody else. It is easily shown that *UC+CRS is equivalent to GUC+SRS*, so the feasibility result of [13] would apply to the "global SRS" setup. Of course, such a "session reference string" model is very unrealistic and difficult to implement, and one may wonder if a *truly global* CRS setup would suffice as well. Unfortunately, [7] showed that the (global) CRS model (as well as other global setup which only provides *public* information, such as the random oracle model [22]) is *not* enough to sidestep the impossibility results of [6, 9, 12]. (In particular, the protocols of [13, 22] are insecure in the GUC framework with the global CRS/random oracle.) This means that any setup sufficient for GUC feasibility must provide some unpublished information, as was the case with the SRS model (where the SRS was hidden from the environment and other protocols).

ACRS MODEL. Luckily, Canetti et al. [7] introduced a new setup assumption, called *Augmented CRS* (ACRS), and showed that it can be used to GUC-realize commitment and ZK (and, thus, any other functionality), even in the presence of adaptive adversaries.[2] The ACRS model is very close to the (global) CRS model, but is (necessarily) augmented so as to circumvent the impossibility result for plain CRS. As in the CRS setup, all parties have access to a short reference string that is taken from a pre-determined distribution. In addition, the ACRS setup allows corrupted parties to obtain "personalized" secret keys that are derived from the reference string, their public identities, and some "global secret" that is related to the public string and remains unknown. It is stressed that *only corrupted parties* may obtain their secret keys. This may sound strange at first, but is actually a huge advantage of the ACRS model over the more traditional

[2] [7] also showed similar results in a variant of a PKI-like "key registration with knowledge (KRK)" setup from [3]. However, since the ACRS model is more minimal and all our results easily extend to the KRK model, we only concentrate on the ACRS model.

"identity-based" setup, where even honest parties *need* to obtain (and, therefore, safeguard) their keys. Namely, the ACRS setup implies that the protocol may not include instructions that require knowledge of the secret keys, and, thus, honest parties do not need their secret keys. In fact, they can only lose *their own* security by obtaining these keys and using them carelessly. This is consistent with any secret-key cryptosystem, where a party will lose its security by publishing its secret key. Luckily, though, the ACRS model permits the luxury of never worrying about losing one's secret key, since one should not get it in the first place. In contrast, malicious parties provably cannot gain anything by obtaining their keys (*i.e.*, they cannot break the security of honest parties). Hence, as a practical matter, one expects that ACRS model is very similar to the CRS model, where parties cannot access any secret information. However, the *mere ability* to get such information is what gives us security, even though we expect that a "rational" party, *either honest or malicious*, will not utilize this ability: honest parties do not need it, and malicious parties do not gain from it.

Of course, one may justifiably criticize the ACRS model because of the need for a trusted party who is always available, as opposed to the (global) CRS model, where no party is needed after the CRS is generated. Indeed, it is a non-trivial setup to realize (although *much* more natural than the SRS model, and seemingly minimal in light of the impossibility result mentioned above). However, as pointed out by [8], the ACRS model has the following "win-win" guarantee. Assume that one proves some protocol secure in the GUC+ACRS model, but in reality the trusted party will only generate a CRS, and will be unavailable afterwards. Then, from a syntactic point of view, *we are back in the (global) CRS model.* In particular, the protocol is still secure in the "old UC+CRS" setting! On an intuitive level, however, it seems to be *more secure* than a protocol proven secure in the "old UC+CRS" setting. This is because the simulator does not need to know a global trapdoor (which is deadly for the security of *honest* parties in the *real* model), but only the secret keys of the *corrupted* parties, which are guaranteed to never hurt the security of honest parties in the real model. For example, the CRS can be safely reused by other protocols, completely solving the "restricted composition" problem of UC that we mentioned earlier. Essentially, properties associated with deniability/non-transferability appear to be the only security properties lost by "downgrading" ACRS into CRS.

Efficiency in the GUC Framework. Thus, from the security and functionality perspectives, the GUC+ACRS model appears to be strictly superior to the UC+CRS model. The question, however, is what is the price in terms of efficiency? Unfortunately, the GUC-feasibility results of [7] are quite inefficient: the commitment scheme committed to the message in a bit-by-bit manner, while the zero-knowledge proof for a relation R was implemented using the generic Cook-Levin reduction to a canonical NP-complete problem. Thus, now that the GUC-feasibility of secure computation has been established, it is natural to ask if one can build *efficient*, GUC-secure commitment and ZK proofs in the ACRS (resp. KRK; see Footnote 2) model. In this paper, we provide such efficient GUC-secure commitment and ZK proofs which are secure against adaptive corruptions, therefore making the ARCS model an attractive alternative to the CRS model on nearly (see below) all fronts.

The only drawback of our solution is that we rely on *data erasures*, which is not the case for most efficient UC protocols, such as that of Damgård and Nielsen [17] (or the inefficient GUC feasibility results of [7]). However, unlike sacrificing adaptive security, which is a *critical* concern (addressed in our work) given the highly dynamic nature of protocols concurrently running on the Internet,[3] we believe that the assumption of data erasures is realistic. Furthermore, this assumption is widely used in practice (for example, for analyzing most key exchange protocols, such as Diffie-Hellman), and was already used in several works on UC security as well (*e.g.*, [10, 11, 21, 24], although it was often hidden deep within the paper). Coupled with the fact that erasures allow us to obtain dramatically more efficient (in fact, *practical*) protocols, we believe that use of this assumption here is justified. Of course, we hope that future research will remove/weaken this restriction, and comment on this more in the last paragraph of the introduction, where we discuss the random oracle model.

Our Results on GUC ZK. We present an efficient compiler giving a direct, efficient, constant-round and GUC-secure ZK proof (GUC ZK) for any NP relation R from any "dense Ω-protocol" [21] for R. The notion of Ω-protocols was introduced by Garay, MacKenzie and Yang [21]. Briefly, Ω-protocols are Σ-protocols (*i.e.*, they satisfy special soundness and ZK properties of Σ-protocols), with an extra property that one can generate the public parameter ρ of the system together with a trapdoor information τ, such that knowledge of τ allows one to extract the witness from any valid conversation between the prover and the verifier (as opposed to the usual special soundness, where one needs two different transcripts with the same first flow). [21, 24] used Ω-protocols for the similar task of building UC-secure ZK proofs in the CRS model, which are inherently not GUC-secure. As a result, our compiler is *considerably* more involved than the compiler of [21, 24] (which also used erasures). For example, in the GUC setting the simulator is not allowed to know τ, so we have to sample the public ρ in the ACRS model using a special coin-flipping protocol introduced by [7]. As a result, our compiler requires Ω-protocols whose reference parameters are "dense" (*i.e.*, indistinguishable from random), and none of the previous Ω-protocols of [21, 24] are suitable for our purpose.

Thus, of independent interest, we show several novel *dense* Ω-protocols. First, we show how to build a direct, but only semi-efficient dense Ω-protocol for any NP relation R from any Σ-protocol for R. Although this Ω-protocol uses the cut-and-choose technique (somewhat similar to the technique of Pass [26], but in a very different setting), it is quite general and gives a much more efficient Ω-protocol than the technique of [7, 13] which requires a generic Cook-Levin reduction. Second, we show a *very efficient* number-theoretically based dense Ω-protocol for proving knowledge of a discrete log representation. Once again, this Ω-protocol had to use some interesting additional tools on top on the prior

[3] We remark that adaptive security with erasures trivially implies static security, and is usually much harder to achieve than the latter.

"non-dense" Ω-protocol of [21], such as a special "projective Paillier encryption" of Cramer and Shoup [15]. As a result, we get a semi-efficient GUC ZK for any R having an efficient Σ-protocol, and a very efficient GUC ZK for proving the knowledge of discrete log representation.

Our Results on GUC Commitments. Using the techniques developed for ZK, we proceed to build the first *constant-rate* (and constant-round) GUC-secure commitments (GUCC) in the ACRS model. In spirit our result is similar to the result of Damgård and Nielsen [17], who constructed the first constant-rate UC-secure commitments in the "old" CRS framework. However, our techniques are very different, and it seems hopeless to adapt the protocol of [17] to the GUC framework. Instead, we essentially notice that the required GUCC would easily follow from our techniques for GUC ZK, provided we can build an efficient Ω-protocol for a special relation on R on *identity-based trapdoor commitments* (IBTCs) — a notion introduced by [7] to implement the ACRS setup. Intuitively, a prover needs to show that he knows the message being committed by a value c (w.r.t. a particular identity). In particular, if one can build an IBTC scheme where the required relation R would involve the proof of knowledge of some discrete log representation, our previous GUC ZK protocol would complete the job. Unfortunately, the IBTCs constructed by [7] had a much more complicated form. Therefore, of independent interest, we build a new IBTC scheme which is based on Water's signature [28]. The resulting IBTC not only has the needed form for its relation R, but is also much simpler and more efficient than prior IBTCs built in the standard model. Combining these results, we finally build the required GUCC.

Results on modeling Random Oracle in GUC. Finally, we briefly comment on using the random oracle (RO) model in conjunction with the GUC framework. The RO is simply modeled as a shared functionality available both in the real and in the ideal model. As such, the simulator cannot "reprogram" the RO. Even more counter-intuitively, it cannot even "prematurely extract" the values used by the real-model attacker! This is because we can assume that all such queries are made by the environment (which the simulator cannot control), and the inputs are only given to the attacker on a "need-to-know" basis. Correspondingly, the RO model is much more restricted in the GUC framework (in particular, by itself it is provably insufficient to GUC-realize most functionalities [7, 8]). However, we still show that one *can* meaningfully use it in the conjunction with the ACRS model, because we *are allowed* to extract and reprogram the RO in the proof of security. In particular, by applying the Fiat-Shamir heuristic to our GUC ZK protocols, we obtain an *efficient*, two-round (which we show is optimal; see Theorem 4), straight-line extractable and simulatable (in fact, GUC-secure!) ZK proof for any relation R having an efficient dense Ω-protocol. Moreover, in this protocol one only needs to erase some short data during a *local computation* (*i.e.*, no sensitive data needs to be stored while waiting for some network traffic), making the need for data erasures extremely minimal. Of course, we can get a less efficient 2-round GUC ZK protocol with these properties that does *not* rely on data erasures at all, by applying the Fiat-Shamir heuristics to the inefficient protocol of [7]. This means that we get a general feasibility of round-optimal GUC ZK for NP in the ACRS+RO model which does not rely on data erasures.

We briefly compare the resulting deniable ZK protocol to previous related work on deniable ZK (*e.g.*, [23, 26]) in Section 6.

2 Definitions and Tools

2.1 GUC Security

At a high level, the UC security framework formalizes the following emulation requirement:

> *A protocol π that emulates protocol ϕ does not affect the security of anything else in the environment differently than ϕ would have – even when π is composed with arbitrary other protocols that may run concurrently with π.*

Unfortunately, the UC security framework requires that parties running in a session of π do not share state with any other protocol sessions at all, limiting the legitimate applicability of that framework. In particular, *global setups* such as a Common Reference String (CRS) or Public Key Infrastructure (PKI) are not accurately modeled. The GUC security framework, introduced in [7], formalizes the same intuitive emulation requirement as the UC framework. However, the GUC framework does so even for protocols π that make use of shared state information that is common to multiple sessions of π, as well as other protocols in the environment (running concurrently with π).

More formally, the security framework of [6] defines a notion called "UC-emulation". A protocol π is said to UC-emulate another protocol ϕ if, for every *adversary* $\mathcal{A}$ attacking ϕ, there exists a *simulator* $\mathcal{S}$ attacking π such that no *environment* $\mathcal{Z}$ can distinguish between $\mathcal{A}$ attacking ϕ, and $\mathcal{S}$ attacking π. In the distinguishing experiment, the environment is *constrained* to interact only with parties participating in a single session of a challenge protocol (either π or ϕ), along with its corresponding attacker (either $\mathcal{A}$ or $\mathcal{S}$, respectively) in a "black-box" manner. This limited interaction prevents the model from capturing protocols that share state with other protocols running in the environment, since the distinguishing experiment does not allow the environment to access any state information used by the parties it is interacting with.

The Generalized Universal Composability (GUC) security framework of [7] extends the original UC security framework of [6] to incorporate the modeling of protocols that share state in an arbitrary fashion. In particular, the GUC framework provides mechanisms to support direct modeling of global setups such as a CRS or PKI. This is done by first defining the notion of *shared functionalities* that can maintain state and are accessible to any party, in any protocol session. The distinguishing experiment of GUC allows the environment to access

Functionality $\mathcal{G}_{\text{acrs}}$

Initialization Phase: At the first activation, run an algorithm Setup to generate a public key/master secret key pair (PK, MSK).

Providing the public value: When activated by any party requesting the CRS, return PK to the requesting party and to the adversary.

Dormant Phase: Upon receipt of a message $(\mathtt{retrieve}, \mathsf{sid}, ID)$ from a *corrupt* party P whose identity is ID, return the value $SK_{ID} \leftarrow \mathsf{Extract}(PK, ID, MSK)$ to P. (Receipt of this message from honest parties is ignored.)

Fig. 1. The Identity-Based Augmented CRS Functionality

any shared functionalities. GUC also removes the constraint on the protocols invoked by the environment, allowing it to interact with any (polynomial) number of parties running arbitrary protocols, including multiple sessions of the protocol being attacked. That is, GUC allows the environment to directly invoke and observe arbitrary protocols that run alongside the distinguishing "challenge protocol" – and those arbitrary protocols may even share state information with the challenge protocol and with the environment itself (via shared functionalities). If a protocol π (that may share state in this fashion) "UC-emulates" a protocol ϕ with respect to such *unconstrained environments*, we say that π GUC-emulates ϕ. We say that a protocol π is a GUC-secure *realization* of a particular functionality $\mathcal{F}$ if π GUC-emulates the ideal protocol for $\mathcal{F}$. Further details of the formal modeling for UC and GUC security can be found in [6] and [7, 27]. In this work, we will focus on the construction of efficient GUC-secure realizations of commitments and zero knowledge, with security even against adversaries capable of adaptive corruptions. As is common throughout the UC literature, we will assume the availability of secure (*i.e.*, private and authenticated) channels. The realization of such secured channels over insecure networks (such as the Internet) is a non-trivial problem studied in [27], but is beyond the scope of this work.

2.2 The ACRS Model

Unfortunately, it is impossible to GUC-realize most useful two-party functionalities in the plain model, or even in the CRS model (see [7]). To avoid this impossibility, we make use of a special *Augmented Common Reference String* (ACRS) trusted setup (which we denote by the functionality $\mathcal{G}_{\text{acrs}}$), as was first proposed in [7]. Another possible alternative would be to use a PKI model supporting "Key Registration with Knowledge" [3, 7] (which we denote by the functionality $\mathcal{G}_{\text{krk}}$) – indeed, our efficient protocols can easily be transformed to use the $\mathcal{G}_{\text{krk}}$ setup – but the more minimal ACRS model suffices and is clearly less costly to implement than a PKI. Thus, we will focus on the ACRS setting. The shared functionality $\mathcal{G}_{\text{acrs}}$ describing ACRS setup, which is parameterized by the algorithms Setup and $\mathsf{Extract}$, is given in Figure 1.

Intuitively, the ACRS setup provides a simple CRS to all parties, and also agrees to supply an identity-based trapdoor for identity P to any "corrupt" party P that asks for one. The provision that only corrupt parties can get their trapdoors is used to model the restriction that protocols run by honest parties should not use the trapdoor – *i.e.* honest parties should never *have* to obtain their trapdoors in order to run protocols. In reality, a trusted party will perform the ACRS initialization phase, and then supply the trapdoor for P to any party P that asks for its trapdoor. Of course, in practice, most parties will never bother to request their trapdoors since the trapdoors are not useful for running protocols. (Ultimately, these trapdoors will be used to enable corrupt parties to simulate attacks by using $\mathcal{S}$, a task that no honest party should need to perform.)

In the following sections, we show how to construct efficient GUC-secure realizations of commitments and zero knowledge using this instantiation of the $\mathcal{G}_{\text{acrs}}$ shared functionality. (As explained in Section 4 of [8], this is enough to GUC-realize any other well-formed functionality.) We then show how to optimize the round complexity of these protocols by using $\mathcal{G}_{\text{acrs}}$ in conjunction with the RO model.

2.3 Omega Protocols

The notion of an Ω-protocol was introduced in [21], and we recall the basic idea here. While our notion of an Ω-protocol is the same in spirit as that in [21], we also introduce some new properties, and there are a few points where the technical details of our definition differ. Details can be found in the full version [18].

Let *ParamGen* be an efficient probabilistic algorithm that takes as input 1^λ, where λ is a security parameter, and outputs a *system parameter* Λ. The system parameter Λ determines finite sets X, $L \subset X$, W, and a relation $R \subset L \times W$, where for all $x \in L$, we have $(x, w) \in R$ for some $w \in W$. The sets X and W, and the relation R should be efficiently recognizable (given Λ). An element $x \in X$ is called an *instance*, and for $(x, w) \in R$, w is called a *witness* for x.

There is also an efficient probabilistic algorithm *RefGen* that takes as input a system parameter Λ and outputs a pair (ρ, τ), where ρ is called a *reference parameter*, and τ is called a *trapdoor*.

An Ω-protocol Π is played between a *prover* P and a *verifier* V. Both P and V take as common input a system parameter Λ, a reference parameter ρ, and an instance $x \in X$. An honest prover P is only run for $x \in L$, and always takes a witness w for x as an additional, private input. Execution runs in three steps: in the first step, P sends a message a to V; in the second, V sends a random challenge c to P; in the third, P sends a response z to V, whereupon V either *accepts* or *rejects* the *conversation* (a, c, z).

Of course, there is a basic *completeness* requirement, which says that if both prover and verifier follow the protocol then the verifier always accepts.

We say that Π is *trapdoor sound* if there exists an efficient *trapdoor extractor algorithm* $\mathcal{E}_{\text{td}}$ such that the following holds: for every efficient cheating prover $\tilde{P}$, it should be infeasible for $\tilde{P}$ (given input (Λ, ρ)) to make V (given input (Λ, ρ, x)) accept a conversation (a, c, z) for an instance x such that execution of $\mathcal{E}_{\text{td}}$ on input $(\Lambda, \tau, x, a, c, z)$ fails to produce witness w for x. Here, (Λ, ρ, τ) are generated by the algorithms *ParamGen* and *RefGen*; c is generated by V; and x, a, and z are generated adversarially.

We shall also make use of the following variant of trapdoor soundness. Very roughly, we say that Π is *partial trapdoor sound for a function* f, if it is a proof of knowledge (in the traditional, rewinding sense) of a witness w of the instance x, such that the value calculated by the trapdoor extractor $\mathcal{E}_{\mathrm{td}}$ (on the same inputs as above) is equal to $f(w)$. As we will see, partial trapdoor soundness is sufficient for some applications, and can be realized using a somewhat more efficient protocol.

We say that Π is *honest verifier zero-knowledge (HVZK)* if there is a *simulator algorithm* ZKSim that on input (Λ, ρ, x, c) can produce a simulation of the conversation (a, c, z) that would arise from an interaction between an honest prover P with input (Λ, ρ, x, w), and a cheating verifier $\tilde{V}$, subject to the constraint that $\tilde{V}$'s challenge c must be generated before it sees a. Here, (Λ, ρ) are generated by the algorithms *ParamGen* and *RefGen*; and x, w, and c are generated by $\tilde{V}$. The requirement is that $\tilde{V}$ should not be able to distinguish the output of the simulator from the output of the real prover.

We note that the notion of an Ω-protocol extends that of a Σ-protocol ([16, 17]). The distinguishing feature is the reference parameter, and the trapdoor soundness property that says that a witness may be extracted using a trapdoor in the reference parameter, rather than by rewinding. The notion of trapdoor soundness is closely related to that of *verifiable encryption* [1, 5]. Indeed, all known constructions of Ω-protocols boil down to using a public key for a semantically secure encryption scheme as reference parameter, where the trapdoor is the secret key; the prover encrypts a witness, and then proves that it did so using a Σ-protocol.

For our application to GUC ZK and GUC commitments, we introduce an additional property that we require of an Ω-protocol. A given system parameter Λ determines a set $\widehat{\Phi}$ of possible reference parameters. Suppose there is some set Φ that contains $\widehat{\Phi}$, with the following properties: (i) the uniform distribution on Φ is efficiently samplable; (ii) membership in Φ is efficiently determined; (iii) Φ is an abelian group (which we write multiplicatively), such that the group and group inverse operations are efficiently computable; (iv) it is hard to distinguish a random element of Φ (generated uniformly), from a random element of $\widehat{\Phi}$ (as generated by *RefGen*). If all of these conditions obtain, we say Π has *dense reference parameters*, and we call Φ the set of *extended reference parameters*.

2.4 Identity-Based Trapdoor Commitments

The notion of an identity-based trapdoor commitment scheme (IBTC) was introduced in [2] (as ID-based Chameleon Hash functions), with some additional refinements appearing in [7]. We recall the basic idea here, leaving the formal definition to the full version [18].

An IBTC scheme has a Setup algorithm that takes as input 1^λ, where λ is the security parameter, and outputs a *public key* PK and a *master secret key* MSK. The public key PK determines a set $\mathcal{D}$ of *decommitment values*. To generate a commitment to a message m, a user computes $d \overset{\$}{\leftarrow} \mathcal{D}$ and $\kappa \leftarrow \mathsf{Com}_{ID}(d, m)$. Here, Com_{ID} is a deterministic algorithm (which implicitly takes

Functionality $\mathcal{F}_{\mathrm{zk}}^{R}$

$\mathcal{F}_{\mathrm{zk}}$, parameterized by a binary relation R and running with a prover P, a verifier V, and an adversary $\mathcal{S}$, proceeds as follows upon receipt of a message $(\texttt{ZK-prover}, sid, P, V, x, w)$ from the prover P:

If $(x, w) \in R$, then send $(\texttt{ZK-proof}, sid, P, V, x)$ to V and $\mathcal{S}$ and halt. Otherwise halt.

Fig. 2. The Zero-Knowledge Functionality for Relation R

PK as a parameter, but we shall in general omit this). The value κ is called a *commitment* to m, while the pair (d, m) is called an *opening* of κ.

Like any commitment, a IBTC should be *binding*: it should be hard to open a commitment under some ID to two different messages; that is, it should be hard to find ID, d, m, d', m' such that $m \neq m'$ and $\mathsf{Com}_{ID}(d, m) = \mathsf{Com}_{ID}(d', m')$. In addition, there should be an *identity-based trapdoor*, which allows for *identity-based equivocation* of commitments. More precisely, there are three algorithms $\mathsf{Extract}$, ECom, and Eqv, which work as follows. Given (PK, ID, MSK) as input, $\mathsf{Extract}$ computes a trapdoor SK_{ID} for the identity ID. Using this trapdoor, algorithm ECom may be invoked with input (PK, ID, SK_{ID}) to produce a pair (κ, α), where κ is a "fake" commitment, and α is a trapdoor specifically tuned to κ. Finally, running algorithm Eqv on input $(PK, ID, SK_{ID}, \kappa, \alpha, m)$ for any message m produces a decommitment d, such that (d, m) is an opening of κ. The security property for equivocation is that is should be hard to distinguish a value d produced in this way from a random decommitment. Moreover, this equivocation property should not interfere with the binding property *for identities whose trapdoors have not been extracted.*

3 GUC Zero-Knowledge in the ACRS Model

The ideal Zero-Knowledge functionality for relation R, $\mathcal{F}_{\mathrm{zk}}$, is described in Figure 2.[4]

Here we give a general transformation from any Ω-protocol Π for a relation R to a GUC-secure zero-knowledge proof for the relation R in the augmented CRS ($\mathcal{G}_{\mathrm{acrs}}$) model. We need to assume that the Ω-protocol satisfies the correctness, trapdoor soundness, honest verifier zero knowledge (HVZK), and dense reference parameters properties. We denote by Φ the space of extended reference parameters for Π. We also need an identity-based trapdoor commitment (IBTC) scheme. Commitments in this scheme are written $\mathsf{Com}_{ID}(d, m)$.

The augmented CRS is instantiated using the public key (and trapdoor extractor) of the IBTC. In addition, any system parameters Λ for the Ω-protocol

[4] Technically, the relation R may be determined by system parameters, which form part of a CRS. Here, we note that the same CRS must be used in both the "ideal" and "real" settings.

are placed in the public value of the augmented CRS. Note that there is no trapdoor associated with the system parameter for the Ω-protocol, so this system parameter is essentially a "standard" CRS. A critical difference between our approach and that of Garay et al. [21] is that the reference parameter for the Ω-protocol are not placed in the CRS; rather, a fresh reference parameter ρ is generated with every run of the protocol using a three-move "coin toss" protocol (which, in turn, makes use of the IBTC).

Here is how the GUC ZK protocol between a prover P and verifier V works. The common input is an instance x (in addition to PK and the identities of the players). Of course, P also has a witness w for x as a private input.

1. V computes $\rho_1 \overset{\$}{\leftarrow} \Phi$, forms commitment $\kappa_1 = \mathsf{Com}_P(d_1, \rho_1)$, and sends κ_1 to P.
2. P computes $\rho_2 \overset{\$}{\leftarrow} \Phi$ and sends ρ_2 to V.
3. V first verifies that $\rho_2 \in \Phi$, and then sends the opening (d_1, ρ_1) to P.
4. P verifies that (d_1, ρ_1) is a valid opening of κ_1, and that $\rho_1 \in \Phi$.
   Both P and V locally compute $\rho \leftarrow \rho_1 \cdot \rho_2$.
5. P initiates the Ω-protocol Π, in the role of prover, using its witness w for x. P computes the first message a of that protocol, forms the commitment $\kappa' = \mathsf{Com}_V(d', a)$, and sends κ' to V.
6. V sends P a challenge c for protocol Π.
7. P computes a response z to V's challenge c, and sends (d', a, z) to V.
   P then **erases** the random coins used by Π.
8. V verifies that (d', a) is a valid opening of κ' and that (a, c, z) is an accepting conversation for Π.

Theorem 1. *The protocol described above GUC-emulates the $\mathcal{F}_{\text{zk}}^{R}$ functionality in the secure-channels model, with security against adaptive corruptions (with erasures).*

Proof (sketch). We first observe that the protocol above only makes use of a single shared functionality, $\mathcal{G}_{\text{acrs}}$. Therefore, we are free to make use of the equivalence theorem and EUC model of [7]. This allows us to prove the GUC security of the protocol using the familiar techniques of the UC framework, with only a single (but crucial) modification – we will allow the environment access to the shared functionality.

Let $\mathcal{A}$ be any PPT adversary attacking the above protocol. We describe an ideal adversary $\mathcal{S}$ attacking the ideal protocol for $\mathcal{F}_{\text{zk}}^{R}$ that is indistinguishable from $\mathcal{A}$ to any distinguishing environment $\mathcal{Z}$, in the presence of a shared setup $\mathcal{G}_{\text{acrs}}$. In standard fashion, $\mathcal{S}$ will run a copy of $\mathcal{A}$ internally. We now formally describe how $\mathcal{S}$ interacts with its internal copy of $\mathcal{A}$. We focus here on the non-trivial aspects of the simulator.

Simulating a proof between an honest P and corrupt V. The following simulation strategy is employed whenever P is honest and V is corrupted at any point prior to, or during, the execution of the protocol. $\mathcal{S}$, upon notification from $\mathcal{F}_{\text{zk}}^{R}$ of a successful proof from P of statement x, proceeds as follows. First, acting on behalf of the corrupt party V, $\mathcal{S}$ obtains the trapdoor SK_V from $\mathcal{G}_{\text{acrs}}$.

Next, $\mathcal{S}$ runs the coin-tossing phase of the protocol with the corrupt party V (being controlled by $\mathcal{S}$'s internal copy of $\mathcal{A}$) normally. Upon completion of the coin-tossing phase at Step 5, rather than sending a commitment to the first message sent by Π (which would require the witness w as an input) as per the protocol specification, $\mathcal{S}$ obeys the following procedure for the next 3 steps of the protocol:

5. $\mathcal{S}$ computes $(\hat{\kappa}', \alpha) \leftarrow \mathsf{ECom}(V, SK_V)$. $\mathcal{S}$ then sends the equivocable commitment $\hat{\kappa}'$ to the corrupt verifier V (which is part of $\mathcal{S}$'s internal simulation of $\mathcal{A}$).
6. $\mathcal{S}$ receives a challenge c from the corrupt verifier V.
7. $\mathcal{S}$ runs the HVZK simulator ZKSim for protocol Π on input (Λ, ρ, x, c), obtaining messages a and z. $\mathcal{S}$ then equivocates $\hat{\kappa}'$, by computing $d' \leftarrow \mathsf{Eqv}(V, SK_V, \hat{\kappa}', \alpha, a)$, and sends d', a, z to the corrupt verifier V.

Observe that this simulation is done entirely in a straight-line fashion, and requires only the trapdoor SK_V belonging to corrupt party V.

If P is also corrupted at some point during this simulation, $\mathcal{S}$ must generate P's internal state information and provide it to $\mathcal{A}$. If P is corrupted prior to Step 5, then $\mathcal{S}$ can easily provide the random coins used by P in all previous steps of the protocol (since those are simply executed by $\mathcal{S}$ honestly). A corruption after Step 5 but before Step 7 is handled by creating an honest run of protocol Π using witness w (which was revealed to $\mathcal{S}$ immediately upon the corruption of P), and computing the internal value d' via $d' \leftarrow \mathsf{Eqv}(V, SK_V, \kappa', \alpha, a)$, where a is now the honestly generated first message of Π. Finally, if corruption of P occurs after Step 7 of the simulation, the internal state is easily generated to be consistent with observed protocol flows, since they already contain all relevant random coins, given the erasure that occurs at the end of Step 7.

Intuitively, the faithfulness of this simulation follows from the equivocability and binding properties of commitments, and the HVZK and dense reference parameters properties of the Ω-protocol Π. We stress that while the *proof* of this requires a rewinding argument (see the full version [18]), the simulation itself is straight-line.

Simulating a proof between a corrupt P and honest V. The following simulation strategy is employed whenever V is honest, and P is corrupted at any point prior to or during the execution of the protocol. First, acting on behalf of the corrupt party P, $\mathcal{S}$ obtains the trapdoor SK_P from $\mathcal{G}_{\text{acrs}}$. Then $\mathcal{S}$ generates a pair (ρ, τ) using the *RefGen* algorithm for Π, and "rigs" the coin-tossing phase of the protocol by playing the role of V (communicating with the internal simulation of the corrupt party P) and modifying the initial steps of the protocol as follows:

1. $\mathcal{S}$ computes $(\hat{\kappa}_1, \alpha) \leftarrow \mathsf{ECom}(P, SK_P)$, and sends $\hat{\kappa}_1$ to P.
2. P replies by sending some string ρ_2 to V.
3. $\mathcal{S}$ computes $\rho_1 \leftarrow \rho \cdot \rho_2^{-1}$, and $d_1 \leftarrow \mathsf{Eqv}(P, SK_P, \hat{\kappa}_1, \alpha, \rho_1)$.
   $\mathcal{S}$ first verifies that $\rho_2 \in \Phi$. Then $\mathcal{S}$ sends the opening (d_1, ρ_1) to P.

The remainder of the protocol is simulated honestly.

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

Functionality $\mathcal{F}_{\text{com}}$ proceeds as follows, with committer P and recipient V. .

Commit Phase: Upon receiving a message $(\texttt{commit}, sid, P, V, m)$ from party P, record the value m and send the message $(\texttt{receipt}, sid, P, V)$ to V and the adversary. Ignore any future `commit` messages.

Reveal Phase: Upon receiving a message $(\texttt{reveal}, sid)$ from P: If a value m was previously recorded, then send the message $(\texttt{reveal}, sid, m)$ to V and the adversary and halt. Otherwise, ignore.

Fig. 3. The Commitment Functionality $\mathcal{F}_{\text{com}}$ (see [9])

Observe that the outcome of this coin-flipping phase will be the same ρ generated by $\mathcal{S}$ at the start of the protocol (along with its corresponding trapdoor information τ). If and when the verifier accepts, $\mathcal{S}$ runs the trapdoor extractor $\mathcal{E}_{\text{td}}$ for Π on input $(\Lambda, \tau, x, a, c, z)$ to obtain a witness w for x. $\mathcal{S}$ then sends the pair (x, w) to the ideal functionality $\mathcal{F}_{\text{zk}}^{R}$ on behalf of the corrupt prover P.

In the event that V is also corrupted at any point prior to completion of the protocol, $\mathcal{S}$ simply produces internal state for V consistent with the visible random coins in the transcript (none of the verifier's random coins are hidden by the honest protocol).

Intuitively, the faithfulness of this simulation follows from the equivocability and binding properties of commitments, and the trapdoor soundness and dense reference parameters properties of the Ω-protocol Π. Again, we stress that while the *proof* of this requires a rewinding argument (*e.g.*, the Reset Lemma of [4]), the simulation itself is straight-line.

Now that we have fully described the behavior of $\mathcal{S}$, it remains to prove that $\mathcal{S}$ interacting with $\mathcal{F}_{\text{zk}}^{R}$ (the ideal world interaction) is indistinguishable from $\mathcal{A}$ interacting with the protocol (the real-world interaction), from the standpoint of any environment $\mathcal{Z}$ with access to $\mathcal{G}_{\text{acrs}}$. We stress that even $\mathcal{Z}$ cannot obtain trapdoor information from $\mathcal{G}_{\text{acrs}}$ for any honest parties, since $\mathcal{G}_{\text{acrs}}$ will not respond to requests for such trapdoors. The proof of indistinguishability follows from a relatively straightforward argument, using the security properties of the IBTC and Ω-protocol. See the full version [18].

4 GUC Commitments in the ACRS Model

The ideal functionality for a commitment scheme is shown in Figure 3. Messages m may be restricted to some particular *message space*.

Our protocol makes use of an Ω-protocol for the IBTC opening relation; here, a witness for a commitment κ with respect to an identity ID is a valid opening (d, m) (*i.e.*, $\mathsf{Com}_{ID}(d, m) = \kappa$). Instead of trapdoor soundness, we only require partial trapdoor soundness with respect to the function $f(d, m) := m$.

Our new GUC commitment protocol has two phases. The commit phase is the same as the ZK protocol in the previous section, except that Step 5 now runs as follows:

5.′ P generates a commitment $\kappa = \mathsf{Com}_V(d, m)$, and then initiates the Ω-protocol Π, in the role of prover, using its witness (d, m).
P computes the first message a of that protocol, forms the commitment $\kappa' = \mathsf{Com}_V(d', a)$, and sends κ and κ' to V.

In the reveal phase, P simply sends the opening (d, m) to V, who verifies that (d, m) is a valid opening of κ.

Theorem 2. *The protocol described above GUC-emulates the $\mathcal{F}_{\text{com}}$ functionality in the secure-channels model, with security against adaptive corruptions (with erasures).*

The proof is analogous to that of our zero knowledge protocol, but entails some minor changes that include the partial trapdoor soundness requirement for Π. See [18] for more detail.

5 Efficient Implementations

5.1 Constructing Ω Protocols from Σ Protocols

We now briefly sketch how to efficiently construct an Ω-protocol Π for a relation R, given any efficient Σ-protocol Ψ for relation R. Intuitively, we must ensure that the dense reference parameter and trapdoor extractability properties of Π will hold, in addition to carrying over Σ-protocol Ψ's existing properties.

Let the reference parameter for Π be the public key pk for a "dense" semantically secure encryption E (where the dense property of the encryption scheme simply satisfies the requirements of the Dense Reference Parameter property of Ω protocols). Standard El-Gamal encryption will suffice for this purpose (under the DDH assumption). Let $\psi = E_{pk}(s, m)$ denote an encryption of message m with random coins s.

Let a, z^c denote the first and last messages (respectively) of the prover in protocol Ψ when operating on input (x, w, r) and with challenge c, where $(x, w) \in R$ and r denotes the random coins of the prover. The three messages to be sent in protocol Π will be denoted as a', c', z'.

Intuitively, we will use a cut-and-choose technique to provide extractability, and then amplify the soundness by parallel repetition k times. The first message a' of Π is constructed as follows:

1. For $i = 1, \ldots, k$, choose random coins r_i and compute a_i, z_i^0, and z_i^1 using the prover input (x, w, r_i).
2. For $i = 1, \ldots, k$, compute ciphertexts $\psi_i^0 = E_{pk}(s_i^0, z_i^0)$ and $\psi_i^1 = E_{pk}(s_i^1, z_i^1)$.
3. Set $a' := (\psi_1^0, \psi_1^1, \ldots, \psi_k^0, \psi_k^1)$.

The challenge c' sent to the prover in Π is a k-bit string $c' = c_1' c_2' \ldots c_k'$. The last message z' of protocol Π is then constructed as follows.

1. For $i = 1, \ldots, k$, set $z_i' := (s_i^{c_i'}, z_i^{c_i'})$.
2. Set $z' := (z_1', \ldots, z_k')$.

The verifier's algorithm for Π is simply constructed accordingly, verifying that all the ciphertexts were correctly constructed, and that the corresponding conversations for Ψ are valid. The proof of the following theorem is standard and is therefore omitted.

Theorem 3. *Π constructed as above is an Ω-protocol for relation R, provided that Ψ is a Σ-protocol for relation R and E is a dense one-time semantically secure public key encryption scheme.*

5.2 An Efficient Identity-Based Trapdoor Commitment with Ω-Protocol

While the protocol in §5.1 is certainly much more efficient than that in [7] (at least for languages with efficient Σ-protocols), we would like to get an even more efficient protocol that avoids the cut-and-choose paradigm altogether. In this section, we briefly show how we can obtain such a protocol for GUC commitments. Unlike the GUC commitment scheme in [7], which could commit bits, our GUC commitment scheme can be used to commit to values in a much larger set. Moreover, because of the special algebraic structure of the scheme, our GUC commitment protocol can be combined with other, well-known protocols for proving properties on committed values (*e.g.*, the that product of two committed integers is equal to a third committed integer).

To achieve this goal, we need an IBTC scheme that supports an efficient Ω-protocol, so that we can use this scheme as in §4. As observed in [7], based on a variation of an idea in [19], to build an IBTC scheme, one can use a secure signature scheme, along with a Σ-protocol for proof of knowledge of a signature on a given message. Here, the message to be signed is an identity ID. Assuming the Σ-protocol is HVZK, we can turn it into a commitment scheme, as follows. For a conversation (a, c, z), the commitment is a, the value committed to is c, and the decommitment is z. To commit to a value c, one runs the HVZK simulator. The trapdoor for a given ID is a signature on ID, and using this signature, one can generate equivocable commitments just by running the actual Σ-protocol.

For our purposes, we suggest using the Waters signature scheme [28]. Let $\mathbb{G}$ and $\mathbb{H}$ be a groups of prime order q, let $e : \mathbb{G} \to \mathbb{H}$ be an efficiently computable, non-degenerate bilinear map, and let $\mathbb{G}^* := \mathbb{G} \setminus \{1\}$. A public reference parameter consists of random group elements $\mathbf{g}_1, \mathbf{g}_2, \mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{G}$, a description of a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^k$, and a group element $\mathbf{h}_1$. A signature on a message m is a pair $(\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{G} \times \mathbb{G}$, such that $e(\mathbf{s}_1, \tilde{\mathbf{u}}_m^{-1}) \cdot e(\mathbf{s}_2, \mathbf{g}_1) = e(\mathbf{h}_1, \mathbf{g}_2)$, where $\tilde{\mathbf{u}}_m := \mathbf{u}_0 \prod_{b_i=1} \mathbf{u}_i$ and $H(m) = b_1 \cdots b_k \in \{0,1\}^k$. Waters' signature is secure assuming the CDH for the group $\mathbb{G}$. With overwhelming probability, the signing algorithm will produce a signature $(\mathbf{s}_1, \mathbf{s}_2)$ where neither $\mathbf{s}_1$ nor $\mathbf{s}_2$ are 1, so we can effectively assume this is always the case.

To prove knowledge of a Waters signature $(\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{G} \times \mathbb{G}$ on a message $m \in \{0,1\}^*$, we may use the following protocol. The prover chooses $w_1, w_2 \in \mathbb{Z}_q^*$ at random, and computes $\bar{\mathbf{s}}_1 \leftarrow \mathbf{s}_1^{1/w_1}$ and $\bar{\mathbf{s}}_2 \leftarrow \mathbf{s}_2^{1/w_2}$. The prover then sends

$\bar{\mathbf{s}}_1$ and $\bar{\mathbf{s}}_2$ to the verifier, and uses a standard Σ-protocol to prove knowledge of exponents $w_1, w_2 \in \mathbb{Z}_q$ such that $\gamma_1^{w_1}\gamma_2^{w_2} = \gamma$ where $\gamma_1 := e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_m^{-1})$, $\gamma_2 := e(\bar{\mathbf{s}}_2, \mathbf{g}_1)$, and $\gamma := e(\mathbf{h}_1, \mathbf{g}_2)$.

The identity-based commitment scheme derived from the above Σ-protocol works as follows. Let $ID \in \{0,1\}^*$ be the identity, and let $m \in \mathbb{Z}_q$ be the message to be committed. The commitment is computed as follows: $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2 \overset{\$}{\leftarrow} \mathbb{G}^*$, $d_1, d_2 \overset{\$}{\leftarrow} \mathbb{Z}_q$, $\gamma_1 \leftarrow e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_{ID}^{-1})$, $\gamma_2 \leftarrow e(\bar{\mathbf{s}}_2, \mathbf{g}_1)$, $\gamma \leftarrow e(\mathbf{h}_1, \mathbf{g}_2)$, $\bar{\gamma} \leftarrow \gamma_1^{d_1}\gamma_2^{d_2}\gamma^m$. The commitment is $(\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\gamma})$.

A commitment $(\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\gamma}) \in \mathbb{G}^* \times \mathbb{G}^* \times \mathbb{H}$ is opened by revealing d_1, d_2, m that satisfies the equation $\gamma_1^{d_1}\gamma_2^{d_2}\gamma^m = \bar{\gamma}$, where $\gamma_1, \gamma_2, \gamma$ are computed as in the commitment algorithm, using the given values $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$.

The trapdoor for such a commitment is a Waters signature on the identity *ID*. Using such a signature, one can just run the Σ-protocol, and open the commitment to any value. The commitment will look the same as an ordinary commitment, unless either component of the signature is the identity element, which happens with negligible probability.

As the opening of a commitment is essentially just a representation of a group element relative to three bases, there is a standard Σ-protocol for proving knowledge of an opening of a given commitment. Moreover, using techniques from Camenisch and Shoup [5], we can actually build an Ω-protocol for such a proof of knowledge, which avoids the cut-and-choose paradigm.

Garay et al. [21] give an Ω-protocol for a very similar task, which could easily be adapted for our purposes, except that the protocol in [21] does not satisfy the dense reference parameters property, which is crucial for our construction of a GUC commitment. To appreciate the technical difficulty, the MacKenzie et al. protocol is based on Paillier encryption, using an RSA modulus N. The secret key for this encryption scheme is the factorization of N, and this is used as "global" trapdoor to a CRS in their proof of security in the UC/CRS model. However, in the GUC framework, we cannot have such a global trapdoor, which is why we make use of Camenisch and Shoup's approach.[5]

The Camenisch and Shoup approach is based on a variant of Paillier encryption, introduced in Cramer and Shoup [15], which we call here *projective Paillier encryption*. While the goal in [5] and [15] was to build a chosen ciphertext secure encryption scheme, and we only require semantic security, it turns out that their schemes do not require the factorization of the RSA modulus N to be a part of the secret key. Indeed, the modulus N can be generated by a trusted party, who then erases the factorization and goes away, leaving N to be used as a shared system parameter. We can easily "strip down" the scheme in [5], so that it only provides semantic security. The resulting Ω-protocol will satisfy all the properties we need to build a GUC commitment, under standard assumptions (the Quadratic Residuosity, Decision Composite Residuosity, and Strong RSA).

[5] It should be noted that the "mixed commitments" of Damgård and Nielsen [17] also have a very similar global extraction trapdoor, which is why we also cannot use them to build GUC commitments.

Due to lack of space, all the remaining details for the IBTC scheme and the Ω-protocol for proof of knowledge of a representation are relegated to the full version [18].

6 Achieving Optimal Round Complexity with Random Oracles

While our constructions for GUC zero knowledge and commitments are efficient in both computational and communication complexity, and the constant round complexity of 6 messages is reasonable, it would be nice improve the round complexity, and possibly weaken the data erasure assumption. In this section we address the question if such improvements are possible in the random oracle (RO) model. We first remark that even the RO model, without any additional setup, does not suffice for realizing GUC commitments or zero knowledge (see [7, 8]). However, we may still obtain some additional efficiency benefits by combining the ACRS and RO models. Ideally, we would like to achieve non-interactive zero knowledge (NIZK), and, similarly, a non-interactive commitment. Unfortunately, this is not possible if we insist upon adaptive security, even if we combine the ACRS or KRK setup models with a random oracle.

Theorem 4. *There do not exist* adaptively secure *and* non-interactive *protocols for GUC-realizing $\mathcal{F}_{\text{com}}$ and $\mathcal{F}_{\text{zk}}^{R}$ (for most natural and non-trivial NP relations R) in the ACRS or KRK setup models. This impossibility holds even if we combine the setup with the random oracle model, and even if we allow erasures.*

We give a more formal statement and proof of this result in the full version [18]. Intuitively, there are two conflicting simulation requirements for GUC-secure commitments/ZK proofs that pose a difficulty here: a) given knowledge of the sender/prover's secret key, they must be "extractable" to the simulator, yet b) given knowledge of the recipient/verifier's secret key, they must be "simulatable" by the simulator. It is impossible for a single fixed message to simultaneously satisfy *both* of these conflicting requirements, so an adversary who can later obtain both of the relevant secret keys via an adaptive corruption will be able to test them and see which of these requirements was satisfied. This reveals a distinction between simulated interactions and real interactions, so we must resort to an interactive protocol if we wish to prevent the adversary from being able to detect this distinction. Accordingly, we will now show that it is possible to achieve *optimal* 2-round ZK and commitment protocols in the GUC setting using both the ACRS and RO setups.

Round-Optimal ZK using Random Oracles. We achieve our goal by simply applying the Fiat-Shamir heuristic [20] to our efficient zero knowledge and commitment protocols, replacing the first three and last three messages of each protocol with a single message. We defer a more formal discussion and analysis of GUC security in the combined ACRS and RO model with the Fiat-Shamir heuristic to the full version [18] (additional details can be also be found in [27]), but briefly comment on three important points. First, note that the only erasure required by our protocols now occurs entirely during a *single local computation*, without delay – namely, during the computation of the second message, where an entire run of three-round protocol is computed and the local randomness used to generate that run is then immediately erased. Thus, the need for data erasures is really minimal for these protocols.

Second, the proof of security for the modified protocols is virtually unaltered by the use of the Fiat-Shamir heuristic. In particular, observe that the GUC simulator $\mathcal{S}$ uses identical simulation strategies, and *does not* need to have access to a transcript of oracle queries, nor does it require the ability to "program" oracle responses. Thus, only in the *proof of security* (namely, that the environment cannot tell the real and the ideal worlds) do we use the usual "extractability" and "programmability" tricks conventionally used in the RO model.

Third, we stress that since the GUC modeling of a random oracle (accurately) allows the oracle to be accessed directly by all entities – including the environment – the aforementioned feature that $\mathcal{S}$ does not require a transcript of all oracle queries, nor the ability to program oracle responses, is *crucial* for deniability. It was already observed by Pass [26] that deniable zero knowledge simulators must not program oracle queries. However, we observe that even using a "non-programmable random oracle" for the simulator is still not sufficient to ensure truly deniable zero knowledge. In particular, if the modeling allows the simulator to observe interactions with the random oracle (even without altering any responses to oracle queries), this can lead to attacks on deniability. In fact, there is a very practical attack (sketched in Appendix 6) stemming from precisely this issue that will break the deniability of the protocols proposed by Pass [26]. Our GUC security modeling precludes the possibility of any such attacks.

Of course, unlike the model of [26], we superimpose the ACRS model on the RO model, providing all parties with implicit secret keys. This bears a strong resemblance to the model of [23], which employs the following intuitive approach to provide deniability for the prover P: instead proving the statement, P will prove "either the statement is true, or I know the verifier's secret key". Indeed, our approach is quite similar in spirit. However, we achieve a much stronger notion of deniability than that of [23]. Our zero knowledge protocols are the first constant round protocols to simultaneously achieve straight-line extractability (required for concurrent composability) and deniability against an adversary who can perform adaptive corruptions. In contrast, the protocol of [23] is not straight-line extractable, and is not deniable against adaptive corruptions (this is easy to see directly, but also follows from Theorem 4, by applying the Fiat-Shamir heuristics to the 3-round protocol of [23]).

Finally, if one does not care about efficiency, applying our techniques to the inefficient protocols of [7], we get a general, round-optimal feasibility result for all of NP:

Theorem 5. *Under standard cryptographic assumptions, there exists a (deniable) 2-round GUC ZK protocol for any language in NP in the ACRS+RO model, which does not rely on data erasures.*

References

1. Asokan, N., Shoup, V., Waidner, M.: Optimistic Fair Exchange of Digital Signatures. In: Proc. of Eurocrypyt (1998)
2. Ateniese, G., de Medeiros, B.: Identity-based Chameleon Hash and Applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110. Springer, Heidelberg (2004)
3. Barak, B., Canetti, R., Nielsen, J., Pass, R.: Universally Composable Protocols with Relaxed Set-up Assumptions. In: Proc. of FOCS (2004)
4. Bellare, M., Palacio, A.: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442. Springer, Heidelberg (2002)
5. Camenisch, J., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729. Springer, Heidelberg (2003)
6. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Proc. of FOCS (2001)
7. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universal Composability with Global Setup. In: Proc. of TCC (2007)
8. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universal Composability with Global Setup (full version), `http://eprint.iacr.org/2006/432`
9. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139. Springer, Heidelberg (2001)
10. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
11. Canetti, R., Krawczyk, H.: Universally Composable Notions of Key Exchange and Secure Channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332. Springer, Heidelberg (2002)
12. Canetti, R., Kushilevitz, E., Lindell, Y.: On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
13. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-Party and Multi-Party Secure Computation. In: Proc. of STOC (2002)
14. Canetti, R., Rabin, T.: Universal Composition with Joint State. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729. Springer, Heidelberg (2003)
15. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332. Springer, Heidelberg (2002)
16. Damgård, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807. Springer, Heidelberg (2000)
17. Damgård, I., Nielsen, J.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442. Springer, Heidelberg (2002)
18. Dodis, Y., Shoup, V., Walfish, S.: Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs (full version), `http://www.shoup.net/papers/gucc.pdf`
19. Feige, U.: Alternative Models for Zero Knowledge Interactive Proofs. Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel (1990)

20. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Proc. of Crypto (1987)
21. Garay, J., MacKenzie, P., Yang, K.: Strengthening Zero-Knowledge Protocols Using Signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
22. Hofheinz, D., Muller-Quade, J.: Universally Composable Commitments Using Random Oracles. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951. Springer, Heidelberg (2004)
23. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070. Springer, Heidelberg (1996)
24. MacKenzie, P., Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027. Springer, Heidelberg (2004)
25. Paillier, P.: Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592. Springer, Heidelberg (1999)
26. Pass, R.: On Deniabililty in the Common Reference String and Random Oracle Model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729. Springer, Heidelberg (2003)
27. Walfish, S.: Enhanced Security Models for Network Protocols. Ph.D. thesis. New York University (2007), `http://www.cs.nyu.edu/web/Research/Theses/walfish_shabsi.pdf`
28. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)

A An Attack on “Deniable” ZK Protocol of [26]

Consider a prover P, a verifier V, and a third party Z who wishes to obtain evidence that P has interacted with V in the 2-round “deniable” ZK protocol of [26]. The third party Z uses RO to prepare a valid verifier's first message α for the protocol, asks V to forward α to P, and then relay back P's response β. In this case, its clear that V cannot know the transcript of RO queries issued by Z during the creation of α, and therefore V cannot run the zero knowledge simulator of [26]. In fact, the soundness of the protocol ensures that V cannot efficiently construct an accepting reply β to α without P's help. Therefore, if V is later able to obtain a valid response β, Z is correctly convinced that P has interacted with V, and P cannot deny that the interaction took place. Thus, the protocol is not deniable in “real life”, despite meeting the definition of [26].