Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs

Yevgeniy Dodis* Victor Shoup† Shabsi Walfish‡

May 6, 2008

Abstract

Canetti et al. [11] 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. While arguing that none of the existing solutions achieved the desired level of security in the GUC-framework, the authors 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 [11] used rather inefficient constructions: the commitment scheme was bit-by-bit, while the zero-knowledge proof for a relation R was implemented using the generic Cook-Levin reduction to a canonical NP-complete problem.

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 [11], we build

- a direct and efficient constant-round GUC ZK for R from any "dense" Ω-protocol [31] 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 the knowledge of discrete log representation.
- the first *constant-rate* (and constant-round) GUCC scheme.

Additionally, we show how to properly model a random oracle (RO) in the GUC framework without losing *deniability*, which is one of the attractive features of the GUC framework. As an application, by adding the random oracle to the setup assumptions used by [11], we build the first two-round (which we show is optimal), deniable, straight-line extractable and simulatable ZK proof for any NP relation R.

*Computer Science Dept. NYU. dodis@cs.nyu.edu
†Computer Science Dept. NYU. shoup@cs.nyu.edu
‡Google, Inc. walfish@cs.nyu.edu

1 Introduction

UC FRAMEWORK. The *Universal Composability* (UC) framework introduced by Canetti [10] is a growingly popular framework for analyzing cryptographic protocols which are expected to be concurrently executed with other, possibly malicious. The UC framework has many very attractive properties, one of which is a very strong composition theorem, enabling one to split the design of a complex protocol into that of simpler sub-protocols. In particular, Canetti, Lindell, Ostrovsky and Sahai [17] 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 [10, 13, 16] 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 consequences of UC-security, such as polynomial-time simulation [39, 8]). To address this issue, the original UC framework was augmented to allow trusted setup. However, until the recent work of [11], 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 a freedom to choose its own CRS and embed some trapdoor information into it. As was argued in a series of papers [13, 19, 4, 11], this modeling creates several serious problems not present in the "plain UC". Two 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 [11] 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 give a simple and convincing example of this phenomenon (illustrated for ZK proofs in the CRS model) in Appendix A, but refer the reader to [11, 41], where the problems of local setup are discussed in more detail.

GUC FRAMEWORK. Motivated by solving the problems caused by modeling the setup as a local subroutine, Canetti et al. [11] 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 shown by [11] 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 Appendix B 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 (see Definition 6 and Theorem 8 for the precise definition and statement).

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

[1] Although [17] presented their results in the common reference string (CRS) model using the JUC theorem [19], one can extract a general implication which is independent of the CRS and does not use JUC. See page 131 of Walfish's thesis [41] for details.

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 "*secret* 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 instance, but nobody else. However, on a technical level, *UC+CRS is equivalent to GUC+SRS*, so the feasibility result of [17] would apply to the "global SRS" setup. Of course, such a "secret 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, [11] showed that the (global) CRS model (as well as other global setup which only provides *public* information, such as the random oracle model [32]) is *not* enough to sidestep the impossibility results of [10, 13, 16]. (In particular, the protocols of [17, 32] are insecure in the GUC framework with the global CRS/random oracle.) This means that any setup sufficient for GUC feasibility must provide some secret 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. [11] introduced a new setup assumption, called *Augmented CRS* (ACRS), and demonstrated how to GUC-realize commitment and ZK (and, thus, any other functionality) in the ACRS model, 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 "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 loose 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 [12], 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 the CRS, but 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 and the "restricted composition" problem of UC (see Appendix A) is also resolved, so 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

[2][11] also showed similar results in a variant of a PKI-like "key registration with knowledge (KRK)" setup from [4]. 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.

of efficiency? Unfortunately, the GUC-feasibility results of [11] 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 Damgard and Nielsen [25] (or the inefficient GUC feasibility results of [11]). 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 very realistic. In particular, 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.*, [15, 31, 36, 14], although there it was 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 this assumption 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, when 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" [31] for R. The notion of Ω-protocol's was introduced by Garay, MacKenzie and Yang [31]. Briefly, Ω-protocol's are usual Σ-protocol's (i.e., they satisfy special soundness and ZK properties of Σ-protocol's), with an extra property that one can generate the public parameter ρ of the system together with a trapdoor information τ, such that the 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). [31, 36] used Ω-protocol's for a similar task of building UC-secure ZK proofs in the CRS model (which was modeled in the "unfair" way mentioned earlier and is not GUC-secure). As a result, our compiler is *considerably* more involved than the compiler of [31, 36] (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 [11]. As a result, our compiler requires Ω-protocol's whose reference parameters are "dense" (i.e., indistinguishable from random), and none of the previous Ω-protocol's of [31, 36] is suitable for our purpose.

Thus, of independent interest, we show several novel *dense* Ω-protocol's. 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 [38], but in a very different setting), it is very general and gives a much more efficient Ω-protocol than the technique of [17, 11] requiring a generic Cook-Levin reduction. Second, we show a *very efficient*, number-theoretic based dense Ω-protocol for proving the knowledge of discrete log representation. Once again, this Ω-protocol had to use some interesting additional tools on top on the "non-dense" prior Ω-protocol of [31], such as a special "projective Paillier encryption" of Cramer and Shoup [20]. 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 Damgard and Nielsen [25], 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 [25] 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 [11] 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

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

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 [11] had a much more complicated form. Therefore, of independent interest, we build a new IBTC scheme which is based on Water's signature [42]. 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 the "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 [11, 12]). 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 heuristics 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 (see above for examples of such Ω-protocols). 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).[4] This makes the need for data erasures really minimal. We briefly compare the resulting deniable ZK protocol to previous related work on deniable ZK (e.g., [38, 34]) 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 modeled. The GUC security framework, introduced in [11], 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 [10] 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). This limited interaction prevents the model from capturing protocols that may share state with other protocols that might be running within the environment, since the environment in the distinguishing experiment cannot access the state of the parties it is interacting with.

The Generalized Universal Composability (GUC) security framework of [11] extends the original UC security framework of [10] 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,

[4] Of course, we can get a less efficient 2-round GUC ZK protocol with these properties, which does *not* rely on data erasures at all, by applying the Fiat-Shamir heuristics to the inefficient protocol of [11]. 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.

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

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 $(\texttt{retrieve}, \mathsf{sid}, ID)$ 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.)

Figure 1: The Identity-Based Augmented CRS Functionality

in any protocol session. GUC then allows the environment to access 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) in addition to the usual UC model interactions with the challenge protocol and its attacker. That is, we allow the environment to directly invoke and observe arbitrary protocols that run alongside the challenge protocol – and the arbitrary protocols may even share state information with the challenge protocol and the environment via shared functionalities. If a protocol π (that may share state with other protocols) "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 [10] and [11, 41]. 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 further detail in [41], 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 [11]). To avoid this impossibility, we make use of a special *Augmented Common Reference String* (ACRS) trusted setup (which we denote by the functionality $\mathcal{G}_{\rm acrs}$), as was first proposed in [11]. Another possible alternative would be to use a PKI model supporting "Key Registration with Knowledge" [4, 11] (which we denote by the functionality $\mathcal{G}_{\rm krk}$) – indeed, our efficient protocols can easily be transformed to use the $\mathcal{G}_{\rm krk}$ 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 $\mathcal{G}_{\rm acrs}$ 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 $\mathcal{G}_{\rm acrs}$ shared functionality. (As explained in Section 4 of [12], 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 $\mathcal{G}_{\rm acrs}$ in conjunction with the RO model.

2.3 Omega Protocols. The notion of an Ω-protocol was introduced in [31]. We recall the basic idea here. While our notion of an Ω-protocol is the same in spirit as that in [31], we also introduce some new properties, and there are a few points where the technical details of our definition differ. Details are in Appendix C.

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. Then 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) 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 $\mathcal{E}_{\text{td}}$ (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 ([22, 25]). 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, 21]. 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 [11]. We recall the basic idea here. Details in Appendix D.

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 $d \xleftarrow{\$} \mathcal{D}$ and $\kappa \leftarrow \mathsf{Com}_{ID}(d, m)$. Here, Com_{ID} is a deterministic algorithm (which implicitly takes 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.

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

$\mathcal{F}_{\rm zk}$, 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.

Figure 2: The Zero-Knowledge Functionality for Relation R

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 Extract, ECom, and Eqv, which work as follows. Given (PK, ID, MSK) as input, 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, $\mathcal{F}_{\rm zk}$, is described in Figure 2.[5]

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 ($\mathcal{G}_{\rm acrs}$) 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 IBTC. In addition, any system parameters Λ for the Ω-protocol 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 [31] 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 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 $\rho_1 \stackrel{\$}{\leftarrow} \Phi$, forms a commitment $\kappa_1 = \mathsf{Com}_P(d_1, \rho_1)$, and sends κ_1 to P.
2. P computes $\rho_2 \stackrel{\$}{\leftarrow} \Phi$ 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 $\mathcal{F}_{\rm zk}^{R}$ functionality in the secure-channels model, with security against adaptive corruptions (with erasures).*

[5]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.

Proof (sketch). We first observe that the protocol above only makes use of a single shared functionality, $\mathcal{G}_{\rm acrs}$. Therefore, we are free to make use of the equivalence theorem and EUC model of [11]. 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 $\mathcal{F}_{\rm zk}^{R}$ that is indistinguishable from $\mathcal{A}$ to any distinguishing environment $\mathcal{Z}$, in the presence of a shared setup $\mathcal{G}_{\rm acrs}$. 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 $\mathcal{F}_{\rm zk}^{R}$ 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 $\mathcal{G}_{\rm acrs}$.

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 $d' \leftarrow \mathsf{Eqv}(V, SK_V, a)$. κ', 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 (specifically, see Appendix F), 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 $\mathcal{G}_{\rm acrs}$.

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.

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

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.

Figure 3: The Commitment Functionality $\mathcal{F}_{\text{com}}$ (see [13])

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 [5]), 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. Details are in Appendix E.

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. The details are sketched in Appendix G.

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 ElGamal 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.

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.*

This is a standard argument, and we omit the proof for lack of space.

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 [11], 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 [11], 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 [11], based on a variation of an idea in [27], 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 [42]. 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: $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2 \xleftarrow{\$} \mathbb{G}^*$, $d_1, d_2 \xleftarrow{\$} \mathbb{Z}_q$, $\gamma_1 \leftarrow e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_{ID}^{-1})$, $\gamma_2 \leftarrow e(\bar{\mathbf{s}}_2, \mathbf{g}_1)$, $\gamma \leftarrow e(\mathbf{h}_1, \mathbf{g}_2)$, $\bar{\gamma} \leftarrow \gamma_1^{d_1}\gamma_2^{d_2}\gamma^m$. 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 [21], we can actually build an Ω-protocol for such a proof of knowledge, which avoids the cut-and-choose paradigm.

Garay et al [31] give an Ω-protocol for a very similar task, which could easily be adapted for our purposes, except that the protocol in [31] 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.[6]

The Camenisch and Shoup approach is based on a variant of Paillier encryption, introduced in Cramer and Shoup [20], which we call here *projective Paillier encryption*. While the goal in [21] and [20] was to build a chosen ciphertext secure encryption scheme, and we only require semantic security, it turns out their schemes do not require that the factorization of the RSA modulus N 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 [21], 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).

Due to lack of space, all these details are relegated to appendices: Appendix H for the IBTC scheme, and Appendix I for the Ω-protocol for proof of knowledge of a representation.

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 [6]. We first remark that even the RO model, without any additional setup, does not suffice for realizing GUC commitments or zero knowledge (see [11, 12]). 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 PKI setup models with a random oracle.

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

[6]It should be noted that the "mixed commitments" of Damgard and Nielsen [25] also have a very similar global extraction trapdoor, which is why we also cannot use them to build GUC commitments.

We give a more formal statement and proof of this result in Appendix K. 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 [28] 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 full version[7] of the paper, 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 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 [38] 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 stemming from precisely this issue that will break the deniability of the protocols proposed by Pass [38] (see Appendix J). Our GUC security modeling precludes the possibility of any such attacks.[8]

Of course, unlike the model of [38], 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 [34], 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 [34]. 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 [34] 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 [34]).

Finally, if one does not care about efficiency, applying our techniques to the inefficient protocols of [11], 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.*

[7] Additional details can be found in [41] as well.

[8] Similarly, the modeling of [33] also rules out such attacks. However, their protocols make use of special hardware based "signature cards" and require more than 2 rounds. They also do not consider the issue of adaptive corruptions.

References

[1] N. Asokan and V. Shoup and M. Waidner. Optimistic fair exchange of digital signatures. In *Eurocrypyt '98*. 1998.

[2] G. Ateniese and B. de Medeiros. Identity-based Chameleon Hash and Applications. *Proc. of Financial Cryptography*, 2004. Available at `http://eprint.iacr.org/2003/167/`.

[3] D. Beaver. Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. in *J. Cryptology*, vol 4., pp. 75–122, 1991.

[4] B. Barak, R. Canetti, J. Nielsen and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. of FOCS*, 2004.

[5] M. Bellare and A. Palacio. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *Proc. of Crypto*, Springer-Verlag (LNCS 2442), 2002.

[6] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS*, pp. 62–73, 1993.

[7] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. of STOC*, pp. 544–553, 1994.

[8] B. Barak and A. Sahai, How To Play Almost Any Mental Game Over the Net - Concurrent Composition via Super-Polynomial Simulation. In *Proc. of FOCS*, 2005.

[9] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, Vol. 13, No. 1, winter 2000.

[10] R. Canetti. Universally Composable Security: A New paradigm for Cryptographic Protocols. In *Proc. of FOCS*, pages 136–145, 2001.

[11] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universal Composability with Global Setup. In *Proc. of TCC*, 2007.

[12] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universal Composability with Global Setup (full version). Available at *Eprint Archive*, `http://eprint.iacr.org/2006/432`.

[13] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Proc. of Crypto*, pages 19–40, 2001.

[14] Ran Canetti and Shai Halevi and Jonathan Katz and Yehuda Lindell and Philip MacKenzie. Universally Composable Password-Based Key Exchange. In *Proc. of EUROCRYPT*, pp 404–421, 2005.

[15] Ran Canetti and Hugo Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Proc. of EUROCRYPT*, 2002.

[16] R. Canetti, E. Kushilevitz and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In *Proc. of Eurocrypt*, Springer-Verlag (LNCS 2656), pp. 68–86, 2003.

[17] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proc. of STOC*, pp. 494–503, 2002.

[18] R. Canetti, a. shelat and R. Pass. Cryptography from sunspots: how to use an imperfect reference string.. In *Proc. of FOCS*, 2007.

[19] R. Canetti and T. Rabin. Universal Composition with Joint State. In *Proc. of Crypto 2003*, Springer-Verlag, pp. 265-281, 2003.

[20] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public key encryption. In *Eurocrypt 2002*. 2002.

[21] J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In *Proc. of Crypto*, 2003.

[22] Ivan Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *Proc. of Eurocrypt*, Springer-Verlag (LNCS 1807), pp. 418–430, 2000.

[23] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G.Persiano and A. Sahai. Robust Non-Interactive Zero Knowledge. In *Proc. of Crypto*, pp. 566–598, 2001.

[24] Y. Dodis and S. Micali. Parallel Reducibility for Information-Theoretically Secure Computation. In *Proc. of Crypto*, Springer-Verlag (LNCS 1880), pp. 74–92, 2000.

[25] I. Damgård and J. Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In *Proc. of Crypto*, Springer-Verlag, pp. 581–596, 2002.

[26] C. Dwork, M. Naor and A. Sahai. Concurrent zero-knowledge. In *J. ACM*, 51(6):851–898, 2004.

[27] U. Feige. Alternative Models for Zero Knowledge Interactive Proofs. Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel, 1990.

[28] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proc. of Crypto*, Springer-Verlag (LNCS 263), pp. 181–187, 1987.

[29] S. Goldwasser, and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. *CRYPTO '90, LNCS 537*, 1990.

[30] O. Goldreich, S. Micali, and A. Wigderson. How to Solve any Protocol Problem. In *Proc.of STOC*, 1987.

[31] J. Garay, P. MacKenzie, and K. Yang. Strengthening Zero-Knowledge Protocols Using Signatures. In *Proc. of Eurocrypt*, Springer-Verlag (LNCS 2656), pp. 177–194, 2003.

[32] D. Hofheinz, J. Muller-Quade. Universally Composable Commitments Using Random Oracles. In *Proc. of Theory of Cryptography Conference*, pp. 58–76, 2004.

[33] D. Hofheinz, J. Muller-Quade, and D. Unruh. Universally Composable Zero-Knowledge Arguments and Commitments from Signature Cards. In *Proc. of the 5th Central European Conference on Cryptology MoraviaCrypt 2005*, June 2005.

[34] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and their Applications. In *Proc. of Eurocrypt*, Springer-Verlag, 1996.

[35] S. Micali and P. Rogaway. Secure Computation. unpublished manuscript, 1992. Preliminary version in *CRYPTO '91, LNCS 576*, 1991.

[36] P. MacKenzie and K. Yang. On Simulation-Sound Trapdoor Commitments. In *Proc. of Eurocrypt*, Springer-Verlag (LNCS 3027), pp. 382–400, 2004.

[37] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. Eurocrypt '99*. 1999.

[38] R. Pass. On Deniabililty in the Common Reference String and Random Oracle Model. In *Proc. of Crypto*, LNCS 2729, pp. 216–337, 2003.

[39] M. Prabhakaran and A. Sahai. New Notions of Security: Achieving Universal Composability without Trusted Setup. In *Proc. of STOC*, 2004.

[40] B. Pfitzmann and M. Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In *Proc. of ACM CCS*, pages 245–254, 2000.

[41] S. Walfish. Enhanced Security Models for Network Protocols. Ph.D. thesis, New York University, 2007. Available online at `http://www.cs.nyu.edu/web/Research/Theses/walfish_shabsi.pdf`.

[42] B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Proc. of Eurocrypt*, Springer-Verlag (LNCS 3494), pp 114–127, 2005.

A Deniability and Composability Problems of UC with Local Setup

In this section we show two main problems of the original UC framework when used with a local modeling of the setup. First, it does not preserve deniability, and, second, it restricts the kind of composition which is safe to do. We illustrate both of these problems using a zero-knowledge functionality analyzed in the UC model with the (local) CRS setup. As was shown by [23], appropriately-designed *non-interactive* ZK proofs of knowledge can be shown to be UC-secure (against static corruptions) in the CRS model. Let us call this non-interactive protocol π. In particular, on input x, witness w and CRS i, in produces a proof $\sigma = \pi(x, w, i)$. This means that there exists an efficiently verifiable relation $R = R_{x,i}$ (which depends on the CRS i but not on its trapdoor!) such that, without knowing a witness for w for x it is computationally hard to produce a non-interactive proof σ satisfying $R(\sigma) = 1$.

DENIABILITY. Although we give the formal definition of deniable ZK in Appendix B, the failure of UC in terms on deniability is obvious. In the ideal ZK functionality, if a prover proves some hard statement to a verifier, the verifier has no way to convince a third party of this fact. Thus, the prover can later deny proving the theorem to the verifier. Needless to say, this property no longer holds with a non-interactive proof: the proof σ can be transfered to any third party who will be convinced that the statement is true, even if the prover wants to deny it.

COMPOSITION. As for the composition, using a local CRS leaves one with two options. Either a brand new CRS must be created for each fresh protocol run, which is obviously impractical, or one can use the Joint-state UC (JUC) theorem of Canetti and Rabin [19]. The JUC theorem allows one to reuse the same setup, but restricts one to only use *specially-designed protocols*. In particular, no guarantees are provided if even one of the composed protocols can depend on the true reference string. As a consequence, composability falls prey to *chosen-protocol attacks*, when a maliciously designed secure protocol (which depends on the true setup) can suddenly become insecure when composed with another secure protocol.

For example, assume some party P is willing to prove in zero-knowledge some statement x that only P knows the witness w of. Second, let us consider a functionality $\mathcal{F}$, parameterized by our relation R above, which does nothing except if a party can provide a satisfying proof σ making R true. In this case, this party would learn P's sensitive witness w, but otherwise would learn nothing. We remark, in the ideal model, $\mathcal{F}$ is secure for P: since only P knows w and nobody but P can fake a valid proof σ of this fact, nobody should learn the value of w. Moreover, this should hold even if P is willing to run an ideal ZK functionality together with $\mathcal{F}$. However, the moment we implement the ideal ZK functionality by our UC-secure protocol π, the ideal security of $\mathcal{F}$ is suddenly broken: the verifier will obtain a valid non-interactive proof σ which will pass π, and then can extract the witness w of x. Notice, in this example

- The description of $\mathcal{F}$ depends on the CRS, but not on the trapdoor (which nobody knows).
- The users of $\mathcal{F}$ (in particular, P) might not even realize that the CRS is used in the description of $\mathcal{F}$. From their perspective, the environment just designed a protocol which is secure for P in the ideal model.
- A maliciously designed *secure* protocol became insecure, when composed with the UC-secure protocol proven so using the *local* setup modeling.

- The security of $\mathcal{F}$ breaks even if $\mathcal{F}$ is implemented using a trusted party, completely independent of the real-model implementation. In particular, we do not need to implement $\mathcal{F}$ using the CRS model to get the break. In fact, we already mentioned that the users of $\mathcal{F}$ need not even know about the CRS.

B GUC Security Implies Deniability

In this section we give a formal definition of a very strong type of deniability for zero-knowledge proofs, which we call *on-line deniability*. Roughly, it implies that an attacker (called the *informant* in this context) cannot convince the "judge" that a zero-knowledge proof is taking place, even if the attacker corrupts either the prover or the verifier, and even if the informant is constantly connected to an on-line judge. As far as we know, this is the strongest known definition of deniable zero-knowledge.[9] Nevertheless, we show that this extremely demanding definition[10] almost trivially follows from GUC-security, for any trusted setup which is modeled as a shared functionality (i.e., if the setup is *global*). This implication solidifies the (convincing but) informal claim of [11] that the GUC-framework is naturally equipped to provide deniability.

THE "PLAYERS". We start by introducing the relevant parties. We will have a *prover* P who is presumably proving a true statement x to a *verifier* V (for some language L), a *judge* $\mathcal{J}$ who will eventually rule whether or not the proof was attempted, an *informant* $\mathcal{I}$ who witnessed the proof and is trying to convince the judge, and a *misinformant* $\mathcal{M}$ who did not witness any proof whatsoever, but still wants to convince the judge that it did. Jumping ahead, the judge will not know whether it is talking to a true informant $\mathcal{I}$ or a "smart enough" misinformant $\mathcal{M}$ (who simply made up fake "evidence"), and $\mathcal{J}$'s task will be actually to determine which of the two it is talking to. In other words, the judge does not trust the (mis)informant, but is willing to work with it together in order to "incriminate" an honest prover or verifier.

THE "RULES". We assume that the prover and the verifier are part of some network environment, which might include some trusted parties (i.e., trusted setup like PKI) and some means of communication (i.e., a direct secure channel or a pair of secure channels to some trusted party). The exact details of this environment are not important for now. What is important, however, is that we assume that the judge should have a direct, private and "always-on" line to the (mis)informant (whichever it is talking to). Intuitively, this on-line channel between the judge and the (mis)informant, coupled with the fact that $\mathcal{J}$ cannot be "rewound", will guarantee us the *on-line deniability* property that we are after.[11] Additionally, we assume that the judge does not have a direct access to the players (for example, it does not learn the players' outputs), except through the (mis)informant and trusted setup (for example, it can know the CRS available to all the parties, or their public keys if a PKI is used).[12] Both the informant $\mathcal{I}$ and the misinformant $\mathcal{M}$ will have the capability of adaptively corrupting either the prover P or the verifier V (who start as honest) at any moment during the computation, and learning the entire state of the corrupted party following the corruption. Additionally, once either P or V is corrupt, the judge learns about the corruption, while the (mis)informant can totally control the actions of this party going forward. We assume the (mis)informant cannot corrupt the trusted setup: for example, in the case of a CRS, the misinformant cannot replace the CRS with a fake one (say, for which it knows a trapdoor). Finally, depending on the network structure, the (mis)informant

[9] For example, much stronger than concurrent zero-knowledge introduced by [26] (which required rewinding in the plain model) or "deniable ZK" in the RO model of [38] (which required extractability of RO). Of course, this means that in order to realize our notion much stronger setup assumptions are needed as well. Nevertheless, *the definition itself* appears to be the strongest and most demanding known in the context of deniability.

[10] Note, since this is not a paper about deniability, we tried to make the simplest possible definition which is already stronger than previous definitions. It might be possible to give even stronger definitions, as we hint in the sequel, but this was not the goal of this work.

[11] We notice that even if the real-world judge is actually "off-line" during the protocol run, the informant can still make use of some online resource, *e.g.*, some on-line bulletin board. Such bulletin board can record messages posted and provide unique identification numbers to them. Since the board cannot be rewound, by using these random numbers that the board (unknowingly) provided to the informant, the informant is effective creating a "judge" which cannot be rewound. In particular, prior technique for "off-line" deniability will not be sufficient in this pretty realistic scenario.

[12] In some situations, we might want an even stronger guarantee allowing the judge to have some limited interaction with the players, but we do not explore this possibility here. Thus, depending on the exact formalization of such partial interaction, our definition may or may not satisfy such stronger requirements.

might have partial control over the network. In our setting, where we envision secure channels, this means that it can only delay or block some messages (not knowing what they are), but other settings can be modeled as well.

THE "GAME". Now, assume we have a protocol π in our network which presumably implements a deniable zero-knowledge proof from P to V. In terms of correctness, we require that for any true statement x for which (honest) P has a valid witness w, if the informant $\mathcal{I}$ is passive, then V will always accept the proof. Additionally, the probability that a dishonest P^* can make an honest verifier V accept a false statement x is negligible.
As for (on-line) deniability, we define it as follows:

Definition 6. *We say that a protocol π achieves* on-line deniability *for zero-knowledge if for any efficient informant $\mathcal{I}$ there exists an efficient misinformant $\mathcal{M}$ such that that no efficient judge $\mathcal{J}$ can distinguish the following two experiments with non-negligible probability. In both experiments, after getting access to the setup (i.e., a CRS), the judge $\mathcal{J}$ chooses an arbitrary pair (x, w), such that w is a valid witness for x.*

1. **Informant Experiment.** *P gets an input (x, w), V and $\mathcal{I}$ get input x, and P and V are instructed to run the protocol π on their inputs against an informant $\mathcal{I}$ (who, in turn, interacts with the judge $\mathcal{J}$ on-line).*
2. **Misinformant Experiment.** *P gets an input (x, w), V and $\mathcal{I}$ get input x, but P and V are* not *instructed to run the protocol π. Instead, they run an "empty" protocol against a misinformant $\mathcal{M}$ (who, in turn, interacts with the judge $\mathcal{J}$ on-line).*

We make a few comments about this definition. First, without loss of generality the "real" informant $\mathcal{I}$ can simply be a "dummy" attacker[13] who blindly follows the instructions of the judge and truthfully reports back everything it sees. Second, the fact that the input x and a witness w are selected by the judge simultaneously serves two purposes. On the one hand, it ensures that the informant cannot incriminate an honest party even if the entire instance (x, w) is *adversarially chosen*. On the other hand, it ensures that the potential incrimination would happen *because P and V really ran π*, and not because *x in itself* has some incriminating information impossible to obtain otherwise (i.e., *irrespective of whether or not π was actually run*). Also, we gave the prover P the witness w even in the misinformant experiment, since we are not trying to deny that P knows the witness (maybe judge knows that P does), but rather that *P proved the statement to V* (who may or may not know its truth). Finally, we remark that although our definition is extremely strong (e.g., stronger than previously proposed models of deniable ZK), by itself it only protects the deniability of parties during the time that they *honestly followed the protocol π*. In particular, after a corruption, only *past* ZK proofs of a given party are guaranteed to be "deniable".[14]

DENIABILITY OF IDEAL ZK. Although this is not needed for our proof that GUC-security implies deniability, it is instructive to consider an *ideal ZK functionality* $\mathcal{F}_{\text{zk}}$ (for some relation R) described in Figure 2. Informally, $\mathcal{F}_{\text{zk}}$ is "deniable" because, despite informing the adversary that the statement x is true , it does not provide the adversary with any useful "evidence" of it. Using Definition 6, we can easily formalize this intuitive claim. Indeed, assume P and V have access to a trusted party T implementing $\mathcal{F}_{\text{zk}}$ (using private authenticated channels between the players and T), and consider a canonical "ideal-model" protocol ϕ, where P and V simply use T to implement message authentication. It is almost immediately clear that this protocol satisfies Definition 6 (irrespective if additional setup is available), formalizing the fact that $\mathcal{F}_{\text{zk}}$ is "deniable".

Lemma 7. *The canonical protocol ϕ achieves on-line deniability.*

Proof. Given the simplicity of ϕ, the misinformant $\mathcal{M}$ only has to report the state of a corrupted prover or verifier back to the judge. For the prover, it learns the witness w after the corruption, so it can just pretend that P activated $\mathcal{F}_{\text{zk}}$ on input (x, w). As for the verifier, it appends a fake receipt of the proof from $\mathcal{F}_{\text{zk}}$ (which does not depend on

[13]This is analogous to a result in [10], which shows that UC security has an equivalent formulation with respect to such a "dummy" adversary.

[14]At this stage, we do not even know how to *define* the deniability of a *corrupt* party *after* the corruption (which could be similar in spirit to the notion of "receipt-freeness" in electronic voting [7]). Depending on such a future formalization, our notion may or may not be applicable to this stronger setting.

w) only if the verifier was corrupted after the receipt of this receipt from T. Clearly, the view of $\mathcal{J}$ is identical in both experiments. □

DOES UC IMPLY DENIABILITY? Informally, since we just established the deniability of the ideal functionality $\mathcal{F}_{\text{zk}}$ in Lemma 7, one would imagine that if it were *realized* with a "secure protocol" analyzed via a sufficiently strong security definition/framework, such a realization of the ideal functionality $\mathcal{F}_{\text{zk}}$ would also be deniable. For example, the strong notion of security captured by the *Universal Composability* (UC) framework of Canetti [10] naturally appears to provide *exactly* this sort of guarantee. (The remainder of our discussion assumes some basic familiarity with the UC framework.) However, the UC framework lacks any mechanism for directly modeling *global setup*, such as a CRS. Therefore, in the past, UC secure protocols which make use of a CRS have simply modeled it as a *local setup* instead. This approach to modeling allows the UC simulator (*i.e.*, the adversary attacking the ideal functionality) to choose its own CRS. Clearly, this modeling does not capture the deniability concern, since such protocols can only be simulated if the simulation procedure is allowed to control the CRS (which is publicly visible in the real world, and therefore cannot be plausibly controlled by anyone other than the trusted authority generating the CRS).

SOLVING DENIABILITY WITH GUC. Luckily, a recently proposed extension to the UC framework allows us to directly model *global setup*. The *Generalized Universal Composability* (GUC) framework of [11] introduces the notion of a *shared functionality*. Such functionality can be shared by multiple protocols, and, as a result, the environment effectively has direct access to the shared functionality – meaning that the simulator is not empowered to control it. Thus, modeling global setup as a shared functionality allows us to properly capture additional security concerns, including deniability, with a UC-style security definition. We state it formally as follows.

Theorem 8. *Consider a real-model protocol π which utilizes some trusted setup* modeled as a shared functionality in the GUC framework. *Then, if π is a GUC-secure implementation of $\mathcal{F}_{\text{zk}}$ with respect to this setup, then π is on-line deniable (according to Definition 6) with respect to this setup.*

Proof. Assume π is GUC-secure. This means that there exists a simulator $\mathcal{S}$ which can fool any environment $\mathcal{Z}$ thinking that it is interacting with the "dummy" attacker $\mathcal{A}$ when parties run π. We define our misinformant $\mathcal{M}$ (for the "dummy" informant $\mathcal{I}$) to simply run $\mathcal{S}$ in its head, by pretending that the prover P activated the ideal functionality with the message $(\texttt{ZK-prover}, sid, P, V, x, w)$ (see Figure 2). Notice, $\mathcal{S}$ does not need to know the witness w to start working (since it does not learn it in the ideal model, unless the prover is corrupt at the start, or the environment tells it w). However, if P gets corrupt, $\mathcal{S}$ would expect to learn the witness w of P, which $\mathcal{M}$ can provide to $\mathcal{S}$ according to our definition of the misinformant experiment. We stress, however, that $\mathcal{M}$ can run $\mathcal{S}$ in relation to the setup as well, because the setup is modeled as a shared functionality, as we explained earlier. As $\mathcal{S}$ generates the simulated view of $\mathcal{A}$, $\mathcal{M}$ pretends that this is the view of the informant $\mathcal{I}$. By the GUC-security of π, it means that the simulated view of $\mathcal{A}$ should be indistinguishable from the real view of (dummy) $\mathcal{A}$ interacting with any $\mathcal{Z}$ which actually initiated P with the input $(\texttt{ZK-prover}, sid, P, V, x, w)$ in the real model, such as the judge $\mathcal{J}$ in its experiment with the actual informant $\mathcal{I}$. Thus, if we define $\mathcal{Z}$ to be mimicking the judge $\mathcal{J}$ except it also initiates a ZK proof from P to V, then the view of $\mathcal{J}$ in the informant/misinformant experiments is exactly the same as the view of $\mathcal{Z}$ in the real/ideal experiment experiment above (except without a possible output of V $(\texttt{ZK-proof}, sid, P, V, x)$). This completes the proof. □

C Ω-protocols

Let Π be an Ω-protocol, where algorithm $ParamGen$ generates a system parameter Λ, and algorithm $RefGen$ generates a reference parameter ρ. Recall that a given system parameter Λ determines X, L, W, R, as described in §2.3.

C.1 Soundness. We say Π satisfies the *special soundness* condition, if there exists an efficient algorithm $\mathcal{E}_{\text{rw}}$, called a *rewinding extractor*, such that every efficient adversary $\mathcal{A}$ wins the following game with negligible probability:

1. The challenger generates a system parameter Λ and a reference parameter ρ, and sends (Λ, ρ) to $\mathcal{A}$. Note that Λ defines X, L, W, R as above.

2. $\mathcal{A}$ computes $x \in X$, along with two accepting conversations (a, c, z) and (a, c', z') for x, where $c \neq c'$, and gives these to the challenger.
3. The challenger then runs $\mathcal{E}_{\text{rw}}$ on input $(\Lambda, \rho, x, a, c, z, c', z')$, obtaining $w \in W$.
4. $\mathcal{A}$ wins if w is not a witness for x.

We say that Π satisfies the *trapdoor soundness* condition if there exists an efficient algorithm $\mathcal{E}_{\text{td}}$, called the trapdoor extractor, such that every efficient adversary $\mathcal{A}$ wins the following game with negligible probability:

1. The challenger generates a system parameter Λ and a reference parameter/trapdoor pair (ρ, τ), and sends (Λ, ρ) to $\mathcal{A}$. Note that Λ defines X, L, W, R as above.
2. $\mathcal{A}$ computes $x \in X$, and sends x to the challenger.
3. The challenger generates a random challenge c, and sends this to $\mathcal{A}$.
4. $\mathcal{A}$ generates a response z, and sends this to the challenger.
5. The challenger runs $\mathcal{E}_{\text{td}}$ on input $(\Lambda, \tau, x, a, c, z)$, obtaining a value w.
6. $\mathcal{A}$ wins if (a, c, z) is an accepting conversation for x, but w is not a witness for x.

We say that Π satisfies the *special trapdoor soundness* condition if there exists an efficient algorithm $\mathcal{E}_{\text{td}}$, called the trapdoor extractor, such that every efficient adversary $\mathcal{A}$ wins the following game with negligible probability:

1. The challenger generates a system parameter Λ and a reference parameter/trapdoor pair (ρ, τ), and sends (Λ, ρ) to $\mathcal{A}$. Note that Λ defines X, L, W, R as above.
2. $\mathcal{A}$ computes $x \in X$, along with two accepting conversations (a, c, z) and (a, c', z') for x, where $c \neq c'$, and gives these to the challenger.
3. The challenger runs $\mathcal{E}_{\text{td}}$ on input $(\Lambda, \tau, x, a, c, z)$, obtaining a value w.
4. $\mathcal{A}$ wins if w is not a witness for x.

Using a standard rewinding argument ([5]), it is easy to show that special trapdoor soundness property implies the trapdoor soundness property, assuming the size of the challenge space is large (i.e., super-polynomial).

We say that Π is *partial trapdoor sound with respect to a function f* if the challenge space is large, and if there exist efficient algorithms $\mathcal{E}_{\text{rw}}$ and $\mathcal{E}_{\text{td}}$, such that every efficient adversary $\mathcal{A}$ wins the following game with negligible probability:

1. The challenger generates a system parameter Λ and a reference parameter/trapdoor pair (ρ, τ), and sends (Λ, ρ) to $\mathcal{A}$. Note that Λ defines X, L, W, R as above.
2. $\mathcal{A}$ computes $x \in X$, along with two accepting conversations (a, c, z) and (a, c', z') for x, where $c \neq c'$, and gives these to the challenger.
3. The challenger then runs $\mathcal{E}_{\text{rw}}$ on input $(\Lambda, \rho, x, a, c, z, c', z')$, obtaining $w \in W$.
   The challenger also runs $\mathcal{E}_{\text{td}}$ on input $(\Lambda, \tau, x, a, c, z)$, obtaining a value v.
4. $\mathcal{A}$ wins if w is not a witness for x, or if $v \neq f(w)$.

These definitions of special soundness and special trapdoor soundness are essentially the same as in Garay, et al [31], except for a the properties are stated in terms of attack games, rather than universal quantifiers; actually, one cannot use Strong-RSA-style arguments otherwise.

C.2 Honest Verifier Zero Knowledge. We say that Π is *honest verifier zero knowledge (HVZK)* if there exists an efficient algorithm ZKSim, called a *simulator*, such that every efficient adversary $\mathcal{A}$ has negligible advantage in the following game:

1. The challenger generates a system parameter Λ and a reference parameter ρ, and sends (Λ, ρ) to $\mathcal{A}$. Note that Λ defines X, L, W, R as above.
2. $\mathcal{A}$ computes $(x, w) \in R$, along with a challenge c, and sends (x, w, c) to the challenger.
3. The challenger chooses $b \in \{0, 1\}$ at random, and computes messages a and z in one of two ways, depending on b:

- if $b = 0$, then a and z are obtained by running the protocol, using the prover P with inputs (Λ, ρ, x, w), and using c as the challenge;
- if $b = 1$, then a and z are computed as the output of algorithm ZKSim on input (Λ, ρ, x, c).

The challenger sends (a, z) to $\mathcal{A}$.

4. $\mathcal{A}$ outputs $\hat{b} \in \{0, 1\}$.
5. $\mathcal{A}$'s advantage is defined to be $|\Pr[b = \hat{b}] - 1/2|$.

C.3 Dense Reference Parameters. A given system parameter Λ determines a set $\widehat{\Phi}$ of possible reference parameters. Let Φ be some larger set, also determined by Λ. We call elements of Φ *extended reference parameters*. Further suppose that:

- we have an efficient algorithm that samples the uniform distribution on Φ — this algorithm takes Λ as input;
- we have an efficient algorithm that determines membership in Φ — this algorithm also takes Λ as input;
- we have an efficiently computable binary operation on Φ that makes Φ into an *abelian group*; the inverse operation of the group should also be efficiently computable;
- it is computationally infeasible to distinguish a random element of Φ from a random element of $\widehat{\Phi}$.

If all of these conditions are met, we say that Π satisfies the *dense reference parameter property.*

The last condition may be stated more precisely as saying that every efficient adversary $\mathcal{A}$ has negligible advantage in the following game:

1. The challenger generates a system parameter Λ. This determines sets $\widehat{\Phi}$ and Φ as above.
2. The challenger chooses $b \in \{0, 1\}$ at random, and computes an extended reference parameter ρ in one of two ways, depending on b:
   - if $b = 0$, then $\rho \leftarrow \mathit{RefGen}(\Lambda)$;
   - if $b = 1$, then $\rho \xleftarrow{\$} \Phi$.

   The challenger sends ρ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $\hat{b} \in \{0, 1\}$.
4. $\mathcal{A}$'s advantage is defined to be $|\Pr[b = \hat{b}] - 1/2|$.

C.4 Σ-protocols. A Σ-protocol is just special type of Ω-protocol, in which there is no reference parameter. The notions of special soundness and HVZK carry over *verbatim* to Σ-protocols, while the various notions of trapdoor soundness, and the notion of dense reference parameters, do not apply to Σ-protocols.

D Identity-based trapdoor commitments

We define IBTCs similarly to [11], with the additional restriction of requiring the input to the commitment algorithm to serve as a decommitment (this removes the need to specify an opening algorithm with the scheme).

Definition 9 (Identity-Based Trapdoor Commitment). *An IBTC scheme IC is given by a 5-tuple of poly-time algorithms, $IC = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Com}, \mathsf{ECom}, \mathsf{Eqv})$, with the following basic properties:*

- Setup**:** Generates a public key PK and a master secret key MSK. We may omit explicit mention of PK (which is always used as an input for the remaining algorithms) as a notational convenience.
- $\mathsf{Extract}$**:** On input (PK, ID, MSK) outputs a trapdoor SK_{ID} for identity ID. The $\mathsf{Extract}$ algorithm may also be randomized.
- Com**:** On input (PK, ID, d, m) outputs a commitment κ for message m under identity ID using a decommitment value d that belongs to some set $\mathcal{D}$ (determined by PK). This is a *deterministic* algorithm (although d will be randomly generated). As a shorthand, we will write $\mathsf{Com}_{ID}(d, m)$ to denote such a commitment.
- ECom**:** On input (PK, ID, SK_{ID}) outputs a pair (κ, α), to be used with Eqv.
- Eqv**:** On input $(PK, ID, SK_{ID}, \kappa, \alpha, m)$ produces a decommitment $d \in \mathcal{D}$ such that $\mathsf{Com}_{ID}(d, m) = \kappa$.

IBTC schemes must satisfy the following security requirements:

- **Binding -** Every efficient adversary $\mathcal{A}$ wins the following game with negligible probability:
  1. The challenger generates (PK, MSK) using the Setup algorithm, and sends PK to $\mathcal{A}$.
  2. $\mathcal{A}$ queries an oracle for $\mathsf{Extract}(PK, \cdot, MSK)$ (many times).
  3. $\mathcal{A}$ outputs ID, d, m, d', m'.
  4. $\mathcal{A}$ wins if ID was not submitted to the $\mathsf{Extract}$ oracle in Step 2, $m \neq m'$, and $\mathsf{Com}_{ID}(d, m) = \mathsf{Com}_{ID}(d', m')$.
- **Equivocability -** The advantage of every efficient adversary $\mathcal{A}$ in the following game is negligible:
  1. The challenger generates (PK, MSK) using the Setup algorithm, and sends MSK to $\mathcal{A}$.
  2. $\mathcal{A}$ chooses an identity ID and a message m, and sends (ID, m) to the challenger.
  3. The challenger chooses $b \in \{0, 1\}$ at random, and computes $d \in \mathcal{D}$ in one of two ways, depending on b:
     - if $b = 0$, then $d \xleftarrow{\$} \mathcal{D}$;
     - if $b = 1$, then

     $$SK_{ID} \leftarrow \mathsf{Extract}(PK, ID, MSK),\ (\kappa, \alpha) \leftarrow \mathsf{ECom}(PK, ID, SK_{ID}),\ d \leftarrow \mathsf{Eqv}(PK, ID, SK_{ID}, \kappa, \alpha, m).$$

     The challenger then sends d to $\mathcal{A}$.
  4. $\mathcal{A}$ outputs $\hat{b} \in \{0, 1\}$.
  5. $\mathcal{A}$'s advantage is defined to be $|\Pr[\hat{b} = b] - 1/2|$.

E Details of GUC ZK analysis

First, we give the remaining details of the simulator, which are essentially standard fare in the UC literature.

Initialization. All parties are assumed to be initialized with a copy of the common reference string PK published by $\mathcal{G}_{\text{acrs}}$ during its global initialization phase. If the parties have not already been so initialized, $\mathcal{S}$ activates a copy of the $\mathcal{G}_{\text{acrs}}$ shared functionality, which then proceeds with the initialization. (Notice, an external copy of the globally shared $\mathcal{G}_{\text{acrs}}$ functionality is actually being invoked by $\mathcal{S}$, and $\mathcal{S}$ does not attempt to initialize any parties directly.)

Simulating communication with $\mathcal{Z}$. $\mathcal{S}$ simply forwards all communications between its internal copy of $\mathcal{A}$ and $\mathcal{Z}$.

Simulating communication with $\mathcal{G}_{\text{acrs}}$. $\mathcal{S}$ simply forwards all communications between its internal copy of $\mathcal{A}$ and $\mathcal{G}_{\text{acrs}}$.

Simulating a proof between two honest parties, P and V. Since we are in the secure channels model, $\mathcal{S}$ simply notifies $\mathcal{A}$ that communications (with messages of appropriate length for a proof protocol) have taken place between P and V. If $\mathcal{A}$ blocks any communications, $\mathcal{S}$ blocks V from receiving the output of $\mathcal{F}_{\text{zk}}^{R}$.

If either P or V is corrupted during the execution of the protocol, or subsequent to its completion, the protocol transcript preceding the corruption event is generated using the corresponding technique described below (including provisions for the internal state of the corrupt party).

Next, we present here a more detailed proof of the claim that the simulated execution of the GUC ZK protocol in §3 is indistinguishable from the real protocol. We structure our proof as a sequence of games, starting with the unaltered real-world interaction and proceeding by steps towards the ideal world interaction.

I_0 - **Real-world interaction** The original protocol runs with adversary $\mathcal{A}$.

I_1 - **Simulating interactions between two honest parties** This interaction is the same as I_0, only the computation of the actual protocol messages between two honest parties is delayed until one of them becomes corrupted (at which point, $\mathcal{A}$ expects to learn the corrupted party's history via examination of its internal state).

Given that we are in the secure channels model (which implies that any messages sent between honest parties remains entirely private until one of them is corrupted) this is only a conceptual change to I_0, so the distributions of these two games are trivially identical.

I_2 - **Modifying** (κ', d') **sent to corrupt verifier** When the verifier is corrupt but the prover is honest, we have the honest prover replace the commitment κ' to be sent in Step 5 of the protocol with an equivocable commitment opened to the same value. That is, we provide the honest prover with the trapdoor information SK_V of the corrupt verifier, and we modify Steps 5-7 of the protocol as follows:

5. P starts the Ω-protocol Π for relation R, with common input (Λ, ρ, x). As usual, P plays the role of the prover in Π and computes the first message a.
   P computes $(\hat{\kappa}', \alpha) \leftarrow \mathsf{ECom}(V, SK_V)$.
   P then sends the equivocable commitment $\hat{\kappa}'$ to the corrupt verifier V.
6. P receives a challenge c from the corrupt verifier V.
7. P computes a response z to V's challenge c. P computes $d' \leftarrow \mathsf{Eqv}(V, SK_V, \hat{\kappa}', \alpha, a)$, and sends d', a, z to the corrupt verifier V.

I_3 - **Modifying** (a, z) **sent to a corrupt verifier** Once again, this change affects only the scenario where the prover is honest and the verifier is corrupt.

This interaction is the same as I_2, only the values of a, z sent by the prover are generated using the HVZK Simulator for Π, rather than using Π directly.

That is, modify Step 7 of the protocol as follows:

7. P runs the HVZK simulator ZKSim for protocol Π on input (Λ, ρ, x, c), obtaining simulated messages a and z (this values are used instead of those that would have been generated via Π).
   P computes $d' \leftarrow \mathsf{Eqv}(V, SK_V, \hat{\kappa}', \alpha, a)$, and sends d', a, z to the corrupt verifier V.

I_4 - **Modifying the coin-toss commitment sent to corrupt provers** This interaction is the same as I_3 in case the verifier is corrupt. However, in the event that the prover is corrupt, we modify the coin-flipping stage of the protocol to replace the commitment sent by the honest verifier with an equivocable commitment opened to the same value.

That is, we provide the honest verifier with the trapdoor information SK_P of the corrupt prover, and we modify Steps 1-3 of the protocol as follows:

1. V computes $\rho_1 \xleftarrow{\$} \Phi$.
   V 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. V computes $d_1 \leftarrow \mathsf{Eqv}(P, SK_P, \hat{\kappa}_1, \alpha, \rho_1)$.
   V first verifies that $\rho_2 \in \Phi$. Then V sends the opening (d_1, ρ_1) to P.

I_5 - **Rigging the coin-flipping for corrupt provers** This interaction is the same I_4, only in the case where the prover is corrupt we further modify the coin-flipping phase of the protocol by changing the honest verifier's opening in Step 3 in order to "rig" the outcome of the coin-flipping to a pre-specified choice of reference parameter ρ.

Specifically, we make the following change:

3. P generates a pair $(\rho, \tau) \leftarrow \mathit{RefGen}(\Lambda)$, and sets $\rho_1 = \rho \cdot \rho_2^{-1}$ (rather than choosing ρ_1 at random).
   V computes $d_1 \leftarrow \mathsf{Eqv}(P, SK_P, \hat{\kappa}_1, \alpha, \rho_1)$.
   V first verifies that $\rho_2 \in \Phi$. Then V sends the opening (d_1, ρ_1) to P.

I_6 - **The ideal world** The most significant difference between I_5 and the final, simulated interaction in the ideal world is that the simulator uses the rigged coin-flipping technique to "trapdoor extract" a witness when the prover is corrupt – and then the honest verifier's output is taken from the $\mathcal{F}_{\mathrm{zk}}^R$ functionality, rather than a direct verification of the protocol messages. There is a minor difference when the verifier is corrupt – now it is the simulator who generates the protocol messages of the honest prover, rather than the prover itself. There are also corresponding changes in message delivery, none of which are ultimately visible to the environment or the (now, internally simulated) adversary.

Provided that the trapdoor extraction procedure produces a valid witness whenever the corrupt prover succeeds in convincing an honest verifier, I_5 and I_6 are identically distributed.

Claim 10. *I_2 is indistinguishable from I_1.*

Proof. The proof follows directly from the equivocability property of IBTCs. Namely, any environment that can distinguish I_2 from I_3 can easily be made to distinguish equivocable commitments from honest commitments to the same value by simply plugging in the challenge commitments appropriately (and using $\mathcal{G}_{\mathrm{acrs}}$ to provide the same public setup parameters as the challenger's IBTC system). □

Claim 11. *I_3 is indistinguishable from I_2.*

Proof. The proof follows by observing that if I_3 can be distinguished from I_2, then either (a) extended reference parameters can be distinguished from reference parameters, or (b) real conversations for Π can be distinguished from simulated conversations. The reduction follows by an application of Theorem 15, which allows us to "rig" the coin-tossing phase of the protocol (which is taking place in either interaction I_3 or I_2) to yield the same ρ specified by the challenger in the HVZK attack game. (Observe that we require the dense reference parameter property of the Ω-protocol, to satisfy the requirements of the Lemma.) Since we can now rest assured that the challenge reference parameter is being used for the Ω protocol in our interaction, we simply send (x, w, c) to the HVZK game challenger (where w is taken from the input to the honest prover P in our interaction), and replace the honest prover P's choice of (a, c) by the response from the challenger. Distinguishing I_3 from I_2 now corresponds precisely to guessing the HVZK challenge bit b. □

Claim 12. *I_4 is indistinguishable from I_3.*

Proof. This is a straightforward reduction to equivocability of IBTCs, as before. □

Claim 13. *I_5 is indistinguishable from I_4.*

Proof. We begin by considering a modified interaction I_4 where V computes ρ_1 by first selecting ρ uniformly at random, and then computing $\rho_1 \leftarrow \rho \cdot \rho_2^{-1}$. It is easy to see that the distribution of ρ_1 is unaltered, and thus we have made only a conceptual change to I_4.

Given this new view of I_4, it is easy to see that if we modify I_4 as per game I_5, the *only difference* is that the value ρ used by V is no longer random, but is instead chosen according to $RefGen$. From here, it is straightforward to reduce the distinguishing game to the Dense Reference Parameter property of the Ω-protocol. □

Claim 14. *I_6 is indistinguishable from I_5.*

Proof. This proof is by reduction to trapdoor extractability property of the Ω-protocol. Recall that the "rigging" of the reference string is already taken care of by the technique of I_5 (so we may easily arrange for the same ρ selected by the challenger in the extraction attack game of the Ω-protocol). The trapdoor soundness property for Π guarantees that we get a witness with overwhelming probability. □

Combining the preceding claims yields the desired proof that the real interaction I_0 is indistinguishable from the ideal interaction I_6.

F A coin-tossing lemma

First, we consider a generic indistinguishability task T (this models encryptions, commitments, HVZK, etc.) The task T has a system parameter Λ and reference parameter ρ, which are generated by some given algorithms.

Further, we assume that reference parameters belong to some abelian group Φ, in which the group operation (which we write multiplicatively) and the group inverse operation are efficiently computable. Membership in Φ should also be efficiently decidable. We assume that the reference parameter generation algorithm generates the uniform distribution on Φ. We also assume that T specifies a probabilistic algorithm E that takes as input a system parameter Λ, a reference parameter ρ, a bit string x, and single "selection" bit b.

The indistinguishability property is defined by the following attack game:

Attack Game 1. *This is a game played between an adversary $\mathcal{A}$ and a challenger.*

1. *The challenger generates a system parameter Λ and a reference parameter ρ, and sends (Λ, ρ) to $\mathcal{A}$.*
2. *$\mathcal{A}$ computes a value $x \in \{0,1\}^*$, and sends x to the challenger.*
3. *The challenger chooses $b \in \{0,1\}$ at random, and computes $y \leftarrow E(\Lambda, \rho, x, b)$, and sends y to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $\hat{b} \in \{0,1\}$.*

The adversary's advantage is defined to be $|\Pr[b = \hat{b}] - 1/2|$.

We say that T is *computationally indistinguishable* if every efficient adversary has negligible advantage in Attack Game 1.

Now suppose that instead of generating the reference parameter ρ at random, we use a "coin tossing" protocol, as described in the following attack game. Assume we have a computationally binding commitment scheme Com, which may have system parameters Λ_{Com}.

Attack Game 2. *This is a game played between an adversary $\mathcal{A}$ and a challenger.*

1. *The challenger generates a system parameter Λ for T and a system parameter Λ_{Com} for Com, and sends $(\Lambda, \Lambda_{\mathsf{Com}})$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends a commitment κ_1 to the challenger.*
3. *The challenger generates a random reference parameter $\rho_2 \in \Phi$, and sends ρ_2 to $\mathcal{A}$.*
4. *$\mathcal{A}$ computes a value $x \in \{0,1\}^*$, and sends x, along with an opening (d_1, ρ_1) of κ_1 to the challenger.*
5. *The challenger verifies that (d_1, ρ_1) is a valid opening of κ_1, and that $\rho_1 \in \Phi$.*
   The challenger sets $\rho \leftarrow \rho_1 \cdot \rho_2$.
   The challenger chooses $b \in \{0,1\}$ at random, and computes $y \leftarrow E(\Lambda, \rho, x, b)$, and sends y to $\mathcal{A}$.
6. *$\mathcal{A}$ outputs $\hat{b} \in \{0,1\}$.*

The adversary's advantage is defined to be $|\Pr[b = \hat{b}] - 1/2|$.

Theorem 15. *If T is computationally indistinguishable and Com is computationally binding, then the advantage of every efficient adversary in Attack Game 2 is negligible.*

Proof. Suppose $\mathcal{A}$ is an efficient adversary playing in Attack Game 2 with advantage α, where $\alpha \geq 1/P$ for some polynomial P (and for infinitely many values of the security parameter). We construct an efficient adversary $\mathcal{A}'$ that contradicts the assumed computational indistinguishability of T, as in Attack Game 1.

Adversary $\mathcal{A}'$ runs as follows:

1. After receiving Λ and ρ from its challenger in Step 1 of Attack Game 1, $\mathcal{A}'$ generates a system parameter Λ_{Com} for the commitment scheme, and sends $(\Lambda, \Lambda_{\mathsf{Com}})$ to $\mathcal{A}$, as in Step 1 of Attack Game 2.
2. After receiving the commitment κ_1 from $\mathcal{A}$ as in Step 2 of Attack Game 2, $\mathcal{A}'$ generates $\rho_2^* \in \Phi$ at random and sends this value to $\mathcal{A}$, as in Step 2 of Attack Game 2.
3. If $\mathcal{A}$ does not respond with a valid opening, as in Step 2 of Attack Game 2, then $\mathcal{A}'$ outputs 0 and halts.
   Otherwise, if $\mathcal{A}$ responds with x^* and a valid opening (d_1, ρ_1), $\mathcal{A}'$ proceeds as follows:

$t \leftarrow P$
$\omega \xleftarrow{\$} \{1, \ldots, t\}$
for $i \leftarrow 1$ to t do
  if $i = \omega$ // *"plug and pray"*
    then $\rho_2 \leftarrow \rho \cdot \rho_1^{-1}$
    else $\rho_2 \xleftarrow{\$} \Phi$
  rewind $\mathcal{A}$ back to Step 2 of Attack Game 2 and send ρ_2 to $\mathcal{A}$
  if $\mathcal{A}$ responds with x and a valid opening (d_1', ρ_1') of κ_1 then
    if $i \neq \omega$ then output 0 and halt // *prayers unanswered* :(
    if $\rho_1' \neq \rho_1$ then output 0 and halt // *commitment broken!*
    send x to the challenger in Step 3 of Attack Game 1
    forward challenger's response y to $\mathcal{A}$
    when $\mathcal{A}$ outputs $\hat{b}$, then output $\hat{b}$ and halt
output 0 and halt // *rewinding failed to produce second opening* :(

Analysis. We claim that $\mathcal{A}'$ has advantage at least $1/4P^2$ in Attack Game 1, for infinitely many values of the security parameter. To see this, first consider the following "unbounded rewinding version" of Attack Game 2. In this game, if $\mathcal{A}$ does not open his commitment in step 4, the game halts (and we assume $\hat{b}$ is set to 0). Otherwise, if $\mathcal{A}$ opens his commitment, the challenger rewinds $\mathcal{A}$ to step 3, feeding him fresh, random values of ρ_2, until $\mathcal{A}$ opens his commitment for a second time. This goes on for as long as it takes. When the adversary does open for the second time, the game continues, exactly as in Attack Game 2, but using the values ρ_2, x, d_1, ρ_1 obtained in the second opening.

It is easy to argue that the advantage of $\mathcal{A}$ in the unbounded rewinding game is equal to α. Moreover, the expected number of attempts to obtain a second opening is easily calculated to be ≤ 1. Therefore, by Markov's inequality, the probability that the number of attempts exceeds t is at most $1/t$.

Now consider a t-bounded rewinding version of Attack Game 2, where $t := P$, in which the challenger aborts the rewinding after t attempts (setting $\hat{b}$ to 0 in this case). Since $t = P$ and the probability that $b = \hat{b}$ is $1/2$ if the challenger aborts, it follows that the advantage of $\mathcal{A}$ in the t-bounded rewinding game is $\geq \alpha - (1/2)(1/t) \geq 1/P - 1/2P = 1/2P$.

We make one more change to the t-bounded rewinding game: if the second opening does not agree with the first, the game is also aborted. Suppose that the probability of abort is ϵ. Under the assumption that the commitment scheme is binding, ϵ must be negligible, and so for all sufficiently large values of the security parameter, $\epsilon \leq 1/4P$. It follows that the advantage of $\mathcal{A}$ in this game is at least $1/2P - 1/4P = 1/4P$ for sufficiently large values of the security parameter.

We obtain $\mathcal{A}'$ from this last game using a standard "plug and pray" argument, which reduces the advantage by a factor of t, from which we obtain the bound $1/4P^2$. □

G Details of GUC commitment analysis

Most of the details for the analysis of the GUC commitment protocol are the same as the analysis of the GUC protocol (see Appendix E). The main difference is that a slightly more specialized argument is needed to prove that I_6 is indistinguishable from I_5. In I_6, the simulator uses the trapdoor extractor $\mathcal{E}_{\text{td}}$ during the commit phase, when P is corrupted, to extract a value m to pass to the ideal functionality. Later, during the reveal phase, P may open the commitment κ inconsistently as $(\hat{d}, \hat{m})$, where $\hat{m} \neq m$; we want to argue that this happens with only negligible probability, using the partial trapdoor soundness property for Π, relative to the function $f(d, m) := m$. Suppose to the contrary that the adversary succeeds in making such an inconsistent opening with non-negligible probability, even though V accepted the conversation (a, c, z) in the Ω-protocol. Then using the binding property of the IBTC scheme (applied to the commitment κ'), we can rewind the adversary to get a second accepting conversation (a, c', z'), where $c' \neq c$, also with non-negligible probability (e.g., the Reset Lemma of [5]). The partial trapdoor soundness property will guarantee that the rewinding extractor $\mathcal{E}_{\text{rw}}$, applied to these two conversations, will yield an opening of κ of the form (d, m). Now we have two openings of κ, $(\hat{d}, \hat{m})$ and (d, m), where $\hat{m} \neq m$, which breaks the binding property of the IBTC scheme — a contradiction.

H An efficient identity-based commitment scheme

We present an efficient identity-based commitment scheme for which an efficient Ω-protocol for proof of possession of an opening may be readily constructed.

H.1 Waters signature scheme. Our starting point is Waters signature scheme, which we review here. 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 $\mathbb{G}^* := \mathbb{G} := \{1\}$.

system parameters: a description of $\mathbb{G}$, $\mathbb{H}$, and e, along with

- 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 $H : \{0,1\}^* \to \{0,1\}^k$.

key generation: a random $x \in \mathbb{Z}_q$ is chosen, $\mathbf{h}_1 \in \mathbb{G}$ is computed as $\mathbf{h}_1 \leftarrow \mathbf{g}_1^x$, and $\mathbf{h}_2 \in \mathbb{G}$ is computed as $\mathbf{h}_2 \leftarrow \mathbf{g}_2^x$; the public key is $\mathbf{h}_1$, the secret key is $\mathbf{h}_2$.

signing: to sign a message $m \in \{0,1\}^*$, the hash $H(m) = b_1 \cdots b_k$ is computed (where each $b_i \in \{0,1\}$), a random $r \in \mathbb{Z}_q$ is chosen, and the signature $(\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{G} \times \mathbb{G}$ is computed as follows:

$$\mathbf{s}_1 \leftarrow \mathbf{g}_1^r, \;\; \mathbf{s}_2 \leftarrow \mathbf{h}_2 \tilde{\mathbf{u}}_m^r,$$

where

$$\tilde{\mathbf{u}}_m := \mathbf{u}_0 \prod_{b_i = 1} \mathbf{u}_i.$$

verification: given a message $m \in \{0,1\}^*$ and a signature $(\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{G} \times \mathbb{G}$, the verification algorithm checks 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 $\tilde{\mathbf{u}}_m$ is as above.

The Waters signature is secure under the *computational Diffie-Hellman (CDH)* assumption in $\mathbb{G}$, together with the assumption that H is collision resistant.

H.2 Proof of knowledge of a Waters signature. 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

$$\bar{\mathbf{s}}_1 \leftarrow \mathbf{s}_1^{1/w_1} \;\text{ and }\; \bar{\mathbf{s}}_2 \leftarrow \mathbf{s}_2^{1/w_2}.$$

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

$$\gamma_1 := e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_m^{-1}), \;\; \gamma_2 := e(\bar{\mathbf{s}}_2, \mathbf{g}_1), \;\; \text{and} \;\; \gamma := e(\mathbf{h}_1, \mathbf{g}_2).$$

The details are as follows:

1. The prover chooses $w_1, w_2 \in \mathbb{Z}_q^*$ at random, and computes
$$\bar{\mathbf{s}}_1 \leftarrow \mathbf{s}_1^{1/w_1} \text{ and } \bar{\mathbf{s}}_2 \leftarrow \mathbf{s}_2^{1/w_2}.$$
Let
$$\gamma_1 := e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_m^{-1}), \ \ \gamma_2 := e(\bar{\mathbf{s}}_2, \mathbf{g}_1), \ \text{ and } \ \gamma := e(\mathbf{h}_1, \mathbf{g}_2). \tag{1}$$
The prover then chooses $\bar{w}_1, \bar{w}_2 \in \mathbb{Z}_q$ at random, and computes $\bar{\gamma} \leftarrow \gamma_1^{\bar{w}_1}\gamma_2^{\bar{w}_2}$.
The prover sends the values
$$\bar{\mathbf{s}}_1 \in \mathbb{G}, \ \ \bar{\mathbf{s}}_2 \in \mathbb{G}, \ \ \bar{\gamma} \in \mathbb{H}$$
to the verifier.
2. The verifier chooses a challenge $c \in \mathbb{Z}_q$ at random, and sends c to the prover.
3. The prover computes
$$\hat{w}_1 \leftarrow \bar{w}_1 - cw_1 \text{ and } \hat{w}_2 \leftarrow \bar{w}_2 - cw_2$$
and sends the values
$$\hat{w}_1 \in \mathbb{Z}_q, \ \ \hat{w}_2 \in \mathbb{Z}_q$$
to the verifier.
4. The verifier checks that
$$\gamma_1^{\hat{w}_1}\gamma_2^{\hat{w}_2}\gamma^c = \bar{\gamma},$$
where $\gamma_1, \gamma_2, \gamma$ are as defined in (1).

It is easily verified that this Σ-protocol is HVZK, at least with respect to signatures of the form $(\mathbf{s}_1, \mathbf{s}_2)$, where $\mathbf{s}_1 \neq 1$ and $\mathbf{s}_2 \neq 1$. Indeed, for such a signature, $\bar{\mathbf{s}}_1$ and $\bar{\mathbf{s}}_2$ are independent and uniformly distributed over $\mathbb{G}^*$, and the rest of the protocol may be simulated using standard techniques. Since signatures output by the signing algorithm are of this form with overwhelming probability, this is sufficient for our purposes.

Also, this Σ-protocol satisfies the special soundness property. Indeed, given two accepting conversations with the same first flow, $(\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\gamma})$, one obtains $w_1, w_2 \in \mathbb{Z}_q$ such that
$$e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_m^{-1})^{w_1} \cdot e(\bar{\mathbf{s}}_2, \mathbf{g}_1)^{w_2} = e(\mathbf{h}_1, \mathbf{g}_2),$$
and since
$$e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_m^{-1})^{w_1} = e(\bar{\mathbf{s}}_1^{w_1}, \tilde{\mathbf{u}}_m^{-1}) \text{ and } e(\bar{\mathbf{s}}_2, \mathbf{g}_1)^{w_2} = e(\bar{\mathbf{s}}_2^{w_2}, \mathbf{g}_1),$$
it follows that $(\bar{\mathbf{s}}_1^{w_1}, \bar{\mathbf{s}}_2^{w_2})$ is a valid Waters signature on m.

H.3 An identity-based commitment scheme. The identity-based commitment scheme derived from the above Σ-protocol works as follows. Let $ID \in \{0,1\}^*$ be the identity to be associated with the commitment, and let $m \in \mathbb{Z}_q$ be the message to be committed. The commitment is computed as follows:

$\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2 \xleftarrow{\$} \mathbb{G}^*, d_1, d_2 \xleftarrow{\$} \mathbb{Z}_q$
$\gamma_1 \leftarrow e(\bar{\mathbf{s}}_1, \tilde{\mathbf{u}}_{ID}^{-1}), \gamma_2 \leftarrow e(\bar{\mathbf{s}}_2, \mathbf{g}_1), \gamma \leftarrow e(\mathbf{h}_1, \mathbf{g}_2)$
$\bar{\gamma} \leftarrow \gamma_1^{d_1}\gamma_2^{d_2}\gamma^m$
output $(\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\gamma})$

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 public group element with respect to public bases, we can easily build a Σ-protocol for proving knowledge of an opening of a given commitment. Indeed, we will show how to build an efficient Ω-protocol, where the message m is trapdoor extractable.

I An efficient Ω-protocol for proving knowledge of a representation

I.1 Number theory background. Let N be a positive integer.

- $[N]$ denotes the set $\{0, \ldots, N-1\}$;
- for $a \in \mathbb{Z}$, $a \bmod N$ denotes the unique integer $x \in [N]$ such that $a \equiv x \pmod{N}$;
- more generally, if $a, b \in \mathbb{Z}$ with $b \neq 0$ and $\gcd(b, N) = 1$, $(a/b) \bmod N$ denotes the unique integer $x \in [N]$ such that $a \equiv xb \pmod{N}$;
- $\mathbb{Z}_N$ denotes the ring of integers modulo N, and $\mathbb{Z}_N^*$ the multiplicative group of units;
- for $a \in \mathbb{Z}$, $[a]_N \in \mathbb{Z}_N$ denotes the residue class modulo N containing a.

The schemes we shall present below use as a system parameter an RSA modulus of the form $N = PQ$, where P and Q are large, distinct, "strong primes," i.e., primes of the form $P = 2P' + 1$ and $Q = 2Q' + 1$, where P' and Q' are odd primes. Define $N' := P'Q'$.

Note that in all applications, no entity is required to know the factorization of N — not even a simulator in a security proof. We assume N is generated by a trusted party who immediately disappears, taking the factorization of N with it.

We shall make use of the two abelian groups $\mathbb{Z}_N^*$ and $\mathbb{Z}_{N^2}^*$. We recall some basic facts:

- $\mathbb{Z}_N^*$ is isomorphic to $\mathbb{Z}_{N'} \times \mathbb{Z}_2 \times \mathbb{Z}_2$;
- if $j_N := \{[a]_N : (a \mid N) = 1\}$, where $(\cdot \mid \cdot)$ is the Jacobi symbol, then this definition of j_N is unambiguous, and j_N is a subgroup of index 2 in $\mathbb{Z}_N^*$; observe that $[-1]_N \in j_N$;
- the subgroup of squares $(\mathbb{Z}_N^*)^2$ has index 2 in j_N; note that $[-1]_N \notin (\mathbb{Z}_N^*)^2$;
- $\mathbb{Z}_{N^2}^*$ is isomorphic to $\mathbb{Z}_N \times \mathbb{Z}_{N'} \times \mathbb{Z}_2 \times \mathbb{Z}_2$;
- the special element $\mathfrak{w} := [1 + N]_{N^2} \in \mathbb{Z}_{N^2}^*$ has order N, and moreover, for each $m \in \mathbb{Z}$, we have $\mathfrak{w}^m = [1 + Nm]_{N^2}$;
- if $J_N := \{[a]_{N^2} : (a \mid N) = 1\}$, then this definition of J_N is unambiguous, and J_N is a subgroup of index 2 in $\mathbb{Z}_{N^2}^*$; observe that $[-1]_{N^2} \in J_N$;
- the subgroup of squares $(\mathbb{Z}_{N^2}^*)^2$ has index 2 in J_N; moreover, for all $a \in \mathbb{Z}$, we have $[a]_{N^2} \in (\mathbb{Z}_{N^2}^*)^2$ if and only if $[a]_N \in (\mathbb{Z}_N^*)^2$; in particular, $[-1]_{N^2} \notin (\mathbb{Z}_{N^2}^*)^2$;
- the subgroup of Nth powers $(\mathbb{Z}_{N^2}^*)^N$ has index N in $\mathbb{Z}_{N^2}^*$.

Now we state the intractability assumptions we will need:

- The *Strong RSA assumption* says that given a random $\mathsf{h} \in \mathbb{Z}_N^*$, it is hard to find $\mathsf{g} \in \mathbb{Z}_N^*$ and an integer $e > 1$ such that $\mathsf{g}^e = \mathsf{h}$.
- The *Quadratic Residuosity (QR) assumption* says that it is hard to distinguish a random element of j_N from a random element of $(\mathbb{Z}_N^*)^2$.
- The *Decision Composite Residuosity (DCR) assumption* says that it is hard to distinguish a random element of $\mathbb{Z}_{N^2}^*$ from a random element of $(\mathbb{Z}_{N^2}^*)^N$.

Another convenient fact is the uniform distribution on $[N/4]$ is statistically indistinguishable from the uniform distribution on $[N']$. Similarly, the uniform distribution on $[N^2/4]$ is statistically indistinguishable from the uniform distribution on $[NN']$.

Some consequences:

Lemma 16. *Under the QR assumption, it is hard to distinguish a random element of J_N from a random element of $(\mathbb{Z}_{N^2}^*)^2$. Under the DCR assumption, it is hard to distinguish a random element of $(\mathbb{Z}_{N^2}^*)^2$ from a random element of $(\mathbb{Z}_{N^2}^*)^{2N}$. Under the QR and DCR assumptions, it is hard to distinguish a random element of J_N from a random element of $(\mathbb{Z}_{N^2}^*)^{2N}$.*

Proof. Easy. □

Lemma 17. *Under the strong RSA assumption, given random elements* $\mathsf{h}_1, \ldots, \mathsf{h}_k \in (\mathbb{Z}_N^*)^2$*, it is hard to find* $\mathsf{g} \in \mathbb{Z}_N^*$*, along with integers* $c, d_1, \ldots, d_k$*, such that*

$$\mathsf{g}^c = \mathsf{h}_1^{d_1} \cdots \mathsf{h}_k^{d_k} \text{ and } c \nmid d_i \text{ for some } i = 1, \ldots, k.$$

Proof. This is a simple generalization of a lemma in Camenisch and Shoup [21]. □

I.2 Projective Paillier Encryption. Cramer and Shoup [20] proposed a variation of Paillier encryption [37]. Although their motivation was completely different than ours (constructing a CCA2-secure encryption scheme), it turns out that some the ideas can be utilized here. The same ideas were also used to similar effect by Camenisch and Shoup [21], although again, their motivation was somewhat different than ours.

In a nutshell, we present a variation of Paillier encryption that is semantically secure under the DCR assumption, and preserves essentially the same homomorphic properties of Paillier encryption; however, unlike the original Paillier scheme, the scheme we present here has a dense set of public-keys, in a sense corresponding to that in §C.3. Following the terminology in Cramer and Shoup [20], let us call this scheme the *Projective Paillier encryption scheme*.

system parameters: in addition to the RSA modulus N (of the form described in §I.1), the system parameters also include a random element

$$\mathfrak{g} \in (\mathbb{Z}_{N^2}^*)^{2N};$$

note that $\mathfrak{g}$ has order dividing N', and this order is equal to N' with overwhelming probability;

recall that $\mathfrak{w} := [1+N]_{N^2} \in \mathbb{Z}_{N^2}^*$ is the special element of order N;

key generation: compute $t \xleftarrow{\$} [N/4]$ and $\mathfrak{h} \leftarrow \mathfrak{g}^t$; the public key is $\mathfrak{h}$ and the secret key is t;

encryption: to encrypt a message $m \in [N]$ using a public key $\mathfrak{h}$, the encryption algorithm runs as follows:

$$r \xleftarrow{\$} [N/4], \ \ \mathfrak{u} \leftarrow \mathfrak{g}^r, \ \ \mathfrak{v} \leftarrow \mathfrak{h}^r \mathfrak{w}^m;$$

the ciphertext is $(\mathfrak{u}, \mathfrak{v})$;

decryption: given a ciphertext $(\mathfrak{u}, \mathfrak{v})$ and a secret key t, the decryption algorithm computes

$$\mathfrak{w}' \leftarrow \mathfrak{v}/\mathfrak{u}^t;$$

if $\mathfrak{w}'$ is of the form $[1+Nm]_{N^2}$ for some $m \in [N]$, then the algorithm outputs m, and otherwise, it outputs "reject."

Lemma 18. *Under the DCR assumption, the Projective Paillier encryption scheme is semantically secure.*

Proof. This follows from results in Cramer and Shoup [20]; however, we sketch the idea directly, as follows. Suppose we encrypt a message m as $(\mathfrak{u}, \mathfrak{v}) := (\mathfrak{g}^r, \mathfrak{h}^r \mathfrak{w}^m)$, where r is chosen at random from $[N/4]$. Certainly, we may instead choose r at random $[N^2/4]$ without affecting security. Under the DCR assumption (see Lemma 16), we may instead choose $\mathfrak{h}$ of the form $\mathfrak{g}^t \mathfrak{w}^s$, where s is chosen at random from $[N]$, subject to $\gcd(s, N) = 1$, without affecting security. Now suppose we instead choose r at random from $[NN']$, which also does not affect security. Writing $r = r_0 + N' r_1$, we see that r_1 is uniformly distributed over $[N]$ and is independent of $\mathfrak{u} = \mathfrak{g}^r = \mathfrak{g}^{r_0}$. But now the ciphertext perfectly hides m, since $\mathfrak{v} = \mathfrak{g}^{r_0 t} \mathfrak{w}^{(r_0 + N' r_1)s + m}$. □

I.3 An Ω-protocol. We now describe our Ω-protocol Π for proving knowledge of a representation. Our protocol works for any abelian group $\mathbb{H}$ of prime order q. The protocol will prove knowledge of a representation relative to k bases, allowing trapdoor extraction of $\ell \leq k$ of the exponents. In our application to commitments based on Waters signatures, $k = 3$ and $\ell = 1$.

In addition to a description of $\mathbb{H}$, the system parameters for Π consist of the RSA modulus N (as described in §I.1), along with the system parameter $\mathfrak{g} \in (\mathbb{Z}_{N^2}^*)^{2N}$ used for Projective Paillier encryption. Recall that $\mathfrak{w} := [1+N]_{N^2} \in \mathbb{Z}_{N^2}^*$ is the special group element of order N. In addition, the system parameters include random group elements

$$\mathsf{g}_0, \mathsf{g}_1, \ldots, \mathsf{g}_\ell \in (\mathbb{Z}_N^*)^2.$$

We need two more parameters, $B_{\rm c}$ and $B_{\rm p}$. Here, $B_{\rm c}$ is a bound on the size of the challenge space, and $B_{\rm p}$ is a "padding bound." The property required is that $1/B_{\rm c}$ and $1/B_{\rm p}$ are negligible. In addition, we require that

$$B_{\rm c}B_{\rm p}q \le N/2 \ \text{ and } \ B_{\rm c} \le \min\{q, P', Q'\}. \tag{2}$$

The reference parameter generation algorithm for Π is the key generation algorithm for the Projective Paillier encryption scheme. A reference parameter is a public key $\mathfrak{h} \in (\mathbb{Z}_{N^2}^*)^{2N}$ for the encryption scheme, and the corresponding trapdoor is the secret key $t \in [N/4]$, where $\mathfrak{g}^t = \mathfrak{h}$.

Now let $\gamma_1, \ldots, \gamma_k, \gamma \in \mathbb{H}$ and $w_1, \ldots, w_k \in [q]$, where

$$\gamma_1^{w_1} \cdots \gamma_k^{w_k} = \gamma. \tag{3}$$

The common inputs to the prover and verifier are the group elements $\gamma_1, \ldots, \gamma_k, \gamma$. The prover also gets the tuple $(w_1, \ldots, w_k)$ as a witness. Our protocol will prove knowledge of values $w_1, \ldots, w_k \in [q]$ satisfying (3), with the values $w_1, \ldots, w_\ell$ being trapdoor extractable. More precisely, our protocol will satisfy the partial trapdoor soundness property relative to the function $f(w_1, \ldots, w_\ell) := (w_1, \ldots, w_k)$.

The protocol Π runs as follows:

1. The prover computes

   $r_1, \ldots, r_\ell, s \xleftarrow{\$} [N/4]$
   for $i \leftarrow 1$ to ℓ: $\mathfrak{u}_i \leftarrow \mathfrak{g}^{r_i}$, $\mathfrak{v}_i \leftarrow \mathfrak{h}^{r_i}\mathfrak{w}_i^{w_i}$
   $\mathsf{h} \leftarrow \mathsf{g}_0^s \mathsf{g}_1^{w_1} \cdots \mathsf{g}_k^{w_\ell}$
   $\bar{r}_1, \ldots, \bar{r}_\ell, \bar{s} \xleftarrow{\$} [B_{\rm p}B_{\rm c}N/4] \setminus [B_{\rm c}N/4]$
   $\bar{w}_1, \ldots, \bar{w}_k \xleftarrow{\$} [B_{\rm p}B_{\rm c}q] \setminus [B_{\rm c}q]$
   $\bar{\gamma} \leftarrow \gamma_1^{\bar{w}_1} \cdots \gamma_k^{\bar{w}_k}$
   for $i \leftarrow 1$ to ℓ: $\bar{\mathfrak{u}}_i \leftarrow \mathfrak{g}^{\bar{r}_i}$, $\bar{\mathfrak{v}}_i \leftarrow \mathfrak{h}^{\bar{r}_i}\mathfrak{w}_i^{\bar{w}_i}$
   $\bar{\mathsf{h}} \leftarrow \mathsf{g}_0^{\bar{s}} \mathsf{g}_1^{\bar{w}_1} \cdots \mathsf{g}_k^{\bar{w}_\ell}$

   and sends

   $$\{(\mathfrak{u}_i, \mathfrak{v}_i, \bar{\mathfrak{u}}_i, \bar{\mathfrak{v}}_i)\}_{i=1}^{\ell}, \ \ \bar{\gamma}, \ \ \mathsf{h}, \bar{\mathsf{h}}$$

   to the verifier.
2. The verifier chooses a random challenge $c \in [B_{\rm c}]$.
3. The prover computes

   for $i \leftarrow 1$ to k: $\hat{w}_i \leftarrow \bar{w}_i - cw_i$
   for $i \leftarrow 1$ to ℓ: $\hat{r}_i \leftarrow \bar{r}_i - cr_i$
   $\hat{s} \leftarrow \bar{s} - cs$

   and sends

   $$\{\hat{w}_i\}_{i=1}^{k}, \ \ \{\hat{r}_i\}_{i=1}^{\ell}, \ \ \hat{s}$$

   to the verifier.
4. The verifier checks that

   $$\hat{w}_i \in [N/2] \ \text{ for } i = 1, \ldots, \ell,$$

   and verifies the following relations:

   $$\bar{\gamma} = \gamma^c \cdot \prod_{i=1}^{k} \gamma_i^{\hat{w}_i}, \qquad \bar{\mathsf{h}} = \mathsf{h}^c \cdot \mathsf{g}_0^{\hat{s}} \prod_{i=1}^{\ell} \mathsf{g}_i^{\hat{w}_i},$$
   $$\bar{\mathfrak{u}}_i = \mathfrak{u}_i^c \cdot \mathfrak{g}^{\hat{r}_i} \ \ (i = 1, \ldots, \ell), \qquad \bar{\mathfrak{v}}_i = \mathfrak{v}_i^c \cdot \mathfrak{h}^{\hat{r}_i}\mathfrak{w}^{\hat{w}_i} \ \ (i = 1, \ldots, \ell).$$

I.3.1 Analysis

In the attack game for partial trapdoor soundness, we assume an adversary has produced two accepting conversations

$$\{(\mathfrak{u}_i, \mathfrak{v}_i, \bar{\mathfrak{u}}_i, \bar{\mathfrak{v}}_i)\}_{i=1}^{\ell}, \bar{\gamma}, \mathsf{h}, \bar{\mathsf{h}}, \quad c, \quad \{\hat{w}_i\}_{i=1}^{k}, \{\hat{r}_i\}_{i=1}^{\ell}, \hat{s},$$
$$\{(\mathfrak{u}_i, \mathfrak{v}_i, \bar{\mathfrak{u}}_i, \bar{\mathfrak{v}}_i)\}_{i=1}^{\ell}, \bar{\gamma}, \mathsf{h}, \bar{\mathsf{h}}, \quad c', \quad \{\hat{w}'_i\}_{i=1}^{k}, \{\hat{r}'_i\}_{i=1}^{\ell}, \hat{s}',$$

where $c \neq c'$. Both conversations are fed into the rewinding extractor, while the first conversation, together with the trapdoor t, is fed into the trapdoor extractor. Let us define

$$\Delta c := c' - c, \qquad \Delta w_i := \hat{w}_i - \hat{w}'_i \ (i = 1, \ldots, k),$$
$$\Delta r_i := \hat{r}_i - \hat{r}'_i \ (i = 1, \ldots, \ell), \qquad \Delta s := \hat{s} - \hat{s}'.$$

¿From the verification relations, we have

$$|\Delta w_i| < N/2 \ \ (i = 1, \ldots, \ell) \tag{4}$$
$$\gamma^{\Delta c} = \prod_{i=1}^{k} \gamma_i^{\Delta w_i}, \tag{5}$$
$$\mathsf{h}^{\Delta c} = \mathsf{g}_0^{\Delta s} \prod_{i=1}^{\ell} \mathsf{g}_i^{\Delta w_i}, \tag{6}$$
$$\mathfrak{u}_i^{\Delta c} = \mathfrak{g}^{\Delta r_i} \ \ (i = 1, \ldots, \ell), \tag{7}$$
$$\mathfrak{v}_i^{\Delta c} = \mathfrak{h}^{\Delta r_i} \mathfrak{w}^{\Delta w_i} \ \ (i = 1, \ldots, \ell). \tag{8}$$

We also know that $|\Delta c| < B_{\mathrm{c}}$.

The rewinding extractor. Give two accepting conversations as above, since $0 < |\Delta c| < q$, the rewinding extractor may compute

$$w_i \leftarrow (\Delta w_i / \Delta c) \bmod q \ \ (i = 1, \ldots, k).$$

¿From (5), it is clear that $(w_1, \ldots, w_k)$ is indeed a valid witness, i.e., $\gamma = \prod_{i=1}^{k} \gamma_i^{w_i}$.

The trapdoor extractor. Given an accepting conversation as above, together with the trapdoor t, the trapdoor extractor runs as follows:

for $i \leftarrow 1$ to ℓ do
  $\mathfrak{w}'_i \leftarrow (\mathfrak{v}_i / \mathfrak{u}_i^t)^2$
  if $\mathfrak{w}'_i = [1 + N z_i]_{N^2}$ for some $z_i \in [N]$ then
    $z_i \leftarrow (z_i/2) \bmod N$
    if $z_i \geq N/2$ then $z_i \leftarrow z_i - N$ *// compute a "balanced" remainder*
    $w_i \leftarrow z_i \bmod q$
  else
    $w_i \leftarrow 0$ *// this is an error*

Lemma 19. *With the given rewinding and trapdoor extractors, under the Strong RSA assumption, protocol* Π *satisfies the trapdoor f-extractable property, where* $f(w_1, \ldots, w_k) := (w_1, \ldots, w_\ell)$.

Proof. This follows the same line of reasoning as in Camenisch and Shoup [21]. Given two valid conversation as above, as we already argued, the rewinding extractor always produces a valid witness $(w_1, \ldots, w_k)$, where

$$w_i := (\Delta w_i / \Delta c) \bmod q \ \ (i = 1, \ldots, k).$$

We want to show that the trapdoor extractor outputs $(w_1, \ldots, w_\ell)$ with overwhelming probability. ¿From the identity (6), with overwhelming probability, we have $\Delta w_i / \Delta c \in \mathbb{Z}$ for each $i = 1, \ldots, \ell$. This is where we use the

Strong RSA assumption (see Lemma 17). Moreover, from (4), we have $|\Delta w_i/\Delta c| < N/2$ for each $i = 1, \ldots, \ell$. ¿From (7) and (8), and the relation $\mathfrak{h} = \mathfrak{g}^t$, one obtains

$$\Big(\frac{\mathfrak{v}_i/\mathfrak{u}_i^t}{\mathfrak{w}^{\Delta w_i/\Delta c}}\Big)^{\Delta c} = 1 \ \ (i = 1, \ldots, \ell).$$

Now, the group $\mathbb{Z}_{N^2}^*$ has exponent $2NN'$, and since $|\Delta c| < B_\mathrm{c} \le \min\{P', Q'\}$, it follows that $\gcd(\Delta c, 2NN') \in \{1, 2\}$, which implies that

$$\Big(\frac{\mathfrak{v}_i/\mathfrak{u}_i^t}{\mathfrak{w}^{\Delta w_i/\Delta c}}\Big)^{2} = 1 \ \ (i = 1, \ldots, \ell).$$

This, together with the fact that $|\Delta w_i/\Delta c| < N/2$, implies that the output of the trapdoor extractor agrees with the output of the rewinding extractor. □

The zero knowledge simulator. Given a challenge c, the simulator runs as follows:

$r_1, \ldots, r_\ell, s \xleftarrow{\$} [N/4]$
for $i \leftarrow 1$ to ℓ: $\mathfrak{u}_i \leftarrow \mathfrak{g}^{r_i}$, $\mathfrak{v}_i \leftarrow \mathfrak{h}^{r_i}$
$\mathsf{h} \leftarrow \mathsf{g}_0^s$
$\hat{r}_1, \ldots, \hat{r}_\ell, \hat{s} \xleftarrow{\$} [B_\mathrm{p} B_\mathrm{c} N/4]$
$\hat{w}_1, \ldots, \hat{w}_k \xleftarrow{\$} [B_\mathrm{p} B_\mathrm{c} q]$
$\bar{\gamma} \leftarrow \gamma^c \cdot \prod_{i=1}^k \gamma_i^{\hat{w}_i}$
$\bar{\mathsf{h}} \leftarrow \mathsf{h}^c \cdot \mathsf{g}_0^{\hat{s}} \prod_{i=1}^{\ell} \mathsf{g}_i^{\hat{w}_i}$
for $i \leftarrow 1$ to ℓ do
    $\bar{\mathfrak{u}}_i \leftarrow \mathfrak{u}_i^c \cdot \mathfrak{g}^{\hat{r}_i}$, $\bar{\mathfrak{v}}_i \leftarrow \mathfrak{v}_i^c \cdot \mathfrak{h}^{\hat{r}_i} \mathfrak{w}^{\hat{w}_i}$

The first flow of the simulated conversation is

$$\{(\mathfrak{u}_i, \mathfrak{v}_i, \bar{\mathfrak{u}}_i, \bar{\mathfrak{v}}_i)\}_{i=1}^{\ell}, \ \ \bar{\gamma}, \ \ \mathsf{h}, \bar{\mathsf{h}},$$

while the third flow is

$$\{\hat{w}_i\}_{i=1}^{k}, \ \ \{\hat{r}_i\}_{i=1}^{\ell}, \ \ \hat{s}.$$

Lemma 20. *With the given simulator, under the DCR assumption, protocol* Π *satisfies the special HVZK property.*

Proof. This follows from the semantic security of Projective Paillier, and standard statistical distance arguments. □

Dense reference parameters. The set of reference parameters is suitably dense, in the sense of §C.3. Namely, under the QR and DCR assumptions, a randomly generated public key $\mathfrak{h}$ is computationally indistinguishable from a random element of the subgroup J_N of $\mathbb{Z}_{N^2}^*$; this follows from Lemma 16. Moreover, the set J_N is efficiently recognizable (just evaluate a Jacobi symbol) and the uniform distribution on J_N is efficiently samplable; indeed, one may generate a random element of J_N as follows:

$b \xleftarrow{\$} \{0, 1\}$, $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
output $(-1)^b \mathfrak{r}^2$

J An Attack on "Deniable" Zero Knowledge in the Random Oracle Model

Consider the following simple scenario, involving a prover P, a verifier V, and a third party Z (who wishes to obtain evidence that P has interacted with V). The third party Z constructs a verifier's first message α for the protocol. Z then asks the verifier V to supply evidence of interaction with P by simply forwarding α to P and relaying the response. In this case, its clear that V cannot know the transcript of random oracle queries issued by Z during the creation of α, and therefore V cannot run the zero knowledge simulator of [38]. Indeed, it is easy to show that V cannot efficiently construct an accepting reply to α without P's help. Therefore, if V is later able to obtain a valid response, Z is correctly convinced that P has interacted with V. This implies that the interaction between P and V is not truly "deniable zero knowledge", since it enables V to convince Z of the interaction.

K A Lower Bound on Round Complexity in the GUC Model

Here we will show that, for all practical intents and purposes, GUC-secure non-interactive commitment schemes and NIZK proof systems with adaptive security (or even mere *forward security*!) are impossible to achieve. First, we present a very general impossibility result for non-interactive commitment schemes with forward security in the GUC framework.

Theorem 21. *We say that an "oracle" (or Interactive Turing Machine) is* monotonically consistent *if it always returns the same response to party P when queried repeatedly on the same input by party P (even across separate sessions), except that it may choose not to respond to some queries when P is honest (otherwise, consistency holds independently of P's corruption status). Let $\mathcal{O}$ denote any PPT monotonically consistent oracle (whose outputs may depend on the* pid *of the querying party, but not the* sid*).*

There exists no non-interactive (single message), terminating protocol π that GUC-realizes $\mathcal{F}_{\mathrm{com}}$ with forward security (even in the erasure model), using only the shared functionality for $\mathcal{O}$. This holds even if the communication is ideally authentic. (In particular, we note that $\mathcal{O} = \mathcal{G}_{\mathrm{acrs}}$ and $\mathcal{O} = \mathcal{G}_{\mathrm{krk}}$ are efficient monotonically consistent oracles, even *if they are also combined with a shared functionality for random oracles.)*

Proof. Following the conventions established by [12], suppose there exists a non-interactive protocol π GUC-realizing $\mathcal{F}_{\mathrm{com}}$ in the $\mathcal{O}$ shared hybrid model. Then, in particular, there must be a simulator $\mathcal{S}$ such that $\mathsf{EXEC}_{\mathcal{F}_{\mathrm{com}},\mathcal{S},\mathcal{Z}} \approx \mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ for a particular adversary $\mathcal{A}$ and $\mathcal{O}$-externally constrained environment $\mathcal{Z}$, which are constructed as follows.

Let $\mathcal{A}$ be a "dummy adversary" that simply forwards protocol flows between corrupt parties and the environment (*i.e.*, when $\mathcal{A}$ receives a message from $\mathcal{Z}$, it will send the message on behalf of some specified corrupt party; similarly, whenever a corrupt party receives a message, $\mathcal{A}$ simply forwards it to $\mathcal{Z}$). Let $\mathcal{Z}$ operate by first corrupting party P (the committer), then choosing a random bit $b \xleftarrow{\$} \{0,1\}$ and running the commit phase of π on behalf of P in order to obtain commitment κ. Wherever π makes oracle queries to $\mathcal{O}$, $\mathcal{Z}$ issues the same queries on behalf of P (relying on monotonic consistency to be sure that it will obtain at least the same information as an honest P would). $\mathcal{Z}$ sends κ to $\mathcal{A}$, and waits for party V to output $(\mathtt{receipt}, \ldots)$. Next, $\mathcal{Z}$ runs the reveal phase of π on behalf of P (again, issuing queries to $\mathcal{O}$ where necessary) and forwards the corresponding messages through $\mathcal{A}$. Finally, $\mathcal{Z}$ waits for V to output $(\mathtt{reveal}, \mathsf{sid}, \hat{b})$ and if $b = \hat{b}$ then $\mathcal{Z}$ outputs 1; otherwise, $\mathcal{Z}$ outputs 0.

Clearly, if the GUC experiments above must remain indistinguishable, $\mathcal{S}$ must cause V to output $\hat{b} = b$ with overwhelming probability. Since $\mathcal{S}$ is interacting with $\mathcal{F}_{\mathrm{com}}$, it must specify the value of $\hat{b}$ to $\mathcal{F}_{\mathrm{com}}$ *prior to* the point where V outputs $(\mathtt{receipt}, \ldots)$, which always occurs before $\mathcal{Z}$ has initiated the reveal phase of π. That is, when $\mathcal{A}$ feeds $\mathcal{S}$ with an honestly generated commitment κ for a bit b, $\mathcal{S}$ will immediately compute a bit $\hat{b}$ such that $\hat{b} = b$ with overwhelming probability. Therefore, $\mathcal{S}$ acts like an "extractor" for commitments. However, we stress that while computing $\hat{b}$, $\mathcal{S}$ expects to have access to the oracle $\mathcal{O}$ – and, in particular, we note that party P is *corrupt* so that $\mathcal{S}$ may ask queries for P which would not be answered when P is honest (we will see how this matters shortly).

Intuitively, we have just shown that $\mathcal{S}$ can be used to extract a commitment sent by honest parties, violating the natural "hiding" property of the commitment, although this extractor requires access to the private oracle on behalf of the committer. Indeed, this "extractor" requires access to the private oracle for a *corrupt* committer, and therefore one might think this extractor is potentially "harmless" since it only violates the security of honest parties *after* they become corrupt. However, security against adaptive corruptions requires that *past transcripts* sent by honest parties who later become corrupt remain indistinguishable from *simulated transcripts* (which were created while the party was still honest). Of course, the simulator does not know the inputs of honest parties, so simulated commitments must be *independent* of the actual bit being committed to – and herein lies the contradiction. If there is an extractor that can open honest commitments to reveal the committed bit with overwhelming probability (when the committing party has later become corrupt), then this extractor distinguishes honest commitments from simulated commitments (where the extractor can only be correct/incorrect with probability $1/2$ for a commitment to a random bit, assuming it even generates an output).

More formally, we will show that the existence of the simulator $\mathcal{S}$ above contradicts the security of π against adaptive corruptions, by creating a particular environment $\mathcal{Z}'$ which succeeds in distinguishing $\mathsf{EXEC}_{\mathcal{F}_{\mathrm{com}},\mathcal{S}',\mathcal{Z}'}$ from $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}'}$ after an adaptive corruption operation for *any* simulator $\mathcal{S}'$ (as before, $\mathcal{A}$ is just a "dummy adversary"). As a notational convenience, we will write $\mathcal{S}^{\mathcal{O}}(P,\kappa)$ to denote the output bit $\hat{b}$ produced by the simulation above, when running on (honestly generated) commitment κ sent by P – recalling that $\mathcal{S}$ can only be run when P is corrupt.

Our $\mathcal{Z}'$ proceeds by corrupting V at the outset, and then choosing random a bit $b \xleftarrow{\$} \{0,1\}$, which it gives as input to the honest party P. It then expects to obtain κ (the output of the commit phase) from the adversary. After receiving κ, $\mathcal{Z}'$ instructs the honest party to reveal b, completing the protocol. In accordance with the forward security corruption model, $\mathcal{Z}'$ is now allowed to corrupt P, which will enable to $\mathcal{Z}'$ to obtain complete access to $\mathcal{O}$ for P. Once this access has been obtained, $\mathcal{Z}'$ is free to compute $\hat{b} \leftarrow \mathcal{S}^{\mathcal{O}}(P,\kappa)$. In the real world experiment (where protocol π is being attacked by the dummy adversary), the distribution of κ is exactly identical to its distribution in the original setting above where $\mathcal{S}$ outputs $\hat{b} = b$ with overwhelming probability. On the other hand, in the ideal world experiment (where $\mathcal{F}_{\mathrm{com}}$ is being attacked by $\mathcal{S}'$), we know that $\mathcal{S}'$ must produce κ independently of the bit b (since b is the hidden input of the honest party, sent only to $\mathcal{F}_{\mathrm{com}}$ which hides it from $\mathcal{S}$ information theoretically). This means that in the ideal world, we must have that $\hat{b} = b$ with probability at most $1/2$, since the entire computation of $\hat{b}$ is independent of b! Therefore, $\mathcal{Z}'$ can distinguish between the real world and ideal world experiments with probability at least $1/2 - \mathsf{negl}(\lambda)$, contradicting our assumption that π is GUC-secure. □

Note that the class of shared functionalities modeled by $\mathcal{O}$ is very large indeed, making this impossibility result quite strong. Not only do *all* the natural global setups mentioned thus far (ACRS, PKI, Random Oracle) fit the modeling requirements of $\mathcal{O}$, they still fit the requirements of $\mathcal{O}$ even if they are all are combined together. Indeed,it seems likely that this impossibility result will hold for virtually any natural setup assumption. Again, this impossibility result holds even in the authenticated links model.

Next, we will prove that the same impossibility also extends to NIZK proofs for many natural NP-relations. More formally, we describe the ideal Zero-Knowledge functionality for relation R, $\mathcal{F}_{\mathrm{zk}}^{R}$, is described in Figure 2.[15] Our impossibility result shows that it is impossible to have forward secure non-interactive GUC-realizations of $\mathcal{F}_{\mathrm{zk}}^{R}$ for non-trivial relations R (that are not already trivialized by the shared functionality for the global setup[16]).

Theorem 22. *We say that an "oracle" (or Interactive Turing Machine) is* monotonically consistent *if it always returns the same response to party P when queried repeatedly on the same input by party P (even across separate sessions), except that it may choose not to respond to some queries when P is honest (otherwise, consistency holds independently of P's corruption status). Let $\mathcal{O}$ denote any PPT monotonically consistent oracle (whose outputs may depend on the* pid *of the querying party, but not the* sid*).*

Further, we say that an NP-relation R defining some language L is non-trivial *if we believe that no PPT algorithm efficiently decides membership in L (*i.e.*, $L \notin BPP$). In particular, R is* non-trivial with respect to oracle $\mathcal{O}$ *if there is no PPT algorithm for efficiently deciding membership in L even when given oracle access to $\mathcal{O}$ (for arbitrary party identifiers, and even with all parties being marked as corrupt).*

There exists no non-interactive (single message), terminating protocol π that GUC-realizes $\mathcal{F}_{\mathrm{zk}}^{R}$ with forward security (even in the erasure model), using only the shared functionality for $\mathcal{O}$, for any NP-relation R that is non-trivial with respect to $\mathcal{O}$. This holds even if the communication is ideally authentic. (In particular, we note that $\mathcal{O} = \mathcal{G}_{\mathrm{acrs}}$ and $\mathcal{O} = \mathcal{G}_{\mathrm{krk}}$ are efficient monotonically consistent oracles, even if they are combined with the shared functionality for a random oracle.)

[15] Technically, the relation R might 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 (*e.g.*, using a *global* CRS modeling).

[16] Of course, it is easy to see how one might achieve non-interactive proofs for certain languages related to the global setup. For example, if the global setup is a PKI that uses key registration with knowledge, parties can trivially prove the statement that their public keys are "well-formed" (without even communicating at all!) since the global setup already asserts the verity of this statement on their behalf. Therefore, our impossibility result does not *necessarily* extend to cases where the relation R to be proved is determined by system parameters, but we are focusing on realizing zero-knowledge for natural relations that are *not* trivialized by the presence of the system parameters (where the impossibility result applies).

Proof. The proof is entirely analogous to the proof of Theorem 21, and therefore we will only sketch it at a high level and direct the reader to the previous proof for further details. Here will call the prover P and the verifier V.

Assuming there is a non-interactive and GUC-secure realization of $\mathcal{F}_{\mathrm{zk}}^{R}$, we first follow the approach of Theorem 22 in order show that (using a similar shorthand notation) there exists an extracting simulator $\mathcal{S}^{\mathcal{O}}(P, x, \psi)$. For any $x \in L$, this extracting simulator is capable of computing a witness w such that $(x, w) \in R$ if ψ is an honestly generated non-interactive proof according to protocol π. However, $\mathcal{S}^{\mathcal{O}}(P, x, \psi)$ expects to be run *after* the corruption of P, and it we are guaranteed that it will succeed in extracting a valid witness w (from any honestly generated proof ψ) with overwhelming probability in that scenario.

Then we construct an environment $\mathcal{Z}'$ which, parameterized by any $(x, w) \in R$, first feeds (x, w) to an honest prover P, and then obtains the resulting protocol flow ψ. Note that ψ is the protocol flow that is either observed by the dummy adversary running in the real world experiment, or is being "faked" by some simulator in the ideal model. The environment then corrupts the honest prover (after completion of the proof protocol), and runs $\mathcal{S}^{\mathcal{O}}(P, x, \psi)$ to obtain w. In particular, since w must be valid with overwhelming probability in the real world, it must also be valid with overwhelming probability in the ideal world running with *some* (efficient) simulator $\mathcal{S}'$ (or else the environment can successfully distinguish the two experiments, contradicting the claimed GUC-security of the protocol). However, the value of w is information theoretically hidden from $\mathcal{S}'$ by $\mathcal{F}_{\mathrm{zk}}^{R}$, so its clear that $\mathcal{S}'$ must output ψ given only x and access to the $\mathcal{O}$ oracle (in particular, while V is corrupt and P is honest).

To conclude the proof, we show how to obtain a witness w for statement x using only a $\mathcal{O}$ oracle, contradicting the non-triviality of L with respect to $\mathcal{O}$. Given any statement x, we first pick some party P to act as the prover, and V to act as the verifier. Then we run $\mathcal{S}'^{\mathcal{O}}(x)$ to produce a "fake" proof ψ. Finally, we run $\mathcal{S}^{\mathcal{O}}(P, x, \psi)$ to obtain w such that $(x, w) \in R$. Since this entire procedure produces a valid witness w for any $x \in L$ while using only oracle access to $\mathcal{O}$, we have successfully contradicted the non-triviality of L with respect to $\mathcal{O}$. $\square$