Composability and On-Line Deniability of Authentication

Yevgeniy Dodis[1⋆], Jonathan Katz[2⋆⋆], Adam Smith[3⋆⋆⋆], and Shabsi Walfish[4†]

[1] Dept. of Computer Science, New York University.
[2] Dept. of Computer Science, University of Maryland.
[3] Dept. of Computer Science and Engineering, Pennsylvania State University.
[4] Google, Inc.

Abstract. Protocols for *deniable authentication* achieve seemingly paradoxical guarantees: upon completion of the protocol the receiver is convinced that the sender authenticated the message, but neither party can convince anyone else that the other party took part in the protocol. Here, we introduce and study *on-line deniability*, where deniability should hold even when one of the parties colludes with a third party during execution of the protocol. This turns out to generalize several realistic scenarios that are outside the scope of previous models.

We show that a protocol achieves our definition of on-line deniability if and only if it realizes the message authentication functionality in the *generalized universal composability* framework; any protocol satisfying our definition thus automatically inherits strong composability guarantees. Unfortunately, we show that our definition is impossible to realize in the PKI model if adaptive corruptions are allowed (even if secure erasure is assumed). On the other hand, we show feasibility with respect to static corruptions (giving the first separation in terms of *feasibility* between the static and adaptive setting), and show how to realize a relaxation termed *deniability with incriminating abort* under adaptive corruptions.

1 Introduction

Message authentication allows a sender S to authenticate a message m to a receiver R. If S has a public key, message authentication is usually handled using digital signatures. A well-known drawback of digital signatures, however, is that they leave a trace of the communication and, in particular, allow R (or, in fact, any eavesdropper) to prove to a third party that S authenticated the message in question. In some scenarios such non-repudiation is essential, but in many other cases *deniability* is desired.

Deniable authentication, introduced in [16, 18] and studied extensively since then, achieves the seemingly-paradoxical guarantees that (1) the receiver is convinced that the message originated from the sender, yet (2) the receiver, even if

⋆ A portion of this work was done while visiting Harvard University.
⋆⋆ Supported by NSF CNS-0447075 and NSF CNS-0627306.
⋆⋆⋆ Supported by NSF TF-0747294 and NSF CAREER award 0729171.
† A portion of this work was done while at New York University.

malicious, cannot *prove* to anyone else that the sender authenticated the given message. Furthermore, (3) the receiver cannot be incriminated either, even by a malicious sender. (This is a meaningful strengthening whenever the receiver has a public key, as will be the case in our work; see further below.)

Deniability is a fundamental concept in cryptography. First, because non-repudiability is crucial for the free exchange of ideas: without the assurance of remaining off the record, individuals are discouraged from discussing subversive (or even simply embarrassing) topics. Second, more technically, deniability is intimately tied to the *simulation paradigm* that is central to our understanding of of cryptographic protocols.

Indeed, deniability is typically formalized via the simulation paradigm introduced in the context of zero-knowledge (ZK) proofs [20]. Zero-knowledge proofs, however, do not automatically provide deniability. Pass [27] points out that *non-interactive* ZK proofs are not deniable, nor are many existing ZK proofs in the *random oracle model*. Furthermore, ZK proofs for which simulation requires rewinding may not suffice to achieve strong notions of deniability, especially *on-line deniability* which protects each party even when the other party colludes with an on-line entity that cannot be "rewound" (see below for an example).[1] Looking ahead, we note that on-line deniability is only potentially feasible if receivers hold public keys, and we assume this to be the case in our work. Once receivers have public keys, however, protocols can realize the stronger semantics by which a sender can authenticate a message *for a specific receiver* R but not for anyone else.

One might question whether requiring on-line deniability is too strong. To see why it might be essential, consider a setting in which Bob talks to Alice while relaying all messages from Alice to an external party, such as a law-enforcement agent, who supplies Bob with replies. Ideally, a deniable authentication protocol would not permit the agent to distinguish a real conversation with Alice from one that Bob invents online. Previous, off-line models of deniability do not extend to this setting since they require that Bob "rewind" the conversation. Alternatively, imagine a publicly-readable and -writeable bulletin board (e.g., a wiki) where all entries are time-stamped and assigned unpredictable identifiers. A corrupt receiver running a protocol with an honest sender can post all the protocol messages it receives to the bulletin board, and then generate its responses based on the identifiers assigned to the resulting posts. Again, off-line deniability would not suffice since the (external) bulletin board cannot be rewound; in this case, the contents of the bulletin board would prove that the interaction occurred.

With the above motivation in mind, we introduce a strong notion of deniability that, in particular, implies on-line deniability. We then show that a protocol satisfies our definition if and only if it securely realizes the message authentication functionality $\mathcal{F}_{auth}$ in the recently introduced *generalized UC* (GUC) framework [7]. (This is an extension of Canetti's UC framework [5] that pro-

[1] In contrast, previous notions of deniable authentication only guarantee *off-line deniability* which protects against a malicious party who records the transcript and shows it to a third party after the fact.

vides a more expressive model of globally-available, "external" functionalities like a common reference string, PKI, etc.) Protocols proven secure with respect to our definition thus inherit all the strong composability properties of the UC framework. We stress that protocols realizing $\mathcal{F}_{auth}$ in the UC framework do *not* necessarily provide deniability (even with *joint state* [12]); in particular, digital signatures — which are clearly not deniable — realize $\mathcal{F}_{auth}$ in the UC setting [6]. Similarly, protocols realizing $\mathcal{F}_{auth}$ in the UC framework may be problematic when composed with other protocols that are allowed to depend on parties' public keys (see Section 2.3). In both cases, the reason is that the UC framework treats public keys as *local* to a particular session. When this condition is enforced, the expected security properties hold; when public keys are truly public, the expected security properties may not hold.

1.1 Our Results

Our first contribution is a definition of deniable authentication which, in comparison to prior work, guarantees stronger security properties such as on-line deniability and security under concurrent executions with arbitrary other protocols. Unfortunately, we show that achieving our notion of deniable authentication is *impossible* in the PKI model if *adaptive* corruptions are allowed. This holds even if secure erasure is assumed; if we are unwilling to allow erasure then we can rule out even the weaker notion of forward security (where, informally, parties are always honest during protocol execution, but their secret keys and memory might be compromised after completion of the protocol).

Our impossibility result is very different from prior impossibility results in the UC setting [5, 9, 11, 7]. Previous impossibility results assume secure channels as a primitive, and show that additional setup assumptions (such as a PKI) are necessary to realize other, more advanced functionalities. Here, we show that the basic functionality of authenticated channels cannot be realized even given the setup assumption of a PKI.

Faced with this strong negative result, we ask whether relaxed definitions of deniable authentication are achievable. In this direction, we show several positive results based on standard assumptions (and without random oracles).

First, we observe that our definition is achievable with respect to *static* adversaries. This appears to give the first separation between the static and adaptive settings with regard to *feasibility*.[2]

Second, we observe that our definition of deniable authentication can be achieved, with respect to adaptive corruptions, in the *symmetric-key* setting, where a trusted authority dispenses a secret keys to all pairs of parties. The protocol is straightforward, but it yields as a corollary a reduction from deniable authentication to deniable key exchange (the converse reduction is also possible, using non-committing encryption). Thus, our initial impossibility result implies

[2] Nielsen [26] shows a separation between the static and adaptive settings with regard to *round complexity*, but not feasibility.

that deniable key exchange is impossible (with regard to adaptive corruptions) in this setting.

As a partial solution, we suggest that symmetric keys for deniable authentication could be established using a weak form of deniable key exchange termed *key exchange with incriminating abort* (KEIA). Intuitively, KEIA guarantees deniability as long as the protocol terminates successfully. In case a malicious party P_2 aborts the protocol, however, this party may be able to obtain some incriminating evidence against the other party P_1 (namely, that P_1 was attempting to establish a key with P_2). The KEIA model is strictly stronger that static security. In light of our impossibility result, realizing KEIA (and hence a weak form of deniable authentication) seems to be a reasonable compromise.

As our third and most technically interesting feasibility result, we show how to realize KEIA in the PKI model with respect to *semi-adaptive* adversaries who corrupt parties either before or after an execution of the KEIA protocol (but not during). We stress that once a shared key is established, deniability is guaranteed even if corruptions later occur at any time. The security proof does not assume that honest parties can erase local information.

THIS EXTENDED ABSTRACT. Due to space constraints, this extended abstract discusses the proposed modeling of online deniability and states our main results. Proofs and many detailed discussions are deferred to the full version.

1.2 Previous Work in Relation to Our Own

Deniable authentication was formally introduced by Dwork, Naor, and Sahai [18] (though it was also mentioned in [16]), and it, along with several extensions and generalizations, has received significant attention since then [4, 31, 3, 18, 19, 17, 24, 27, 28, 14, 7, 25]. This prior work all assumes that only the sender has registered a public key; thus, it appears that prior work implicitly assumes that "guaranteed delivery" of messages to a specific, intended recipient is possible, and/or that the sender is willing to authenticate a given message for anyone, in which case on-line deniability does not make sense. Since we are specifically interested in on-line deniability, we consider the setting where the receiver has also registered a public key. (This setting was also considered in concurrent work done independently of our own [23, 32].) Once the receiver holds a public key, the sender can meaningfully authenticate a message *for a particular receiver* without being willing to authenticate the message for all other parties.

As previously mentioned, our definition implies very strong notions of deniability. In particular, our protocols remain secure under concurrent composition, something that was an explicit goal of prior work [18, 19, 24, 28]. To the best of our knowledge all prior constructions achieving concurrent security use timing assumptions [18] which, though reasonable, seem preferable to avoid.[3] Our

[3] It is not clear whether plugging a generic concurrent ZK proof [29] into any existing deniable authentication protocol would yield a concurrently-secure protocol. In any case, concurrent ZK proofs require super-logarithmic round complexity [10] and thus do not result in efficient protocols.

protocols also remain secure when executed concurrently with arbitrary *other* protocols, something not addressed by previous work.

Designated-verifier proofs [22] and two-party ring signatures [30, 2] also provide authentication without non-repudiation. These primitives do not provide deniable authentication, however, since they incriminate the sender S and receiver R *jointly*; that is, they *do* leave evidence that either S or R was involved in authenticating some message. Deniable authentication, in contrast, does not implicate either party. Similarly, although there has been extensive work constructing and analyzing various deniable key-exchange protocols such as as SIGMA, SKEME, and HMQV (see [13, 14]), none of these protocols meets deniability. For example, SIGMA leaves a trace that the sender and receiver communicated, even if it does not reveal exactly what message was authenticated. (HMQV might satisfy our definition with respect to static adversaries, though we have not verified the details. The HMQV protocol is not, however, forward-secure unless erasure by honest parties is allowed)

2 Defining Deniable Authentication

In this section we define our notion of deniable authentication. We begin by giving a self-contained definition whose primary aim is to model *on-line* deniability. Our definition is based on an interactive distinguisher, much like the "environment" in the UC framework. Indeed, we observe that a protocol satisfies our definition if and only if it securely realizes the message authentication functionality $\mathcal{F}_{auth}$ in the GUC framework. This means that any protocol satisfying our definition automatically inherits the strong composability guarantees of the UC framework, and also provides some justification of the claim of Canetti et al. [7] that the GUC-framework naturally models deniability.

2.1 The Basic Definition

We start by introducing the relevant parties. We have a *sender* S who is presumably sending a message m to a *receiver* R, a *judge* $\mathcal{J}$ who will eventually rule whether or not the transmission was attempted, an *informant* $\mathcal{I}$ who witnesses the message transmission and is trying to convince the judge, and a *misinformant* $\mathcal{M}$ who did not witness any message transmission but still wants to convince the judge that one occurred. Jumping ahead, we will say a protocol is secure if the judge is unable to distinguish whether it is talking to a true informant $\mathcal{I}$ (interacting with S and R while they are running the protocol), or a misinformant $\mathcal{M}$.

We assume the sender and the receiver are part of some network environment, which includes some trusted parties (e.g., trusted setup like the PKI), some means of communication between the sender and receiver (e.g., a direct unauthenticated channel), and a direct, private channel between the judge and the informant (or misinformant, depending on the setting). Intuitively, this on-line channel, coupled with the fact that $\mathcal{J}$ cannot be "rewound", is what guarantees on-line deniability. Additionally, we assume that the judge does not have

direct access to the players (in particular, the judge does not know whether or not S really intended to send a message, and whether or not R really received one); instead, the judge must obtain information about the parties through the (mis)informant. However, the judge $\mathcal{J}$ *does* have direct access to any global setup (for example, in the case of a PKI it can reliably obtain the public keys of S and R), and so the misinformant cannot necessarily lie arbitrarily without being caught. Both the informant $\mathcal{I}$ and the misinformant $\mathcal{M}$ will have the capability of adaptively corrupting either the sender S or the receiver R at any moment during the computation, and learning the entire state of the corrupted party following the corruption (e.g., in the case of the PKI, this would include the secret key of this party). Additionally, once either S or R 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 global setup: for example, in the case of a PKI, the (mis)informant will not know the secret keys of any uncorrupted party (but can know all the public keys). Finally the (mis)informant has partial control over the network: it can totally control all unauthenticated links, and can block messages from authenticated links.

Now, assume we have a protocol π such that, for any message m, if the informant $\mathcal{I}$ is passive, then R will always accept and output the correct message m. Roughly, protocol π achieves *on-line deniable authentication* 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.

1. **Informant Experiment.** S and R run π in the presence of the informant $\mathcal{I}$ (which in turn interacts with the judge $\mathcal{J}$ in an on-line manner). R informs $\mathcal{J}$ upon receiving any message m' from S (which *may not* be the same as the input to S, *e.g.*, if S is corrupt and ignores its input).
2. **Misinformant Experiment.** S and R do nothing, and $\mathcal{J}$ only interacts with the misinformant $\mathcal{M}$. In particular, $\mathcal{M}$ is also allowed to falsely inform $\mathcal{J}$ that R received some arbitrary message m' from S.

See the full version for a precise definition. Note that, as in the UC model of security, the "real" informant $\mathcal{I}$ can, w.l.o.g., simply be a "dummy" attacker that blindly follows the instructions of the judge and truthfully reports back everything it sees.

2.2 Deniable Authentication in the GUC Framework

Before continuing, we refer the reader to the description of $\mathcal{F}_{auth}$, the ideal message authentication functionality, in Figure 1. (As in all our ideal functionalities, delivery of "delayed" messages is scheduled by the adversary.) The functionality $\mathcal{F}_{auth}$, essentially from [5], is "deniable" because, although the adversary learns that a message transmission took place, the adversary is not provided with any "evidence" of this fact that would be convincing to a third party. Since $\mathcal{F}_{auth}$ is deniable, one would imagine that if it were *realized* by a protocol Φ (using

Functionality $\mathcal{F}_{auth}$

1. Upon receiving an input $(\mathtt{send}, \mathsf{sid}, m)$ from party S, do: If $\mathsf{sid} = (S, R, \mathsf{sid}')$ for some R, then output $(\mathtt{sent}, \mathsf{sid}, m)$ to the adversary, and, after a delay, provide the same output to R and halt. Else ignore the input. (The adversary controls the channel delay.)
2. Upon receiving $(\mathtt{corruptsend}, \mathsf{sid}, m')$ from the adversary, if S is corrupt and $(\mathtt{sent}, \mathsf{sid}, m)$ was not yet delivered to R, then output $(\mathtt{sent}, \mathsf{sid}, m')$ to R and halt.

Fig. 1. The message authentication functionality of [5].

a sufficiently-strong notion of "realizing"), that Φ would be deniable as well. The strong notion of security captured by the *Universal Composability* (UC) [5] appears to provide exactly this sort of guarantee. However, the UC framework lacks a mechanism for directly modeling *global setup* such as a PKI. Instead, external features such as a PKI are modeled as being *local* to a particular session. Modeling a PKI in this way does not capture our intuitive notion of deniability, since we naturally picture a PKI as something *global* that can be accessed by anyone, independent of any particular session. As clear evidence of this, recall that ordinary signatures realize $\mathcal{F}_{auth}$ in the UC framework [6], though they are non-repudiable.

Thus, we turn instead to a recently-proposed extension to the UC framework called *generalized universal composability* (GUC) [7] which enables direct modeling of *global setup*. Canetti et al. claim [7], informally, that modeling global setup in this way provides a means of capturing additional security concerns, including deniability, within a UC-style security framework. The main benefit of such a definition is that protocols proven secure in the GUC framework can be composed *arbitrarily* while maintaining the same security guarantees. The following result is straightforward (albeit technical) to prove:

Proposition 1. *Protocol π achieves* on-line deniable authentication *if and only if it realizes $\mathcal{F}_{auth}$ in the GUC framework.*

Similarly to our treatment of authentication, one can prove that GUC-security of the identification ($\mathcal{F}_{id}$) and key-exchange ($\mathcal{F}_{ke}$) functionalities is equivalent to an appropriately-defined notion of "deniability". See the full version.

2.3 PKI Setup and Comparison with Prior Models

We model a PKI as a shared functionality that requires parties to register with a central authority who generates the public and secret keys for them unless the party is corrupt, in which case it can choose an arbitrary, but consistent, pair of keys. The central registration authority knows the secret keys of all parties (including the corrupt parties), and therefore the model is referred to as "Key Registration with Knowledge" [1]. This extra knowledge property only strengthens our impossibility result (Theorem 1). It is also provably *necessary* for our

feasibility results (Proposition 2). Finally, we require that honest parties protect their secret keys by using them only in the specified authentication protocol Φ (if honest parties could re-use the keys in arbitrary protocols, they might run a protocol which leaked their secret keys). There are several ways to model this requirement. For concreteness, we chose to parameterize the key registration functionality with a description of Φ, and allow honest parties to run Φ via calls to the key registration functionality. In real life, honest parties will actually have their secret key but must restrict its use. Corrupt parties are still allowed to retrieve (or set) their secret keys and use them in an arbitrary manner. The functionality $\mathcal{F}_{krk}^{\Phi}$ is defined in Figure 2. Again, the parameterization with Φ is simply a formalization of the constraint that parties use the key only for this protocol. It does not prevent keys from being stored locally.

Comparison to Prior Modeling of PKI. Our PKI functionality is defined similarly to that of [1]. However, unlike in [1], we restrict honest parties to only use their secret keys with the protocol Φ, and a single instance of $\mathcal{F}_{krk}$ persists across multiple sessions. In the terminology of [7], $\mathcal{F}_{krk}$ is a *shared functionality*, as opposed to a *local* one. The shared nature of the functionality implies, in particular, that environment has direct access to the public keys of all the parties (and the secret keys of corrupted parties).

Local setup, in contrast, is not adequate for capturing deniability properties. For example, standard digital signatures yield a UC-secure implementation of $\mathcal{F}_{auth}$ [6]. (Roughly, this is because the simulator can choose the keys of honest players and sign the messages on their behalf without the environment noticing.) Signatures, however, are not deniable. As argued in [7], local setup is also not satisfactory from the perspective of composition. Indeed, the local modeling of the PKI leaves one with two (known) options: either one needs to have a fresh instance of the PKI for every protocol run, which is impractical, or can use the "joint" UC (JUC) theorem of [12]. Unfortunately, the latter option only guarantees security when composing an authentication protocol with a restricted class of protocols. These protocols cannot depend on the public keys of the honest players– a condition that is to be hard to enforce. In the full version, we give an example of a protocol, which depends on the public keys, that is provably secure in the joint UC when composed with $\mathcal{F}_{auth}$, but not with the signature-based realization of $\mathcal{F}_{auth}$.

Thus, there are considerable advantages in modeling the PKI in a global manner, as done in this paper, both for deniability and for composition. Unfortunately, our main result in Theorem 1 shows that the most natural modeling of a PKI-based authentication is impossible in the face of adaptive attackers. This motivates us to describe several relaxations of security, which are still considerably better than signatures for our purposes.

2.4 Flavors of Protocols/Attackers

We consider several kind of attackers. So far, we considered an *adaptive* attacker that can corrupt the parties before, during and after the protocols. A

Shared Functionality $\mathcal{F}_{krk}^{\Phi}$

Parameterized by a security parameter λ, a protocol (or, more generally, a list of protocols) Φ, and a (deterministic) key generation function Gen, shared functionality $\mathcal{F}_{krk}$ proceeds thusly when running with parties $P_1, \ldots, P_n$:

Registration: When receiving a message ($\texttt{register}$) from an honest party P_i that has not previously registered, sample $r \leftarrow \{0,1\}^\lambda$ then compute $(PK_i, SK_i) \leftarrow \mathsf{Gen}^\lambda(r)$ and record the tuple (P_i, PK_i, SK_i).

Corrupt Registration: When receiving a message $(\texttt{register}, r)$ from a corrupt party P_i that has not previously registered, compute $(PK_i, SK_i) \leftarrow \mathsf{Gen}^\lambda(r)$ and record the tuple (P_i, PK_i, SK_i).

Public Key Retrieval: When receiving a message $(\texttt{retrieve}, P_i)$ from any party P_j (where $i = j$ is allowed), if there is a previously recorded tuple of the form (P_i, PK_i, SK_i), then return (P_i, PK_i) to P_j. Otherwise return $(P_i, \perp)$ to P_j.

Secret Key Retrieval: When receiving a message $(\texttt{retrievesecret}, P_i)$ from a party P_i that is either *corrupt* or honestly running the protocol code for Φ, if there is a previously recorded tuple of the form (P_i, PK_i, SK_i) then return (P_i, PK_i, SK_i) to P_i. In all other cases, return $(P_i, \perp)$.

Fig. 2. The Φ-Key Registration with Knowledge shared functionality.

static attacker can only corrupt parties before the beginning of the protocol. A *semi-adaptive* attacker can arbitrarily corrupt the parties before and after, but not during some particular phase of the protocol run. Finally, a *forward-secure attacker* can only corrupt the parties after the end of the protocol. We will distinguish between protocols which allow (honest) parties to securely erase their information (protocols with *data erasures*), and more realistic protocols protocols with *no erasures* (which is our default notion unless stated otherwise). Protocols with erasures are easier to design since the simulator does not need to fake the information erased prior to corruption (but harder to implement since modern hardware makes secure erasure difficult). Erasures do not affect the model for static attackers, but are meaningful for semi-adaptive, forward-secure and fully adaptive attackers.

3 Impossibility Result

In this section we prove our main result: adaptively-secure deniable authentication is impossible in the PKI model, even if erasures are allowed, even if each secret key is used only once, and even if no man-in-the-middle attacks are allowed. In the non-erasure model, we can also rule out even deniable protocols which are only forward-secure.

First Attempts. Before stating the precise impossibility results, it is instructive to consider some naïve strategies for ruling out adaptively-secure online deniable authentication. At first, it appears that since the behavior of the sender

S can be simulated in a straight-line manner *without* its secret key SK_S, there is an attacker who can impersonate the sender to the recipient R (by running the simulator). Of course, one of the reasons why this does not have to work is that R might use its own secret key SK_R for verification. In particular, a simulated transcript might look different than the real one to R, since R can employ knowledge of SK_R to distinguish the transcript (whereas the adversary does not have this ability). One "fix" to this problem is to (adaptively) corrupt R and then check the simulated transcript from R's viewpoint. Unfortunately, if R is corrupted too early (say, at the very beginning), it could be the case that knowledge of R's secret key is subsequently employed by the simulator in order to simulate the proper transcript (without talking to S or obtaining SK_S). Notice that such a simulation does not contradict soundness, since, in the real world, R would know that he is not simulating the conversation with S. On the other hand, if R is corrupted too late (say, at the very end), the initial flows from the "then-honest" party R were also chosen by the simulator, so there is no guarantee that they correspond to the behavior of a real R interacting with the sender's simulator.

Theorem 1. *There does not exist an adaptively-secure protocol Π for realizing the deniable authentication functionality $\mathcal{F}_{auth}$ in the $\mathcal{F}_{krk}^{\Pi}$-hybrid model. Moreover, the impossibility holds even under the following additional assumptions/constraints:*

- *Secure data erasures are allowed.*
- *Each honest party P will not use its secret key sk_P for more than one identification session (either as a sender or as a recipient).*
- *The attacker $\mathcal{A}$ will either try to impersonate the sender to an honest recipient exactly once, or will try to impersonate the recipient to an honest sender exactly once. In particular, $\mathcal{A}$ will not mount the man-in-the-middle attack against two honest parties.*

We also show how to extend the above impossibility result for semi-adaptive security to encompass even mere forward security, if erasures are not allowed.

Theorem 2. *If data erasures are not allowed, it is impossible to realize authentication ($\mathcal{F}_{auth}$) with forward security in any $\mathcal{F}_{krk}$-hybrid model. Moreover, impossibility holds even under the remaining constraints of Theorem 1.*

The proofs of the results above also rule out the possibility of key exchange $\mathcal{F}_{ke}$ and identification $\mathcal{F}_{id}$. (The result for $\mathcal{F}_{ke}$ is implied since key exchange can be used to implement authentication. The result for identification requires an examination of the impossibility proofs.)

Finally, we note that the weaker "bare public key" or "bulletin board" PKI setup notions (in which parties are allowed to post public keys without providing any secret keys for them) are not sufficient to realize these functionalities at all, even with mere static security. This seems to imply that key registration with knowledge is an unavoidable requirement for the PKI setup.

Proposition 2. *It is impossible to realize the identification, authentication, or key exchange functionalities in either the bare public key or bulletin board setup models. This impossibility holds even in the static corruption model.*

3.1 Proof Sketch for Impossibility (Theorem 1)

At a high level, the proof is an inductive argument showing that each round of the protocol either incriminates one of the parties, or can be simulated entirely (from either side) without knowledge of *any* secret keys. Of course, if either party can simulate the *entire* protocol without knowledge of any secret keys, it cannot be sound (*i.e.*, an attacker without S's key can identify himself as S to R). Thus, we show that either the protocol is not deniable, or it is not sound, contradicting our security requirements. The difficult part of the proof is the inductive step, which requires a delicate series of hybrid arguments. In particular, one must be careful about the issues mentioned above involving the order of corruptions in the various hybrids.

More formally, let Π be any protocol for deniable identification using $r = r(n)$ rounds, and assume toward a contradiction that Π is adaptively secure. Without loss of generality, we assume that the receiver goes first, and that the final message of the protocol is sent by the sender. In particular, we let $\alpha_1, \alpha_2, \ldots, \alpha_r$ denote the messages sent by the receiver R and $\beta_1, \ldots, \beta_r$ denote the response messages sent by the sender S. For convenience, we let α_{r+1} denote the binary decision bit of the receiver indicating whether or not R accepted. Throughout the protocol, we denote the current state of the sender and the receiver by ω_S and ω_R, respectively. This evolving state will include all the information currently stored by the given party, except for its secret key. Because we allow erasures, the current state does not include any information previously erased by this party.

We already stated that we only consider two kinds of attackers: sender impersonator $\mathcal{A}_S$ and receiver impersonator $\mathcal{A}_R$. The sender impersonator $\mathcal{A}_S$ will talk with an honest receiver R, while the receiver impersonator $\mathcal{A}_R$ will talk to an honest sender S. By assumption, there exists efficient simulators Sim_R and Sim_S for $\mathcal{A}_S$ and $\mathcal{A}_R$, respectively: the job of Sim_R is to simulate the behavior of R when talking to $\mathcal{A}_S$, while the job of Sim_S is to simulate the behavior of S when talking to $\mathcal{A}_R$. Moreover, the GUC security of Π implies that Sim_S and Sim_R have to work given only oracle access to R and S, respectively.[4] In particular, this means that in each round $1 \leq i \leq r$,

- As long as neither S or R is corrupted, Sim_S (resp. Sim_R) will receive some arbitrary message α_i (resp. β_i) and have to generate a "good-looking" response β_i (resp. α_{i+1}). Moreover, it has to do so *without the knowledge of the secret keys SK_S and SK_R or any of the future messages $\alpha_{i+1}, \ldots$ (resp. $\beta_{i+1}, \ldots$).*

[4] This is because, without loss of generality, $\mathcal{A}_S$ and $\mathcal{A}_R$ are simply the dummy parties forwarding the messages of the environment, and the simulator has to work for any environment. In fact, this property follows whenever there is an external "judge" with whom the adversary may interact when gathering evidence of protocol interactions.

- If S (resp. R) is corrupted, Sim_S (resp. Sim_R) will be given the secret SK_S (resp. SK_R), and will then be responsible to generate a "consistent-looking" internal state ω_S (resp. ω_R) for the corresponding party at round i. The pair (SK_S, ω_S) (resp. (SK_R, ω_R)) will then be given to the attacker and the environment.

From this description, we make our first key observation: as long as S and R are not corrupted, it is within the power of our attackers $\mathcal{A}_S$ and $\mathcal{A}_R$ to internally run the simulators Sim_S and Sim_R, respectively. In particular, we can make meaningful experiments where $\mathcal{A}_S$ runs Sim_S against an honest receiver R, or $\mathcal{A}_R$ runs Sim_R against an honest sender S. Of course, *a priori* it is unclear what happens during these experiments, since Sim_S was only designed to work against attackers $\mathcal{A}_R$ who *do not know* SK_R (as opposed to R itself, who certainly knows it), and similarly for Sim_R. The bulk of the proof consists of showing that such "unintended" usages of Sim_S and Sim_R nevertheless result in the "expected" behavior. We give a sequence of hybrid experiments which show that, without knowing the secret key of the sender, the simulator Sim_S can still successfully imitate the sender to an honest receiver, contradicting the soundness of identification. The details are given in the full version.

4 Circumventing the Impossibility Result

In this section, we discuss several positive results that circumvent the impossibility result of the previous section. We exhibit:

- a 1-message deniable authentication protocol tolerating *adaptive* corruptions, assuming a symmetric key infrastructure (*i.e.*, a symmetric key shared between the sender and receiver);
- a 1-message deniable authentication protocol tolerating *static* corruptions, assuming a PKI;
- a 4-message protocol for a relaxed notion of deniable authentication, which we dub *incriminating abort*, that tolerates *semi-adaptive* corruptions assuming a PKI. The protocol we give also satisfies the non-relaxed definition when the adversary is static.

Key exchange and deniable authentication are equivalent in the ideal model, and so the results above also imply the feasibility of corresponding notions of deniable key exchange.

The first two results above are quite simple (in the symmetric setting, a regular MAC suffices). The results serve mainly to illustrate the gap in complexity between the simpler settings (symmetric keys and static corruptions) and the most realistic setting of public keys and adaptive corruptions. The last feasibility result is significantly more involved. We feel it represents an interesting and reasonable compromise between realistic modeling and feasibility.

Deniability with Symmetric Keys Suppose for a moment that players have access to a *symmetric* key infrastructure, i.e. every pair of participants shares a

uniformly random long-term key that is unknown to other participants. Then S can authenticate a message m to R by appending a MAC (message authentication code) tag computed on the input (sid, S, R, m), where sid is a fresh random nonce. This is deniable roughly because the simulator for the protocol can make up a key for every pair of communicating players, and generate tags using the made-up key. In case of an adaptive corruption, the simulator can include the made-up key in the corrupted player's simulated memory contents.

This can be formalized in terms of the ideal functionalities $\mathcal{F}_{auth}$ and $\mathcal{F}_{ke}$. The SKI corresponds to granting every player of players one-time use of $\mathcal{F}_{ke}$, with key re-use modeled via the UC with joint state theorem [12]. The use of a MAC shows that $\mathcal{F}_{auth}$ can be reduced to a one-time $\mathcal{F}_{ke}$. In fact, the converse is also true: one can realize $\mathcal{F}_{ke}$ by encrypting a key using a protocol that is secure against *adaptive* but *passive* adversaries (known as *non-committing encryption* [8]). The flows of this protocol can be authenticated using $\mathcal{F}_{auth}$ to make it resistant to active attacks.

Lemma 1 ($\mathcal{F}_{auth} \iff \mathcal{F}_{ke}$). *If one way functions exist, then there exists a protocol that UC-realizes the natural multi-session extension of $\mathcal{F}_{auth}$ in the $\mathcal{F}_{ke}$ hybrid model, requiring only a single call to $\mathcal{F}_{ke}$. Conversely, if non-committing encryption exists, then there exists a protocol that UC-realizes $\mathcal{F}_{ke}$ in the $\mathcal{F}_{auth}$-hybrid model. These reduction hold even against adaptive adversaries.*

Static Security with a PKI We now turn to the public-key model. For certain types of public keys, players can use a PKI to generate a symmetric key k *non-interactively*. The idea of the protocol is then to use k to compute tags on messages as above. The authenticity of the message is derived from the authentication inherent in the PKI. The non-interactive key generation is not adaptively secure, and so the resulting protocols are only secure against static adversaries.

For example, suppose we operate in a cyclic group G generated by a generator g where the Decisional Diffie-Hellman (DDH) assumption holds. If each party P_i has a secret key $x_i \in \mathcal{Z}_q$ and a public key $y_i = g^{x_i}$, then P_i and P_j non-interactively share a key $k = g^{x_i x_j} = y_i^{x_j} = y_j^{x_i}$. Under the DDH asumption, k looks like a random group element to an attacker who only knows the public keys y_i and y_j. Either one of P_i or P_j can then use k as a MAC key to authenticate messages to the other.

This type of key exchange is abstracted as *non-interactive authenticated key exchange*. We model the PKI via the *registered keys with knowledge* functionality $\mathcal{F}_{krk}^{\Phi}$ (Figure 2) described in Section 2. (Key knowledge is necessary even for static security—see Proposition 2).

Theorem 3 (Static Security with a PKI). *Assuming the existence of non-interactive authenticated key exchange, there exists an efficient protocol Φ such that $\mathcal{F}_{auth}$ can be UC-realized in the $\mathcal{F}_{krk}^{\Phi}$-hybrid model, with static security.*

In contrast to the theorem, the impossibility result of Section 3 rules out adaptively secure GUC realizations. To the best of our knowledge, this is the first example of a task that provably cannot be realized at all in the presence

of adaptive adversaries, but for which statically-secure protocols are possible. In a result of a similar flavor, Nielsen [26] showed that adaptively-secure public key encryption (that is, *non-committing encryption* [8]) requires interaction in the standard public-key model. There are also many tasks for which known adaptively-secure protocols are less efficient or require stronger assumptions than their adaptively secure counterparts. However, our results provide the first example of a cryptographic task where static security is not just a matter of simplicity or efficiency, but one of feasibility.

4.1 Deniability with Incriminating Abort

Given the impossibility result of Section 3 and the possibility of PKI-based deniable authentication for static adversaries, it is natural to ask just how strong a notion of deniability can be achieved in the public key setting. We show here that one can guarantee deniability as long as *(a)* the protocol does not abort, and *(b)* there is one round during which neither the sender S nor the receiver R is adaptively corrupted. If the protocol aborts, the adversary can learn unsimulatable information depending on the secret keys of S and R—potentially enough to prove that one of them was trying to talk to the other. We call this notion *deniability with incriminating abort*.

We refer to an adversary that makes no corruptions during the run of the protocol as *semi-adaptive*. In particular, such an adversary will not corrupt any players during the protocol's single vulnerable round. However, the restriction to semi-adaptive security is also necessary to make the notion of abort meaningful: a fully adaptive adversary could always ensure that a protocol does not abort by corrupting a party immediately before it complains. Note that semi-adaptive security implies *forward security*, that is, a conversation that completes successfully is later deniable even if parties are forced to reveal the contents of their memories.

We phrase our results in terms of key exchange. This implies the corresponding feasibility results for authentication. However, forward security is especially meaningful for key exchange, because the protocol need only be run once, at setup time, for every pair of participants. If the key exchange protocol succeeds (with no adaptive corruption occurring during the protocol execution), then we can still use adaptively secure protocols realized in the $\mathcal{F}_{ke}$-hybrid model, and they will retain their adaptive security. In other words, the new protocol *almost* represents a deniable realization of $\mathcal{F}_{ke}$: if we could somehow guarantee that the protocol never aborts, then it would GUC-realize $\mathcal{F}_{ke}$.

Modeling Incriminating Abort We model the PKI via a shared, "registered keys with knowledge" functionality $\mathcal{F}_{krk}^{\Phi}$ (Figure 2), as in the protocols for static adversaries. The key exchange with incriminating abort functionality $\mathcal{F}_{\mathsf{keia}}$ is similar to $\mathcal{F}_{ke}$ except that the ideal-model adversary may explicitly request the protocol to abort, and in such a case the functionality will provide evidence that one of S and R was trying to talk to the other. It would be intuitively appealing to leak a single bit to the environment stating that a conversation occurred. We

Functionality $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$

$\mathcal{F}_{\mathsf{keia}}$, which is parameterized by an "incrimination procedure" $\mathsf{IncProc}$, and a security parameter λ proceeds as follows, when running in the $\mathcal{F}_{krk}$-hybrid model with parties S and R (who have already registered secret keys SK_S and SK_R, respectively) and adversary $\mathcal{S}$:

1. Upon receiving a message of the form $(S, \mathtt{keyexchange}, \mathsf{sid}, S, R, SK_S)$ from party S, if there are no previous records, then record the value $(\mathtt{keyexchange}, \mathsf{sid}, S, R, SK_S)$, mark S "active", and a send public delayed output $(\mathtt{keyexchange}, \mathsf{sid}, S, R)$ to R. (Otherwise, ignore the message.)
2. Upon receiving a message of the form $(R, \mathtt{keyexchange}, \mathsf{sid}, S, R, SK_R)$ from party R, if R is not yet "active", mark R as "active" and send a public delayed output $(\mathtt{active}, \mathsf{sid}, S, R)$ to S. (Otherwise, ignore the message.)
3. Upon receiving a message of the form $(\mathtt{setkey}, \mathsf{sid}, S, R, k')$ from $\mathcal{S}$, if R is corrupt and S is "active", then output $(\mathtt{setkey}, \mathsf{sid}, S, R, k')$ to S and R, and halt. If R is "active" but not corrupt, then sample a fresh key $k \leftarrow \{0,1\}^\lambda$ and send the message $(\mathtt{setkey}, \mathsf{sid}, S, R, k)$ to R. Furthermore, if S is "active", then send the delayed message $(\mathtt{setkey}, \mathsf{sid}, S, R, k)$ to S as well. In all cases, this completes the protocol, and the functionality halts.
4. Upon receiving a message of the form $(\mathtt{abort}, \mathsf{sid}, S, R)$ from $\mathcal{S}$, if S is "active", send $(\mathtt{abort}, \mathsf{sid}, S, R)$ as a delayed message to S and mark S "aborted". If R is "active", send $(\mathtt{abort}, \mathsf{sid}, S, R)$ as a delayed message to R (*i.e.*, $\mathcal{S}$ need not notify either party that the protocol was aborted, and may still cause R to output a key using a $\mathtt{setkey}$ message, but cannot cause S to output a key once an abort has occurred).
5. Upon receiving a message of the form $(\mathtt{incriminate}, \mathsf{sid}, S)$ from $\mathcal{S}$, if this is the first time receiving such a message and S is currently "aborted" and honest, then run the procedure $\mathsf{IncProc}(\mathsf{sid}, S, R, PK_S, PK_R, SK_S)$.

Fig. 3. The ideal functionality for Key Exchange with Incriminating Abort, parameterized by an incrimination procedure $\mathsf{IncProc}$ which runs only if the key exchange is aborted by the adversary.

do not know of a way to ensure such limited leakage and besides this gives up too much information: as we will see, our protocol only compromises deniability if the protocol aborts *and* one of S or R is corrupted at a later time. Instead, we parametrize $\mathcal{F}_{\mathsf{keia}}$ with an "incrimination procedure" $\mathsf{IncProc}$. In the case of an abort, $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ allows the adversary to interact with $\mathsf{IncProc}(SK_S, ...)$, which essentially represents the potentially non-simulatable flows of the protocol. If the protocol doesn't abort, then a fresh symmetric key is distributed to S and R and nothing is leaked to the adversary. The functionality $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ is described in Figure 3.

The incrimination procedure may at first be hard to interpret, and so we highlight some of the properties of protocols which realize $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$. First, if no abort occurs then the ideal protocol is perfectly forward secure, since the symmetric key is random and unconnected to other quantities in the protocol.

Hence, if a protocol π GUC-realizes $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ for any procedure $\mathsf{IncProc}$ then π is forward-secure.

Second, if an abort does occur and the adversary learns information, the information depends only on the sender's secret key and public information. Because secret keys are useless in the ideal model, *this has no impact on the security of other protocols.* In particular, the incrimination information cannot be used to fake authenticated messages in other conversations, or to convince the environment that S talked to anyone other than R. This last observation implies that not all incrimination oracles can be realized by real protocols: for example, if $\mathsf{IncProc}$ gives away the sender's secret key, then a real adversary would subsequently be able to fake arbitrary messages from the sender to other parties, contradicting the indistinguishability from the ideal model. We prove below that there exists a procedure $\mathsf{IncProc}$ for which $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ can, in fact, be realized.

Construction At the core of our constructions for realizing $\mathcal{F}_{\mathsf{keia}}$ is a chosen-ciphertext (CCA) secure variant of *Dual Receiver Encryption* (DRE) [15]. DRE allows anyone to encrypt a message to two parties with a single ciphertext, with the guarantee that attempts to decrypt a ciphertext by either of the two recipients will produce the *same result.* In our protocol, this means that certain actions can be simulated using either S or R's secret key. We formally define CCA-secure DRE in the full version, and describe a DRE scheme that is similar to the plaintext-aware encryption of [21]. Our protocol also uses a 2-round non-committing encryption (NCE) scheme [8].

Our 4-message protocol for realizing $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ is summarized in Figure 4. Intuitively, the incrimination procedure $\mathsf{IncProc}$ will expect the adversary to supply a ciphertext that matches the form of the second flow (ψ_1) in the protocol below, and will then use S's secret key to decrypt ψ_1 and compute a corresponding third flow (ψ_2). The incrimination procedure hands ψ_2 to the adversary, along with the random coins used by a non-committing encryption scheme.

Notably, although we only realize $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ against a semi-adaptive adversary, *the same protocol* is also a *statically secure* realization of $\mathcal{F}_{ke}$. Therefore, we have achieved a strictly stronger notion of security than that achieved by the one-message protocol using NI-AKE and MACs, or HMQV. Honest parties are always guaranteed complete deniability when the protocol succeeds, and even if the protocol aborts, deniability is maintained until some future corruption of a party occurs. It is an open question whether this notion of deniability can be further improved upon.

Theorem 4. *Assuming the existence of dual-receiver and non-committing encryption, the protocol Φ_{dre} in Figure 4*

1. *realizes $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}_{dre}}$ in the $\mathcal{F}_{krk}^{\Phi_{dre}}$-hybrid model with semi-adaptive security, for a suitable procedure $\mathsf{IncProc}_{dre}$ (defined in the full version); and*
2. *realizes $\mathcal{F}_{ke}$ in the $\mathcal{F}_{krk}^{\Phi_{dre}}$-hybrid model with static security.*

Moreover, the output of $\mathsf{IncProc}_{dre}$ can be simulated using the secret key of R instead of S.

As mentioned above, realizing $\mathcal{F}_{\mathsf{keia}}^{\mathsf{IncProc}}$ is meaningful for any procedure IncProc. However, the particular incrimination procedure of IncProc has additional properties. First, it can be faked without knowing either secret key, and the fake distribution is indistinguishable from the real one to a distinguisher who knows neither key. Second, it can be simulated exactly using either of the secret keys. These properties together mean that Φ_{dre} is statically secure (as stated in the theorem) and that even in the case of an abort with a subsequent corruption, it is only possible to incriminate one of the pair $\{S, R\}$, not S specifically.

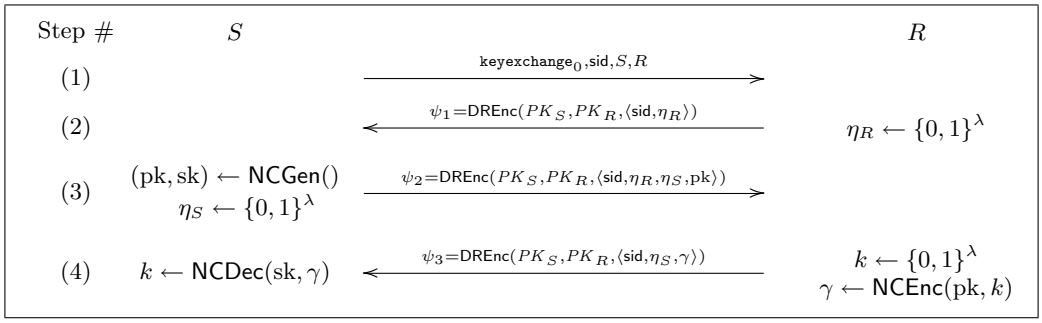

Fig. 4. A graphical illustration of Protocol Φ_{dre} for realizing $\mathcal{F}_{\mathsf{keia}}$. S and R check consistency of each flow immediately upon receiving it; if the flow is not consistent, the protocol is aborted. Notation: (Gen, DREnc, DRDec) is a Dual Receiver Encryption scheme and (NCGen, NCEnc, NCDec, NCSim, NCEqv) is a Non-Committing Encryption scheme (see full version).

References

1. B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195. IEEE Computer Society, 2004.
2. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography Conference*, pages 60–79, 2006.
3. N. Borisov, I. Goldberg, and E.A. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES*, pages 77–84. ACM, 2004.
4. C. Boyd, W. Mao, and K.G. Paterson. Deniable authenticated key establishment for internet protocols. In *Security Protocols Workshop*, volume 3364 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.
5. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
6. R. Canetti. Universally composable signatures, certification, and authentication. In *Computer Security Foundations Workshop*, 2004.
7. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

8. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC 1996*.
9. R. Canetti and M. Fischlin. Universally composable commitments. In *Advances in Cryptology — Crypto 2001*.
10. R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. *SIAM J. Comput.*, 32(1):1–47, 2002.
11. R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.
12. R. Canetti and T. Rabin. Universal composition with joint state. In *Advances in Cryptology — Crypto 2003*.
13. M. Di Raimondo, R. Gennaro, and H. Krawczyk. Secure off-the-record messaging. In *WPES*, pages 81–89. ACM, 2005.
14. M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In A. Juels, R. Wright, and S. De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 400–409. ACM, 2006.
15. Theodore Diament, Homin K. Lee, Angelos D. Keromytis, and Moti Yung. The dual receiver cryptosystem and its applications. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 330–343. ACM, 2004.
16. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. Preliminary version in STOC 1991.
17. C. Dwork and M. Naor. Zaps and their applications. *SIAM J. Computing*, 36(6):1513–1543, 2007. Preliminary version in FOCS 2000.
18. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004. Preliminary version in STOC 1998.
19. C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In *Advances in Cryptology — Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 442–457. Springer, 1998. Full version available from the second author's webpage.
20. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
21. J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In *Advances in Cryptology — Crypto 2003*.
22. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154, 1996.
23. S. Jiang. Deniable authentication on the internet. Cryptology ePrint Archive, Report 2007/082, 2007. `http://eprint.iacr.org/`.
24. J. Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2003.
25. M.-H. Lim, S. Lee, Y. Park, and S. Moon. Secure deniable authenticated key establishment for internet protocols. Cryptology ePrint Archive, Report 2007/163, 2007. `http://eprint.iacr.org/`.
26. J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2002.

27. R. Pass. On deniability in the common reference string and random oracle model. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337. Springer, 2003.
28. M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM Conference on Computer and Communications Security*, pages 112–121. ACM, 2005.
29. R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *Advances in Cryptology — Eurocrypt '99*.
30. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology — Asiacrypt 2001*, pages 552–565.
31. W. Susilo and Y. Mu. Non-interactive deniable ring authentication. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 386–401. Springer, 2003.
32. A. C.-C. Yao, F. Yao, Y. Zhao, and B. Zhu. Deniable internet key-exchange. Cryptology ePrint Archive, Report 2007/191, 2007. `http://eprint.iacr.org/`.