Notions of Deniable Message Authentication

Marc Fischlin
Cryptoplexity
Technische Universität Darmstadt, Germany
marc.fischlin@cryptoplexity.de

Sogol Mazaheri
Cryptoplexity
Technische Universität Darmstadt, Germany
sogol.mazaheri@cased.de

ABSTRACT

Deniable message authentication has drawn significant attention since it was first formalized by Dwork, Naor, and Sahai (STOC 1998). Since then, multiple notions of deniability have been introduced that vary in the considered adversary model and the required level of deniability. Most of the previous works concentrate on fairly strong notions of deniability, allowing the prover to even dispute that an interaction took place. In practice, however, weaker forms of deniability may suffice, such as being able to deny that a certain message has been transmitted at a certain point in time. Our work here thus introduces alternative notions of deniable message authentication, including for example content deniability (where one can deny the actual message) and context deniability (where one can claim that the allegedly transmitted message is taken out of context). We then analyze existing approaches, carving out the deniability properties these protocols achieve. In particular, we investigate the off-the-record messaging protocol (OTR) of Borisov, Goldberg, and Brewer (WPES 2004), which lists deniability of authentication as one of its explicit goals, but escapes the strong notions of deniability in the literature.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*Privacy*; K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection—*Authentication*

Keywords

deniability, authentication, zero-knowledge, privacy, OTR

1. INTRODUCTION

Message authentication enables us to verify the origin and the integrity of messages that we receive and to assure our communication partners of the authenticity of messages that we send. Over the years, cryptography has offered many solutions for message authentication. Many of these, including classical digital signatures, combine message authentication with the non-repudiation property, which means that authenticators are not able to subsequently deny having authenticated messages. While non-repudiation is essential in certain applications such as commerce transactions and conducting contracts, in various other scenarios it is not only unnecessary, but also undesired. In order to prevent malicious verifiers, or anyone who witnesses authenticated interactions, from provably disclosing them to others, it is essential that the whole authentication process or at least certain aspects, such as the authenticated message, its context, or the authentication time can later be denied.

WPES'15 October 12 2015, Denver, CO, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3820-2/15/10 ...$15.00
DOI: http://dx.doi.org/10.1145/2808138.2808143

1.1 Deniable Message Authentication

Deniable message authentication [10] is a significant step towards solving the conflict between authenticity and privacy. The formalization of this notion is based on the common simulation paradigm, saying that an ideal-model simulator could have produced the communication transcript between the actual sender (often called prover) and the recipient (often called verifier). The simulator's output should be indistinguishable from a real transcript to any distinguisher, often called judge in this context. The idea is that the existence of such a simulator allows the prover to claim that the transcript could have been produced by someone else, even hiding the fact that an interaction took place. As such, the notion for instance prohibits any usage of common, non-repudiable digital signature schemes in such protocols.

While the above notion of deniability provides strong security guarantees, it may sometimes be overly restrictive, labeling otherwise sound approaches to be insecure. As a concrete example consider the off-the-record messaging protocol (OTR) [3], in which users initially sign ephemeral Diffie-Hellman public keys, such that this protocol certainly cannot provide full deniability. Yet, since the signed keys are self-chosen random values, a malicious party does not seem to be able to extract more proof than the fact that the user has been involved in some interaction. In particular, the adversary cannot slip any "semantic" into the signed value such as a newspaper headline for fixing some date. Similarly, the recent technological standard of the International Civil Aviation Organization for machine-readable travel documents [17] touches the question of avoiding so-called challenge semantics, but only those.

Whereas many previous works on deniable message authentication (see Section 1.3) aim at introducing strong notions, or on constructions achieving such notions, one can easily identify several alternative intuitive notions for deniable message authentication, which all provide some mean of dispute for the prover:

- *Content deniability*: Allowing the prover to deny having sent a certain message, but not necessarily hiding the fact that an interaction occurred.
- *Context deniability*: The ability to deny that a message has been transmitted in a certain context.
- *Time deniability*: The ability to dispute that a message has been authenticated at a certain point in time.
- *Source deniability*: The possibility of the prover to blame another party as the origin.
- *Destination deniability*: Allowing the prover to claim that a message has been sent to another recipient.

We note that in the scenario of file or disk encryption one already encounters different forms of deniability, usually all subsumed under the term plausible deniability. To some extent these notions resemble the ideas for deniable authentication above. Namely, deniable encryption can for example either mean the inability to distinguish encrypted data from random noise without the key, or it can refer to the ability to present some key enabling decryption to an arbitrary message. The former roughly corresponds to full deniability, whereas the latter can be interpreted as some kind of content deniability. Since our focus is message authentication we do not further investigate notions for encryption.

1.2 Contributions

Our starting point for defining deniable message authentication are the above five intuitive notions, serving as alternatives to full deniability. We give precise definitions of all notions via the simulation-based approach, where a simulator should be able to produce a transcript which is indistinguishable from a real interaction to a judge. It turns out that the differences in the notions above can be captured by the support the simulator gets, for instance, being allowed to interact with different provers and verifiers on certain messages.

In our definitions we consider so-called *offline* judges which have to decide for a transcript after completion. In principle, we could also consider *online* judges [7], who cannot be rewound and can actively interfere with the running executions, e.g., providing data which the malicious verifier should use in the execution. But since achieving security against such online judges is significantly harder, and our goal is to provide relaxed notions of deniability covering broader classes of protocols, we refrain from doing so.

Nonetheless, we take into account different attack possibilities for the verifier. We allow both malicious and honest verifiers, where in the first case the adversary controls the verifier from the beginning (and knows its secret keys), whereas in the latter case the adversary is merely an outsider, trying to assemble some convincing proof for the judge from the observed communication. Outsider adversaries are, however, easy to thwart by simply encrypting the network communication. We therefore consider an intermediate form, called a spy or trojan, where the adversary gets to see the unencrypted network communication. The scenario we have in mind here is that the message authentication scheme may be immune, e.g., runs on a trusted platform module (TPM), but the network encryption is vulnerable to virus or trojan attacks on the verifier's computer.[1]

We then discuss the relationship between our notions of deniability and how they can be combined. Afterwards, we exemplify our notions by analyzing several protocols including classical digital signatures, chameleon signatures, ring signatures, and designated verifier signatures. For each scheme we specify which of our relaxed notions (or full deniability) it achieves. We then revisit the OTR protocol [3] with respect to our definitions, showing that it is directly context, content, and time deniable, as well as destination deniable against malicious verifiers.

1.3 Related Work

Although first mentioned in 1991 [8], deniable message authentication was not formalized until 1998 by Dwork, Naor, and Sahai [10]. Since then, it has been extensively studied. Most previous works concentrate on a fairly strong notion of deniability, which allows provers to even dispute that an interaction took place. We refer to this notion as *full deniability*, as also suggested in several related works [4,14,20,21]. It is also often simply referred to as deniability [9,10,16,25].

Some previous works have already identified that a weaker level of deniability would be required to evaluate protocols. An example is the work of Raimondo, Gennaro, and Krawczyk [20] about the OTR protocol, where they discuss a weaker but intuitively sufficient notion of deniability, which they later formally introduced as *partial deniability* in the context of key exchange for analyzing SIGMA protocols [21]. Other relaxed notions are *peer* and *peer-and-time deniability* for key exchange, defined by Cremers and Feltz [5] to analyze their one-round protocol. To allow for an independent evaluation of deniability levels, instead of notions such as peer-and-time deniability, we define, among others, separate notions for time deniability (inspired by peer-and-*time* deniability), source deniability, and destination deniability (both inspired by partial deniability and peer deniability).

Regarding the adversary model, in addition to malicious verifiers, considered in most previous work [10, 14, 15, 21, 27], and outsiders [3, 6, 7, 19], we also consider spies as an intermediate form. We outline the capabilities of all three adversary types in order to simplify the deniability analysis of concrete message authentication protocols.

The issues caused when concurrently executing protocols involving zero-knowledge proofs were first discussed in [10] and motivated the investigation of deniable message authentication. Similar to some other work [7, 10, 21], we also consider deniability in a concurrent setting, where provers are willing to concurrently authenticate polynomially many messages for verifiers.

Deniable message authentication is mostly deployed to protect provers, since in most applications, the risk of being disclosed and incriminated is more critical for provers than for verifiers. There is, nonetheless, some work on protecting verifiers against malicious provers, such as *forward deniability* [19], and a general notion for protecting provers as well as verifiers in [7].

[1]We note that formalizing this idea is delicate, as it would require to precisely rule out that the authentication scheme encrypts itself; we thus formally still allow this form of internal encryption and leave it to the common sense to identify such solutions as trivially secure against spies.

We revisit the OTR protocol introduced in 2004 by Borisov, Goldberg, and Brewer [3] with respect to our relaxed deniability notions. A general security analysis including some improvement suggestion can be found in [20]. The current improved version of OTR, OTRv3, was introduced in 2007 [1].

2. SECURE INTERACTIVE MESSAGE AUTHENTICATION

In this section we discuss the notion of interactive message authentication protocols and their unforgeability notions.

2.1 Interactive Message Authentication

Our definition of interactive message authentication protocols is similar to the one introduced by Raimondo *et al.* [21], but modified to cover more general protocols where both the prover and the verifier can have key pairs. Prover and verifier instances can have an internal state used for authenticating multiple messages. For instance, in a protocol consisting of a key exchange and a MAC computation phase the key exchange does not have to be repeated every time a new message is being authenticated. This is also important for our notion of context deniability, where the authentication order and completeness of messages can change the context. Below we do not mention the states explicitly, but assume that each party stores its state internally.

Definition 2.1 (Interactive Message Authentication Protocol)**.** *An interactive message authentication protocol* Π *consists of a triple* $(\mathsf{KGen}, \mathsf{P}, \mathsf{V})$*, where:*

- KGen *is a probabilistic polynomial time (*PPT*) key generation algorithm. On input* $(1^n, \mathrm{P})$ *it outputs a key pair* $(\mathsf{sk}_\mathrm{P}, \mathsf{pk}_\mathrm{P})$ *for the prover, and on input* $(1^n, \mathrm{V})$ *it outputs a key pair* $(\mathsf{sk}_\mathrm{V}, \mathsf{pk}_\mathrm{V})$ *for the verifier.*
- P *is an interactive, stateful* PPT *Turing machine, called the prover algorithm, which runs on inputs* sk_P*,* pk_V*, and a message* m*.*
- V *is an interactive, stateful* PPT *Turing machine, called the verifier algorithm, running on inputs* sk_V *and* pk_P*.*
- *The output of the interaction between* P *and* V *on a message* m *is denoted by* $\langle \mathsf{P}(\mathsf{sk}_\mathrm{P}, \mathsf{pk}_\mathrm{V}, m), \mathsf{V}(\mathsf{sk}_\mathrm{V}, \mathsf{pk}_\mathrm{P})\rangle$*. This output is returned by* V *and is either the message* m*, indicating a successful verification of authenticity, or it is* $\perp$*, indicating a failed verification. Both parties may update their state after or during the interaction.*

Completeness.

Intuitively, completeness requires that honest provers can always successfully authenticate messages for verifiers. For this we call the pair consisting of the prover's state and the verifier's state *genuine* if it has been derived through a sequence of genuine executions of the two parties on genuine data, including the initial states, genuine keys, and admissible messages. Then for all messages m and every genuine state pair, we have:

$$\Pr[\langle \mathsf{P}(\mathsf{sk}_\mathrm{P}, \mathsf{pk}_\mathrm{V}, m), \mathsf{V}(\mathsf{sk}_\mathrm{V}, \mathsf{pk}_\mathrm{P})\rangle = m] = 1,$$

where the probability is over the random coin tosses of P, V, and KGen.

2.2 Unforgeability

Existential unforgeability under chosen-message attacks may come in various flavors. The main idea in all cases is that the adversary, called forger F here, simultaneously interacts with multiple instances of a prover and a verifier, and tries to make some verifier instance output a message m such that this message is "fresh". We refer to the two main branches as *global unforgeability*, in which case an adversary succeeds already if the verifier with public key pk_V outputs m such that no prover instance has been queried about the pair $(\mathsf{pk}_\mathrm{V}, m)$, and *local unforgeability*, where the adversary succeeds only if m has not been queried to the prover for any $\mathsf{pk}_{\mathrm{V}'}$. That is, in local unforgeability the adversary is for example disallowed to redirect a prover's message sent to some other verifier V' now to V in order to win. Jumping ahead we note that one can usually turn locally unforgeable schemes into globally unforgeable ones by adding the verifier's identity like its public key to the message. This, however, may conflict with destination deniability.

Additionally, in the interactive setting one may only charge "queries" to the prover oracle if the execution is completed, or even if it has started. We opt for the latter for the sake of simplicity. As for notation, we write $\mathsf{F}^{\mathsf{P}(\mathsf{sk}_\mathrm{P},\cdot,\cdot),\mathsf{V}(\mathsf{sk}_\mathrm{V},\mathsf{pk}_\mathrm{P})}$ to denote that F can spawn multiple instances of the interactive machines of the prover and the verifier, where each instance uses fresh randomness and keeps its current state.

Definition 2.2 (Unforgeability)**.** *An interactive message authentication scheme* $\Pi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ *is called (globally or locally) unforgeable if for every interactive* PPT *Turing machine* F*, called forger, the probability that some verifier instance outputs* $m \neq \perp$ *in the run* $\mathsf{F}^{\mathsf{P}(\mathsf{sk}_\mathrm{P},\cdot,\cdot),\mathsf{V}(\mathsf{sk}_\mathrm{V},\mathsf{pk}_\mathrm{P})}(\mathsf{pk}_\mathrm{P}, \mathsf{pk}_\mathrm{V})$*, such that no prover instance has started an interaction for* $(\mathsf{pk}_\mathrm{V}, m)$ *(in case of global unforgeability) resp.* m *(in case of local unforgeability) before, is negligible. Here the probability is taken over the random choices of the keys and the internal randomness of the algorithms resp. instances.*

3. NOTIONS OF DENIABILITY

In this section, we formally introduce the attack model and our notions of deniability. Afterwards, we discuss the relationship between these notions and the possibility of combining them. We note that for our definition of time deniability we need a notion of time in our model. For this we assume that there is a trusted party which maintains some (discrete) notion of time, initialized to 0 when the real or ideal experiment starts, and which clocks all the parties by scheduling computational steps to each party in a round-robin fashion and providing the parties with the current time.[2] We omit mentioning this part explicitly and instead refer to an abstract notion of time.

3.1 Attack Model

Real-world attacks.

We follow the classical simulation-based approach to define our notions of deniability. That is, we assume an adversary $\mathcal{A}$ which can interact with multiple instances of the prover algorithm P, all initialized with the same key sk_P,

[2]The offline judge does not receive the actual time each experiment ran, else it would most likely be trivial to distinguish the two worlds.

and multiple instances of the verifier algorithm V, all initialized with the same key sk_{V}. The adversary receives the corresponding public keys pk_{P} and pk_{V}, and also a message vector $\boldsymbol{m}$ for which it tries to create some evidence that the prover authenticated these messages to the verifier.

The adversary can start a polynomial number of new instances of the prover and the verifier algorithms with fresh randomness, and concurrently interact with these instances to authenticate a given message vector $\boldsymbol{m}$. In particular, the adversary may interact with a prover instance to authenticate messages $\boldsymbol{m}$ in this order, sequentially with respect to this instance, of course. We note that the adversary has full control over the order of transmissions, i.e., can decide to which instance it delivers the next message for which it then immediately obtains a reply.

We assume that the adversary can statically decide to corrupt the verifier, at the outset of the attack, in which case it also receives the secret key sk_{V} and from there on controls the party. We then speak of deniability *against malicious verifiers.* If the verifier remains honest, and the secret key remains hidden from $\mathcal{A}$, then we call this form deniability *against honest verifiers* or *outsider deniability.* As explained in the introduction, outsider deniability is usually easy to achieve via network encryption, such that we often rather refer to deniability against *spies* (or *trojans*) where the adversary gets to learn the unencrypted communication. As mentioned before, formalizing the notion of unencrypted communication is intricate, even though the intuition is that the adversary then holds the secret decryption key. For most of the schemes we discuss, deniability against spies and honest-verifier deniability coincide such that we do not define the notion formally here. We simply denote by ks the corresponding set of input keys $\mathcal{A}$ receives at the beginning, depending on the choice of corruption.

We finally note that we do not consider the corruption of the prover, as knowledge of the prover's secret key in all known solutions allows to trivially authenticate any message to any verifier. Even for time deniability, where the corruption time may matter, one cannot guarantee that an adaptive corruption cannot be used in authenticating a message for an earlier time (unless one uses a forward-secure solution with key updates). Hence, the interesting case appears to be the setting in which the prover's secret key is still intact.

Ideal-world attacks.

We compare the output behavior of our adversary, observing genuine protocol executions, with the one of a simulator $\mathcal{S}$ who gets the public keys of the parties (and the verifier's secret key in case of malicious-verifier deniability) and input messages $\boldsymbol{m}$, which was also given to the adversary. If the simulator's task is to provide an indistinguishable output to an efficient judge, $\mathcal{J}$, given only the keys and messages as input, then we are in the setting of full deniability.

To cover the other notions, we provide $\mathcal{S}$ with a set of "helper" prover and verifier oracles $\mathcal{H}_\delta$ which, depending on the aspired deniability notion δ, provides some limited form of help. For various choices of $\mathcal{H}_\delta$ we thus obtain the general notion of δ-deniability as follows:

Definition 3.1 (δ-deniability)**.** *An interactive message authentication protocol* $\Pi := (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ *is* δ-deniable *if for every adversary* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$*, such that for every* PPT *judge* $\mathcal{J}$ *and every message vector* $\boldsymbol{m}$ *there is a negligible function* negl *such that:*

$$\Big|\Pr\left[\mathcal{J}(1^n, ks, \boldsymbol{m}, \mathcal{S}^{\mathcal{H}_\delta}(1^n, ks, \boldsymbol{m})) = 1\right] - \Pr\left[\mathcal{J}(1^n, ks, \boldsymbol{m}, \mathcal{A}^{\mathsf{P}(\mathsf{sk}_{\mathrm{P}}, \mathsf{pk}_{\mathrm{V}}, \cdot), \mathsf{V}(\mathsf{sk}_{\mathrm{V}}, \mathsf{pk}_{\mathrm{P}})}(1^n, ks, \boldsymbol{m})) = 1\right]\Big| \le \mathsf{negl}(n),$$

where the probability is over the random coin tosses of KGen*,* P*,* V*,* $\mathcal{J}$*, and* $\mathcal{A}$ *resp.* $\mathcal{S}$*. The set of auxiliary oracles* $\mathcal{H}_\delta$ *is given in Definition 3.2.*

If we work in the random oracle model, then we assume that all parties, including the adversary, the simulator, and the judge, get access to a given random oracle. We emphasize that in the context of deniability it is important that the simulator does not get to program the random oracle [18].

3.2 Capturing Different Notions

The above definition can be applied to different sets of auxiliary oracles to derive concrete notions of deniability. In the definition below, oracles that are defined with a constraint intuitively do not respond whenever their constraint is not fulfilled. For instance if the queried messages must not be equal to the messages for which an evidence is being simulated (i.e., different from m_i for any m_i in $\boldsymbol{m}$), no prover oracle instance can be started on m_i. We give an intuitive description of our deniability notions followed by a formal definition of their set of auxiliary oracles $\mathcal{H}_\delta$.

- *Full deniability*: Evidences can be simulated using only the known keys.
- *Content deniability*: Evidences for a message vector $\boldsymbol{m}$ can be simulated using the known keys and with oracle access for interactions on messages m_i that are not contained in $\boldsymbol{m}$.
- *Context deniability*: Evidences of interactions between P and V on a message vector $\boldsymbol{m}$ can be simulated using the known keys, and with the help of interactions with P and V on different messages, more precisely, by querying the prover oracle on a sequence $\mathcal{M}_\mathcal{S}$ of messages that differ from $\boldsymbol{m}$ in at least one message entry, as well as with interactions with P for different verifiers, described through different keys $\mathsf{pk}_{\mathrm{V'}} \neq \mathsf{pk}_{\mathrm{V}}$. This may, or may not, implicitly assume an underlying PKI where such keys $\mathsf{pk}_{\mathrm{V'}}$ need to be registered.

  Note that $\mathcal{M}_\mathcal{S}$ is already considered to be different from $\boldsymbol{m}$ if it contains an additional entry, as this final entry could invalidate all the previous messages in the sequence and the prover could claim of having sent this final message. It is convenient to define the sequence containing those messages that are queried to a prover instance for the verifier public key pk_{V} by $\mathcal{M}_\mathcal{S}^{\mathsf{pk}_{\mathrm{V}}}$, and the set of such queries over all instances by $\{\mathcal{M}_\mathcal{S}^{\mathsf{pk}_{\mathrm{V}}}\}$.
- *Time deniability in* $[\tau, \tau']$: Let τ and τ' be two points in time, and $[\tau, \tau']$ be a time window spanning from τ to τ'. Evidences can be simulated using the known keys and interactions with P and V, where P has only been active at points in time different than $[\tau, \tau']$. We usually say a scheme is time deniable if it is time deniable for arbitrary time intervals.[3]

[3]Note, however, that the simulator may still depend on the specific interval; this is necessary to allow the simulator to call the prover at all, outside of the interval.

The notion of time has some interesting side effect, due to the entanglement with other events outside of our model. Assume for example that in the protocol the prover signs some data with a regular signature scheme. If the signed data is, say, a self-chosen random value (as in case of OTR), then this signature intuitively cannot violate time deniability, as it could have been signed at any point in time. Now consider the case that the data is partially chosen by the adversary and contains, for instance, today's newspaper headline. Then the signature may easily serve as a proof that the execution took place today (or later). Since talking about semantics of arbitrary data appearing in an execution is hard, we do not consider such differences here in our model. We note, however, that we make a more fine-grained distinction when discussing time deniability for some concrete schemes.

- *Source deniability*: Evidence of an interaction between P and V can be simulated using the known keys and V's interactions with other provers, captured by assuming different prover keys $\mathsf{pk}_{\mathrm{P}'}$. In a more liberal notion one could allow to query $\mathrm{V}(\mathsf{sk}_{\mathrm{V}}, \cdot)$ also about the actual prover's key pk_{P}, but since the simulator misses the P's secret key, unforgeability indicates that such queries should not facilitate the simulator's task.
- *Destination deniability*: Evidence of an interaction between P and V can be simulated using the known keys and interactions with P instances for different verifiers.

In the following definition we write $\mathrm{P}(\cdots, \cdot[C])$ to denote the prover oracle that can take arbitrary arguments, but will only respond if condition C is met by the argument in question. Similarly, we write $\mathrm{P}^{C}(\cdots)$ to denote the prover's oracle which only interacts if some global condition C (usually about the time) is true. We also write $\mathrm{P}(\cdots)|C \Rightarrow \bot$ to denote the fact that the simulator eventually outputs $\bot$ if condition C in one of its query sequences is met. All notions straightforwardly apply to the verifier oracle as well.

Definition 3.2 (Oracles $\mathcal{H}_\delta$ for δ-deniability)**.** *The set of auxiliary oracles $\mathcal{H}_\delta$ for δ-deniability of a message authentication protocol $\Pi := (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ when simulating evidence for a communication between a prover* P *and a verifier* V *on messages $\boldsymbol{m} := (m_1, \ldots, m_n)$ is defined as follows.*

δ-deniability	*auxiliary oracles $\mathcal{H}_\delta$*
full deniability	$\mathcal{H}_\delta := \emptyset$
content deniability	$\mathcal{H}_\delta := \big(\mathsf{P}(\mathsf{sk}_{\mathrm{P}}, \mathsf{pk}_{\mathrm{V}}, \cdot[\notin \{m_1, \ldots, m_n\}]), \mathsf{V}(\mathsf{sk}_{\mathrm{V}}, \cdot)\big)$
context deniability	$\mathcal{H}_\delta := \big(\mathsf{P}(\mathsf{sk}_{\mathrm{P}}, \mathsf{pk}_{\mathrm{V}}, \cdot)\vert\boldsymbol{m} \in \{\mathcal{M}_{\mathcal{S}}^{\mathsf{pk_V}}\} \Rightarrow \bot, \mathsf{P}(\mathsf{sk}_{\mathrm{P}}, \cdot[\neq \mathsf{pk}_{\mathrm{V}}], \cdot), \mathsf{V}(\mathsf{sk}_{\mathrm{V}}, \cdot)\big)$
time deniability in $[\tau, \tau']$	$\mathcal{H}_\delta := \big(\mathsf{P}^{time \notin [\tau, \tau']}(\mathsf{sk}_{\mathrm{P}}, \mathsf{pk}_{\mathrm{V}}, \cdot), \mathsf{V}(\mathsf{sk}_{\mathrm{V}}, \cdot)\big)$
source deniability	$\mathcal{H}_\delta := \mathsf{V}(\mathsf{sk}_{\mathrm{V}}, \cdot[\neq \mathsf{pk}_{\mathrm{P}}])$
destination deniability	$\mathcal{H}_\delta := \mathsf{P}(\mathsf{sk}_{\mathrm{P}}, \cdot[\neq \mathsf{pk}_{\mathrm{V}}], \cdot)$

3.3 Primary and Combined Notions

Different notions are, due to different auxiliary oracles, often incomparable. In the following we give a few similarities and relationships between our primary notions and discuss how these notions can be combined.

Roughly speaking, if a simulator can simulate indistinguishable evidence with less auxiliary oracles, it can still do so when having more. Consequently, full deniability implies all other notions. Furthermore, context deniability is by definition implied by destination deniability. This reflects our understanding that the ability to deny the intended verifier also means to be able to deny some context. Additionally, content deniability implies context deniability, since content deniability allows adversaries to use a communication on one message to simulate evidence for another message, which is also understood as a type of context deniability.

If a simulator needs more auxiliary oracles than available in any of the primary notions alone, we have the possibility of combining notions. When combining two deniability definitions with auxiliary oracles $\mathcal{H}_\delta$ and $\mathcal{H}_{\delta'}$, respectively, the combined notion claims the existence of a successful simulator using $\mathcal{H}_\delta \cup \mathcal{H}_{\delta'}$ as its set of auxiliary oracles. The combined notion is at most as strong as —and usually weaker than— the original notions, since it usually requires more support during simulation. We can combine for instance context with content deniability, or context with source deniability, especially if the content or the source can only be denied if some other context is provided to the simulator. A *context-and-source deniable* protocol is not necessarily source deniable, as we will see in Section 5 in our analysis of the OTR protocol.

4. DENIABLE AUTHENTICATION PROTOCOLS

In this section, we informally discuss the level of deniability in several common message authentication protocols, both against malicious verifiers and spies. A formalization is easy to derive from our discussions. A summary of the analysis results is given in a table following each protocol, where a checkmark ✓ indicates that the deniability notion holds. For time deniability we sometimes use (✓) to indicate that, depending on the authenticated message and the chosen time window, time deniability may not be fulfilled. To illustrate this point, consider the case that the prover signs a self-chosen random message, as in the OTR protocol, versus the case that the prover signs an adversarially chosen message which may contain for example today's newspaper headline.

In the protocol descriptions below, the keys used by each party, either in a real protocol execution or for simulation, are shown as inputs after the party's role in parentheses. We usually do not touch the issue of unforgeability explicitly, but remark that all candidates achieve local unforgeability.

4.1 Classical Digital Signatures

Classical digital signatures, such as DSA and RSA-based signatures, are the most common way of authenticating messages over the Internet. Because of their non-repudiation property, digital signatures are not source deniable, and hence whenever full deniability is a goal, we are advised against using digital signatures. Nevertheless, they do have some deniable aspects. Below, we see an abstract classical digital signature scheme, where a prover signs a message using a secret signing key Sk_{P}, and the signature can be verified with the matching public verification key Vk_{P}.

Prover $(\mathsf{Sk}_{\mathrm{P}})$ | **Verifier** $(\mathsf{Vk}_{\mathrm{P}})$

$\sigma := \mathsf{Sig}(\mathsf{Sk}_{\mathrm{P}}, m)$

$$\xrightarrow{m, \sigma}$$

$\mathsf{Vf}(\mathsf{Vk}_{\mathrm{P}}, \sigma, m)$

digital signatures	full	ctnt	ctxt	time	src	dest
malicious verifier			✓	(✓)		✓
spy			✓	(✓)		✓

Table 1: Deniability of classical digital signatures

Context deniability follows as the simulator can, with the help of its auxiliary oracle $\mathcal{H}$, sign a message m for a different verifier V$'$ and claim that it has been signed for V. Formally, this allows the simulator to run the real-world adversary $\mathcal{A}$ and simulate $\mathcal{A}$'s oracles through $\mathcal{H}$, such that the output of the simulator is perfectly indistinguishable. Destination deniability holds similarly because P's signatures sent to different verifiers are perfectly indistinguishable. That is, $\mathcal{A}$'s prover oracle is simulatable through the (restricted) prover oracle in $\mathcal{H}$, and the verifier's oracle is trivially simulatable since the verifier only relies on public keys. Furthermore, as long as the signed message m is not time-specific, signatures created at different times are indistinguishable from one another. For arbitrary messages, time deniability depends on the desired time window. For sake of illustration we also remark that it would be straightforward to violate time deniability formally, by simply having the prover prepend the current time to the message before signing it.

Note that, if we let P include the verifier's identity V or public key pk_{V} in the signed message, then we gain the stronger notion of global unforgeability, but lose destination deniability as well as context deniability. The latter follows as the simulator cannot produce a signature for the intended verifier V, using signatures for different verifiers, anymore.

4.2 Chameleon Signatures

Chameleon signatures were introduced by Krawczyk and Rabin as a combination of a chameleon hash function and a standard signature scheme [13]. The chameleon hash function $\mathsf{ChamHash}(\mathsf{Hk}_{\mathrm{V}}, m, r)$ computes a hash value of a message m using a random value r and a special public hashing key Hk_{V}. It is collision-resistant unless a secret trapdoor key Tk_{V} is known, which enables its holder (the verifier) to efficiently find collisions, i.e., given m and r, for any message m', a value r' can be found such that the hash value for m' and r' is the same as for m and r. Furthermore, for random r the derived value r' has the same distribution.

Prover $(\mathsf{Sk}_{\mathrm{P}}, \mathsf{Hk}_{\mathrm{V}})$ | **Verifier** $(\mathsf{Tk}_{\mathrm{V}}, \mathsf{Hk}_{\mathrm{V}}, \mathsf{Vk}_{\mathrm{P}})$

$r \leftarrow_{\$} \{0,1\}^*$
$h := \mathsf{ChamHash}(\mathsf{Hk}_{\mathrm{V}}, m, r)$
$\sigma := \mathsf{Sig}(\mathsf{Sk}_{\mathrm{P}}, h)$

$$\xrightarrow{m, r, \sigma}$$

$h := \mathsf{ChamHash}(\mathsf{Hk}_{\mathrm{V}}, m, r)$
$\mathsf{Vf}(\mathsf{Vk}_{\mathrm{P}}, \sigma, h)$

chameleon signature	full	ctnt	ctxt	time	src	dest
malicious verifier		✓	✓	✓		
spy				(✓)		

Table 2: Deniability for chameleon signatures

Because of the provers' signatures, this protocol is clearly not source deniable and hence, also not fully deniable. The most interesting result about chameleon signatures is their content deniability against malicious verifiers, which follows from the fact that the simulator, receiving the verifier's secret key Tk_{V}, can ask its auxiliary oracle for a signature for a message m', different from m, and then find collisions for each hash value with the help of Tk_{V} to create chameleon signatures with the same distribution. This implies also context deniability and time deniability against malicious verifiers, because one can transform signatures on different messages or from other points in time.

Destination deniability against malicious verifiers should not hold, but to some extent this depends on the chameleon hash scheme. If the hash scheme is such that one can transform given hash values for m, r under some verifier's hash key to a hash value for m', r', and a different verifier, with help of its trapdoor key, then one can claim destination deniability. This, however, seems unlikely given the current chameleon hash schemes.

In contrast to malicious verifiers, spies do not hold the trapdoor key for the chameleon hash function. Consequently, chameleon signatures are not content deniable against spies, because in order to simulate signatures, a simulator would need to either forge a signature for a new hash value, or find collisions for the chameleon hash without knowledge of the trapdoor key. Depending on the authenticated message and the time window, chameleon signatures can be time deniable against spies. Since hash values are computed using a verifier's public hashing key, and then signed by the prover, they cannot be obtained from different verifiers with different hash keys. Therefore, chameleon signatures are not destination deniable against spies. Furthermore, since chameleon signatures cannot be simulated by spies using other context, i.e., communications with different verifiers, or on different messages, they are not context deniable.

4.3 Ring Signatures

Ring signatures were formally introduced in [22] by Rivest, Shamir, and Tauman as a way of leaking secrets, such that the authenticity of a message is *only* verifiable as being signed by someone among a set of signers. In other words, in a set of signers, signatures created by one signer are indistinguishable from signatures of any other signer. This property is referred to as *anonymity* of ring signatures. Ring signatures can be used for constructing deniable authentication protocols [2, 16], especially two-party ring signatures with only one prover and one verifier as the possible signers.

The level of anonymity usually has an effect upon the level of deniability provided by ring signature schemes. Bender, Katz, and Morselli discuss three different levels of anonymity in ring signatures [2]. For our analysis here, we are interested in their definition of *anonymity against full key exposure* (or *against attribution attacks*). Intuitively, it requires that adversaries cannot identify the real signer, even if they have access to a signing oracle, and some of the users have maliciously generated keys, and others reveal their secret keys to the adversary, as long as at least two users have honestly generated keys. In our scenario, both parties generate their keys honestly, while malicious verifiers reveal their (still honestly generated) keys to the judge (the adversary in the anonymity attack).

Ring signature schemes are setup-free, i.e., provers do not need the knowledge or consent of the other potential signers to authenticate a message. The only things that a prover needs for creating a deniable ring signature is the own signing key (Sk_P), the matching verification key (Vk_P), and the verifier's verification key (Vk_V).

Prover $(\mathsf{Sk}_\mathrm{P}, \mathsf{Vk}_\mathrm{P}, \mathsf{Vk}_\mathrm{V})$ **Verifier** $(\mathsf{Sk}_\mathrm{V}, \mathsf{Vk}_\mathrm{V}, \mathsf{Vk}_\mathrm{P})$

$\sigma := \mathsf{RingSig}(\mathsf{Sk}_\mathrm{P}, \mathsf{Vk}_\mathrm{P}, \mathsf{Vk}_\mathrm{V}, m)$

$\xrightarrow{m, \sigma}$

$\mathsf{RingVf}(\mathsf{Vk}_\mathrm{P}, \mathsf{Vk}_\mathrm{V}, \sigma, m)$

ring signatures	full	ctnt	ctxt	time	src	dest
malicious verifier	✓	✓	✓	✓	✓	✓
spy				(✓)		

Table 3: Deniability in two-party ring signatures

Ring signatures with anonymity against full key exposure are fully deniable against malicious verifiers, since not even the knowledge of V's secret key can help with identifying the real signer. As a result, a judge who is presented with some evidence including a signature and even V's secret key can only guess the real signer's identity with a probability that is negligibly close to $\frac{1}{2}$.

However, against spies, ring signatures are not content deniable, since P's signature can only be forged by V and not by a spy. Depending on the desired time window of deniability and the messages, they can be time deniable against spies. Ring signatures are not source deniable, because they cannot be simulated by a spy using signatures of other provers for the same verifier. Similarly, destination deniability does not hold, since signatures cannot be simulated by a spy using the prover's signatures for other verifiers. Without having the possibility to transform signatures to signatures for other messages, or to create signatures for specific verifiers from signatures for other verifiers, context deniability against spies cannot be achieved either.

4.4 Designated Verifier Signatures

Verifier designation is a general term used for proofs that can be verified only by designated verifiers, specified by provers. Designated verifier proofs were introduced by Jakobsson, Sako, and Impagliazzo [12]. Other concrete protocols such as [23, 24, 26] were later introduced.

Here, we analyze the designated verifier signature scheme (DVS) from [26] by Yang an Liao. In DVS, verifiers need their secret keys to verify received signatures. Moreover, the same secret keys allow them to forge signatures. Below, x and y are Diffie-Hellman long term secret keys and X and Y are the corresponding public keys in a cyclic group $\mathbb{G}$. A long term symmetric secret key can be computed as $\mathsf{k} := Y^x$ or $\mathsf{k} := X^y$ by the prover or the verifier, respectively. Furthermore, a keyed hash function H is used for computing a MAC, where H is modeled as a random oracle.

Prover (x, Y) **Verifier** (y, X)

$t \leftarrow\!\!\$\ \mathbb{G}$

$s := \mathsf{H}(\mathsf{k}, m||t)$

$r := (m||t) \cdot \mathsf{k}^s$

$\xrightarrow{s, r}$

$m||t := r \cdot \mathsf{k}^{-s}$

verify $\mathsf{H}(\mathsf{k}, m||t) = s$

DVS	full	ctnt	ctxt	time	src	dest
malicious verifier	✓	✓	✓	✓	✓	✓
spy	✓	✓	✓	✓	✓	✓

Table 4: Deniability in DVS from [26]

DVS is fully deniable against malicious verifiers, and under the CDH assumption also against spies. Assuming that the transcript of an alleged authentication for a message m between a prover P and a verifier V is presented to a judge, which consists of $s = \mathsf{H}(\mathsf{k}, m||t)$ and $r = (m||t) \cdot \mathsf{k}^s$ for a randomly chosen t, the symmetric key k can be used by V to generate a signature on any message m exactly the way the prover P would.

A spy can choose a key k' at random and use it for simulating evidence by computing $s = \mathsf{H}(\mathsf{k}', m||t)$ for a random t and $r = (m||t) \cdot \mathsf{k}'^s$. For verifying the transcript, a judge needs to compute the real key k and hash it with the random oracle H, which is infeasible according to CDH assumption. As a result, the evidence simulated by the spy is indistinguishable to any efficient judge from real evidence.

4.5 Key Exchange and MAC

The combination of an authenticated key exchange protocol and a MAC scheme is a classical construction to achieve deniability [3, 11]. Since both parties share a key after the key exchange, the verifier can create a MAC for any message, providing at least content deniability. We distinguish between key exchange protocols that are deniable for both parties, denoted by dKE, and other key exchange protocols, denoted by KE. Assuming a previously shared key k, the message authentication protocol looks simply like below.

Prover (k) **Verifier** (k)

$t := \mathsf{MAC}(\mathsf{k}, m)$ $\xrightarrow{m, t}$

verify $t = \mathsf{MAC}(\mathsf{k}, m)$

(d)KE&MAC	full	ctnt	ctxt	time	src	dest
verifier (dKE)	✓	✓	✓	✓	✓	✓
spy (dKE)	✓	✓	✓	✓	✓	✓
verifier (KE)		✓	✓	✓		
spy (KE)		✓	✓	✓		

Table 5: Deniability in KE and MACs

A key exchange protocol is called *key indistinguishable* if given a protocol transcript, the real key is computationally indistinguishable from a random value from the same distribution. Key indistinguishability is a very basic security property and hence we only consider here key exchange protocols that satisfy this property. The combination of a deniable key exchange protocol that is deniable for provers and verifiers against malicious verifiers and spies and a MAC scheme is fully deniable against malicious verifiers and spies. In particular, because of key indistinguishability, a spy can randomly sample a key and use it for computing MACs. However, for an arbitrary key exchange protocol, which is not necessarily deniable for provers or verifiers, the above protocol is not necessarily source and destination deniable.

5. DENIABILITY IN OFF-THE-RECORD PROTOCOL

In this section we analyze deniability of the off-the-record messaging protocol (OTR), which was introduced by Borisov, Goldberg, and Brewer [3] as a better suited protocol for casual conversations than for instance PGP and S/MIME. It enables message authentication with certain level of deniability, as well as end-to-end encryption and forward secrecy. OTR is deployed in various instant messaging clients, including Pidgin, Kopete, and Cryptocat. The current version, OTRv3 [1], uses a SIGMA protocol for key-exchange, which still contains signatures and is therefore not fully deniable. In the following we discuss that the simpler original version OTRv1, despite using signatures, still provides some reasonable deniability guarantees.

Below, we see an execution of OTRv1 between two users. For our analysis, however, we consider message authentication only in one direction, where a user P is always the prover and the other user V always the verifier. For confidentiality, messages are encrypted with a symmetric key, and for authentication, MACs are used. To provide forward secrecy, keys are regularly renewed. The current MAC key is always the hash value of the current encryption key using a one-way hash function H. An initial Diffie-Hellman key exchange is performed in a cyclic group $\mathbb{G}$ generated by a generator g. This key exchange is authenticated by sending signatures on the ephemeral DH values g^{x_1} and g^{y_1} along. Afterwards, P and V share a DH key k_{11} as the encryption key and its hash value $\mathsf{H}(\mathsf{k}_{11})$ as a MAC key. Before an old encryption key is securely erased, a new one is exchanged, where the ephemeral DH values are, contrary to the computation of the initial key, not signed, but MACed with the previous MAC key. In the simplified protocol, below, only the initial key exchange and the first two messages are shown. We neither include the revelation of the MAC key after transmissions. Furthermore, x_i and y_j are chosen uniformly at random by P and V, respectively, and the resulted shared encryption key is denoted by k_{ij}.

Prover $(\mathsf{Sk}_{\mathrm{P}}, \mathsf{Vk}_{\mathrm{V}})$ **Verifier** $(\mathsf{Sk}_{\mathrm{V}}, \mathsf{Vk}_{\mathrm{P}})$

$$\xrightarrow{g^{x_1},\ \mathsf{Sig}(\mathsf{Sk}_{\mathrm{P}}, g^{x_1})}$$

$$\xleftarrow{g^{y_1},\ \mathsf{Sig}(\mathsf{Sk}_{\mathrm{V}}, g^{y_1})}$$

$$\xrightarrow{g^{x_2},\ \mathsf{Enc}(\mathsf{k}_{11}, m_1),\ \mathsf{MAC}(\mathsf{H}(\mathsf{k}_{11}), g^{x_2}, \mathsf{Enc}(\mathsf{k}_{11}, m_1))}$$

$$\xleftarrow{g^{y_2},\ \mathsf{Enc}(\mathsf{k}_{21}, m_2),\ \mathsf{MAC}(\mathsf{H}(\mathsf{k}_{21}), g^{y_2}, \mathsf{Enc}(\mathsf{k}_{21}, m_2))}$$

OTRv1	full	ctnt	ctxt	time	src	dest
verifier		✓	✓	✓		✓
spy		✓	✓	✓		

Table 6: Deniability in OTRv1

Discussing unforgeability of OTRv1 is beyond the scope of our work here. For one, for a comprehensive treatment one would need to argue about scenarios in which both parties act as senders and receivers, potentially also taking into account considerations like causality. The other point is that the keys are used in a non-standard way (as also pointed out in [20]), because the encryption key k_{11} and its deployment infringe with the MAC key $\mathsf{H}(\mathsf{k}_{11})$. Providing such an unforgeability analysis would be very interesting.

Theorem 5.1. *OTRv1 is context, content, time, and destination deniable against malicious verifiers. Under the CDH assumption, context, content, and time deniability hold also against spies.*

We discuss informally, why the above theorem holds. Let tr be the transcript of an alleged interaction between P and V, where only P authenticates messages. It starts with an initial signed DH key exchange and may contain further key exchanges if more than one message was authenticated.

Regardless of the adversary, the authenticity of all messages m_i other than the first message m_1 can be *fully* denied. In other words, the adversary can simulate the part of tr after the initial key exchange by itself. The reason behind this is that a malicious verifier can easily use the initial shared key k_{11} to simulate the rest of tr. A spy, who does not know k_{11}, can pick a random value $\mathsf{k}' \leftarrow_{\$} \mathbb{G}$ and use it as the symmetric encryption key to simulate the rest of the transcript. Judges cannot compute the real key k_{11} from the g^{x_1} and g^{y_2} in the transcript to verify the computed MACs, otherwise they would violate the CDH assumption. Consequently, it is sufficient to only analyze deniability of the key exchange and the first authenticated message:

For a malicious verifier, V:

- context deniability: V can obtain P's signature on some g^{x_1} using old communications with P or from another user, who can communicate with P. The verifier V then chooses y_1, computes g^{y_1}, the key k_{11} and its hash $\mathsf{H}(\mathsf{k}_{11})$ which is then used to compute MACs on messages. For multiple messages, OTRv1 remains context deniable, since after the initial key exchange, V can simulate the rest of the transcript arbitrarily. This can be formalized accordingly by having the simulator emulate $\mathcal{A}$'s attack, assembling a prover oracle with its oracle $\mathrm{P}(\mathsf{sk}_{\mathrm{P}}, \cdot[\neq \mathsf{pk}_{\mathrm{V}}], \cdot)$ to generate a signed g^{x_1} value, then creating its own y_1 and signing it with the known signature key. The verifier oracle is already given such that $\mathcal{S}$ can generate the same output as $\mathcal{A}$.
- content deniability: Given a real communication with P authenticating some message, V can compute a secret key k_{11} which matches P's g^{x_1}. This allows V to compute a MAC for any other message using the key $\mathsf{H}(\mathsf{k}_{11})$ and simulate evidence that is indistinguishable from evidence of a real interaction. The formal simulation is as in case of context deniability.

- time deniability: At the beginning of a communication P signs a random value. This signature is indistinguishable from P's signatures computed at different times. Since V can compute MACs with the key $\mathsf{H}(\mathsf{k}_{11})$, the content of authenticated messages and, therefore, the time of the interaction is deniable.

- destination deniability: V can obtain P's signature from another user's interaction with P. For the simulation, V chooses some value y_1, computes g^{y_1}, the key k_{11} and its hash $\mathsf{H}(\mathsf{k}_{11})$. Then, V signs g^{y_1} and has everything needed to compute MACs on arbitrary messages and forge interactions that are indistinguishable to a judge. The formalization is again straightforward.

For a spy, S:

- context deniability: S can obtain P's signature on some g^{x_1}, as well as V's signature on some value g^{y_1} from P and V's communication on a different message. Then, S chooses some random value k'_{11} from $\mathbb{G}$ as a shared key and uses it to encrypt the first message and the hash value $\mathsf{H}(\mathsf{k}'_{11})$ to compute a MAC on the ciphertext. Since the judge does not obtain k'_{11} from S, who is not supposed to have this key, and cannot compute the real key k_{11} in polynomial time by the CDH assumption, it is not possible to verify the ciphertext nor the MAC, and the simulated evidence is indistinguishable from an honest one to the judge. For multiple messages, OTRv1 remains context deniable, since after the initial key exchange, V can simulate the rest of the transcript arbitrarily.

- content deniability: Follows as context deniability.

- time deniability: At the beginning of a communication P signs a random value. This signature is indistinguishable from P's signatures at different times. Because of content deniability, S can compute indistinguishable MACs for arbitrary messages. Therefore, even if the authenticated message is time dependent, the execution remains time deniable.

Since OTRv1 is not source deniable, provers cannot claim that they have never before participated in any communication. However, users that have talked at least once before with another user have also signed at least one random value for an initial key exchange. Because of context deniability, provers can argue that their signatures were obtained from some communication transcripts and were then used to simulate others. The same is true for destination deniability against spies. This way, an acceptable level of deniability in off-the-record messaging is offered. More formally, the combined notions of *context-and-source deniability* and *context-and-destination deniability* are satisfied, since a simulator can use the auxiliary oracles for context deniability to obtain signatures of provers and verifiers on random values and simulate the rest, i.e., encrypting and authenticating messages with simulated ephemeral encryption and MAC-keys.

6. DISCUSSION

Adversary classification.

Regarding a message authentication protocol which is deniable against malicious verifiers but not against spies, it is beneficial to malicious verifiers to impersonate spies and present judges with evidence of an interaction which does not contain their secret keys. Judges would have to decide, possibly using some circumstantial evidence, whether they believe the adversaries on their identity and the claim of not having the verifiers' secret keys. This might be hard or in cases even impossible.

Unforgeability and deniability trade-off.

Depending on the authentication protocol, there can be a trade-off between deniability and unforgeability. Although forgeable protocols are fully deniable, unforgeability can and should not be sacrificed completely. We observed a simple example of such a trade-off in our analysis of classical digital signature schemes. When simply signing a message, such a scheme achieves destination deniability, without being globally unforgeable since a prover's signatures sent to different verifiers are indistinguishable. While a verifier designation by appending the intended verifier's public key results in a globally unforgeable scheme, this solution costs the scheme its destination deniability. When designing or using a message authentication protocol, it is crucial to analyze the scenario and clarify, which levels of deniability and unforgeability are desired.

Deniability in practice.

Formally, successfully denying an authenticated interaction relies among others on two basic assumptions. First, judges need flawless proofs in order to favor an adversary's evidence over a prover's denial and do not settle for less, e.g., log-files, IP-addresses, or the like. Secondly, judges are not biased, do not care about the adversary's motive to simulate an interaction, and do not fully trust the adversary. In practice, however, these assumptions may be too strong. Moreover, judges can be more powerful than assumed here. They may be able to communicate with adversaries in an online manner as mentioned in the introduction, force provers to reveal their randomness, and observe the network communication.

Outlook.

Our notions for deniable message authentication each capture, with the exception of a few dependencies discussed in Section 3.3, different aspects of deniability. As future work one can formally show the orthogonality of the notions. Moreover, we suggest studying the possibility of combining two or more message authentication protocols that satisfy different deniability notions in order to construct a protocol with stronger deniability guarantees.

7. ACKNOWLEDGMENTS

This work has been co-funded by the DFG as part of project D.2 within the Doctoral College 2050: Privacy and Trust for Mobile Users.

8. REFERENCES

[1] C. Alexander and I. Goldberg. Improved user authentication in off-the-record messaging. In *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007, Alexandria, VA, USA, October 29, 2007*, pages 41–47, 2007.

[2] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 60–79, 2006.

[3] N. Borisov, I. Goldberg, and E. A. Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pages 77–84, 2004.

[4] D. R. L. Brown. Deniable authentication with RSA and multicasting. *IACR Cryptology ePrint Archive*, 2005:56, 2005.

[5] C. J. F. Cremers and M. Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. *IACR Cryptology ePrint Archive*, 2011:300, 2011.

[6] Ö. Dagdelen, M. Fischlin, T. Gagliardoni, G. A. Marson, A. Mittelbach, and C. Onete. A cryptographic analysis of OPACITY - (extended abstract). In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, pages 345–362, 2013.

[7] Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 146–162, 2009.

[8] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 542–552, New York, NY, USA, 1991. ACM.

[9] C. Dwork and M. Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[10] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 409–418, 1998.

[11] L. Harn, C. Lee, C. Lin, and C. Chang. Fully deniable message authentication protocols preserving confidentiality. *Comput. J.*, 54(10):1688–1699, 2011.

[12] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 143–154, 1996.

[13] H. Krawczyk and T. Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*, 2000.

[14] J. Liu, X. Chen, and Z. Han. Full and partial deniability for authentication schemes. *Frontiers of Computer Science in China*, 4(4):516–521, 2010.

[15] R. Lu and Z. Cao. A new deniable authentication protocol from bilinear pairings. *Applied Mathematics and Computation*, 168(2):954–961, 2005.

[16] M. Naor. Deniable ring authentication. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 481–498, 2002.

[17] I. C. A. O. A. G. on Machine Readable Travel Documents. Updated technical report — supplemental access control. Twenty-second Meeting, May 2014, 2014.

[18] R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Berlin, Germany.

[19] M. D. Raimondo and R. Gennaro. New approaches for deniable authentication. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 112–121, 2005.

[20] M. D. Raimondo, R. Gennaro, and H. Krawczyk. Secure off-the-record messaging. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005*, pages 81–89, 2005.

[21] M. D. Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*, pages 400–409, 2006.

[22] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 552–565, 2001.

[23] M. Rjasko and M. Stanek. On designated verifier signature schemes. *IACR Cryptology ePrint Archive*, 2010:191, 2010.

[24] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, pages 523–542, 2003.

[25] C. D. Xin Xiangjun. A secure and efficient deniable authentication protocol. In *Information Engineering, 2009. ICIE '09. WASE International Conference on*, 2009.

[26] F.-Y. Yang and C.-M. Liao. A provably secure and efficient strong designated verifier signature scheme. *International Journal of Network Security*, 10(3):220–224, 2010.

[27] T.-Y. Youn, C. Lee, and Y.-H. Park. An efficient non-interactive deniable authentication scheme based on trapdoor commitment schemes. *Computer Communications*, 34(3):353–357, 2011.