New Approaches for Deniable Authentication*

Mario Di Raimondo † Rosario Gennaro ‡

May 31, 2006

Abstract

Deniable Authentication protocols allow a Sender to authenticate a message for a Receiver, in a way that the Receiver cannot convince a third party that such authentication (or any authentication) ever took place.

We present two new approaches to the problem of deniable authentication. The novelty of our schemes is that they do not require the use of CCA-secure encryption (all previous known solutions did), thus showing a different generic approach to the problem of deniable authentication. These new approaches are practically relevant as they lead to more efficient protocols.

In the process we point out a subtle definitional issue for deniability. In particular we propose the notion of *forward deniability*, which requires that the authentications remain deniable even if the *Sender* wants to later prove that she authenticated a message. We show that a simulation-based definition of deniability, where the simulation can be *computationally indistinguishable* from the real protocol does *not* imply forward deniability. Thus for deniability one needs to restrict the simulation to be perfect (or statistically close). Our new protocols satisfy this stricter requirement.

1 Introduction

Authentication is arguably the most important security goal in cryptography. When communication happens over a real-life network we need to make sure that we are talking to the right person and not with an impostor.

Authentication, thus, has received a lot of attention in the cryptographic literature. Authentication methods follow the usual distinction between private and public key techniques. In a private key scenario, two parties Alice and Bob share secret key k and use that to prove to each other that they are the originators of the messages. Usually this is done by Alice sending a message m to Bob together with a tag t, which is computed as a function of the message m and the key k. The pair m, t is verified by Bob as coming from Alice if the tag matches the one that B can compute on his own, with m and k. The tag is called a *message authentication code* and must satisfy some security properties (namely unforgeability) in order for this technique to be meaningful.

*A preliminary version of this paper appeared in the Proceedings of the 2005 ACM Conference on Computer and Communication Security.

†Dipartimento di Matematica ed Informatica – Università di Catania, Italy. diraimondo@dmi.unict.it Work done while visiting the IBM T.J. Watson Research Center.

‡IBM T.J.Watson Research Center, USA. rosario@watson.ibm.com

On the other hand in the public key scenario, message authentication has been long associated with *digital signatures* [18]. In this case, Alice is publicly associated with a public key pk_A which is matched to a secret key sk_A known only to her. When Alice wants to authenticate a message m, again it computes a tag t as a function of the secret key sk_A and the message m. The tag in this case is called a digital signature, and again must satisfy some meaningful notion of unforgeability (see [29]). The interesting twist is that the tag can be verified by anybody using the public key pk_A.

This last property is a very useful feature of digital signatures, as it provides the crucial *non-repudiation* property. Once Alice signs a message, she is bound to it. Everybody can verify that she signed it. This is very useful when digital signatures are used for contracts or commerce transactions, where conditions must be enforced in case of dispute.

On the other hand, this feature raises important privacy issues. What if Alice wants to say something very private to Bob, in a way that Bob believes it comes from her, but also in a way that Bob cannot convince a third party that Alice said such a thing? Or even that Alice spoke to Bob at all? Clearly digital signatures do not allow Alice to do this.

Notice that message authentication codes do not provide for non-repudiation, as the tag could be easily computed by the receiver. In other words once Bob gets m, t from Alice, he is convinced that it comes from her (as apart from Bob she is the only one who can compute t), but Bob can't show this to Charlie and convince him that it comes from Alice, as Bob could have computed t on his own.

But what if Alice and Bob don't have a shared secret key? They could, in principle, run a *key exchange protocol* (see for example [18, 4]). At the end of such protocol Alice and Bob hold a shared secret key k, and then they could use it to authenticate messages. But since most of the known key exchange protocols use digital signatures to authenticate the parties running them, at the end Bob can still convince Charlie that he spoke to Alice, even if not specifically about the subject of the conversation[1].

Deniable Authentication The issue of *deniability* in public key authentication was brought forward and formalized by Dwork, Naor and Sahai, in their groundbreaking paper on concurrent zero-knowledge [21]. The paradigm suggested in [21] is to replace the non-interactive transmittal of a digital signature, with an interactive communication protocol between Alice and Bob on input a message m. At the end of the protocol Bob is convinced that Alice wants to authenticate m to him, but will not be able to convince a third party as his view of the communication can be easily produced *a posteriori* even without the knowledge of Alice's secret key sk_A. This property is called deniability.

This protocol should maintain some meaningful unforgeability property, i.e. it should be hard for an adversary to convince Bob that Alice wants to authenticate a message m.

Dwork *et al.* point out that, since we are introducing interaction, we should consider what happens in a *concurrent* scenario, i.e. one in which an adversary may schedule executions of protocols and delay messages in arbitrary ways. That is, unforgeability and deniability should still hold against such a powerful attacker. This turned out to be a very powerful attack model, especially when considering zero-knowledge protocols.

The basic solution for deniable authentication based on encryption can be summarized as follows

[1] In fact a careful use of digital signatures in key exchange protocols may prevent Charlie even from proving that he spoke to Alice, though Alice could not deny that she was "alive" and talking to somebody at some point. See [19] for their analysis of the deniability properties of the SIGMA key exchange protocol [37].

(this protocol appeared first in [36], though similar protocols appear also in [20, 21]). Bob chooses a random key k and encrypts it under Alice's public key. Alice decrypts such key and uses it to MAC the message m. Bob's belief that Alice is really authenticating m comes from the fact that she is the only one able to decrypt k. On the other hand, Bob could create the whole transcript on his own, so the authentication is deniable.

Dwork *et al.* prove the unforgeability of the above scheme in a concurrent setting. The assumption required to prove security is that the encryption scheme must be secure against adaptive chosen-ciphertext attack (CCA2) [43, 20]. Informally, an encryption scheme E is said to be CCA2-secure if the secrecy of a message m, encrypted as $c = E_{pk}(m)$, holds up even if an adversary is allowed to obtain decryption of any ciphertext of her choice (except of course c). It should be clear why this property is needed in the above scheme: Alice is in effect acting almost like a decryption oracle, on ciphertexts that are under the control of Bob, whenever asked to authenticate a message (the "almost" comes from the fact that Alice does not answer directly k, but rather $MAC_k(m)$).

Deniability for this basic protocol, however, cannot be proven via a black-box simulation. In order to make the above scheme black-box simulatable, a basic challenge-response sub-protocol is added in [21]. However this introduces a rewinding step in the proof, which causes the deniability property to hold only if copies of the protocol are performed sequentially, and not concurrently. In order to overcome this problem Dwork *et al.* introduce *timing assumptions* on the network to limit the number of concurrent executions that can be performed by the adversary in the network. We refer to the above solution as the CCA-paradigm for deniable authentication.

Other approaches: In the literature there are alternatives that require the receiver/verifier to have a public key: *Designated Verifier Proofs* [34] create signatures that convince only the intended recipient (using his public key); *Ring Signatures* [47] allow a member of an *ad hoc* group to sign a message on behalf of the group, i.e. it is impossible to trace the actual signer inside the group. This solution can be used to create deniable signatures by choosing the sender and the receiver as members of the group: the signature is deniable as the receiver could have created it too. Naor in [44] observes that in ring signatures the public keys should be registered with a proof-of-knowledge of the corresponding secret-key: suppose that the receiver B registers a public key pk_B that is equal (or derived through a suitable one-way function) to A's one. If the ring signature has been created using these two public keys the involvement of A in the signature process is hardly deniable, since B has a way of proving that he does *not* know its own secret key.

Naor in [44] also presents *Deniable Ring Authentication* combining the encryption-based approach of Dwork *et al.* with Ring Signatures: one member (or a proper sub-structure) of a group can sign a message in a deniable way towards a receiver that is not required to have a public key. This solution can be considered as an extension of the CCA-paradigm of Dwork *et al.*

The requirement of a registered public key for both parties creates a less general model that does not fit in all practical applications (e.g. the Internet, where most users do not have public keys). Thus, we can conclude that in the most general setting, where only the prover is required to own a public key, all the known solutions follow the CCA-paradigm.

1.1 What if the Sender changes her mind?

In the definition of deniable authentication we assume that the Sender wants to preserve his privacy, and thus prevent the Receiver from proving to a third party that he received a message from the Sender. However there are scenarios in which deniability is actually a concern to the Receiver's

privacy.

Consider the following example. Alice and Bob are involved in some shady transaction, like drug-dealing or money laundering. Alice wants to make sure that her communications to Bob cannot be later linked to her, so she uses deniable authentication. Bob thinking that such communication is indeed deniable, stores all the messages in his hard disk. Later the operation is busted by the police and Alice and Bob end up in jail, and Bob's computer is seized. Alice is offered a sweet deal in exchange for her cooperation in linking Bob to the crime (Bob is claiming the messages in his hard disk are not coming from Alice, that he never talked to her, actually does not even know her, they are all simulations!!). Alice produces some piece of secret information (her secret key for example) that indeed shows that the transcripts in Bob's hard disk are actually authentic and not simulations. Bob ends up in jail, cursing himself for dropping out of crypto class in graduate school.

The above example shows that deniability is not just a concern of the Sender, but also of the Receiver. What we would like to happen is that if the Sender acts honestly during the protocols, she should not be able at a later stage to claim the messages as authentic. We call this property *forward deniability*, as it has some affinity to the notion of forward secrecy[2].

We would like to point out that the above CCA-based paradigm is indeed forward deniable. However the issue of forward deniability has not been discussed in the literature, and indeed we show that some proposed deniable protocols are not forward deniable (see below).

1.2 Our Contribution

New Approaches. The first question we asked was: "Are there other approaches to concurrent deniable authentication, besides the CCA paradigm?"[3]. We provide a positive answer to this question. We show that deniable authentication can be constructed out of different primitives.

The question is interesting for both theoretical and practical reasons. On the theoretical front, it is not clear why to build authentication protocols, encryption must be needed at all. One of our solutions uses a special kind of non-malleable commitment scheme, thus showing that deniable authentication, while linked to non-malleability, is not linked to encryption. The practical reason is that by creating new paradigms for deniable authentication we may end up with more efficient protocols or protocols based on weaker computational assumptions. This is indeed what we do in this paper.

We present two new schemes for deniable authentication. The first scheme eliminates the need for an encryption scheme altogether. We build deniable authentication protocols, using special kinds of trapdoor commitment schemes (the multi-trapdoor commitments of Gennaro [23]). The protocols using this approach are incredibly simple and efficient: the cost of the protocol is twice that of a regular digital signature [4].

[2] In forward secrecy if a party's key is compromised, only the secrecy of future messages is compromised, while past messages are still safe. Here if the Sender is "compromised" at some time t he will not be able to revoke the deniability from transactions happened before time t.

[3] As stated in Section 1 , we are investigating in the general setting where only the prover is required to have a registered public key.

[4] We note that we are only considering *strong* deniability in which Alice can deny to have ever authenticated anything to Bob. For *weak* deniability, where Bob can prove to have spoken to Alice but not the content of what Alice authenticated, signature schemes are sufficient as we pointed above in the Introduction, and thus weakly deniable schemes can be done at the cost of a single signature. In the following when referring to deniability we will always refer to the strong one.

The second scheme can be seen as an improvement of the CCA-paradigm when implemented with the Cramer-Shoup's CCA schemes in [14]. Namely, we use some specific properties of the *projective hash functions* [14] used in those schemes to build a new kind of deniable authentication. The scheme can still be thought as the encryption of a random key which is then used to MAC the message; however, it is not clear how to argue that the encryption module is CCA-secure. The net result is that we save one modular exponentiation compared to the CCA-paradigm solution, and the transcripts are shorter.

Improved Efficiency and Reductions. Our schemes are very efficient, and require four rounds. They can be proven secure in a concurrent setting, but the proof of deniability requires timing assumptions

On the other hand the proof of unforgeability holds even without timing assumption: while this is not a new feature of our schemes (the CCA-paradigm enjoys it too) it is still remarkable, as this proof in our commitment-based protocol uses rewinding as well, but in a way that does not compromise security in unbounded concurrent executions.

An interesting feature of the commitment-based protocol is that in a realistic multi-user setting (like the one we consider in this paper) the security reduction depends linearly on the number of players in the network while for the CCA-paradigm protocols the dependency is linear in the number of *sessions*. The latter can be a much higher parameter and thus the improvement is not just a mere theoretical feature but it is important in practice as it guarantees security with much smaller parameters and consequently improved efficiency.

Forward Deniability. Finally we present a formal definition of forward deniability. We argue that deniability obtained by proving that the protocol is computational zero-knowledge protocols is *not* forward deniability. We also prove that statistical zero-knowledge protocols (including our new proposals) are forward deniable.

To prove our schemes we present a unifying model to define deniable authentication. To prove that our protocols are secure authenticators we use the model introduced by Bellare *et al.* [4]. We then integrate the notion of deniability and forward deniability to it, by adding the required simulation properties.

1.3 Related Work

As we said above, the first solution is based on the notion of multi-trapdoor commitments [23]. These commitments were inspired by the work on non-malleability by Di Crescenzo, Ishai and Ostrovsky [16], and a series of works that followed their paradigm (e.g. [17, 35, 15, 41]). In our solution we exploit in a novel and original way their non-malleability properties in order to obtain deniable authentication. We note that some of the constructions of simulation-sound commitments presented in [41] (namely the ones that achieve information-theoretic privacy of the message) can also be used in our construction[5].

The second solution exploits the properties of ϵ-universal projective hash functions [14], to relax the requirement on the key establishment mechanism in the CCA paradigm. It is somewhat related to a recent improvement to the Cramer-Shoup CCA encryption proposed by Kurosawa and Desmedt [39]. They consider the hybrid version of the Cramer-Shoup cryptosystem described in [49]. There

[5]If we use the generic construction of simulation-sound commitments based on one-way functions, deniability is only achieved in the computational sense, and thus no forward deniability is obtained. On the other hand the number-theoretic constructions yield statistical ZK and forward deniability.

the encryption scheme is split in a Key Encapsulation Module (KEM) where the Cramer-Shoup encryption scheme is used to encrypt a random key, and a Data Encapsulation Module where the message is encrypted and MAC'ed using the above key. Kurosawa and Desmedt note that in this scenario it is not strictly necessary that the KEM is CCA-secure, and indeed they show how to convert the Cramer-Shoup KEM in one which is still sufficient for the overall CCA-security of the hybrid encryption, yet it is not known to be CCA secure itself. The net result is an improvement in the efficiency of the overall scheme (especially when considering the improved proof of security presented in [25]). In doing so, however Kurosawa and Desmedt introduced a stronger requirement on the projective hash functions used in the scheme.

Our work, which was done independently from [39], follows a related path for the case of deniable authentication. But our solution is conceptually simpler, again because we remove the need to *encrypt* anything. Instead of using the projective hash functions to establish a random key (which is the source of the stronger requirement in [39]), we use it *directly* as a message authenticator. Thus the original notion of ϵ-universality in [14] suffices.

Comparison with Katz's protocols. The CCA-paradigm for deniable authentication can also be considered in the context of *interactive* CCA-secure encryption protocols (exploiting the fact that since deniable authentication already introduces interaction, then we can use interaction also inside the encryption step). We point to the results of Katz [35] in this area. The protocols presented there, exploit the interactive nature of the encryption step, in order to get solutions which are more efficient than the ones based on the basic CCA-paradigm. In order to achieve interactive CCA-security, Katz uses proofs of plaintext knowledge for semantic secure encryptions (or even trapdoor permutations in some cases), combined with a non-malleable commitment scheme.

Our solutions show that if one strengthens the commitments used by Katz to be multi-trapdoor ones, then the commitments themselves yield efficient deniable authentication protocol. Thus we dispense with using encryption altogether. The tradeoff is that we use stronger computational assumptions: while the protocols in [35] can be proven secure based on the regular RSA and Computational Diffie-Hellman assumptions, the efficient instantiations of our protocols require either the Strong RSA, DSA, or Strong DH assumptions.

The efficiency of our protocol[6] is basically the same as the ones in [35]. However our protocol does not require one-time signature schemes, making the communication much shorter than in [35]. Finally our security reduction is more efficient as it depends only on the number of parties in the network, while Katz's depends on the total number of sessions.

Recent Work. After the publication of the preliminary version of this paper, in [19] we proved that the basic authentication protocol based on encryption (i.e. the one from [20, 36] without the challenge-response subprotocol) is *not* deniable under the mere assumption that E is CCA-secure (by showing an example of an encryption scheme E which is CCA-secure but when used inside the protocol yields a non-repudiable proof of authentication). On the other hand [19] proves that the required assumption on E to prove deniability of this protocol is *plaintext-awareness*.

In [19] the notion of deniable key exchange is also presented as an extension of the notion of deniable authentication from [21].

[6] The protocol in Figure 2 when instantiated with concrete number-theoretic commitments compared to the one in [35] based on a similar assumption – i.e. comparing the [35] protocol based on RSA (resp. CDH) with ours based on Strong RSA (resp. Strong DH or DSA).

1.4 Practical Applications

Deniability is an important privacy-enabling feature of cryptographic protocols and as such has many important practical applications.

The practical importance of this concept can be seen by the weight that deniability issues have played in the design of Internet key exchange protocols (see for example [36, 33, 42]).

Another typical example where deniability is important is for electronic elections. There while it is important that both parties (the voting authority and the voter) authenticate each other (for the authority to know that the voter has the right to vote, for the voter to know that her vote will be counted), it is also mandatory to prevent either party from walking away with a non-repudiable proof of what the actual vote was (the message being authenticated). This application, in particular, shows the importance of forward deniability: if the voter (sender) is authenticating her vote to the authority, not only the latter should not be able to prove to a third party how the voter voted, but even more importantly the voter herself should not be able to do so at a later stage, to prevent coercion and vote-selling.

Finally we point out that deniable authentication has applications in electronic commerce as well. As pointed out by Aumann and Rabin [1, 2], the use of deniable authentication, instead of regular signatures, can be used to communicate confidential terms of a transactions (such as price offers) without fear that such terms could be shown to a third party in an effort to obtain better terms (such as a better price offer).

2 The model

This section introduces the model used in the paper for the analysis of the authentication protocols. It was introduced by Bellare *et al.* [4] as a new modular approach to prove the security of authentication and key exchange protocols. Here we reuse and extend that idea for the analysis of protocols for deniable authentication.

This model deals with two kinds of network: an ideal authenticated network and a more realistic unauthenticated network. The former models a simplified peer-to-peer network of authenticated links in which the powers of adversary are limited to manage the delivery of the messages exchanged by the parties (it can't inject or manipulate messages) and to corrupt some of them. The latter has characteristics of a real network (with unauthenticated links) in which the adversary has full powers on the communication channels, so it can change and forge new messages.

The task to prove any kind of properties (like secrecy) of a protocol in a simplified world like our authenticated model is simpler than in a real network. To obtain a version of the protocol that works in a realistic unauthenticated network we can use special protocol compilers named authenticators that, informally, take a protocol for (ideally) authenticated networks and turns it in a protocol that has similar input-output characteristics in an unauthenticated network. This way to proceed permits to modularize the analysis of the properties of protocols. In [4] this was applied to prove the full security of Key Exchange protocols.

2.1 Definitions and main theorems

Here we recall some definitions, the adversarial model and the main theorems from [4]. Further, we introduce an extension to formalize the deniability of a protocol.

MESSAGE-DRIVEN PROTOCOLS. A message-driven protocol π is an iterative process that is initially invoked by a party with some initial state that includes the protocol's input, randomness and the party's identity. Once invoked, the protocols waits for an activation that can happen for the arrival of a message from the network or an external request (from other processes run by the same party). Upon activation, the protocol process the incoming data together with its current internal state, generating a new internal state (like a finite-states machine), as well as generating outgoing messages to the network and external requests to other protocols (or processes) run by the party. In addition, a cumulative output is generated. Once the activation is completed, the protocol waits for the next activation.

THE AUTHENTICATED-LINKS MODEL (AM). There are n parties $P_1, \ldots, P_n$, each running a copy of a message-driven protocol π. The computation consists of a sequence of activation of π within different parties. The adversary $\mathcal{A}$ is a probabilistic polynomial-time algorithm that control and schedule these activation. That is $\mathcal{A}$ can decide which is the next party to activate and which incoming message or external request the activated party is to receive. Upon the completion of an activation, the outgoing messages and external requests and the output generated by the protocol become known to $\mathcal{A}$. The new internal state remains unknown to $\mathcal{A}$.

In the authenticated-links model, $\mathcal{A}$ is restricted to delivering messages faithfully. That is, we assume that each message carries the identities of the sender P_i and of the intended recipient P_j. When a message is sent by a party, it is added to a set M of undelivered messages. Whenever $\mathcal{A}$ activates a party P_j on some incoming message m it must be that m is in the set M and that P_j is the intended recipient in m. Furthermore, m is now deleted from M.[7] Note that $\mathcal{A}$ can change the order of delivery and can choose to not deliver at all some messages.

The adversary $\mathcal{A}$ can corrupt parties as wish. Upon corruption $\mathcal{A}$ learns the entire current state of the corrupted party P_i and can add to the set M any fake messages, as long as P_i is specified as the sender of these messages. A special symbol in the output of P_i is generated to signal his corruption. $\mathcal{A}$ will control all the sequent activations of P_i. We refer to an adversary as described here as an AM-adversary.

Briefly, the global output of running protocol is the concatenation of all the outputs of the parties[8], together with the output of the adversary (derived from all the information gathered during the process). Let $\text{AUTH}_{\pi,\mathcal{A}}(\vec{x}, \vec{r})$ denote the global output of a running of the protocol π with the n parties and the adversary $\mathcal{A}$ with input $\vec{x} = x_1 \ldots x_n$ and random input $\vec{r} = r_0 r_1 \ldots r_n$ (r_0 for $\mathcal{A}$; x_i and r_i for party P_i, $i > 0$). Let $\text{AUTH}_{\pi,\mathcal{A}}(\vec{x})$ denote the random variable describing $\text{AUTH}_{\pi,\mathcal{A}}(\vec{x}, \vec{r})$ when $\vec{r}$ is uniformly chosen.

THE UNAUTHENTICATED-LINKS MODEL (UM). Basically, the unauthenticated-links model of computation is similar to the previous one, with the exception that here the adversary $\mathcal{U}$, referred to as a UM-adversary, is not limited to deliver messages that are in M. Instead, it can activate parties with arbitrary incoming messages (even with fake messages that were never sent).

Further, here the protocol π is augmented with an initialization function I that models an initial phase out-of-band and authenticated information exchange between the parties (like the creation of public and secret keys in asymmetric cryptosystems and the trustful exchange of public keys).

The random variables $\text{UNAUTH}_{\pi,\mathcal{U}}(\vec{x}, \vec{r})$ and $\text{UNAUTH}_{\pi,\mathcal{U}}(\vec{x})$ are defined analogously to $\text{AUTH}_{\pi,\mathcal{A}}(\vec{x}, \vec{r})$ and $\text{AUTH}_{\pi,\mathcal{A}}(\vec{x})$, but with the computation carried out in the unauthenticated-links model.

[7] This implies that no message appears twice. Alternatively, one can add message-ID's to messages to make them unique.

[8] This implies that also the identities of corrupted parties are part of the global output.

EMULATION OF PROTOCOLS. When we say that a protocol π' in the unauthenticated-links model emulates a protocol π in the authenticated-link model we want to capture he idea that 'running π' in an unauthenticated network has the same effect as running π in an authenticated network'. Formally speaking:

Definition 1 *Let π and π' be message-driven protocols for n parties. We say that π'* emulates *π* in unauthenticated networks *if for any* UM*-adversary $\mathcal{U}$ there exists an* AM*-adversary $\mathcal{A}$ such that for all inputs x,*

$$\text{AUTH}_{\pi,\mathcal{A}}(\vec{x}) \overset{\text{c}}{\approx} \text{UNAUTH}_{\pi',\mathcal{U}}(\vec{x}) \tag{1}$$

where $\overset{\text{c}}{\approx}$ denotes 'computationally indistinguishable'.

This definition implies that the combined distributions of the outputs of the parties, the adversary's output and the identities of corrupted parties should be indistinguishable on the two sides of Eq. 1. In general, this condition captures the required notion of "security equivalence" between the protocols in the sense that any consequences of the actions of the strong UM-adversary against executions of the protocol π' can be imitated or achieved by the weaker AM-adversary against the runs of protocol π without requiring the corruption of more (or different) parties.

AUTHENTICATORS. An authenticator is a 'compiler' that takes for input protocols designed for authenticated networks, and turns them in 'equivalent' protocols for unauthenticated networks.

Definition 2 *A* compiler *$\mathcal{C}$ is an algorithm that takes for input descriptions of protocols and outputs descriptions of protocols. An* authenticator *is a compiler $\mathcal{C}$ where for any protocol π, the protocol $\mathcal{C}(\pi)$ emulates π in unauthenticated networks.*

In particular, authenticators translate secure (in some well defined sense) protocols in the authenticated model in secure protocols in the unauthenticated-links model[9].

Bellare *et al.* [4] introduced also a natural way to construct full-fledged authenticators using simpler methods to authenticate a single message. Consider the banal protocol message transmission MT that transport a message from a party to another. Speaking in terms of message-driven protocol: upon activation within P_i on external request (P_j, m), party P_i sends the message (P_i, P_j, m) to party P_j and outputs 'P_i `sent` m `to` P_j'. Upon receipt of a message (P_i, P_j, m), P_j outputs 'P_j `received` m `from` P_i'.

Suppose that λ is a protocol that emulates MT in unauthenticated networks (we call such protocols MT-authenticators). We can construct a compiler $\mathcal{C}_\lambda$ that transform any protocol π in another that use the sub-protocol λ for the transmission of each message. In other words, the sub-protocol λ acts as a layer of transmission: instead of sending messages to the network, λ is activated for delivery of all the messages, and instead of receiving incoming messages from the network, the messages are taken from the λ's output. In [4] there is a theorem that prove that this technique yields valid authenticators.

Theorem 3 *Let λ be a* MT*-authenticator (i.e. λ emulates* MT *in unauthenticated networks), and let $\mathcal{C}_\lambda$ be a compiler constructed based on λ as described above. Then $\mathcal{C}_\lambda$ is an authenticator.*

For further details on the model, see the original paper [4].

[9] In the Introduction we claim that our deniable authentication protocols don't require that both parties have a public key. Sender's public key is sufficient for the aim of authentication of the message. In the ambit of the model, a simulation of the authenticated channels of the UM-model is possible only where the party who sends the message has a public key.

2.2 Extension for deniable methods

This paper focuses on deniable methods of authentication. A protocol for the authentication of a message is a protocol between a party A (who sends the message and proves its identity) and a party B that assures the latter of the integrity of the message and the identity of the sender A. Informally, we say that an authentication protocol is deniable if the transcript of its execution does not allow B to prove to a third party the participation of A. That is, the transcript of the interaction cannot be used as evidence that A took part in the protocol (i.e. A or B can later *deny* that the authentication took place).

For example, digitally signing a message m with the secret key of A is a valid technique for message authentication but it is completely non-deniable: only A could produce a valid signature and it is a proof for third parties.

To define deniable authentication we use the simulatability notion from [21] and combine it with the notion of MT-authenticator:

Definition 4 *We say that an* MT*-authenticator* λ *is* deniable *if for any receiver* B*, there exists a simulator* $\mathcal{S}_\lambda^{(B)}$ *that given a message* m *sent by a party* A *to* B *produces a transcript of a session of* λ *for* m *that is* indistinguishable *from a real one.*

Note that the simulator $\mathcal{S}_\lambda^{(B)}$ can't use the private information of the parties (i.e. private keys). This is enough to prove that the transcript of any session of λ can't be used by third parties to verify the participation of the involved participants. In fact, anyone could produce realistic transcripts of λ using the simulator $\mathcal{S}_\lambda^{(B)}$.

There are, as usual, three flavors of the above definition depending of what kind of indistinguishability the simulator achieves. We say that a deniable authenticator is perfectly or statistically zero-knowledge if the real and simulated transcripts follow distributions which are either identical or statistically close. We say that a deniable authenticator is computational zero-knowledge if the real and simulated transcripts follow distributions which are computationally indistinguishable (see [27, 28] for definitions of the various types of indistinguishability).

Remark: In [21] the authors do not specify what kind of indistinguishability they require, but it is clear from the context that they consider computational zero-knowledge protocols to be deniable. In [45] computational zero-knowledge is explicitly mentioned as sufficient for deniable authentication. On the other hand Katz in [35] explicitly limits deniable authentication protocols to be statistical ZK.

The same concept can be applied to a generic authenticator. Given a deniable MT-authenticator λ we say that the authenticator C_λ built as in Theorem 3 is a deniable authenticator.[10]

2.3 Forward Deniability

If we look at the example in Section 1.1, we see that deniability is not just a concern of the sender, but also of the receiver. In order to ensure forward deniability we need to make sure that at the end of a real execution, the sender does not inherit a "witness" of the fact that the transcript is real.

[10] More formally, we should define the concept of deniable authenticator as in Definition 4 and then straightforwardly prove that an authenticator C_λ built using a deniable MT-authenticator λ is a deniable authenticator.

Basically it must be hard for Alice to present a "witness" of the fact that a particular transcript is real. Given a sender A and a receiver B using a deniable MT-authenticator λ, for Definition 4 there exists a simulator $\mathcal{S}_\lambda^{(B)}$ that produces transcripts that are indistinguishable from the real ones. If this indistinguishability holds in a perfect or statistical way, our authenticator must be forward deniable too. Indeed, there does not exist any witness that the sender A can show to prove that his transcript is real.

Definition 5 *Let λ be a deniable* MT*-authenticator between sender A and a receiver B. We say that λ is* forward deniable *if the indistinguishability of the distributions produced by the deniability simulator is perfect or statistical.*

Remark: *On the honesty of the Sender.* In the definition above, we assume that A behaves honestly during the executions of λ. In particular we assume that A chooses its random input as prescribed by the protocol. A possible way for A to prove that a transcript is real, is to modify its coin tosses in a way that is not detectable from the outside but that will allow her later to prove that those messages were generated by her (for example, she could choose a random string r not by directly sampling it, but by choosing r' and setting $r = f(r')$ where f is a hard-to-invert permutation). This kind of behavior was termed *semi-honest* in [8]. Notice, however, that in the case of deniable authentication A herself is not interested in keeping such a strategy as the presence of such "witnesses", if leaked, may be used to prove that she authenticated a message.

What about computational indistinguishability? In the next remark we show that if the simulated distribution is only computational indistinguishable from the real one, then there exists a witness that the sender can reveal in a later stage (violating forward deniability).

Remark: *Computational ZK and forward deniability.* Consider any computational ZK protocol for an NP-complete problem, e.g. the one for graph 3-colorability [30]. The common input is a 3-colorable graph and the Prover knows such a coloring. In the first message the Prover commits to a random 3-coloring of the graph, i.e. for each vertex v commits to $\pi(col(v))$, where $col(v)$ is the color of v described as an integer in $\{1, 2, 3\}$ and π is a random permutation over the same set. Then the (honest) verifier asks for a random edge and the prover decommits, to the colors of the nodes composing that edge. Under the security of the commitment scheme this is a computational ZK protocol: the simulator for the honest verifier chooses a random edge, commits to different random colors for the nodes on that edge, and then commits to random colors for the other nodes. Note that since the simulator does not know the 3-coloring, the coloring which it commits to is not a correct 3-coloring, and that can be easily detected if all the commitments are opened. However since the commitments are secure, and only the chosen edge is opened, the protocol is computational ZK. However it is not forward deniable[11]. Indeed the prover's state contains the openings of the commitments, and thus the prover can produce information showing that a real transcript is indeed real (it contains a real 3-coloring).

This problem is shared by all the computational ZK protocols we know and thus shared by any protocol that proves deniability by reduction to such problems, and is sufficient to argue that computational zero-knowledge is not sufficient to achieve forward deniability. In Section 3.4, we show a deniable authentication protocol whose security is based on a reduction to a NP-complete language, and for that reason not forward deniable.

[11] Although we defined forward deniability in the context of authentication, the definition can be extended to any two-party protocol.

The only way in which forward deniability could be achieved is if Alice "forgets" about how she computed the commitments. I.e. erases her internal state (apart from her secret key) after the execution of the protocol. From the argument given in the previous Remark, it would appear in Alice's interest to do so. However this makes the assumption on the model that such erasures are indeed possible. This is a very strong assumption to make (e.g. see [31] for a survey on the difficulty of erasing data).

Thus while we assume that Alice behaves honestly during the protocol (and has all the motivation to do so) we also assume that it is hard for her to erase traces of her past executions of the protocol from her memory.

3 MT-authentication using Multi-trapdoor Commitment Schemes

In this section we present a family of deniable MT-authenticators that uses as building blocks *Multi-trapdoor Commitment Schemes*. For each instantiation of this kind of commitments we obtain a different scheme of deniable authentication.

3.1 Multi-trapdoor Commitment Schemes

A *commitment* is the digital equivalent of a "sealed envelope". A party commits to a value by placing it into a sealed envelope (this is the "committing phase"), so that the same party may later reveal the value by opening the envelope (the "opening phase"). Further, the envelope cannot be opened by another party before the opening phase (this is known as "secrecy" or "hiding" property) and its content cannot be altered (this is known as "binding" property).

A *Trapdoor Commitment Scheme* (TCS) is a commitment scheme where there exists a *trapdoor* the knowledge of which allows to open a commitment in any possible way (we will refer to this also as *equivocate* the commitment). Obviously this trapdoor should be hard to compute. In this way the privacy property of the commitment is information-theoretically guaranteed (i.e., given the commitment the receiver, even with infinite computing power, cannot guess the committed message better than at random). On the other hand, the binding property can be only be computational (because of the existence of the trapdoor).

A *Multi-Trapdoor Commitment Scheme*, introduced in [23], consists of a family of TCS with a special binding property. Here we shall use two slightly different definition of these schemes: an *adaptive* one (not presented in [23] and inspired by the notion of *Simulation-Sound Commitments*[12] (SSC) in [22, 41]) and the original definition by Gennaro in [23] which we will name *static*. The first is a stronger definition than the second one, so the instantiations of this kind of schemes are usually less efficient, but the meta-description of the authenticator becomes simpler. The two definitions differ only in the binding property as we shall see.

Let's start presenting the formal definition of adaptive MTC[13]. An *Adaptive Multi-Trapdoor Commitment (AMTC) Scheme* consists of five randomized algorithms: CKG, Sel, Tkg, Com and Equiv with the following properties:

- CKG is the master key generation algorithm: given a security parameter it outputs a pair $(\mathsf{PK}, \mathsf{TK})$, where PK is the *master public key* associated with the family of commitment

[12] They use the name of *Simulation-Sound Trapdoor Commitments* (SSTC) for their schemes but we prefer to not use the attribute "trapdoor" for the observations raised in Section 3.4

[13] We elaborate in Section 3.4 on the differences between this definition and the definition of SSTC in [41].

schemes and TK is the *master trapdoor key*;

- Sel is the algorithm that selects a particular scheme in the family: given PK it outputs a pk that identifies one of the schemes;
- Tkg is the algorithm that generates the trapdoors: given the triple $(\mathsf{PK}, \mathsf{pk}, \mathsf{TK})$ it outputs the trapdoor information tk relative to pk;
- Com is the commitment algorithm: on input PK, pk and a message M it outputs $C(M) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R)$ where R are the coin tosses. To open a commitment the sender reveals (M, R) and the receiver verifies using Com to recompute the commitment;
- Equiv is the algorithm that opens a commitment in any possible way given an original opening and the trapdoor: it takes as input PK, pk, a commitment C of a message M, its opening (M, R), a different message $M' \neq M$ and a value T; if $T = \mathsf{TK}$ or $T = \mathsf{tk}$ then Equiv outputs R', uniformly chosen among all R' such that $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$.

The notion of AMTC requires the following security properties:

Information Theoretic Security For every message pair (M, M') the distributions of the commitments $C(M)$ and $C(M')$ are statistically close;

AMTC Secure Binding Consider the following game: the adversary $\mathcal{A}$ is given a public key PK for a multi-trapdoor commitment family, generated with the same distribution as the ones generated by CKG. Also, $\mathcal{A}$ is given access to an oracle $\mathcal{EQ}$ (for *Equivocator*). This oracle gets as input a string $(C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R), M, R, M')$ with message $M' \neq M$ and outputs a random R' such that $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$ (that is the oracle creates openings with an arbitrary message M'). The adversary $\mathcal{A}$ wins if it outputs $(\mathsf{pk}, M, R, M', R')$ such that $\mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$, $M' \neq M$ and pk is different from all the public keys used during queries to the oracle $\mathcal{EQ}$ (in other words, $\mathcal{A}$ must never have used the oracle $\mathcal{EQ}$ to equivocate a commitment on the scheme with public key pk). We require that for all the efficient algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible in the security parameter.

The notion of (static) MTC in [23] is identical to the previous except for the property of binding that use a different game. Roughly, the adversary must choose the public keys to use with the oracle *before* seeing the master public key PK. More precisely:

MTC Secure Binding Consider the following game: the adversary $\mathcal{A}$ selects k public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ (with k polynomial in the security parameter), then it is given a public key PK for a multi-trapdoor commitment family, generated with the same distribution as the ones generated by CKG. The oracle $\mathcal{EQ}$ still gets as input a string $(C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R), M, R, M')$ with message $M' \neq M$ but it outputs a random R' such that $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$ only if $\mathsf{pk} = \mathsf{pk}_i$ for some i, otherwise it outputs *nil*. The adversary $\mathcal{A}$ wins if it outputs $(\mathsf{pk}, M, R, M', R')$ such that $\mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$, $M' \neq M$ and $\mathsf{pk} \neq \mathsf{pk}_i$ for all i. As before, we require that for all the efficient algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible in the security parameter.

In Appendix B, for the sake of completeness, we show an example of MTC scheme based on the Strong-RSA Assumption.

3.2 AMTC-based MT-authenticators

Here we present a deniable MT-authenticator λ_{AMTC} based on the notion of *Adaptive Multi-Trapdoor Commitments* (AMTC). First we prove that it is a MT-authenticator then we verify the deniability of the scheme.

Let's start from the initialization function I of the protocol λ_{AMTC}. For each party P_i, the master key generation algorithm of the AMTC scheme is invoked obtaining the pair $(\mathsf{PK}_i, \mathsf{TK}_i)$. Further, a hash function $\mathcal{H}_i$ is chosen from the family of UOWHFs such that it outputs strings with the same distribution of the algorithm Sel[14]. The public key of P_i is $PK_i = (\mathsf{PK}_i, \mathcal{H}_i)$ and the secret key is the master trapdoor key TK_i. So, the public information I_0 is simply the collection of all the public keys:

$$I_0 = PK_1, \ldots, PK_n$$

and the secret information of the player P_i is $I_i = \mathsf{TK}_i$.

Next, when activated, within party P_i and with external request to send message m to party P_j, protocol λ_{AMTC} invokes a two-party sub-protocol $\hat{\lambda}_{AMTC}$ between P_i and P_j. Since the sub-protocol $\hat{\lambda}_{AMTC}$ involves only two parties, we use the names A and B instead of P_i and P_j for simplicity. In this context, with $(\mathsf{PK}, \mathcal{H})$ and TK we indicate the public and secret keys of A.

The protocol $\hat{\lambda}_{AMTC}$ works as follow: first A uses the hash function $\mathcal{H}$ to select a specific scheme from the family of AMTC schemes in the following way: $\mathsf{pk} = \mathcal{H}(m, B)$ where m is the message to send and B is the identity of the receiver. After that a random string a is selected from the space of the messages of the AMTC scheme and another random string r is chosen. The commitment algorithm associated to the public key pk is used to commit the string a with coin tosses r obtaining $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, a, r)$. Finally, A sends '`message:`m, C' to B and outputs 'A `sent message` m `to` B'.

Upon receipt of '`message:`m, C' from A, party B chooses a random string c (for *challenge*) from the space of the messages of the AMTC scheme and sends it to A as '`challenge:`m, c'.

Upon receipt of '`challenge:`m, c' from B, party A uses the master trapdoor key TK to equivocate the commitment C so that the message committed becomes the challenge string c. He computes $r' = \mathsf{Equiv}(\mathsf{PK}, \mathsf{pk}, C, a, r, c, \mathsf{TK})$ so that (c, r') becomes another opening of the commitment C (remember that the first opening is (a, r)). A replies to B with '`reply:`m, r''.

When B receives the reply, he simply checks if the pair (c, r') is an opening for the commitment C, that is if $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, c, r')$. Note that B can compute the specific public key pk by himself using the hash function $\mathcal{H}$. If the check is correct, then B accepts m and outputs 'B `received` m `from` A'. Otherwise, B rejects this message and terminates this invocation of $\hat{\lambda}_{AMTC}$ [15]. Note that the length of strings to commit a, c should be long enough so it's infeasible to guess them (for example, 80 bits). A pictorial representation of a complete invocation of $\hat{\lambda}_{AMTC}$ for a message m can be seen in figure 1.

Theorem 6 *If the underlying commitment scheme is an AMTC, then protocol λ_{AMTC} emulates protocol* MT *in unauthenticated networks.*

PROOF: To prove that λ_{AMTC} emulates correctly the protocol MT in an unauthenticated network we need to show that all the things that an adversary can do against λ_{AMTC} in an unauthenticated

[14] For all known AMTC and MTC schemes such hash functions exist and they are efficiently computable.

[15] One does not need to send the message m in each of the flows of the protocol. A can send it in the first flow only, or even in the last flow only.

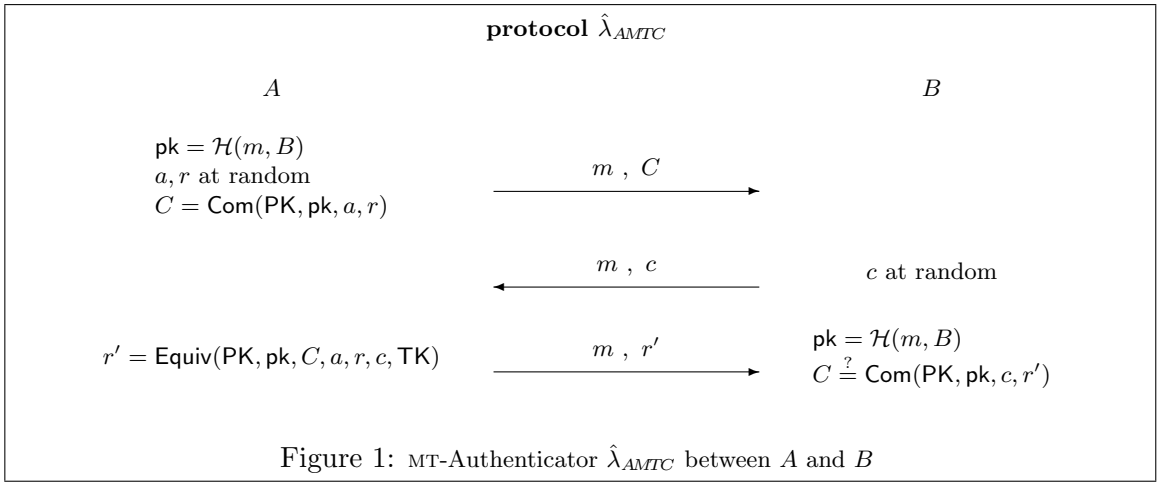

Figure 1: MT-Authenticator $\hat{\lambda}_{AMTC}$ between A and B

world can be done against the simple protocol MT in a hypothetical authenticated environment (the vice versa is straightforward). More specifically, let $\mathcal{U}$ be a UM-adversary that interacts with λ_{AMTC}. We want construct an AM-adversary such that $\text{AUTH}_{\text{MT},\mathcal{A}}() \approx \text{UNAUTH}_{\lambda_{AMTC},\mathcal{U}}()$,[16] that is their respective views are indistinguishable.

Adversary $\mathcal{A}$ runs $\mathcal{U}$ on the following simulated interaction with a set of n "fake" parties $P'_1 \dots P'_n$ running λ_{AMTC}. Note that $\mathcal{A}$ interacts with n "real" parties $P_1 \dots P_n$ running the protocol MT in an authenticated-link environment. First $\mathcal{A}$ invokes the initialization function I of λ_{AMTC} so that each simulated party P'_i has a pair of keys (PK_i, SK_i). Next, when $\mathcal{U}$ activates some imitated party A' for sending a message m to imitated party B', adversary $\mathcal{A}$ activates the dual party A in the authenticated network to send m to B. Moreover, $\mathcal{A}$ continues the interaction between $\mathcal{U}$ and the imitated parties running λ_{AMTC}. When some imitated party B' outputs 'B' `received` m `from` A'', adversary $\mathcal{A}$ activates corresponding party B in the authenticated-links model with incoming message m from A. If $\mathcal{U}$ corrupts some imitated party A' in its world, $\mathcal{A}$ corrupts the dual party A and hands the corresponding information to $\mathcal{U}$ (including the private information as the A's secret key that $\mathcal{A}$ knows). Finally, $\mathcal{A}$ outputs whatever $\mathcal{U}$ outputs.

Now we need to show that the output of $\mathcal{A}$ is indistinguishable from the output of $\mathcal{U}$. It is easy to see that outputs are identical unless the following event happens. Let $\mathcal{B}$ denote the event that imitated party B' outputs 'B' `received` m `from` A'' where A' is uncorrupted and the message (m, A', B') is not currently in the set M of undelivered messages.[17] In other words, $\mathcal{B}$ is the event where B' outputs 'B' `received` m `from` A'', and either A wasn't activated for sending m to B, or B has already had the same output before. In this case we say that $\mathcal{U}$ broke party A'. This is the only thing that our $\mathcal{A}$ cannot simulate without breaking the definition of AM-adversary. If the event $\mathcal{B}$ does not happen, it's straightforward to see that the simulation run by $\mathcal{A}$ is perfect and that $\text{AUTH}_{\text{MT},\mathcal{A}}() \stackrel{\mathsf{d}}{\approx} \text{UNAUTH}_{\lambda_{DDH},\mathcal{U}}()$ (where $\stackrel{\mathsf{d}}{\approx}$ denotes 'equally distributed').

The remaining step is to prove that the probability of the event $\mathcal{B}$ is negligible. Assume that event $\mathcal{B}$ occurs with non negligible probability δ. We construct an adversary $\mathcal{E}$ (for *Equivocator*) that breaks the security of the AMTC scheme with non negligible probability (related to δ).

[16] Note that the MT protocol ignores its input, so we consider only the empty input.

[17] See the Section 2.1 for the details about the definition of an AM-adversary.

CONSTRUCTION OF $\mathcal{E}$. Given the master public key PK^* for a Adaptive Multi-Trapdoor Commitment Scheme and access to relative equivocator oracle $\mathcal{EQ}$, the adversary $\mathcal{E}$ runs $\mathcal{U}$ on the following simulated interaction with a set of parties running λ_{AMTC}. First $\mathcal{E}$ chooses and distributes keys for the imitated parties according to function I, with the exception that the public key associated with some party A^*, chosen at random, is replaced with the pair $(\mathsf{PK}^*, \mathcal{H}^*)$ where PK^* is the input key and $\mathcal{H}^*$ is a hash function as described in I. Next, $\mathcal{E}$ continues the simulated interaction. If during the simulation $\mathcal{U}$ asks to corrupt party A^* then the simulation aborts and $\mathcal{E}$ fails. If A^* is required to reply to a `'challenge:`m, c`'` from a party B (where A^* sent a first message `'message:`m, C`'` using the specific commitment scheme associated with $\mathsf{pk} = \mathcal{H}^*(m, B)$ with opening (a, r)) then $\mathcal{E}$ uses the oracle $\mathcal{EQ}$ to equivocate C. The oracle $\mathcal{EQ}$ is given as input the commitment C, the first opening (a, r) and the challenge c (that with high probability is different than a). It returns a randomness r' such that (c, r') is another opening of C, so A^* can reply with `'reply:`m, r'`'`.

If some party D outputs `'`D `received` m^* `from` A^*`'` and A^* was not activated to send m^* to D (or if D has already output this value before), this means that $\mathcal{U}$ broke on party A^* using the message m^*. Since D outputs `'`D `received` m^* `from` A^*`'`, he collected from the adversary a first message `'message:`m^*, C`'` and a replying message `'reply:`m^*, r'`'` to his challenge c. This means that we have a commitment C (related to the public key $\mathsf{pk} = \mathcal{H}^*(m^*, D)$) and a first opening (c, r'). To obtain a second opening $\mathcal{E}$ can *rewind* the simulator that runs $\mathcal{U}$ (which is under his control) to the point in which D sent its challenge c for the broken session and then use a different `'challenge:`$m^*, \bar{c}$`'` with $\bar{c} \neq c$. Now suppose that $\mathcal{U}$ runs for at most t steps (t polynomial in the security parameter). The two mutually-exclusive events that can occur are:

- the adversary $\mathcal{U}$ breaks another time the same session between A^* and D with message m^* (but this time with challenge $\bar{c}$);
- this time $\mathcal{U}$ doesn't break the same session (or doesn't break anyone). To detect this event the simulator can wait until the running time limit t. In this case $\mathcal{E}$ rewinds another time the simulation and draws a different random challenge $\bar{c}$. This until the previous event occurs.

Later we analyze the expected running time of this simulation. At this point we can assume that $\mathcal{U}$ breaks the same session after the rewind: this means that he forges another `'reply:`$m^*, \bar{r}'$`'` where the pair $(\bar{c}, \bar{r}')$ is a second opening for C. Now $\mathcal{E}$ can stop the simulation and output the commitment C and the two different openings (c, r') and $(\bar{c}, \bar{r}')$. Note that the public key pk of the broken commitment scheme was never used during the invocations to the oracle $\mathcal{EQ}$. In fact, the pair (m, B) is different for each session (recall that all the messages coming from a party are all different) and the hash function $\mathcal{H}^*$ used to draw the AMTC scheme for every session is collision-resistant.

Now we compute the expected-running time of the iterated simulation process. Consider the exact instant in the simulation when the message `'message:`m^*, C`'` is exposed. Related to this instant, there is a fixed well-defined probability p that the adversary $\mathcal{U}$ will break that particular session. We can see it as:

$$p = \frac{\text{number of challenges } c \text{ on which } \mathcal{U} \text{ can reply}}{\text{number possible challenges}}$$

When $\mathcal{E}$ rewinds the simulation this probability related to the session (A^*, m^*, D) is almost the

same.[18] Let p' be this probability, so $p' \sim p$. If the probability that we draw a second challenge on which $\mathcal{E}$ can reply is p', the expected-number of times we need to iterate the rewind is exactly $\frac{1}{p'}$. Follows that the expected-running time of the simulation is

$$t + p\left(\frac{1}{p'}\right)t \sim 2t$$

Another way to see this is the following. If the probability p is small we need to reiterate the rewinds many times but this event can happen only with the same small probability p. If the probability p is big, we need to rewind few times and this happens more probably.

$\mathcal{U}$'s view in the simulation (conditioned to the event that it does not fail) is exactly the same of a real interaction with an unauthenticated network. In fact A^* is randomly chosen. If the number of parties in the simulation is n, the probability that $\mathcal{E}$ guesses the party broken by $\mathcal{U}$ is $\frac{1}{n}$, so the probability of success of the equivocator $\mathcal{E}$ is $\frac{\delta}{n}$ (a non negligible quantity) with an expected-running time of $2t$. This actually breaks the security of the AMTC scheme.

An alternative way to think of the equivocator $\mathcal{E}$ is the following. We say that the message '`message:`m^*, C' is *good* if the associated p is bigger than $1/2$. If the message is good, by rewinding ℓ times (where ℓ is polynomial in the security parameter) we get a different opening with overwhelming probability ($\geq 1 - 2^{-\ell}$). On the other hand the first break will happen on a bad (i.e. not good) message '`message:`m^*, C' with probability at most $1/2$. So we rewind exactly ℓ times, and the success probability is bigger than $\frac{\delta}{2n}(1 - 2^{-\ell})$ which is still non negligible. □

Remark: Note that in the proof of this theorem (exactly as in the following Theorem 8) the equivocator $\mathcal{E}$ has to guess the party broken by $\mathcal{U}$ among n parties; this implies that the probability to break the AMTC is reduced by a factor of $\frac{1}{n}$. This is a better result than other reductions (like in Katz's [35]), where this factor is $\frac{1}{l}$, with l the number of total sessions that all the parties open. Such quantity can be much higher than the number of peers in a real network and this fact leads to more secure protocols and to smaller security parameters.

Implementations. The number-theoretic constructions of simulation-sound commitments in [41] can be shown to be AMTC's and thus can be used in our protocol. They are based on the Strong RSA assumption and the security of the DSA signature scheme. We can't however use the generic construction based on one-way function as that does not satisfy the notion of AMTC (see Section 3.4).

3.2.1 Deniability

λ_{AMTC} is deniable for an honest receiver. Indeed in that case the simulator could: (i) compute the public key pk associated to the particular commitment scheme of the session as $\mathsf{pk} = \mathcal{H}(m, B)$; (ii) choose at random the challenge string c and the randomness r'; (iii) compute the commitment $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, c, r')$.

But for a dishonest verifier the way in which λ_{AMTC} authenticates the messages is actually *not* deniable. Here is a strategy from a dishonest verifier B who tries to get a transcript that A can't later deny. B could compute $c = hash(C)$ for some complicated hash function $hash$ after seeing the original commitment C. Now the above simulator will be in trouble as it chooses c before seeing C.

[18] In fact, in each iteration we need to consider all the challenges on which $\mathcal{U}$ can reply except the challenges used in the previous rewinds.

We modify the protocol in order to make it deniable. We assume that the public key of A contains the public key t for a regular trapdoor commitment scheme. The idea is to have B use t to commit to the challenge in advance. The protocol appears in Figure 2.

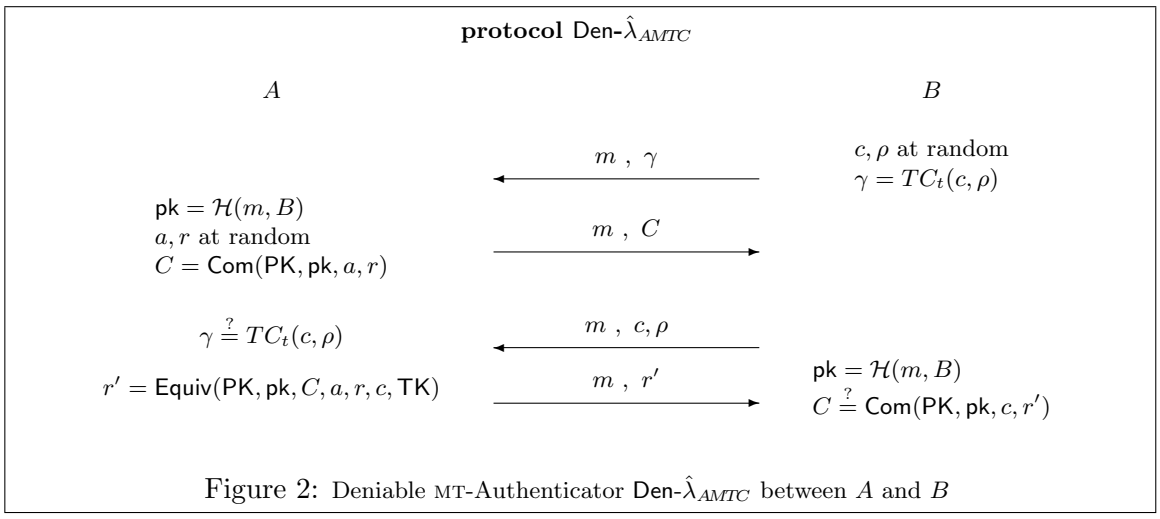

Figure 2: Deniable MT-Authenticator Den-$\hat{\lambda}_{AMTC}$ between A and B

Theorem 7 *Protocol* Den-$\hat{\lambda}_{AMTC}$ *is a forward deniable authenticator if used sequentially.*

PROOF: Before we prove deniability let's make sure that the protocol is still an authenticator. The proof of theorem 6 is identical except for the handling of event $\mathcal{B}$. There we showed that if event $\mathcal{B}$ happened with substantial probability then we could construct an equivocator $\mathcal{E}$ that would break the security of the AMTC.

$\mathcal{E}$ performed an "extraction" procedure in which it asked party A^* two different challenges on the same AMTC C. However this time the challenges are committed in advance using TC, but that's not a problem as we can give to $\mathcal{E}$ the trapdoor t so that it can open the corresponding γ in any way possible. The rest of the proof remains unchanged.

The deniability simulator works using standard zero-knowledge techniques. For any, possibly dishonest, receiver B, it gets m, γ from B and computes $\mathsf{pk} = \mathcal{H}(m, B)$ and $C' = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, a, r)$ for random a, r. Then B opens c. At this point the simulator rewinds it to the previous step and places $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, c, r')$ for random r'. If B does not decommit on this value of C, the rewinding step is repeated[19] until B decommits to c and the simulator places c, r' on the last message. The expected running time of this kind of simulator is polynomial as analyzed in [26].

The simulated transcript is perfectly indistinguishable, so *forward deniability* holds too.

□

Remark: *Concurrent Executions.* First we point out that the modified protocol Den-λ_{AMTC} remains an authenticator even if used in a concurrent setting. This is remarkable, as we are using rewinding in the proof of its unforgeability. On the other hand the rewinding in the proof of deniability is more troublesome and in a concurrent setting the adversary can create a scheduling which will result in a running time exponential in the number of open sessions (see [21]). Thus we can

[19] If B decommits to a different value then we can break the security of TC.

only use the protocol with a logarithmic number of such sessions open at any time, and this can be enforced by using timing assumptions as in [21]. Notice that if the parties are not concerned about deniability then unbounded number of executions can be performed concurrently.

3.3 Some MTC-based MT-authenticators

In Section 3.1 we presented also the original definition of *Static Multi-Trapdoor Commitment* (MTC) schemes. It is a weaker definition than AMTC, in fact the proposed implementations of MTC schemes are more efficient. Gennaro [23] introduced two efficient implementations of MTC schemes: one based on the Strong RSA Assumption and another on a stronger variant of the DDH Assumption introduced by Boneh and Boyen [6] over gap-DDH groups. This schemes are about twice as fast as the AMTC's in [41].

We show that the previous AMTC-based authenticator can be still proven secure using a MTC scheme.

Looking at proof of Theorem 6 we can easily see why such proof does not hold with MTCs. We construct an adversary $\mathcal{E}$ that given access to the adversary $\mathcal{U}$ and to the oracle $\mathcal{EQ}$ can break the security of the commitment scheme. In particular $\mathcal{E}$ uses this oracle $\mathcal{EQ}$ to reply to the opening requests that arrive to party A^* (for which we don't known the master trapdoor key TK).

When given access to an MTC oracle $\mathcal{EQ}$ we need to know in advance the public keys pk_i for these invocations. Remember that the public key pk used during the session is determined by the message m and by the identity of the receiver B as $\mathsf{pk} = \mathcal{H}(m, B)$, thus we can't guess the messages that A^* will be asked to send during the simulation.

The first, most obvious, idea is to use a *chameleon hash* function (see [38]) in the public key of each player. We recall that a chameleon hash function $\mathcal{H}$ is a collision-resistant hash function, with a trapdoor $t_{\mathcal{H}}$ whose knowledge allows one to find arbitrary collisions. More specifically H takes two arguments: a message M and a random string R. Given $h = \mathcal{H}(M, R), M, R, M', t_{\mathcal{H}}$ it is easy to find R' such that (i) $h = \mathcal{H}(M', R')$ and (ii) R' is uniformly distributed among all the random strings with that property.

In the above λ_{AMTC} protocol we use $\mathcal{H}$ as follows. We set $M = (m, B)$ and choose a random R to compute $\mathsf{pk} = \mathcal{H}(M, R)$. We add R to the first message flow. Now we can prove the protocol secure even if we are using an MTC family instead of a AMTC.

Indeed, the proof proceeds similarly to Theorem 6, i.e. we construct an equivocator $\mathcal{E}$ which will use the UM-adversary $\mathcal{U}$.

As in the previous proof: let A^* be the party chosen at random among the n parties. $\mathcal{H}^*$ is the hash function of A^*. The main difference is that $\mathcal{E}$ is given the trapdoor information $t_{\mathcal{H}^*}$.

In the proof of the above theorem, we set the AMTC master key PK^* to be the public key of A^* and then asked to equivocate specific public keys pk^* as they come. Here, instead, since we are using a MTC family we need to fix these specific public key in advance.

Let q be a polynomial upper bound to the number of messages that A^* is requested to send. We choose q random public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_q)$ as $\mathsf{pk}_i = \mathcal{H}^*(\alpha_i, \beta_i)$ where α_i, β_i are randomly chosen. We declare that these are the keys we want to equivocate on, and we get in return PK^* the master key for a MTC scheme, and the oracle $\mathcal{EQ}$ which will equivocate commitments related to the pk_i keys.

When $\mathcal{A}^*$ receives the i^{th} request to send a message m_i to a party B_i, $\mathcal{E}$ uses $t_{\mathcal{H}^*}$ to find randomness R_i for which $\mathcal{H}(M_i, R_i) = \mathsf{pk}_i$ (where again $M_i = (m_i, B_i)$). After, when A^* will

receive the challenge from B_i, $\mathcal{E}$ can invoke the equivocator oracle $\mathcal{EQ}$ to open the commitment as requested. The rest of the simulation proceeds as in the original proof.

Remark: What we described above is basically a generic way to build AMTC commitments out of MTC ones via mapping the specific public keys through a chameleon hash function (indeed the AMTC schemes in [41] can be thought as the composition of a chameleon hash with a MTC scheme).

However, we only know how to build chameleon hashing using expensive public key operations. In Appendix C we show two alternative assumptions that can be made on $\mathcal{H}$ to avoid using chameleon hashing, and without changing the protocol. In one method we use the Random Oracle model (a non-standard approximation of the real model). The other method introduces a strong (but well defined) computational assumption on $\mathcal{H}$ (an assumption modeled after a similar one used in [24]).

Finally in the next section we show another very efficient MT-authenticator using MTC schemes and one-time signatures, which can be proven secure in the standard model and just assuming basic collision-resistance for $\mathcal{H}$.

3.3.1 MT-authenticator based on MTC and one-time signatures schemes

In this section we introduce another version of MTC-based MT-authenticator that, using one-time signatures, can be proven to be secure in the standard model. This construction, together with efficient implementations of MTC schemes, leads to very efficient solutions for the problem of deniable message authentication. The cost to pay is slightly longer transcripts because of the one-time signature scheme. We shall denote this protocol with $\lambda_{MTC\&Sig}$.

One-time Signatures. A signature scheme consists of a triple of algorithms: Gen the key generator which on input the security parameter outputs a pair of keys $(\mathsf{vk}, \mathsf{sk})$; Sign to sign a message using the secret sign key sk and Ver to verify a signature using the public verification key vk. Our construction requires a one-time signature scheme which is secure against chosen message attack. Informally this means that the adversary is given the public verification key vk and a signature on a message of his choice (chosen after seeing vk). Then it is infeasible for the adversary to compute a signature on a different message.

One-time signatures can be constructed more efficiently than general signatures since they usually do not require complex operations like modular operations (see [40] for further details).

The protocol. The initialization function I of the protocol $\lambda_{MTC\&Sig}$ is identical to the λ_{AMTC}'s one at Section 3.2 except that now we are using a Multi-trapdoor Commitment (MTC) scheme. Each party P_i has a public key $(\mathsf{PK}_i, \mathcal{H}_i)$ and a secret key TK_i, where PK_i and TK_i are respectively the master public key and the master trapdoor key of a family of MTCs and $\mathcal{H}_i$ is a chosen UOWHF with output distributed as the output of the algorithm Sel for the selection of a commitment.

The two-party sub-protocol $\hat{\lambda}_{MTC\&Sig}$ between two generic parties A and B is the following. Let $(\mathsf{PK}, \mathcal{H})$ and TK be the public and secret keys of A.

First A uses the algorithm Gen for the generation of the keys of the one-time signature scheme obtaining a pair $(\mathsf{sk}, \mathsf{vk})$. A signature of the pair (m, B) is computed using the secret sign key sk: $\mathsf{sig} = \mathsf{Sign}_{\mathsf{sk}}(m, B)$. Then a specific member of the family of MTC schemes is selected applying the hash function $\mathcal{H}$ to the verification key vk as $\mathsf{pk} = \mathcal{H}(\mathsf{vk})$. Exactly as in the other protocol, a random string a is selected from the space of the messages of the MTC scheme and another random string

r is chosen. A commitment of the string a with coin tosses r is computed as $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, a, r)$. Finally, A sends '`message:`$m, C, \mathsf{vk}, \mathsf{sig}$' to B and outputs 'A `sent message` m `to` B'.

Upon receipt of '`message:`$m, C, \mathsf{vk}, \mathsf{sig}$' from A, party B checks if the signature sig is valid according to the verification key vk. If it is, a random challenge c is chosen from the space of the messages of the MTC scheme and the message '`challenge:`m, c' is sent to A. Otherwise B rejects and closes the session.

The rest of the protocol is identical to $\hat{\lambda}_{AMTC}$, that is: upon receipt of '`challenge:`m, c' from B, party A computes $r' = \mathsf{Equiv}(\mathsf{PK}, \mathsf{pk}, C, a, r, c, \mathsf{TK})$ so that (c, r') is another opening of the commitment C and he replies to B with '`reply:`m, r''. When B receives the reply, he checks if the pair (c, r') is an opening for the commitment C. If it is then B accepts m and outputs 'B `received` m `from` A'. Otherwise, B rejects this message and terminates this invocation of $\hat{\lambda}_{MTC\&Sig}$.

A pictorial representation of a complete invocation of $\hat{\lambda}_{MTC\&Sig}$ for a message m can be seen in figure 3.

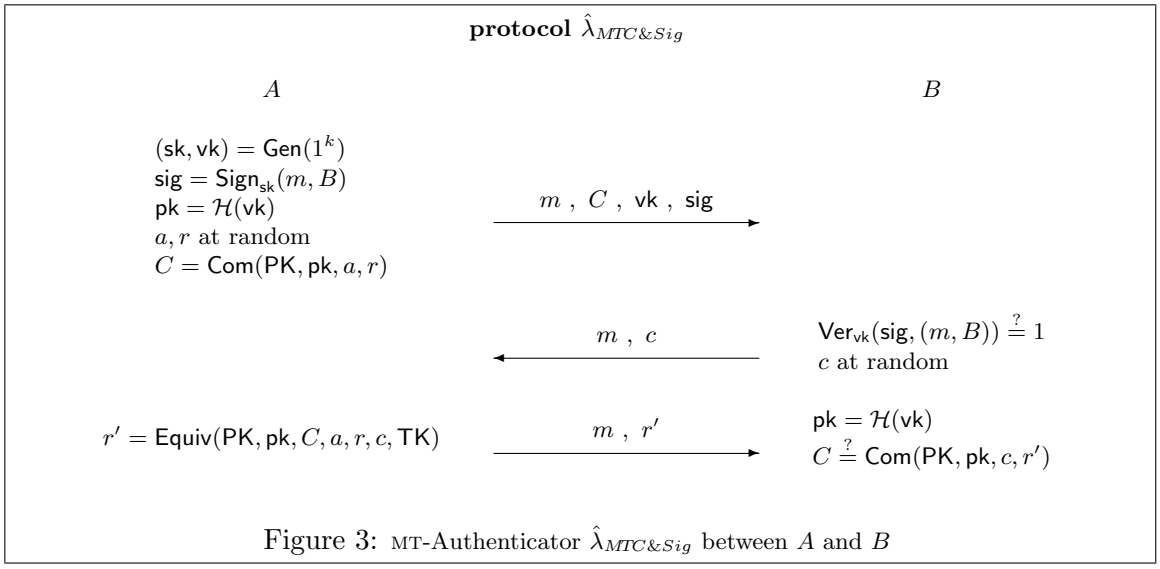

Figure 3: MT-Authenticator $\hat{\lambda}_{MTC\&Sig}$ between A and B

Theorem 8 *If the underlying commitment scheme is a MTC and the one-time signature scheme is secure against chosen message attack, then protocol $\lambda_{MTC\&Sig}$ emulates protocol* MT *in unauthenticated networks.*

PROOF: This proof follows the outline of Theorem 6: supposing that there is a UM-adversary $\mathcal{U}$ that breaks the integrity of the emulation "authenticated world *vs* unauthenticated world" with non negligible probability δ we can build an equivocator $\mathcal{E}$ for the MTC scheme or a forger $\mathcal{F}$ for the one-time signature system.

Let's start with the construction of the equivocator $\mathcal{E}$. He chooses a party A^* among the n parties of the simulation. Let q be a polynomial upper bound to the number of messages that A^* is requested to send. Then $\mathcal{E}$ invokes q times the algorithm Gen for the generation of one-time signature keys obtaining q pairs of keys $(\mathsf{sk}_i, \mathsf{vk}_i)$. Let $\mathcal{H}^*$ be a hash function as described in the specification of $\hat{\lambda}_{MTC\&Sig}$. For each pair $(\mathsf{sk}_i, \mathsf{vk}_i)$ a public key pk_i for a MTC is computed as

$\mathsf{pk}_i = \mathcal{H}^*(\mathsf{vk}_i)$. These q values pk_i are "declared" in the MTC secure binding game so the oracle $\mathcal{EQ}$ can be invoked on them. Now adversary $\mathcal{E}$ receives the master public key PK^* of the MTC family to break.

$\mathcal{E}$ finalizes the setup of the simulator by choosing and distributing keys for the n imitated parties according to function I except for party A^*, for which the pair $(\mathsf{PK}^*, \mathcal{H}^*)$ is used.

In this simulation of the unauthenticated world among parties using protocol $\lambda_{MTC\&Sig}$, $\mathcal{E}$ runs $\mathcal{U}$. If party A^* is invoked to send a message m to a party B, $\mathcal{E}$ draws the first *unused* pair $(\mathsf{sk}_i, \mathsf{vk}_i)$ of one-time signature keys. Thus, for the creation of the first message $\texttt{`message:}m, C\texttt{'}$, we are using the commitment scheme associated with the precomputed public key $\mathsf{pk}_i = \mathcal{H}^*(\mathsf{vk}_i)$. As described in the protocol, $\mathcal{E}$ computes: $\mathsf{sig} = \mathsf{Sign}_{\mathsf{sk}_i}(m, B)$ and $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}_i, a, r)$ with a, r chosen at random. The message sent is $\texttt{`message:}m, C, \mathsf{vk}_i, \mathsf{sig}\texttt{'}$.

When party A^* receives a message $\texttt{`challenge:}m, c\texttt{'}$ from a party B, $\mathcal{E}$ finds the keys $(\mathsf{sk}_i, \mathsf{vk}_i)$ and pk_i used in connection with m. Note that $\mathcal{E}$ is allowed to invoke the oracle $\mathcal{EQ}$ on the key pk_i so it can be used to create the reply to the challenge. Namely, the oracle $\mathcal{EQ}$ is given as input the specific public key pk_i, the commitment C, the first opening (a, r) and the challenge c. It returns randomness r' such that (c, r') is another opening of C, so A^* can reply with $\texttt{`reply:}m, r'\texttt{'}$.

If some party D outputs $\texttt{`}D\ \texttt{received}\ m^*\ \texttt{from}\ A^*\texttt{'}$ and $A^*$ was not activated to send $m^*$ to $D$ (or if $D$ has already output this value before), this means that $\mathcal{U}$ broke party $A^*$ using the message $m^*$. Since $D$ outputs $\texttt{`}D\ \texttt{received}\ m^*\ \texttt{from}\ A^*\texttt{'}$, he collected from A^* a first message $\texttt{`message:}m^*, C, \mathsf{vk}^*, \mathsf{sig}^*\texttt{'}$ and a replying message $\texttt{`reply:}m^*, r'\texttt{'}$ to his challenge c. At this point there are two cases:

- The verification key vk^* used by $\mathcal{U}$ is different than all the verification keys vk_i used by A^*, this means that $\mathcal{U}$ has broken the commitment schemes and to find the pair of openings we can use the rewinding technique seen in Theorem 6. Note that the public key $\mathsf{pk}^* = \mathcal{H}^*(\mathsf{vk}^*)$ is with high probability different than all the public keys for which the oracle $\mathcal{EQ}$ is enabled to answer (this is for the collision-resistance of $\mathcal{H}^*$). Without going into details, the probability of success of $\mathcal{E}$ is the same as in the proof of Theorem 6.

- $\mathsf{vk}^* = \mathsf{vk}_j$ for some j and vk_j was used by A^* to authenticate in an other session a message m' sent to a party B'. For the properties of uniqueness assumed in our model of message-driven protocols, we can deduce that the pair (m, B) is different that (m', B'). This means that $\mathcal{U}$ given the verification key vk_j and a signature on the string (m', B') forged a new signature on a different string (m, B). This breaks the security of the one-time signature system. More specifically given this adversary $\mathcal{U}$ we could build a forger $\mathcal{F}$ for the one-time signature scheme. This construction is really similar to the $\mathcal{E}$'s one of this proof. Starting from a verification key vk^* and a signature on a message of his choice, $\mathcal{F}$ can create a simulation where all the parties have regular public/secret keys. Then he chooses at random a session managed by the simulator where a party A^* send a message m^* to a party B^*. For this session the verification key used is vk^* and the message is signed using the signature oracle. If $\mathcal{U}$ chooses to break a sequent session cloning the verification key vk^* we obtain a valid forger. The probability of success of the forger $\mathcal{F}$ is $\frac{\delta}{l}$ where l is the total number of messages delivered during the simulation.

In both cases we obtained an adversary for one scheme that was supposed to be secure. □

Deniability. As before this protocol is not deniable. But using the same techniques as in Section 3.2.1 it can be made deniable.

3.4 On Simulation Sound Commitments

As introduced in [41] a *Simulation-Sound Commitment* (SSC) scheme is defined as a commitment scheme with a single public key PK, which has a matching trapdoor TK. The commitment algorithm takes as input a message m and a *tag* t, and it outputs $C = \mathsf{Com}(\mathsf{PK}, m, t, r)$ where r is the randomness used. There is also a "fake" committing algorithm that takes as input both the public key and the trapdoor and returns a value $C = \mathsf{FakeCom}(\mathsf{PK}, \mathsf{TK}, m, t, r)$. Fake commitments for a specific tag t can be equivocated, i.e. can be opened as any message efficiently if one knows TK. The crucial properties of SSC's are

- the distribution of fake commitments is indistinguishable from the distribution of real commitments;
- no efficient adversary can create a fake commitment and opening it in two ways for a tag t, even after having access to an oracle that creates fake commitments and arbitrary openings for any tag $t' \neq t$.

We stress that SSCs are *not* necessarily information-theoretic private (indeed the generic construction based on one-way function in [41] is not information-theoretic private, which means that the indistinguishability in the first condition holds in the computational sense); also that the trapdoor may be needed to create and equivocate fake commitments.

Notice that an AMTC is a stronger version of a SSC. Indeed for AMTC (where the tag t is interpreted as a specific public key pk) we have information theoretic security. In particular fake commitments are not needed: the trapdoor information can equivocate real commitments as well (so fake commitments are created identical to real ones).

In [41] there are several constructions of SSCs, one based on one-way functions and the other based on specific number-theoretic assumptions. The number-theoretic constructions are actually AMTCs. But the generic construction based on one-way functions is not an AMTC. Indeed in that scheme, fake commitments are only computationally indistinguishable from real ones. This computational indistinguishability is derived from a reduction to the ZK protocol for Hamiltonicity. More specifically, a fake commitment contains a correct ZK proof for an Hamiltonian graph, while a real commitment contains a simulated proof.

Consider the protocol $\hat{\lambda}_{AMTC}$ when used with SSC. The message being authenticated is mapped into the tag. The value C sent by the sender is computed using the fake commitment algorithm, and then equivocated (both steps require use of the secret key TK). The proof of unforgeability then follows from the fact that during the simulation I can put fake commitments and equivocate them using the oracle, without knowing TK, and an adversary will equivocate a different message and thus a different tag. Indeed the protocol remains a secure authenticator.

The proof of deniability also goes through. Indeed the simulator without knowing TK will compute real commitments which are computational indistinguishable from fake ones, so the transcript is also (computationally) indistinguishable[20].

Forward deniability however cannot be proven. Indeed, the Sender can prove (by revealing how she prepared the fake commitments) that there is a real Hamiltonian cycle contained in the fake commitments.

[20] As in our proof the deniability simulator uses rewinding to create a real commitment to the challenge that the receiver will ask later.

4 A DDH-based MT-authenticator

Here we show a MT-authenticator (later proven deniable) whose security is based on the difficulty of the DDH problem in some groups.

NUMBER THEORY. In the following we denote G_q a cyclic group of prime order q where multiplication and membership test can be perfomed efficiently. An example is to consider two prime numbers p, q such that $q|(p-1)$. Then G_q is the subgroup of Z_p^* of order q. Let g_1, g_2 be two generators[21] for G_q. All computations are in G_q unless otherwise noted.

We are going to assume that the well-known *Decisional Diffie-Hellman Assumption* holds in G_q, namely that the DDH problem is difficult in this group. Consider the following distributions over G_q^4:

$$\mathsf{DDH} = \{(g_1, g_2, g_1^r, g_2^r) \mid r \in_{\mathsf{R}} Z_q\} \ , \ \mathsf{Random} = \{(g_1, g_2, g_1^{r_1}, g_2^{r_2}) \mid r_1, r_2 \in_{\mathsf{R}} Z_q\}$$

The DDH Assumption claims that these two distributions are computationally indistinguishable, in other words no polynomial-time algorithm given as input (g_1, g_2, u_1, u_2) can decide if the input was drawn from DDH or Random.[22]

HASH FUNCTIONS. We shall use two kinds of hash functions. First we will denote with $\mathcal{H}$ a function chosen randomly in a set of *Universal One-way hash functions* (UOWHFs) [43].

Also we consider a hash function $H : G_q \to \{0,1\}^{2k}$, where k is a security parameter, such that 2^{-k} is considered negligible. H must have the following property: the distribution of $H(x)$ when $x \in_{\mathsf{R}} G_q$ should be indistinguishable from the uniform distribution over $\{0,1\}^{2k}$. In this case, we will say that it is a *smooth* hash function. An example of such a function is a function H randomly chosen over a set of 2-universal family [10]: in this case assuming that $|q| > 2k + 2\delta$ we have that the distribution $\{H(x)\}_{x \in_{\mathsf{R}} G_q}$ is $2^{-\delta}$ statistically close to the uniform one over $\{0,1\}^{2k}$. To avoid choosing such a large q, one could use a cryptographic hash function like SHA1, and explicitly assume that the distribution $\{\mathsf{SHA1}(x)\}_{x \in_{\mathsf{R}} G_q}$ is computationally indistinguishable from the uniform one. In the future we shall denote with $\lceil H \rceil(x)$ the first k-bits of $H(x)$ and with $\lfloor H \rfloor(x)$ the remaining k-bits.

THE PROTOCOL. We construct the DDH-based MT-authenticator λ_{DDH}: the choice of the group G_q and of the generators g_1, g_2 can be seen as the first phase of the initialization[23] function I of the protocol λ_{DDH}. To conclude the initialization phase, for each party a pair of keys (PK, SK) is generated as follows. Consider a generic party P_i: random elements $x_1, x_2, y_1, y_2 \in Z_q$ are chosen and the group elements

$$c = g_1^{x_1} g_2^{x_2} \ , \ d = g_1^{y_1} g_2^{y_2}$$

are computed. Further, two hash functions are chosen: a UOWHF $\mathcal{H}$ and a *smooth* one H so that $H : G_q \to \{0,1\}^{2k}$. Finally, the public key of P_i is $PK_i = (c, d, \mathcal{H}, H)$ and the secret key is $SK_i = (x_1, x_2, y_1, y_2)$. The public information is the collection of the information on the underlying group and of all the public keys:

$$I_0 = p, q, g_1, g_2, PK_1, \ldots, PK_n$$

[21] The reciprocal discrete-logs of the two generators must be secret for security reasons.

[22] This formulation is equivalent to several others and in particular, using the substitution $g_1 \to g, g_2 \to g^x, u_1 \to g^y, u_2 \to g^{xy}$, one sees that it's equivalent to distinguish Diffie-Hellman triples (g^x, g^y, g^{xy}) from non-Diffie-Hellman triples (g^x, g^y, g^z).

[23] These values could also be specific to a user's public key, rather than common to all users.

P_i's private information consists of its secret key: $I_i = SK_i$.

Next, when activated, within party P_i and with external request to send message m to party P_j, protocol λ_{DDH} invokes a two-party sub-protocol $\hat{\lambda}_{DDH}$ between P_i and P_j. Since the sub-protocol $\hat{\lambda}_{DDH}$ involves only two parties, we use the names A and B instead of P_i and P_j for simplicity. Going into details: first, A sends '`message:`m' to B (A also outputs 'A `sent message` m `to` B'). Upon receipt of '`message:`m' from A, party B creates a challenge for A as follows: a random $r \in Z_q$ is chosen and then the following values are computed

$$u_1 = g_1^r \ , \ u_2 = g_2^r \ , \ \alpha = \mathcal{H}(m, B) \ , \ v = c^r d^{r\alpha} \ , \ h_1 = \lceil H \rceil(v)$$

the message for A is '`challenge:`m, u_1, u_2, h_1'.[24] Upon receipt of '`challenge:`m, u_1, u_2, h_1' from B, party A:

- checks the validity of the received challenge computing the values $\alpha = \mathcal{H}(m, B)$, $v = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ and checking that $h_1 = \lceil H \rceil(v)$;[25]
- replies to the challenge with '`reply:`m, h_2' where $h_2 = \lfloor H \rfloor(v)$.

Finally, when B receives the message '`reply:`m, h_2' from A he proceeds as follows: if $h_2 = \lfloor H \rfloor(v)$ (using its copy of v) then B accepts m and outputs 'B `received` m `from` A'. Otherwise, B rejects this message and terminates this invocation of $\hat{\lambda}_{DDH}$. A pictorial representation of a complete invocation of $\hat{\lambda}_{DDH}$ for a message m can be seen in figure 4.

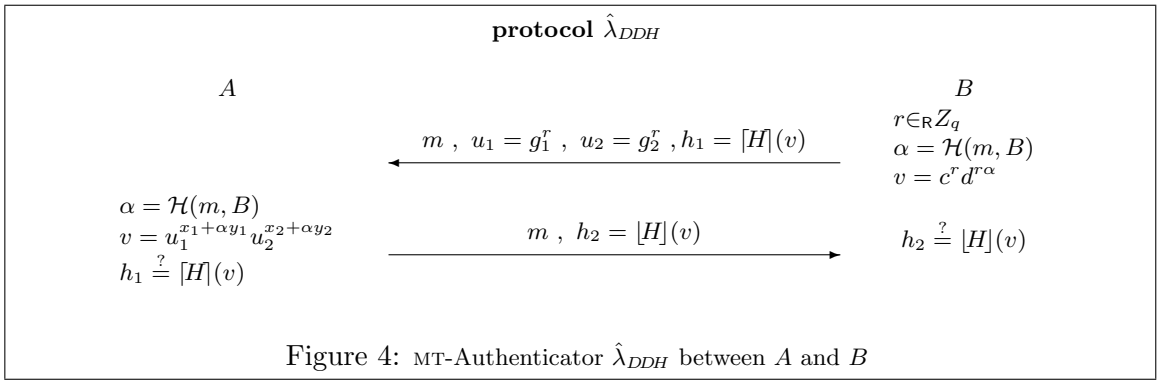

Figure 4: MT-Authenticator $\hat{\lambda}_{DDH}$ between A and B

Theorem 9 *Assume that the DDH assumption holds on the group G_q then protocol λ_{DDH} emulates protocol* MT *in unauthenticated networks.*

PROOF: Up till the definition of event β, the proof is identical to the proof of Theorem 6. We pick the proof from this point, showing that event β occurs only with negligible probability. Assume that event $\mathcal{B}$ occurs with non negligible probability δ. We construct a DDH-distinguisher $\mathcal{D}$ that is capable to recognize DDH-instance with a non negligible probability (related to δ).

CONSTRUCTION OF $\mathcal{D}$. Let $(G_q, g_1, g_2, u_1^*, u_2^*)$ be an instance of the DDH problem, given the adversary $\mathcal{U}$ (that is able to breaks λ_{DDH} with non negligible probability δ) the distinguisher $\mathcal{D}$

[24] Note that the values α, v are computed but not sent, to reduce bandwidth.

[25] It's easy to see that if the challenge is created correctly then A and B compute the same value v.

runs $\mathcal{U}$ on the following simulated interaction with a set of parties running λ_{DDH}. First $\mathcal{D}$ chooses and distributes keys for the imitated parties according to function I with the only exception that the public common information G_q, g_1, g_2 is taken from the DDH-instance. Let A^* be a party chosen at random among the n simulated parties and let

$$PK^* = (c^* = g_1^{x_1^*} g_2^{x_2^*}, d^* = g_1^{y_1^*} g_2^{y_2^*}, \mathcal{H}^*, H^*) \ , \ SK^* = (x_1^*, x_2^*, y_1^*, y_2^*)$$

be its pair of keys, chosen according to the correct distribution. $\mathcal{D}$ chooses at random m^* among all messages m such that some party B was activated with '`message:`m' from A^*.[26] Next, $\mathcal{D}$ continues the simulated interaction. If during the simulation $\mathcal{U}$ asks to corrupt party A^* then the simulation aborts and $\mathcal{D}$ outputs at random '`DDH`' or '`random`'.

If party A^* is activated with a request '`challenge:`m, u_1, u_2, h_1' from B with $m \neq m^*$, then A^* checks the validity of the challenge, by computing $\alpha = \mathcal{H}^*(m, B)$, $v = u_1^{x_1^* + \alpha y_1^*} u_2^{x_2^* + \alpha y_2^*}$ and replies with '`reply:`$m, h_2 = \lfloor H^* \rfloor(v)$'.

When a particular party B^* is activated by $\mathcal{U}$ with incoming '`message:`m^*' then $\mathcal{D}$ computes

$$\alpha^* = \mathcal{H}^*(m^*, B^*) \ , \ v^* = u_1^{*x_1^* + \alpha^* y_1^*} u_2^{*x_2^* + \alpha^* y_2^*} \ , \ h_1^* = \lceil H^* \rceil(v^*)$$

and has B^* respond with '`challenge:`m^*, u_1^*, u_2^*, h_1^*'. We are embedding the instance of the DDH problem into the challenge from B^* for the message m^*.

Next, if A^* is activated with incoming message '`challenge:`m^*, u_1^*, u_2^*, h_1^*' then the simulation aborts and the distinguisher $\mathcal{D}$ outputs at random '`DDH`' or '`random`'.

Finally, if $\mathcal{U}$ activates B^* with incoming message '`reply:`m^*, h_2^*' and $h_2^* = \lfloor H^* \rfloor(v)$ (that is the reply to the challenge is correct), then $\mathcal{U}$ has broken party A^* on the message m^* and the distinguisher $\mathcal{D}$ outputs '`DDH`'. If the simulation terminates normally then $\mathcal{D}$ outputs at random '`DDH`' or '`random`'.

Analysis of $\mathcal{D}$. First, let's highlight what the distinguisher $\mathcal{D}$ was trying to do in its simulation: by choosing A^* and m^* at random it is trying to guess the party that $\mathcal{U}$ will break into and the message that it will use. Indeed the only cases where $\mathcal{D}$ aborts its simulation are when it's clear that he chose the wrong party and/or the wrong message (A^* is corrupted or is activated to reply to the "fake" challenge). If we denote with l the total number of messages that $\mathcal{U}$ delivers in its run, the probability that $\mathcal{D}$ guesses the correct pair (A^*, m^*) is $\frac{1}{l}$.

Now consider the following cases:

- if $(g_1, g_2, u_1^*, u_2^*) \in \mathsf{DDH}$ then we observe that the "forged" message '`challenge:`m^*, u_1^*, u_2^*, h_1^*' is a legitimate challenge, that is $\log_{g_1} u_1^* = \log_{g_2} u_2^*$. It's clear that, considering the view of $\mathcal{U}$, $\mathcal{D}$'s simulation is identical to the real world (also because A^* and m^*, and A^*'s keys, are chosen at random with an uniform distribution in their domains). In this case the distinguisher $\mathcal{D}$ (considering also the case in which it doesn't guess the correct broken pair) outputs '`DDH`' with probability:

$$\mathsf{Prob}(\mathcal{D} = \text{`DDH'}) \quad = \quad \frac{1}{l} \cdot \left(\mathsf{Prob}(\mathcal{B}) \cdot 1 + \mathsf{Prob}(\bar{\mathcal{B}}) \cdot \frac{1}{2} \right) + \left(1 - \frac{1}{l} \right) \cdot \frac{1}{2}$$

[26] This can be done because if the total number l^* of messages that A^* sends and $\mathcal{U}$ delivers is known in advance then $\mathcal{D}$ simply choose the i^{th} message (with $i \in_R [1 \ldots l^*]$). Otherwise, $\mathcal{D}$ chooses m^* in the following way: whenever some party B was activated with '`message:`m' from A^*, distinguisher $\mathcal{D}$ decides to choose $m^* = m$ with an appropriate probability, making sure that by the end of the run all the candidate messages have equal probability to be chosen.

$$
\begin{aligned}
&= \frac{1}{l} \cdot \left(\delta + (1-\delta) \cdot \frac{1}{2}\right) + \left(1 - \frac{1}{l}\right) \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{\delta}{2l}
\end{aligned}
$$

and then:

$$\mathsf{Prob}(\mathcal{D} = \texttt{`random'}) = \frac{1}{2} - \frac{\delta}{2l}$$

where $\frac{\delta}{2l}$ is a non negligible factor.

- if $(g_1, g_2, u_1^*, u_2^*) \in \mathsf{Random}$ we prove in the following Lemma 10 that the distinguisher $\mathcal{D}$ says `'DDH'` with a probability equal to $1/2$ plus a negligible quantity that here we denote with τ, so that:

$$
\begin{aligned}
\mathsf{Prob}(\mathcal{D} = \texttt{`DDH'}) &= \frac{1}{2} + \tau \\
\mathsf{Prob}(\mathcal{D} = \texttt{`random'}) &= \frac{1}{2} - \tau
\end{aligned}
$$

Concluding, the distinguisher $\mathcal{D}$ solves the DDH problem with the non negligible advantage of $\left(\frac{\delta}{2l} - \tau\right)$.

□

Remark: Note that in this proof the distinguisher $\mathcal{D}$ has to guess not only the party broken by $\mathcal{U}$, but also message: in other words, the broken session among l total sessions. This leads to a less efficient reduction (with respect to Theorems 6 and 8) with a factor of $\frac{1}{l}$ (instead of $\frac{1}{n}$).

Lemma 10 *Suppose that* $(g_1, g_2, u_1^*, u_2^*) \in \mathsf{Random}$. *Then, the distinguisher* $\mathcal{D}$ *outputs* `'DDH'` *with probability equal to* $1/2$ *plus a negligible quantity.*

Proof: The distinguisher $\mathcal{D}$ says `'DDH'` always if $\mathcal{U}$ breaks into A^* with message m^* and with probability $1/2$ if $\mathcal{D}$ doesn't guess the correct pair (A^*, m^*) or if $\mathcal{U}$ does not break into anyone.

To create a correct reply for the challenge `'challenge:`m^*, u_1^*, u_2^*, h_1^*`'` $\mathcal{U}$ must guess v^* and then compute $h_2^* = \lfloor H^* \rfloor(v^*)$ or guess directly the value h_2^*.

Let's consider the $\mathcal{U}$'s view on the point $(x_1^*, x_2^*, y_1^*, y_2^*)$, it is the secret key of the party A^*. Considering the information given by his public key, the point satisfies the two equations:

$$\log c^* = x_1^* + \lambda x_2^* \tag{2}$$

$$\log d^* = y_1^* + \lambda y_2^* \tag{3}$$

where 'log' indicates the discrete-logarithm in base g_1 and $\lambda = \log_{g_1} g_2$.

Consider the challenges `'challenge:`m', u_1', u_2', h_1'`'` that $\mathcal{U}$ submits to A^* as party B' during the simulation. We say that such a challenge is *valid* if $r' = \log_{g_1} u_1' = \log_{g_2} u_2'$. Let $v' = H^*((cd^{\alpha'})^{r'})$, where $\alpha' = \mathcal{H}(m', B')$. If A^* answers a valid challenge with $h_2' = \lfloor H^* \rfloor(v')$, this is just giving an extra equation

$$\log v' = r'(x_1^* + \alpha' y_1^*) + r'\lambda(x_2^* + \alpha' y_2^*)$$

which is linearly dependent on Equations 2 and 3. Thus $\mathcal{U}$'s view of the point $(x_1^*, x_2^*, y_1^*, y_2^*)$ remains the same. In Lemma 11 below we show that A^* answers invalid challenges only with negligible probability. Thus for the rest of the proof of this Lemma, we assume A^* never answers such challenges.

Since $(g_1, g_2, u_1^*, u_2^*) \in \mathsf{Random}$ we can suppose that $r_1^* \neq r_2^*$, where $r_1^* = \log_{g_1} u_1^*$ and $r_2^* = \log_{g_2} u_2^*$ (they are equal only with probability $\frac{1}{q}$). Inserting the message '$\texttt{challenge:}m^*, u_1^*, u_2^*, h_1^*$' into the simulation, $\mathcal{D}$ is given some information about the secret point, in particular that the point satisfies also the equation:

$$\log v^* = r_1^*(x_1 + \alpha^* y_1) + r_2^* \lambda (x_2 + \alpha^* y_2) \tag{4}$$

It's clear that equations 2, 3 and 4 are linearly independent. This means that v^* can take any value and it is uniformly distributed over G_q. We are not exposing the value v^* but $\mathcal{U}$ sees $h_1^* = \lceil H^* \rceil(v^*)$ that gives information about v^*. Because of the properties of H^*, the value $H^*(v^*)$ is uniformly distributed over $\{0,1\}^{2k}$ thus seeing the first half, does not help in predicting the second half. Thus $\mathcal{U}$ can guess h_2^* only with negligible probability 2^{-k}. □

Lemma 11 *Even after presenting* '$\texttt{challenge:}m^*, u_1^*, u_2^*, h_1^*$' *to* $\mathcal{U}$, A^* *answers invalid challenges only with negligible probability.*

Proof: Let '$\texttt{challenge:}m', u_1', u_2', h_1'$' be an invalid challenge presented by $\mathcal{U}$ to A^*. Since all the messages that come from a party must be all different it follows that: m' is different than m^* or this challenge comes from a party $B' \neq B^*$. Recall that $\alpha' = \mathcal{H}^*(m', B')$ and $\alpha^* = \mathcal{H}^*(m^*, B^*)$. At the end of the proof we prove that if the event $\alpha' = \alpha^*$ happens with non-negligible probability, then the function $\mathcal{H}^*$ is not a secure hash function. For now, we assume that $\alpha' \neq \alpha^*$. Consider the equation:

$$\log v' = r_1'(x_1 + \alpha' y_1) + r_2' \lambda (x_2 + \alpha' y_2) \tag{5}$$

where $r_1' = \log_{g_1} u_1'$ and $r_2' = \log_{g_2} u_2'$ and $r_1' \neq r_2'$ (because this challenge is invalid) and $v' = (u_1')^{x_1^* + \alpha' y_1^*} (u_2')^{x_2^* + \alpha' y_2^*}$. Since $\alpha' \neq \alpha^*$ and $r_1' \neq r_2'$, it is simple to verify that equations 2, 3, 4 and 5 are linearly independent. This means that v' is uniformly distributed over G_q and that is independent from v^*, and thus $h_1' = \lceil H' \rceil(v')$ is independent from $h_1^* = \lceil H^* \rceil(v^*)$. In other words, the view of $\mathcal{U}$ up to this point, does not help $\mathcal{U}$ in any way to find the correct h_1'. Thus the probability that A^* accepts is the probability that $\mathcal{U}$ guesses h_1' which is negligible.

Bouding the probability that $\alpha' = \alpha^*$. If our hash function $\mathcal{H}^*$ were (fully) collision-resistant, we could conclude immediately that $\alpha' = \mathcal{H}^*(m', B')$ is equal to α^* only with negligible probability, since their inputs are different.

The same conclusion however can be reached using the weaker assumption that $\mathcal{H}^*$ is only a UOWHF[27]. If $\alpha' = \alpha^*$ with non-negligible probability, the adversary $\mathcal{U}$ can be used to break the UOWHF. We modify the simulation: when $\mathcal{D}$ sends '$\texttt{challenge:}m^*, u_1^*, u_2^*, h_1^*$' to A^* (the broken party), it uses $\alpha = \mathcal{H}^*(\hat{m}, B^*)$ instead of $\alpha = \mathcal{H}^*(m^*, B^*)$ where $\hat{m}$ is a random message. Up until the time that a collision occurs, $\mathcal{U}$'s view is statistically indistinguishable from the view in the original simulation, so $\mathcal{U}$ will find a collision with the same probability as before. In this case the choice of $(\hat{m}, B^*)$ is independent from $\mathcal{H}^*$ (we could fix the pair $(\hat{m}, B^*)$ *before* choosing $\mathcal{H}^*$). □

[27] This argument appears already in the original Cramer-Shoup paper [12] and is due to Moni Naor.

4.1 Deniability for λ_{DDH}

As in the previous case, protocol λ_{DDH} can be proven to be a deniable MT-authenticator if the receiver is honest. Indeed the simulator can be defined by choosing at random $r \in_R Z_q$ and computing the appropriate values, like an honest receiver would.

The case of a dishonest receiver is more complicated. Here we do not have an attack, but a black box simulation of the receiver fails (once the dishonest simulator sends us '`challenge:`m, u_1, u_2, h_1', how are we going to simulate the answer h_2?).

We introduce a challenge-response mechanism where A will commit to the answer h_2 and then reveal it only after B shows that he knows h_2 as well. However at this point it is redundant to split $H(v)$ in two pieces and we can just use v alone.

The protocol appears in Figure 5. We assume that A's public key includes an unconditionally binding commitment scheme COM (i.e. a commitment scheme that can be opened in only one way even if you have infinite computing power, but on the other hand its secrecy is computational).

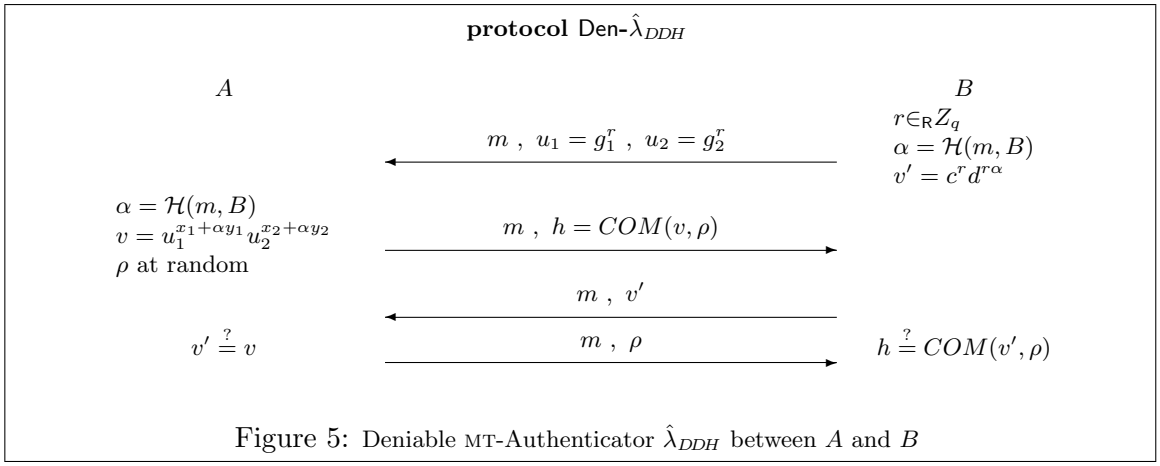

Figure 5: Deniable MT-Authenticator $\hat{\lambda}_{DDH}$ between A and B

Theorem 12 *Protocol* Den-λ_{DDH} *is a forward deniable authenticator if used sequentially.*

PROOF: Let us first check that Den-λ_{DDH} is still a secure authenticator. The proof of Theorem 9 can be carried out with minor modification. Lemma 11 continues to hold as in order for $\mathcal{U}$ to have an invalid ciphertext accepted it must guess v. However in this case $\mathcal{U}$ is given some "help" by seeing $h = COM(v, \rho)$. Thus the proof of Lemma 11 holds computationally (rather than unconditionally) under the security of COM.

The deniability simulator works again by rewinding. After getting '`challenge:`m, u_1, u_2' from the receiver, we commit to a random value, and wait for the receiver to reveal v. The receiver's probability of revealing v must remain close to the one of the real protocol (where the real v is committed by the sender) otherwise we can break COM (this is basically the technique from [26]). Once we see v we rewind the receiver and place a correct commitment to v on the previous round and complete the simulation.

The simulated transcript is perfectly indistinguishable from a real one, so the forward deniability holds. □

Remark: *The first step is malleable.* The above protocol shows how far we have gone from the CCA paradigm. We can think of u_1, u_2 as the "encryption" of a key, and of v as the MAC of the message. But if we do that, then we do not know how to prove the encryption step to be CCA-secure.

Remarks: *Concurrent Executions.* As in the previous section, the unforgeability property of this authenticator holds in a concurrent setting (as in all the authenticators in the CCA paradigm). If we use timing assumptions to force only a logarithmic number of executions to be open at any time, we achieve deniability in the concurrent setting as well.

4.2 Generalizing to Projective Hash Functions

The previous scheme can be generalized to use projective hash functions, a tool introduced by Cramer and Shoup in [14] as a generalization of their previous CCA-secure encryption scheme based on DDH [12].

We briefly recall the notion of projective hash functions families, with some slight change of terminology to fit it to our scenario. Let X be a set and $L \subset X$ an NP-language. Let Y be an arbitrary set. We consider a family of hash function PH_k that maps $X \times Y$ in some set Z where k ranges in some key space K. We assume that PH_k is efficiently computable, i.e. there is some efficient algorithm that computes $PH_k(x, y)$ when given k and x, y.

We say that this family is projective if there exists a projection key that allows to compute PH_k over the subset $L \times Y$ even without knowing k. That is, there exists a projection function $\mu(\cdot)$ that maps keys k into their projections $s = \mu(k)$, such that there is an efficient algorithm that given only $s = \mu(k)$, $y \in Y$, $x \in L$ and w a witness of $x \in L$ (which exists because L is in NP), computes $PH_k(x, y)$. Notice that this algorithm is *not* given k but just $s = \mu(k)$.

An *ϵ-2universal* projective hash function has the additional property that for $x \notin L$, the projection key s actually says *very little* about the value of $PH_k(x, y)$, even after seeing $PH_k(x', y')$ for a $y' \neq y$. More specifically, for every x, x', y, y', z, z', such that $y \neq y'$ we have

$$Prob_k[PH_k(x, y) = z \mid s = \mu(k) \text{ and } PH_k(x', y') = z'] \leq \epsilon$$

In [14] *ϵ-2universal* projective hash function families are constructed under the DDH, Quadratic Residuosity and Higher Residuosity Assumptions.

We now show how the previous scheme can be thought in terms of projective hash functions. The set X is $G_q \times G_q$. The set Y is $\mathcal{M} \times \mathcal{ID}$, where $\mathcal{M}$ is the message space, and $\mathcal{ID}$ is the set of possible user's IDs. The NP-language L is the set of pairs $(u_1 = g_1^r, u_2 = g_2^r)$ for $r \in Z_q$. This is an NP language and the witness for such an element is r. The key space of the hash function is defined as Z_q^4 and thus a key k is a tuple $k = (x_1, x_2, y_1, y_2)$.

$$PH_{x_1,x_2,y_1,y_2}(u_1, u_2, m, B) = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$$

where $\alpha = \mathcal{H}(m, B)$. The projection of the key $k = (x_1, x_2, y_1, y_2)$ are the values $c = g_1^{x_1} g_2^{x_2}$ and $d = g^{y_1} g_2^{y_2}$. Clearly given c, d if (u_1, u_2) is in L we can compute

$$PH_{x_1,x_2,y_1,y_2}(u_1, u_2, m, B) = (cd^{\alpha})^r$$

The proof of the protocol basically contains a proof that thanks to the collision resistance of $\mathcal{H}$, the family PH is *ϵ-2universal*.

The last property we need to construct the protocol is that pairs (x, w) can be efficiently sampled. We denote with L' the set of pairs (x, w) where w is a witness of x.

The protocol above can be generalized as follows. The public key of A is a description of X and L, plus $s = \mu(k)$, while the secret key is k. The protocol is described in Figure 6. Notice that A can compute PH_k as she knows k, while B can compute $PH_k(x, m, B)$ because he sampled x together with w and he knows the projection.

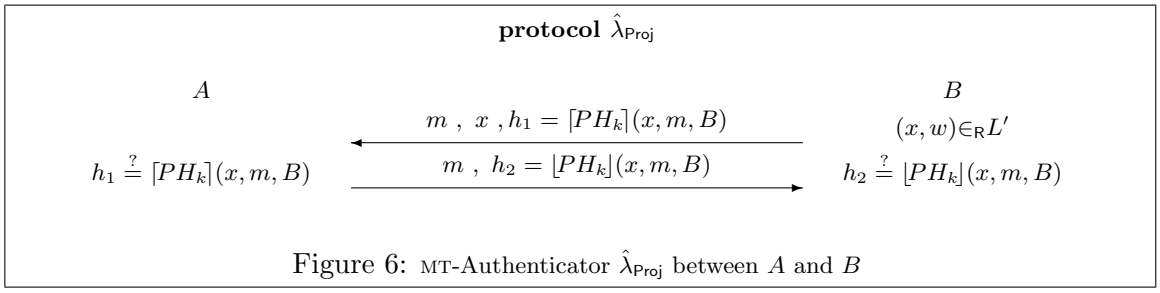

Figure 6: MT-Authenticator $\hat{\lambda}_{\mathsf{Proj}}$ between A and B

Remark: In the Kurosawa-Desmedt encryption scheme [39] a *strongly smooth* projective hash function must be used. In this case the output of the function is required to be uniform not just very unpredictable. This is clearly a stronger property. In [39] this is needed as the output of the function is used as a key to encrypt and MAC a message. In our case, basically, we are using the output of the function directly as a MAC thus good unpredictability is sufficient.

The security of $\hat{\lambda}_{\mathsf{Proj}}$ can be proven by generalizing the proof for $\hat{\lambda}_{DDH}$, as follow:

Theorem 13 *Assume that $L \subset X$ is an hard on the average NP-language and that PH_k is a ϵ-*2universal *projective hash function over X, L (with ϵ negligible), then protocol λ_{Proj} emulates protocol* MT *in unauthenticated networks.*

PROOF: Up till the definition of event β, the proof is identical to the proofs of Theorem 6 and Theorem 9. We pick the proof from this point, showing that event β occurs only with negligible probability. Assume that event $\mathcal{B}$ occurs with non negligible probability δ. We construct an Adversary $\mathcal{D}$ that, given a random element $x^* \in X$, is able to decide if x^* is in the language or not with a non negligible probability (related to δ).

CONSTRUCTION OF $\mathcal{D}$. Given the adversary $\mathcal{U}$ (that is able to break λ_{Proj} with non negligible probability δ) the distinguisher $\mathcal{D}$ runs $\mathcal{U}$ on the following simulated interaction with a set of parties running λ_{Proj}. First $\mathcal{D}$ chooses and distributes keys for the imitated parties according to the protocol λ_{Proj} using the sets L and X. Let A^* be a party chosen at random among the n simulated parties and let k^* his secret key (chosen at random) and $(s^* = \mu(k^*), X, L)$ his public key.

As in the previous proofs, $\mathcal{D}$ chooses at random m^* among all messages m such that some party B was activated with `'message:`m`'` from A^*. If during the simulation $\mathcal{U}$ asks to corrupt party A^* then the simulation aborts and $\mathcal{D}$ outputs at random `'inLanguage'` or `'outLanguage'`.

If party A^* is activated with a request `'challenge:`m, x, h_1`'` from B with $m \neq m^*$, then A^* computes $PH_k(x, m, B)$ using k, so he can check the validity of challenge and reply with `'reply:`$m, h_2 = \lfloor PH^* \rfloor(x, m, B)$`'`.

When a particular party B^* is activated by $\mathcal{U}$ with incoming '`message:`m^*' then $\mathcal{D}$ computes $h_1^* = \lceil PH_{k^*} \rceil (x^*, m^*, B^*)$ using k^* (and not the projection s^* and the witness w^* that $\mathcal{D}$ doesn't know). B^* responds with '`challenge:`m^*, x^*, h_1^*'.

Next, if A^* is activated with incoming message '`challenge:`m^*, x^*, h_1^*' then the simulation aborts and the distinguisher $\mathcal{D}$ outputs at random '`inLanguage`' or '`outLanguage`'.

Finally, if $\mathcal{U}$ activates B^* with incoming message '`reply:`m^*, h_2^*' and $h_2^* = \lfloor PH_{k^*} \rfloor (x^*, m^*, B^*)$ (that is the reply to the challenge is correct), then $\mathcal{U}$ has broken party A^* on the message m^* and the distinguisher $\mathcal{D}$ outputs '`inLanguage`'. If the simulation terminates normally then $\mathcal{D}$ outputs at random '`inLanguage`' or '`outLanguage`'.

Analysis of $\mathcal{D}$. $\mathcal{D}$ tries to guess the party that $\mathcal{U}$ will break and the message that it will use: A^* and m^*. If we denotes with l the total number of the messages that $\mathcal{U}$ delivers in its run, the probability that $\mathcal{D}$ guesses the correct pair (A^*, m^*) is $\frac{1}{l}$.

Now consider the following cases:

- if $x^* \in L$ then we observe that the "forged" message '`challenge:`m^*, h_1^*' is a legitimate challenge. It's clear that, considering the view of $\mathcal{U}$, $\mathcal{D}$'s simulation is identical to the real world. As in the previous Theorem, in this case the distinguisher $\mathcal{D}$ (considering also the case in which it doesn't guess the correct broken pair) outputs '`inLanguage`' with probability:

$$\begin{aligned} \mathsf{Prob}(\mathcal{D} = \text{`inLanguage'}) &= \frac{1}{l} \cdot \left(\mathsf{Prob}(\mathcal{B}) \cdot 1 + \mathsf{Prob}(\bar{\mathcal{B}}) \cdot \frac{1}{2} \right) + \left(1 - \frac{1}{l} \right) \cdot \frac{1}{2} \\ &= \frac{1}{2} + \frac{\delta}{2l} \end{aligned}$$

  and then:

$$\mathsf{Prob}(\mathcal{D} = \text{`outLanguage'}) = \frac{1}{2} - \frac{\delta}{2l}$$

  where $\frac{\delta}{2l}$ is a non negligible factor.

- if $x^* \notin L$ we prove in the following Lemma 14 that the distinguisher $\mathcal{D}$ says '`inLanguage`' with a probability equals to $1/2$ plus a negligible quantity that here we denotes with τ, so that:

$$\begin{aligned} \mathsf{Prob}(\mathcal{D} = \text{`inLanguage'}) &= \frac{1}{2} + \tau \\ \mathsf{Prob}(\mathcal{D} = \text{`outLanguage'}) &= \frac{1}{2} - \tau \end{aligned}$$

Concluding, the distinguisher $\mathcal{D}$ solves the membership problem related to L with the non negligible advantage of $\left(\frac{\delta}{2l} - \tau \right)$.

□

Lemma 14 *Suppose that $x^* \notin L$, the distinguisher $\mathcal{D}$ outputs* '`inLanguage`' *with a probability equals to $1/2$ plus a negligible quantity.*

PROOF: The distinguisher $\mathcal{D}$ says '`inLanguage`' always if $\mathcal{U}$ breaks in A^* with message m^* and with probability 1/2 if $\mathcal{D}$ doesn't guess the correct pair (A^*, m^*) or if $\mathcal{U}$ doesn't break anyone.

To create a correct reply for the challenge '`challenge:`m^*, x^*, h_1^*' $\mathcal{U}$ must guess the value $PH_{k^*}(x^*, m^*, B^*)$ or, at least, the second half of it.

When A^* replies to challenges coming from other parties B', he exposes values like $PH_{k^*}(x', m', B')$, with $x' \neq x$. Thanks to the characteristic property of *ϵ-2universal* projective hash functions, the probability to guess the target hash value is negligible for $\mathcal{U}$.

When $\mathcal{D}$ inserts in the simulation the message '`challenge:`$m^*, x^*, h_1^* = [PH_{k^*}](x^*, m^*, B^*)$', this value doesn't help $\mathcal{U}$ in his task for the uniform distribution of $PH_{k^*}(x^*, m^*, B^*)$. Thus $\mathcal{U}$ can guess h_2^* only with negligible probability. □

Deniability: as before, the protocol can be proven deniable only against an honest verifier. Against a malicious verifier, as before, we need an extra challenge response mechanism (removing the need to split $PH_k(x, m, B)$ in two parts): the final protocol is shown in Figure 7.

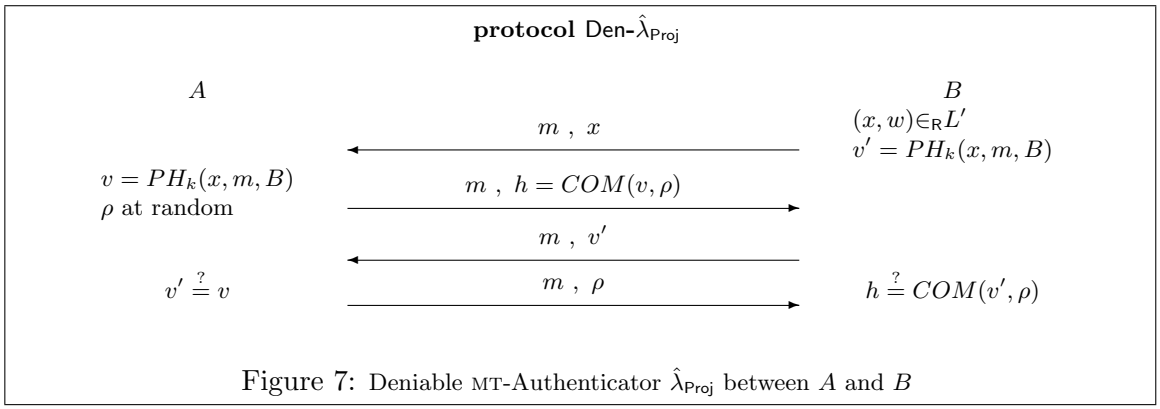

Figure 7: Deniable MT-Authenticator $\hat{\lambda}_{\mathsf{Proj}}$ between A and B

This variant of the protocol is a *forward deniable authenticator* and the proof follows the logic of Theorem 12. The proof still holds if the protocol is used sequentially or in a concurrent setting, using timing assumptions to force a logarithmic number of executions to be open at any time.

References

[1] Y. Aumann and M. Rabin, Authentication, enhanced security and error correcting codes, *Advances in Cryptology – proc. of CRYPTO '98, LNCS 1462*, Springer-Verlag, pp. 299–303, 1998.

[2] Y. Aumann and M. Rabin, Efficient deniable authentication of long messages, in *International Conference on Theoretical Computer Science* in honor of Professor Manuel Blum's 60^{th} birthday, April 20-24, 1998. Available from: `http://www.cs.cityu.edu.hk/dept/video.html`.

[3] N. Barić, and B. Pfitzmann, *Collision-free Accumulators and Fail-stop Signature Schemes without Trees*, *Advances in Cryptology – proceedings of EUROCRYPT '97, LNCS 1233*, Springer-Verlag, pp. 480–494, 1997.

[4] M. Bellare, R. Canetti and H. Krawczyk, A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols, *proc. of* 30^{th} *Symposium on Theory of Computing (STOC)*, ACM, pp. 419–428, 1998.

[5] M. Bellare and A. Palacio, The Knowledge of Exponent Assumption and 3-Rounds Zero-Knowledge Protocols, *Advances in Cryptology – proc. of CRYPTO '04, LNCS 3152*, Springer-Verlag, pp. 273–289, 2004.

[6] D. Boneh and X. Boyen, Short Signatures Without Random Oracles, *Advances in Cryptology – proc. of EUROCRYPT '04, LNCS 3027*, Springer-Verlag, pp. 56–73, 2004.

[7] G. Brassard, D. Chaum and C. Cre'peau, Minimum Disclosure Proofs of Knowledge, *Journal of Computer and System Sciences*, vol. 37, n. 2, pp. 156–189, 1988.

[8] R. Canetti, U. Feige, O. Goldreich and M. Naor, Adaptively Secure Multi-Party Computation, *proc. of* 28^{th} *Symposium on Theory of Computing (STOC)*, ACM, pp. 639–648, 1996.

[9] R. Canetti, J. Kilian, E. Petrank and A. Rosen, Black-Box Concurrent Zero-Knowledge Requires (Almost) Logarithmically Many Rounds, *SIAM Journal on Computing* , vol. 32, n. 1, pp. 1–47, 2002.

[10] L. Carter and M.N. Wegman, Universal Classes of Hash Functions, *Journal of Computer and System Sciences*, vol. 18, n. 2, pp. 143–154, 1979.

[11] R. Cramer and I. Damgard. New Generation of Secure and Practical RSA-based Signatures, *Advances in Cryptology – proceedings of CRYPTO '96, LNCS 1109*, Springer-Verlag, pp. 173–185, 1996.

[12] R. Cramer and V.Shoup, A practical public-key cryptosystem secure against adaptive chosen ciphertexts attacks, *Advances in Cryptology – proc. of CRYPTO '98, LNCS 1462*, Springer-Verlag, pp. 13–25, 1998.

[13] R. Cramer and V. Shoup, Signature Scheme based based on the Strong RSA assumption, *proc. of* 6^{th} *ACM Conference of Computer and Communication Security*, 1999.

[14] R. Cramer and V.Shoup, Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption, *Advances in Cryptology – proc. of EUROCRYPT '02, LNCS 2332*, Springer-Verlag, pp. 45–64, 2002.

[15] I. Damgård, J. Groth. *Non-interactive and reusable non-malleable commitment schemes.* Proc. of 35^{th} ACM Symp. on Theory of Computing (STOC'03), pp.426-437, 2003.

[16] G. Di Crescenzo, Y. Ishai, R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. Proc. of 30^{th} ACM Symp. on Theory of Computing (STOC'98), pp.141–150, 1998.

[17] G.Di Crescenzo, J. Katz, R. Ostrovsky, A. Smith. *Efficient and Non-interactive Non-malleable Commitment.* Proc. of EUROCRYPT 2001, Springer LNCS 2045, pp.40-59.

[18] W. Diffie and M.E. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. 22, n. 6, pp. 644–654, 1976.

[19] M. Di Raimondo, R. Gennaro, and H. Krawczyk, Deniable Authentication and Key Exchange. *Manuscript.*

[20] D. Dolev, C. Dwork, and M. Naor, Non-Malleable cryptography, *SIAM Journal on Computing*, vol. 30, n. 2, pp. 391–437, April 2000.

[21] C. Dwork, M. Naor and A. Sahai, Concurrent Zero-Knowledge. *J. ACM 51(6): 851-898 (2004).* Preliminary version in STOC'98.

[22] J. Garay, P. MacKenzie and K. Yang, Strengthening Zero-Knowledge Protocols Using Signatures, *Advances in Cryptology – proc. of EUROCRYPT '03, LNCS 2656*, Springer-Verlag, pp. 177–194, 2003.

[23] R. Gennaro, Multi-trapdoor Commitments and their Applications to Proofs of Knowledge Secure under Concurrent Man-in-the-middle Attacks, *Advances in Cryptology – proc. of CRYPTO '04, LNCS 3152*, Springer-Verlag, 2004.

[24] R. Gennaro, S. Halevi and T. Rabin, Secure Hash-and-Sign Signatures without the Random Oracle, *Advances in Cryptology – proc. of EUROCRYPT '99, LNCS 1592*, Springer-Verlag, pp. 123–139, 1999.

[25] R. Gennaro and V. Shoup, A Note on An Encryption Scheme of Kurosawa and Desmedt, `http://eprint.iacr.org/2004/194/`.

[26] O. Goldreich and A. Kahan, How to Construct Constant-Round Zero-Knowledge Proof Systems for NP, *Journal of Cryptology*, vol. 9, n. 3, pp. 167–190, 1996.

[27] S. Goldwasser and S. Micali, Probabilistic Encryption, *Journal of Computer and System Sciences*, vol. 28, n. 2, pp. 270–299, 1984.

[28] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof-systems, *SIAM Journal on Computing*, vol. 18, n. 1, pp. 186–208, February 1989.

[29] S. Goldwasser, S. Micali and R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, *SIAM Journal on Computing*, vol. 17, n. 2, pp. 281–308, April 1988.

[30] O. Goldreich, S. Micali and A. Wigderson, Proofs that yield nothing but their validity or all languages in NP have Zero-Knowledge Proof Systems, *in proc. of* 27^{th} *IEEE Annual Symposium on the Foundations of Computer Science*, vol. 38, n. 1, pp. 691–729, July 1991.

[31] P. Gutman, Secure Deletion of Data from Magnetic and Solid-State Memory, *Sixth USENIX Security Symposium Proceedings, San Jose, California*, July 22-25, 1996.

[32] S. Hada and T. Tanaka, On the Existence of 3-Round Zero-Knowledge Protocols, *Advances in Cryptology – proc. of CRYPTO '98, LNCS 1462*, Springer-Verlag, pp. 408–423, 1998.

[33] D. Harkins and D. Carrel, ed., "The Internet Key Exchange (IKE)", *RFC 2409,* Nov. 1998.

[34] M. Jakobsson, K. Sako and R. Impagliazzo, Designated Verifier Proofs and Their Applications, *Advances in Cryptology – proc. of EUROCRYPT '96, LNCS 1070*, Springer-Verlag, pp. 143–154, 1996.

[35] J. Katz, Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications, *Advances in Cryptology – proc. of EUROCRYPT '03, LNCS 2656*, Springer-Verlag, pp. 211–228, 2003.

[36] H. Krawczyk. SKEME: a versatile secure key exchange mechanism for Internet. 1996 IEEE Symposium on Network and Distributed System Security (SNDSS '96).

[37] H. Krawczyk, SIGMA: The 'SiGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols, *Advances in Cryptology – proc. of CRYPTO '03, LNCS 2729*, Springer-Verlag, pp. 400–425, 2003. Available at `http://www.research.ibm.com/security/sigma.ps`

[38] H. Krawczyk and T. Rabin, Chameleon Hashing and Signatures, *proc. of Network and Distributed Systems Security Symposium (NDSS) 2000*, Internet Society, pp. 143–154, 2000.

[39] K. Kurosawa and Y. Desmedt, A New Paradigm of Hybrid Encryption Scheme, *Advances in Cryptology – proc. of CRYPTO '04, LNCS 3152*, Springer-Verlag, pp. 426–442, 2004.

[40] L. Lamport, *Constructing Digital Signatures from a One-Way Function*, Technical Report SRI Intl. CSL 98, 1979.

[41] P. MacKenzie and K. Yang, On Simulation-Sound Commitments, *Advances in Cryptology – proc. of EUROCRYPT '04, LNCS 3027*, Springer-Verlag, pp. 382–400, 2004.

[42] W. Mao and K.G. Paterson. *On the plausible deniability feature of Internet protocols.* Manuscript.

[43] M. Naor and M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks, *proc. of* 22^{nd} *Symposium on Theory of Computing (STOC)*, ACM, pp. 427–437, 1990.

[44] M. Naor, Deniable Ring Authentication, *Advances in Cryptology – proc. of CRYPTO '02, LNCS 2442*, Springer-Verlag, pp. 481–498, 2002.

[45] R. Pass, On Deniability in the Common Reference String and Random Oracle Model, *Advances in Cryptology – proc. of CRYPTO '03, LNCS 2729*, Springer-Verlag, pp. 316–337, 2003.

[46] R. Rivest, A. Shamir and L. Adelman, A Method for Obtaining Digital Signature and Public Key Cryptosystems, *Communications of the ACM*, vol. 21, n. 2, pp. 120–126, 1978.

[47] R. Rivest, A. Shamir and Y. Tauman, How to Leak a Secret, *Advances in Cryptology – proc. of ASIACRYPT '01, LNCS 2248*, Springer-Verlag, pp. 552–565, 2001.

[48] A. Shamir, On the Generation of Cryptographically Strong Pseudorandom Sequences, *ACM Transactions on Computer Systems (TOCS)*, ACM Press, vol. 1, n. 1, pp. 38–44, 1983.

[49] V. Shoup, Using hash functions as a hedge against chosen ciphertext attack, *Advances in Cryptology – proc. of EUROCRYPT '00, LNCS 1807*, Springer-Verlag, pp. 275–288, 2000.

A Alternative model for the analysis

Here we briefly describe the model for the analysis of deniable authentication protocol adopted in [21, 35]. We also show that the model we use (from [4]) is not weaker than the one presented here. That is, we show that a protocol satisfying our definition of deniable authenticator is valid in their model.

MODEL. There are a prover (or authenticator) P and a verifier V. P publishes a public key PK that V knows. V concurrently interacts with P in many sessions. For each session, V following the execution of an authentication protocol decides whether to accept or reject the session as an authentication of a message m. Consider an adversary $\mathcal{M}$ that controls the communication of several copies of a prover P who are not aware of each other and control the verifiers with whom they interact.

The protocol is a *deniable authentication protocol* if it satisfies:

Completeness For any message m, if the prover and verifier follow the protocol for authenticating the message m, then the verifier accepts.

Soundness Suppose that the copies of P are willing to concurrently authenticate any polynomial number of messages $m_1, m_2, \ldots$, which may be chosen adaptively by the adversary $\mathcal{M}$. We say that $\mathcal{M}$ successfully attacks the scheme if a forger C, under control of $\mathcal{M}$ and pretending to be P, succeeds in authenticating to a third party D (running the verifier's V protocol) a message $m \neq m_i, i = 1, 2, \ldots$.

We say that the scheme is unforgeable if the probability that $\mathcal{M}$ successfully attacks it is negligible.

Strong Deniability Consider an adversary $\mathcal{M}$ as above and suppose that the copies of P are willing to concurrently authenticate any polynomial number of messages. Then for each $\mathcal{M}$ there exists a polynomial time simulator that, given black-box access to $\mathcal{M}$, outputs an indistinguishable transcript.

EQUIVALENCE. To prove that our model implies this alternative model we need to show that a protocol in the former model (specifically a deniable MT-authenticator) is also a deniable authentication protocol (using the above definition).

Briefly, let λ be a deniable MT-authenticator. Consider the implicit protocol between the sender P (the prover) and the receiver V (the verifier). The completeness is straightforwardly verified.

For the soundness, suppose there exists an adversary $\mathcal{M}$ who is able to break the soundness property of the protocol. Imagine a parallel simulation between an authenticated world and an unauthenticated one using the MT-authenticator λ as connection using the techniques of Theorem 9. Using the adversary $\mathcal{M}$ (running in the unauthenticated world) we can build an AM-adversary $\mathcal{A}$ for the authenticated side. The parties in the soundness-game (copies of P and verifiers) run in the unauthenticated side of the simulation. We use the convention to attach to each message m the identity of the sender and an incremental id specific for each sender. This way the requirement that each party should send only distinct messages is satisfied.

If $\mathcal{M}$ breaks the scheme in the unauthenticated world, thanks to the fact that MT-authenticators are simulatable, it follows that $\mathcal{A}$ breaks the rules of the authenticated world (sending a message not present in the set M of "undelivered messages").

Finally, for strong deniability it is enough to note that the simulator $\mathcal{S}_{\mathcal{M}}$ is identical to our simulator $S_{\lambda}^{(\mathcal{M})}$.

B Implementations of AMTC/MTC Schemes

For the sake of completeness, we report an MTC scheme from [23] that is based on the Strong-RSA Assumption. It can be extended to an AMTC using Chameleon Hashing: the resulting scheme is quite similar to the construction of a Simulation-sound Commitments scheme in [41]. In the same paper there is another construction based on the security of DSA signature scheme that can be shown to be an AMTC too.

B.1 The RSA and Strong RSA Assumption

Let N be the product of two primes, $N = pq$. With $\phi(N)$ we denote the Euler function of N, i.e. $\phi(N) = (p-1)(q-1)$. With Z_N^* we denote the set of integers between 0 and $N-1$ and relatively prime to N.

Let e be an integer relatively prime to $\phi(N)$. The RSA Assumption [46] states that it infeasible to compute e-roots in Z_N^*. I.e. given a random element $s \in_R Z_N^*$ it is hard to find x such that $x^e = s \bmod N$.

The Strong RSA Assumption (introduced in [3]) states that given a random element s in Z_N^* it is hard to find $x, e \neq 1$ such that $x^e = s \bmod N$. The assumption differs from the traditional RSA assumption in that we allow the adversary to freely choose the exponent e for which she will be able to compute e-roots.

B.2 The Construction of the MTC based on Strong RSA

We first show a basic trapdoor commitment based on RSA (see [11, 13]) and later show how to make it into a MTC. Let N be the product of two large primes p, q. Let e be a prime such that $GCD(e, \phi(N)) = 1$, and s a random element of Z_N^*. The triple (N, s, e) are the public parameters. The commitment scheme is defined over messages in $[1..e-1]$.

We denote the commitment scheme with $\mathsf{Com}_{N,s,e}(\cdot,\cdot)$ and we drop the indices when obvious from the context. To commit to $a \in [1..e-1]$ the sender chooses $r \in_R Z_N^*$ and computes $A = \mathsf{Com}(a, r) = s^a \cdot r^e \bmod N$. To decommit the sender reveals a, r and the previous equation is verified by the receiver.

Under the RSA Assumption this scheme is an unconditionally secret, computationally binding trapdoor commitment scheme. The trapdoor is the value $x = s^{\frac{1}{e}} \bmod N$.

Let us first prove that the scheme is unconditionally secret. Given a value $A = s^a \cdot r^e$ we note that for each value $a' \neq a$ there exists a unique value r' such that $A = s^{a'}(r')^e$. Indeed this value is the e-root of $A \cdot s^{-a'}$.

We now show that the scheme is computationally binding under the RSA Assumption. We show that an adversary $\mathcal{A}$ who is able to open the commitment scheme in two ways can be used to compute e-roots. The proof uses Shamir's GCD-trick [48]. Given as input the values (N, s, e) we want to compute integer x such that $x^e = s \bmod N$. We place (N, s, e) as the public parameters of the commitment scheme and run $\mathcal{A}$. The adversary returns a commitment A and two distinct openings of it (a, r) and (a', r'). Thus

$$A = s^a r^e = s^{a'}(r')^e \implies s^{a-a'} = \left(\frac{r'}{r}\right)^e \tag{6}$$

Let $\delta = a - a'$. Since $a, a' < e$ we have that $GCD(\delta, e) = 1$. We can find integers α, β such that $\alpha\delta + \beta e = 1$. Now we can compute

$$s = s^{\alpha\delta+\beta e} = (s^\delta)^\alpha \cdot s^{\beta e} = \left(\frac{r'}{r}\right)^{\alpha e} s^{\beta e} \tag{7}$$

where we used Equation (6). By taking e-roots on both sides we find that $x = \left(\frac{r'}{r}\right)^\alpha s^\beta$.

Finally we show that if we know x then we can open a commitment (of which we already know one opening), in any way we want. Assume we computed A as $\mathsf{Com}(a, r) = s^a r^e$, and later we want to open it as a'. All we need to do (as shown above for the unconditional security) is to compute the e-root of

$$A \cdot s^{-a'} = s^{a-a'} \cdot r^e \bmod N$$

which clearly is $x^{a-a'} \cdot r \bmod N$.

Remark 1: The commitment scheme can be easily extended to any message domain $\mathcal{M}$, by using a collision-resistant hash function H from $\mathcal{M}$ to $[1..e-1]$. In this case the commitment is computed as $\mathsf{Com}(a, r) = s^{H(a)} r^e$. For example, it is possible to use a collision resistant function like SHA-1 that maps inputs to 160-bit integers and then choose e's larger than 2^{160}.

Remark 2: How to make the scheme into a multi-trapdoor commitment. Notice that the scheme above is really a family of commitment schemes, one for each prime e. The master trapdoor is the factorization of N. The specific trapdoor of each scheme is $s^{1/e} \bmod N$.

We only need to show that the Secure Binding condition holds, under the Strong RSA Assumption. Assume we are given a Strong RSA problem instance N, σ. Let us now run the **MTC Secure Binding** game.

The adversary is going to select k public keys which in this case are k primes, $e_1, \ldots, e_k$. We set $s = \sigma^{\prod_{i=1}^k e_i} \bmod N$ and return N, s as the public key of the MTC family. Now we need to show how to simulate the oracle $\mathcal{EQ}$. But that's easy, as we know the e_i-roots of s, so we actually know the trapdoor of the schemes in the family identified by e_i.

Assume now that the adversary equivocates a commitment scheme in the family identified by a prime $e \neq e_i$. Using the above observation we can then compute $\rho = s^{1/e}$. In turn this allows us to solve the Strong RSA problem instance N, σ by computing an e-root of σ as follows.

Let $E = \prod_{i=1}^{k} e_i$. Then $GCD(e, E) = 1$ which means that we can find integers α, β such that $\alpha e + \beta E = 1$. Then

$$\sigma = \sigma^{\alpha e + \beta E} = [\sigma^{\alpha} \rho^{\beta}]^{e}$$

This prove that this scheme is an MTC scheme.

C Alternative authenticators based on MTC

This section shows methods to make protocol λ_{AMTC} secure when using MTC commitments instead of AMTC ones. Notice that in these cases also in order to make λ_{AMTC} deniable we need to modify it as shown in Section 3.2.1.

C.1 Same protocol with MTC and Random Oracle proof

The only difference between the notion of AMTC and MTC schemes resides in the secure binding properties. In the game that defines the MTC secure binding property the adversary can't see the master public key PK until he declares the public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ on which he wants invoke the equivocator oracle $\mathcal{EQ}$. Instead, in the AMTC game he can invoke $\mathcal{EQ}$ without any limitations.

To deal with this problem in the random oracle model is relatively simple: let A^* be the party chosen at random among the n parties and let q be a polynomial upper bound to the number of messages that A^* is requested to send. Suppose that $\mathcal{H}^*$ is the hash function of the public key of A^*. Note that we need to program a random oracle only for $\mathcal{H}^*$, in fact the simulator knows the master trapdoor keys of the other parties.

We can choose q random public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_q)$ and program the oracle of the hash function $\mathcal{H}^*$ to map the i^{th} different request to the oracle (for example the string (m_i, B_i)) on the string pk_i. Further, in the MTC secure binding game, we declare that we want to be able to invoke the equivocator $\mathcal{EQ}$ on all these public keys except for one pk_j chosen at random. Namely, we bet on this session hoping that it is the broken session.

Finally, we saw in the proof of Theorem 6 that if the adversary $\mathcal{U}$ has probability δ to break the authenticator then the built equivocator $\mathcal{E}$ breaks the security of the multi-trapdoor commitment with probability $\frac{\delta}{n}$. Here, this probability is different but still non negligible, that is $\frac{\delta}{n \cdot q}$.

Remark: Note that here we are using the features of the ideal Random Oracle model only in the proof of unforgeability. The proof of deniability is identical to Theorem 7 and it holds in the standard model.

C.2 MTC-based authenticator without Random Oracle

Now we show that the Random Oracle in the previous construction can be replaced by a hash function which satisfies some strong (but well defined) computational assumptions introduced by Gennaro *et al.* [24].

In the previous section we saw how the adversary $\mathcal{E}$ can program the random oracle to obtain the previously chosen values. In the standard model this is no longer the case: clearly, if $\mathcal{H}$ is deterministic, then the choice of the message m and of the receiver uniquely determines the public key pk and $\mathcal{E}$ has no room to play with these values. But even if $\mathcal{H}$ is randomized this does not help $\mathcal{E}$ due to the fact that $\mathcal{H}$ is one-way. Thus, if the adversary first chooses pk and then sees the pair (m, B), it cannot find randomness r for witch $\mathsf{pk} = \mathcal{H}(r; m, B)$ (even if such r exists).

To deal with this problem we need an oracle associated to the hash function that given the hash Y and the message M returns a randomness R such that $\mathcal{H}(R; M) = Y$. Further, we need to assume that the presence of this oracle does not help in any way to break the security of the commitment scheme. This technique is introduced in [24] where they use this assumption relatively to the problem of Strong RSA. Here we present the same assumption but adapted to the use of a generic commitment scheme.[28]

We say that a family of hash function is *suitable*, relatively to a particular family of commitment schemes, if:

1. the hash functions in the family are collision-resistant;
2. for every function $\mathcal{H}$ in the family and for every two messages M_1, M_2, the distributions $\mathcal{H}(R; M_1)$, $\mathcal{H}(R; M_2)$, induced by the random choice of R, are statistically close;
3. there is an oracle that on input $\mathcal{H}, M, Y$ returns a random R such that $\mathcal{H}(R; M) = Y$;
4. the security of commitment schemes still holds in a model where there exists such an oracle.

Sketch of proof. If we assume that the hash functions used in the MTC-based authenticator are *suitable*, we can still prove that it emulates a protocol MT in unauthenticated networks. The proof proceeds similarly to Theorem 6, i.e. we construct an equivocator $\mathcal{E}$ which will use the UM-adversary $\mathcal{U}$. The main difference is that $\mathcal{E}$ operates in a relativized model, given in addition access to the oracle from Condition 3 of the suitable hash function.

As in the previous sketch of proof: let A^* be the party chosen at random among the n parties and let q be a polynomial upper bound to the number of messages that A^* is requested to send. $\mathcal{H}^*$ is the hash function of A^*. As before, We can choose q random public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_q)$ and, in the MTC secure binding game, we declare that we want to be able to equivocate with oracle $\mathcal{EQ}$ the commitments related to these keys.

When $\mathcal{A}^*$ receives the i^{th} request to send a generic message m_i to a party B_i, $\mathcal{E}$ invokes the randomness-finding oracle for a randomness r_i for which $\mathcal{H}(r_i; m_i, B_i) = \mathsf{pk}_i$. After, when A^* will receive the challenge from B_i, $\mathcal{E}$ can invoke the equivocator oracle $\mathcal{EQ}$ to open the commitment as requested.

The rest of the simulation proceeds as in the original proof, that is rewinding the simulation if $\mathcal{U}$ breaks party A^* and obtaining a double opening of a commitment of the family.[29] It is important to note that because of Condition 2 of the suitable hash function, the distribution that $\mathcal{U}$ sees in this simulation is statistically close to the distribution it sees when interacting with real parties in unauthenticated networks.

Implementations of Suitable Hash Functions. Following the discussion in [24], we argue that adopting this type of non-standard assumptions is reasonable.

First of suitable hash functions can be built under standard assumptions (like Factoring, RSA or Discrete Logarithm), by using Chameleon hashing [7, 38] (the trapdoor property of this hash

[28] If the security of the commitment scheme is base, for example, on the Strong RSA assumption, then our assumption is identical to the one used in the original paper.

[29] The probability that $\mathcal{U}$ uses in the broken session a pair (m, B) with a randomness r such that $\mathcal{H}^*(r; m, B) = \mathsf{pk}_i$ for some i is negligible. This means that the final probability of success of the built adversary is slightly different but still non negligible.

function immediately gives you the inverting oracle). Indeed our Chameleon Hashing solution is basically an istantiation of this paradigm.

But even without resorting to Chameleon Hashing, we can still assume that a very-efficient cryptographic hash function, such as SHA-1, is suitable. Intuitively what this means is that we are assuming that finding collisuions in SHA-1 is a computational task which is "unrelated" to breaking the commitment scheme, and that finding such collisions would not help in say inverting RSA.