Secret-Key Authentication Beyond the Challenge-Response Paradigm: Definitional Issues and New Protocols

Petros Mol * Stefano Tessaro †

December 18, 2012

Abstract

Secret-key authentication is the task of one party proving to another party that they share the same key. The problem has recently attracted widespread interest due to the existence of lightweight protocols amenable to implementation on simple architectures, such as RFID tags. This paper revisits and improves upon the large body of work on secret-key authentication in two different ways.

On the definitional side, we show that the notion of active security, the strongest attained by existing lightweight protocols, is *too weak* and can be satisfied by protocols completely insecure with respect to seemingly much weaker notions. We provide new, more apt definitions of active security, and investigate relations among them, within a new general framework for fine-grained modeling of the security of secret-key authentication protocols of independent interest. Moreover, we prove that previous protocols following the so-called challenge-response paradigm remain secure with respect to our new definitions as long as they are actively secure with respect to the old one.

In the second part of our paper, however, we also provide concrete evidence of the benefits of going beyond this paradigm: We present new generic constructions of authentication protocols which deviate from the challenge-response blueprint. On the one hand, we devise an actively-secure three-round protocol based on a very weak MAC which only needs to be secure against a forger on a *random* message when evaluated on *random* messages. The protocol admits efficient LPN- and CDH-based instantiations. On the other hand, we provide a two-round generic construction of a Man-in-the-Middle secure protocol based on weak MACs which enjoys an efficient instantiation based on the qSDH assumption.

Keywords: Authentication, secret-key cryptography, provable security, lightweight cryptography.

*Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, Email: pmol@cs.ucsd.edu.

†MIT CSAIL, 32 Vassar Street, Cambridge, MA 02139, Email: tessaro@csail.mit.edu.

1 Introduction

Consider two parties, Alice and Bob, sharing a secret key K, and taking the roles of a prover and a verifier, respectively. Alice would like to prove to Bob that she knows the key K, but no adversary Eve, without knowledge of K, should be able to persuade Bob that she knows K. Research on this problem, known as *symmetric authentication*, has recently gained momentum, driven by the discovery of lightweight authentication protocols suitable to implementation on RFID devices [30, 13, 19, 35, 24, 32, 33, 17].

In summary, the contributions of this paper are two-fold: On the one hand, we propose a refined framework to capture security notions for secret-key authentication and exercise it both to surface *problems* with existing security definitions for active security, which we show to capture too weak security goals, and to define better and stronger security goals. On the other hand, for some of these notions, we present new *generic* constructions of authentication protocols at the cost of *weaker* generic assumptions with respect to previous work. Before turning to a detailed overview of our results, let us first discuss some further background.

Secret-key authentication. Theoretical research on authentication protocols has been initially concerned with the public-key setting, where a prover $\mathcal{P}$ in pocession of a public/secret-key pair (pk, sk) wishes to prove its identity to a verifier $\mathcal{V}$ who only knows pk (such protocols have often been called *identification* protocols). Starting from the seminal work of Fiat and Shamir [22], a long series of protocols have been proposed (among others, cf. e.g. [27, 38, 36, 39]) mostly leveraging techniques from zero-knowledge proofs [26, 21].

In the meanwhile, the design of authentication protocols in practice has followed a completely different path. An increasing number of ubiquitous-computing applications (item-labeling, payment systems, proximity cards just to name a few) requires the existence of RFID tags that are capable of authenticating themselves to a reader. But such tags are extremely simple devices, typically circuits with a few thousand gates, and extremely low *hardware* complexity. Unfortunately, implementing public-key cryptography on such devices remains, even to date, beyond reach. Fortunately, *secret*-key authentication is sufficient in many scenarios, and may generally enjoy much more efficient solutions. For instance, the simplest protocol uses a block cipher E (such as DES or AES) with a secret key K and consists of only 2 rounds: in the first round, the verifier sends a random *challenge* R to the prover, which, upon receiving R, replies with $E_K(R)$. The verifier accepts if and only if the prover's response is the unique correct value. (Such protocols are known as *challenge-response protocols*.) Provided the block cipher is a sufficiently strong message-authentication code, this simple protocol achieves the strongest notion of *man-in-the-middle* (MIM) security: Roughly speaking, MIM security demands that an adversary interacting at will with an arbitrary number of both prover and verifier instances cannot later bring a further verifier instance to accept. Very general definitional frameworks modeling MIM security have been first proposed by Bellare and Rogaway [7].[1]

Unfortunately, mainstream block-cipher designs such as AES are not amenable to lightweight hardware implementation. Seeking for alternatives, Juels and Weis [30] were the first to point out that a very simple protocol by Hopper and Blum [29] (called HB) can be implemented with very low hardware complexity, and proven secure under the well-known Learning Parity with Noise (LPN) assumption.[2] Yet, HB happens to only satisfy a fairly weak notion of security, called *passive security*, where an adversary observing transcripts of honest prover-verifier interactions cannot convince a further verifier instance that she knows the key. Every attempt to design HB-like protocols with MIM security [13, 19, 35, 24]

[1] Bellare and Rogaway's work in fact focused on *mutual* authentication, where both parties authenticate to each other; yet their definitions are easily extended to the unilateral setting of interest in this paper.

[2] The (decisional) LPN assumption with error η asserts that for a random secret $\mathbf{s} \in \{0,1\}^n$, it is computationally hard to distinguish random independent $(n+1)$-bit strings from samples $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a} \in \{0,1\}^n$ is random and $e \in \{0,1\}$ is 1 with probability η.

turned out to miss a security proof, which, very often, resulted sooner or later in a fatal flaw being found [23, 37]. All *provably* MIM-secure protocols to date [33, 17] are challenge-response protocols derived from the construction of a suitable MAC.[3] While these elegant constructions do improve upon block-cipher based schemes, their hardware complexity remains far from that of the HB protocol.

THE NEED FOR WEAKER SECURITY: ACTIVE SECURITY. To overcome the above gap, previous work has focused on an intermediate security notion, called *active security*, where one asks that even an adversary which can interact with the prover arbitrarily fails in later convincing a verifier that she knows the key. This is a secret-key version of the standard security notion for public-key identification schemes dating back to Fiat and Shamir [22]. Juels and Weis [30] proposed a three-round challenge-response[4] protocol, called HB^+, which was shown to be secure against active attacks (the proof was later extended to hold for parallel or concurrent executions by Katz, Shin, and Smith [32]) and which is known not to be MIM secure [25]. Active security has then attracted further interest [33, 17, 28]. Simply put, active security appears to have become a de-facto standard security notion, backed by the existence of very efficient protocols achieving it, its widespread acceptance in the public-key setting, and the inherent hardness of achieving anything stronger such as MIM security efficiently.

1.1 Our Contributions

1.1.1 Modeling Security for Secret Key Authentication

Revisiting active security. As our first main contribution, we revisit the notion of active security for secret-key authentication protocols: We show that despite the common belief, and in *sharp* contrast to the public-key setting, active security as formulated in the literature is *too* weak to be considered a valid security target. We present new, stronger notions that should be targeted instead. To illustrate said weakness, let us consider the following toy protocol: The prover and the verifier hold a pair of secret values $K = (K_1, K_2)$, and the verifier starts the protocol by sending K_1 to the prover: If the prover receives the correct value K_1, it then gives back K_2 to the verifier, which accepts if this value is correct. This protocol cannot even be passively secure, yet it is actively secure according to the traditional definition for the following reason: In the first stage, the only way an adversary can "exploit" the prover is by guessing K_1 which is rather unlikely. And without the help of the prover, the adversary needs to guess K_2 in the second stage in order to convince the verifier, which is also information-theoretically hard.

Obviously, it is tempting to dismiss the issue by simply demanding passive security on top of active security. But perhaps surprisingly, we will show that this by itself is *not* sufficient either. We exhibit protocols which are both actively *and* passively secure, yet their security falls apart as soon as an active attacker is given one honest transcript between the prover and the verifier. Moreover, this remains true even if the protocol satisfies an even stronger notion of active security, where the adversary is allowed multiple, alternating (yet non-overlapping), interactions with the prover and the verifier.

Motivated by this observation, we present a new notion which combines active and passive attacks in *a single security game* by incorporating a corresponding transcript oracle, and advocate this to be the appropriate way of defining active security for *secret-key authentication*. Moreover, we distinguish between *one* and *several* alternating interactions with the prover or the verifier, the latter resulting notion being in the following informally referred to as *strong* active security.

[3]With one exception, perhaps, being the protocol where the verifier sends the encryption of a random plaintext to the prover under a OW-CCA-secure encryption scheme, and the prover returns its decryption; to the best of our knowledge, however, no efficient instantiations of this paradigm are known.

[4]In the three-round case, this means that the first two messages are both random.

A flexible definitional framework. The above discussion highlights the importance of precise definitions of security. To this end, we present a general framework to treat security definitions for authentication in secret-key protocols. We use the framework to study relations among security notions, the above issues being only one part of our extensive investigation. Our framework refines the framework of Bellare and Rogaway [7] (developed to provide security definitions in the context of mutual entity authentication and key distribution, and tailored at MIM security) to the case of *unilateral* secret-key authentication. In particular, the framework follows the adversary-is-the-network paradigm initially introduced by Dolev and Yao [18] and subsequently adopted by several other works [8, 6]. Similarly to [7], the adversary is given access to a collection of oracles representing either the parties participating in the protocol (prover or verifier) or executions of the protocol (the latter is modeled with a designated *transcript* oracle).

However, unlike [7], we want to model notions weaker than MIM security, and consequently do not necessarily allow access to all available oracles. Rather, the adversaty might have access to some oracles but not to others. Another difference is that we consider adversaries running in multiple phases (stages). On the contrary, in the model of [7], multiple phases were meaningless since they offered no more power to the adversary. Due to the above modifications, our framework can capture several attacks that can emerge in practice, ranging from simple eavesdropping (passive attacks) to full control of the entire communication (matching the definition of [7]). Even though adopting a conservative viewpoint when defining security goals has several benefits, it is an unavoidable fact that existing protocols with attractive implementation features fall short of achieving the strongest notions; our goal is to precisely address what these protocols achieve instead.[5]

All existing security notions, including the stronger definitions of MIM security due to Vaudenay [40] and Bellare and Rogaway [7] fit nicely into our framework. We also show that an other recurring folklore claim [32, 33, 17], namely that security for attackers making one verification query implies security for arbitrary (polynomial) verification queries, is also incorrect for protocols which are not challenge-response. Once again, this contradicts intuition from the public-key setting. But to our rescue, we will also show that many incorrect claims are indeed correct when restricting focus on challenge-response protocols. For example, we prove that existing protocols [30, 32, 33], claimed to be actively secure, are also secure with respect to the notion of strong active security.

1.1.2 Constructions Beyond the Challenge-Response Paradigm

One may question whether the current state of affairs is as problematic as we depict above. After all, all protocols we are interested in are challenge-response. So, why bother too much? We provide a very pragmatic answer to this question: We give protocols which fall outside the challenge-response paradigm, and which improve on existing protocols proposed in the literature, exhibiting a better efficiency-security trade off. For our new protocols, proving active security with respect to the traditional definition would yield too weak security guarantees.

Active security from LPN and weak MACs. As a case study, we focus on the question of achieving active security. Specifically, we consider protocols secure under the LPN assumption. Such protocols are usually very attractive, being suitable for implementation on lightweight hardware, yet we only know two such schemes to date: The HB^+ [30] protocol and the recent two-round protocol by Kiltz *et al* [33]. From the perspective of *concrete security*, both however suffer from drawbacks: On the one hand, no "tight" security reduction to LPN is known for HB^+. Roughly speaking, if LPN is ϵ-hard for secret length n and complexity t, we can only prove that an active attacker with time complexity

[5] This phenomenon is especially pronounced in the case of RFID protocols where requiring very strong cryptography drives the manufacturing of RFID tags to prohibitively high costs and slows down the widespread deployment of the technology.

(roughly) t cannot break security of HB^+ for key length $2n$ with probability larger than $\sqrt{\epsilon}$. This is problematic. For example, if we have $t = 2^{40}$ and $\epsilon = 2^{-40}$, an adversary attacking 2^{20} independent instances of the protocol may break at least one of them spending overall effort $t' = 2^{60}$, which may still be feasible. Also, we point out that this loss is the inevitable result of using rewinding in the security reduction, and, at least from a theoretical perspective, that this makes it impossible to prove HB^+ security against quantum attackers (based on the quantum hardness of LPN). Kiltz *et al* [33] did take a substantial step towards solving this issue by presenting a protocol which enjoys a tight reduction to LPN in terms of *advantage* ϵ, yet, if we assume as above that LPN is ϵ-hard for secret-size n, for their protocol to be ϵ-secure too, even under the most optimistic instantiation of their parameters, their key size becomes larger than $4n$ bits and the communication complexity is larger than the one of HB^+.

We hence ask the question: *Can we obtain the best of both worlds?* In other words, under the assumption that LPN is ϵ-hard for secret-size n, can we have an ϵ-secure protocol with key size and complexity comparable to HB^+? We answer this in the affirmative as long as we are interested in basic active security (as opposed to *strong* active security). Concretely, we propose a new *generic* approach to obtain the first efficient authentication protocol based on any *weak* MAC, i.e., a MAC which can be evaluated on *random* messages and which must be unforgeable on fresh, *random* messages. This is the weakest generic assumption on which such a protocol can be based, with previous generic constructions being either from stronger MACs or from a Weak PRF [17]. Given such a MAC, our three-round protocol, which we call **DM** (for Double Mac), is extremely simple. The secret key of the protocol consists of two keys K_1, K_2 for the underlying MAC. In the first round, the prover sends a random message r_1 to the verifier, which replies with $(\text{MAC}_{K_1}(r_1), r_2)$, for a random message r_2, in the second round. The prover, upon receiving (τ_1, r_2), subsequently checks whether τ_1 happens to be a valid tag for r_1, and if so, sends $\text{MAC}_{K_2}(r_2)$ back to the verifier, which finally accepts if and only if it receives a valid tag τ_2 for r_2 under key K_2.

When instantiated with an LPN-based weak MAC, **DM** yields a three-round protocol with communication complexity only minimally larger than HB^+, but with the benefit of a tight reduction to LPN. In addition, for the same security level, our protocol has lower communication complexity and at least 2 times smaller keys than the protocol of Kiltz *et al.* Even more, the security of our LPN-based **DM** scales significantly better than both HB^+ and the protocol of Kiltz *et al.* in the face of *multiple verification* attempts where, in the latter two protocols, an adversary essentially increases its success probability by a factor which is *linear* in the number of interactions with the verifier. Finally, **DM** is amenable to instantiation via a CDH-based weak MAC, yielding the most efficient actively secure protocol based on CDH. En passant, we also obtain an efficient protocol based on Ring-LWE [34].

Of course, it is fair to note that a drawback of **DM** compared to the existing challenge-response protocols is that it is not, in general, *strongly* actively secure, a clear advantage of existing challenge-response protocols. However, we point out that, to the best of our knowledge, strong active security was never considered prior to our work, hence indicating that (non-strong) active security is considered sufficient in many settings.

Generic constructions for MIM-security. As explained above, existing efficient protocols secure against man-in-the-middle attacks follow the challenge-response approach where the second message is the MAC of the challenge. Man-in-the-middle security for these protocols requires that the underlying MAC be unforgeable under chosen message attacks. We provide evidence of the potential advantages of moving outside the challenge-response paradigm by presenting a new generic approach to build two-round authentication protocols that resist man-in-the-middle attacks, yet are based on weak MACs which are strongly unforgeable only when evaluated on *random* messages, a strictly weaker assumption than what needed in challenge-response protocols. Once again, our protocol is fairly simple: The prover and the verifier both share two independent MAC keys K_1 and K_2, and the verifier sends $\text{MAC}_{K_1}(r), r$ to the prover, who *first* checks whether $\text{MAC}_{K_1}(r)$ is a valid MAC for r under K_1, and if the check is successful, replies with a valid MAC $\mathrm{MAC}_{K_2}(r)$ for r.

The simple intuition is that a MIM adversary cannot bring the prover to successfully authenticate a challenge r under K_2 unless it can provide a valid tag for r under K_1 in the first place, and the latter is unlikely unless the adversary uses the same r that was output by the verifier. In other words, the first MAC forces the adversary to stick to valid interactions. However, reducing MIM security of the above protocol to the right security notion for MACs is surprisingly subtle. We illustrate an instantiation of our approach based on the qSDH assumption (initially proposed by Boneh and Boyen [11]) which requires only four exponentiations in a cyclic group.

Other important related work. A vast body of literature has focused on privacy concerns related to RFID protocols [2, 16, 4, 31, 40, 15] and especially traceability of tags. Also, protocols for *mutual* authentication [7] have been considered in the past. We do not consider these aspects in this work. Rather, our goal here is to raise awareness with respect to subtleties related to the security of unilateral secret-key authentication protocols.

In terms of security notions, Gilbert *et al* [24] have considered an intermediate model (*aka* GRS-MIM model) in which an adversary can interact with both the tag and the reader in the first phase of the attack, but can only modify messages from the reader. Even though security in the GRS-MIM is strictly stronger than active security, the protocols that are known to achieve the former notion [24] are either inefficient ($RANDOM-HB^{\#}$) or based on assumptions that are not well studied ($HB^{\#}$). It has also been questioned whether there exist real-world attack scenarios in which an attacker can modify messages from the reader but not from the tag – and in the full-fledged MIM case, none of the protocols from [24] is secure [37]. Another interesting line of research studies the security of *distance-bounding* RFID protocols initially introduced by Brands and Chaum [12] as a countermeasure against MIM attacks. In this scenario, a verifier can measure roundtrip times between sending and receiving a message in order to detect MIM attacks. Most related to our work is the framework developed by Durholz *et al* [20] (building upon the work of Avoine *et al* [3]) to model security in distance-bounding protocols. Despite their similarities (mostly due to the fact that both inherit from the Bellare and Rogaway framework [7]), the model from [20] is incomparable to ours, since both the adversarial capabilities and the security goals are different.

Finally, we mention that very recently Heyse *et al* [28] presented Lapin, a simple and elegant 2-round protocol that is secure against active attacks. The security of Lapin relies on the assumption that the Ring-LPN problem, a structured variant of the standard LPN problem, is hard. However, the hardness of Ring-LPN is much less understood[6] than the hardness of LPN and thus, given our current understanding of algorithmic attacks, any comparison with LPN-based protocols is hardly meaningful (see also a recent attack by Bernstein and Lange [10] which exploits the ring structure of Ring-LPN to drastically reduce the resources needed for an active attack).

2 Preliminaries

Notation. We use $\mathbb{Z}, \mathbb{N}$ and $\mathbb{R}$ for the sets of integer, natural and real numbers respectively. We reserve lower case symbols for scalars, upper case for sets, bold lower case for vectors, bold upper case for matrices and sans serif font for polynomials. We also use calligraphic letters for probability distributions and (possibly randomized) algorithms. For a string x of length n, $\mathsf{pref}_k(x)$ $(k \leq n)$ is the prefix of x consisting of the first k symbols appearing in x. For a positive integer m, we write $[m]$ for the set of the first m postive integers, i.e. $[m] = \{1, \ldots, m\}$.

Games and Probability. We will often use games, as defined by Bellare and Rogaway [9], and adopt

[6]Ring-LPN can be also seen as a special case of Ring-LWE [34] with modulus $q = 2$. However, unlike Ring-LWE, Ring-LPN is *not* backed by a worst-case/average-case connection with (ideal) lattices.

Game $\mathsf{PRF}_{\mathsf{F}}^{\mathbf{O}}$	**oracle** $\mathbf{F}(x)$:
proc. main:	Ret $F_K(x)$
$K \xleftarrow{\$} \mathsf{KGen}$	**oracle** $\mathbf{R}(x)$:
for all $x \in \mathcal{D}$	If $V[x] = \perp$
$\quad V[x] \leftarrow \perp$	$\quad V[x] \xleftarrow{\$} \mathcal{R}$
$d \leftarrow \mathcal{A}^{\mathcal{O}}$	Ret $V[x]$

Game SUF-CMA$_{\mathsf{MAC}}$	**oracle Tag**(m):	**oracle Vrfy**(m, τ):
procedure main:	$\tau \leftarrow \mathsf{TAG}_K(m)$	If $(m, \tau) \in T$ then
$K \xleftarrow{\$} \mathsf{KGen}$	$T \leftarrow T \cup (m, \tau)$	$\quad$ Ret $\perp$
$\mathsf{Forge} \leftarrow \mathtt{false}$	Ret (m, τ)	If $\mathsf{VRFY}_K(m, \tau) = 1$ then
$T \leftarrow \emptyset$		$\quad \mathsf{Forge} \leftarrow \mathtt{true}$
Run $\mathcal{A}_{\mathsf{MAC}}^{\mathbf{Tag},\mathbf{Vrfy}}$		$\quad$ Ret 1
Ret Forge		Ret 0

Figure 1: **Definition of cryptographic primitives.** On the left: Games $\mathsf{PRF}_{\mathsf{F}}^{\mathbf{O}}$ for $\mathbf{O} \in \{\mathbf{F}, \mathbf{R}\}$. On the right: Game $\mathsf{MAC}_{\mathsf{MAC}}^{\mathsf{suf\text{-}cma}}$.

their computational model and notational conventions. We write $x \xleftarrow{\$} \mathcal{X}$ for the operation of selecting x according to a probability distribution $\mathcal{X}$ or by running probabilistic algorithm $\mathcal{X}$. For any probability distribution $\mathcal{X}$ over a set X and any value $x \in X$, $\Pr[x \leftarrow \mathcal{X}]$ denotes the probability associated to x by distribution $\mathcal{X}$. When X is a well understood set, we overload notation and write $x \xleftarrow{\$} X$ to mean that x is an element sampled uniformly at random from X. The *support* of a distribution $\mathcal{X}$ is denoted $[\mathcal{X}] = \{x \in X \mid \Pr[\, x \leftarrow \mathcal{X}\,] > 0\}$. We use Ber_η for the *Bernoulli* distribution with parameter η, i.e. Ber_η is a distribution over bits such that $\Pr_{b \leftarrow \mathsf{Ber}_\eta}[\, b = 1\,] = \eta$. Accordingly, Ber_η^m is the distribution over $\{0,1\}^m$ where each bit is independently distributed according to Ber_η. For some of our bounds, we use the *binary entropy function*, defined as $\mathrm{H}_2(p) = -p \cdot \log_2 p - (1-p) \cdot \log_2(1-p)$ as well as the (binary) *relative entropy* function with parameters p and q defined as

$$\mathrm{D}(p \,||\, q) = p \cdot \log_2\left(\frac{p}{q}\right) + (1-p) \cdot \log_2\left(\frac{1-p}{1-q}\right) .$$

$$\Pr[\, X > p \cdot m\,] \leq 2^{-\mathrm{D}(p \,||\, q) \cdot m} . \tag{1}$$

PRFs and MACs. A *pseudorandom function* (PRF) is a pair of algorithms $\mathsf{F} = (\mathsf{KGen}, F)$ where KGen is the randomized *key generation* algorithm, which outputs a key K from the keyspace $\mathcal{K}$, and $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is a function. The security of a PRF is formally defined on the left of Figure 1. For all adversaries $\mathcal{A}$, the prf-advantage is defined as

$$\mathbf{Adv}_{\mathsf{F}}^{\mathsf{prf}}(\mathcal{A}) = \Pr\left[\, (\mathsf{PRF}_{\mathsf{F}}^{\mathbf{F}})^{\mathcal{A}} \Rightarrow 1\,\right] - \Pr\left[\, (\mathsf{PRF}_{\mathsf{F}}^{\mathbf{R}})^{\mathcal{A}} \Rightarrow 1\,\right] .$$

For integers t, q, the advantage function is defined as $\mathbf{Adv}_{\mathsf{F}}^{\mathsf{prf}}(t, q) = \max_{\mathcal{A}}\{\mathbf{Adv}_{\mathsf{F}}^{\mathsf{prf}}(\mathcal{A})\}$ where the maximum is over all adversaries $\mathcal{A}$ running in time t and making q queries to the given oracle $\mathbf{O} \in \{\mathbf{F}, \mathbf{R}\}$. A *message authentication code* is a triple of algorithms $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ where

- KGen is the *key generation* algorithm which outputs a key K from some understood keyspace $\mathcal{K}$,
- TAG is the (possibly randomized) *tagging* algorithm taking as input a key $K \in \mathcal{K}$ and a message m from the understood message space $\mathcal{M}$ and outputting a tag $\mathsf{TAG}(K, m)$ or $\mathsf{TAG}_K(m)$ and
- VRFY is the (deterministic) *verification* algorithm taking as inputs a key $K \in \mathcal{K}$, a message $m \in \mathcal{M}$, as well as a tag τ from the tag space $\mathcal{T}$, and outputting a decision $\mathsf{VRFY}_K(m, \tau) \in \{0, 1\}$.

The *completeness error* of MAC is defined as

$$\epsilon_c = \max_{m \in \mathcal{M}} \Pr\left[\, K \xleftarrow{\$} \mathsf{KGen}, \tau \leftarrow \mathsf{TAG}_K(m) : \mathsf{VRFY}_K(m, \tau) = 0\,\right]$$

and is typically required to be small. The standard security notion for MACs is *strong unforgeability under chosen message attacks* ($\mathsf{suf\text{-}cma}$), formally defined in Figure 1. The corresponding advantage, for an attacker $\mathcal{A}$, is

$$\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{MAC}}(\mathcal{A}) = \Pr\left[\,\text{SUF-CMA}^{\mathcal{A}}_{\mathsf{MAC}} \Rightarrow \mathtt{true}\,\right],$$

and accordingly, we define $\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{MAC}}(t, q_{\mathsf{TAG}}, q_{\mathsf{VRFY}}) = \max_{\mathcal{A}}\{\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{MAC}}(\mathcal{A})\}$, where the maximum is over all adversaries $\mathcal{A}$ running in time t and making q_{TAG} (resp. q_{VRFY}) tag (resp. verification) queries.

3 Security Notions of Authentication Protocols

In Section 3.1, we introduce a framework for modeling security of secret-key authentication protocols refining the framework of Bellare and Rogaway [7] to support fine-grained description of attack models. Existing security notions fit within our framework and so do several new notions that we introduce in this work. Section 3.2 applies the framework both to highlight subtleties and problems with existing definitions and to study how several notions are related to each other. To that end, we show both *implications* and *separations*.

3.1 A Unified Definitional Framework

Algorithms and protocols. A stateful algorithm $\mathcal{A}$ has an initial input, keeps a *state*, and processes messages. Formally, $\mathcal{A}$ is a randomized algorithm $\mathcal{A} : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to (\{0,1\}^* \cup \{\bot\}) \times \{0,1\}^* \times (\{0,1\}^* \cup \{\bot\})$, where $(y, \sigma', msg') \stackrel{\$}{\leftarrow} \mathcal{A}(x, \sigma, msg)$ means that starting from state σ, on initial input x, and upon receipt of message msg, $\mathcal{A}$ changes its internal state to σ', sends message msg' and, if $y \neq \bot$, terminates with output y. Here, $y = \varepsilon$ indicates termination without any output.

An interactive two-party protocol, is a pair $(\mathcal{P}_1, \mathcal{P}_2)$ of interactive algorithms, where exactly one of $\mathcal{P}_1$ and $\mathcal{P}_2$ accepts a special designated message start. (We assume that it is $\mathcal{P}_1$ in the following.) The protocol execution is defined via the following procedure:

$\underline{(\mathcal{P}_1 \leftrightarrow \mathcal{P}_2)(x)}$:
$msg_0 \leftarrow \mathsf{start}$; $i \leftarrow 0$
$y_1, y_2 \leftarrow \bot$; $\sigma_1, \sigma_2 \leftarrow \varepsilon$
While $y_1 = \bot$ or $y_2 = \bot$ do
  If $i = 0 \mod 2$ and $y_1 = \bot$ then
    $(y_1, \sigma_1, msg_{i+1}) \stackrel{\$}{\leftarrow} \mathcal{P}_1(x, \sigma_1, msg_i)$
  Else if $y_2 = \bot$ then
    $(y_2, \sigma_2, msg_{i+1}) \stackrel{\$}{\leftarrow} \mathcal{P}_2(x, \sigma_2, msg_i)$
  $i \leftarrow i + 1$
Ret $\mathtt{true}$

We say that $(\mathcal{P}_1, \mathcal{P}_2)$ is *well-formed* if the above procedure always terminates returning $\mathtt{true}$. Moreover, it is an *r-round protocol* if $i = r + 1$ upon termination. We denote as $(y_1, y_2) \stackrel{\$}{\leftarrow} (\mathcal{P}_1 \leftrightarrow \mathcal{P}_2)(x)$ the process of sampling the outputs of $\mathcal{P}_1$ and $\mathcal{P}_2$ after an interaction. We also overload notation by writing $\text{Tran} \stackrel{\$}{\leftarrow} (\mathcal{P}_1 \leftrightarrow \mathcal{P}_2)(x)$ for the process of sampling the transcript of the interaction between $\mathcal{P}_1$ and $\mathcal{P}_2$, i.e., the sequence consisting of the messages $(msg_1, \ldots, msg_r)$ exchanged. Notice that $msg_0 = \mathsf{start}$ and the very last message are *not* part of the transcript.

Authentication protocols. A (secret-key) authentication protocol is a triple $\Pi = (\mathcal{K}, \mathcal{P}, \mathcal{V})$ such that $\mathcal{K}$ is a randomized *key generation algorithm* that generates a key K, while $\mathcal{P}$ and $\mathcal{V}$ are interactive algorithms, both taking as input a key K in the range of $\mathcal{K}$, and such that $(\mathcal{P}, \mathcal{V})$ is a well-formed interactive protocol. In addition, $\mathcal{P}$ always outputs ε, whereas $\mathcal{V}$ outputs a decision value $d \in \{\mathsf{A}, \mathsf{R}\}$.

Game $\mathrm{AUTH}_{\Pi}^{S_1,\ldots,S_m}$:

procedure main():

$K \xleftarrow{\$} \mathcal{K}$; $T \leftarrow \emptyset$; $\mathsf{ctr} \leftarrow 0$; $\sigma_0 = \bot$

For all $i = 1$ to m do // phase $i \in [m]$

  For all $sid \in \mathbb{N}$ do

    $\mathsf{state}[sid] \leftarrow \varepsilon$; $\mathsf{decision}[sid] \leftarrow \bot$

    $\mathsf{done}[sid] \leftarrow \mathtt{false}$

  $\sigma_i \xleftarrow{\$} \mathcal{A}_i^{\mathbf{S}_i}(\sigma_{i-1})$

If $\exists sid \in SID_{\mathcal{V}}$: $(\mathsf{decision}[sid] = \mathsf{A})$

  $\wedge$ $(\forall sid' \in SID_{\mathcal{P}} : \neg\mathsf{Matching}(T[sid'], T[sid]))$

    Ret $\mathtt{true}$

Ret $\mathtt{false}$

oracle T():

Tran $\xleftarrow{\$} (\mathcal{P} \leftrightarrow \mathcal{V})(K)$

Ret Tran

oracle P(sid, msg):

If $(sid \notin SID_{\mathcal{P}}) \vee \mathsf{done}[sid]$ Ret $\bot$

Else

  $(\mathsf{state}[sid], msg', y') \xleftarrow{\$} \mathcal{P}(K, \mathsf{state}[sid], msg)$

  $T \leftarrow T \cup \{(sid, \mathsf{ctr}, msg, msg')\}$

  $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

  If $y' \neq \bot$ then

    $\mathsf{done}[sid] \leftarrow \mathtt{true}$

  Ret msg'

oracle V(sid, msg):

If $(sid \notin SID_{\mathcal{V}}) \vee \mathsf{done}[sid]$ Ret $\bot$

Else

  $(\mathsf{state}[sid], msg', y') \xleftarrow{\$} \mathcal{V}(K, \mathsf{state}[sid], msg)$

  $T \leftarrow T \cup \{(sid, \mathsf{ctr}, msg, msg')\}$

  $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

  If $y' \neq \bot$ then

    $\mathsf{done}[sid] \leftarrow \mathtt{true}$; $\mathsf{decision}[sid] \leftarrow y'$

    Ret y'

  Ret msg'

Figure 2: **Pseudocode description of Game** $\mathrm{AUTH}_{\Pi}^{S_1,\ldots,S_m}$**:** Here, $S_i \subseteq \{\mathsf{P}, \mathsf{T}, \mathsf{V}\}$ for all $i \in [m]$, $\Pi = (\mathcal{K}, \mathcal{P}, \mathcal{V})$ is an authentication protocol and $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_m)$ is an m-phase adversary. The predicate $\mathsf{Matching}(T[sid'], T[sid])$ returns $\mathtt{true}$ if $sid' \in SID_{\mathcal{P}}$, $sid \in SID_{\mathcal{V}}$ and $T[sid']$, $T[sid]$ are matching.

For any real value $\delta \in [0, 1]$, we say that the protocol Π is δ-*complete* (or has completeness δ) if $\Pr\left[K \xleftarrow{\$} \mathcal{K}, (\varepsilon, d) \xleftarrow{\$} (\mathcal{P} \leftrightarrow \mathcal{V})(K) : d = \mathsf{A} \right] \geq \delta$. We assume without loss of generality that the last message is sent from $\mathcal{P}$ to $\mathcal{V}$, which then terminates with a decision, and does not send any further messages.

Security of authentication protocols. Let $\Pi = (\mathcal{K}, \mathcal{P}, \mathcal{V})$ be an authentication protocol. To model the security of Π, we consider adversaries that run in multiple phases (stages). More concretely, let $S_1, \ldots, S_m$ be such that $S_i \subseteq \{\mathsf{P}, \mathsf{V}, \mathsf{T}\}$ for all $i \in [m]$ and $\mathsf{V} \in S_m$. The security of Π against an adversary running in m phases, is defined via the game $\mathrm{AUTH}_{\Pi}^{S_1,\ldots,S_m}$ shown in Figure 2. The game $\mathrm{AUTH}_{\Pi}^{S_1,\ldots,S_m}$ starts by sampling a key $K \xleftarrow{\$} \mathcal{K}$, and allows the attacker to arbitrarily and concurrently interact with instances of the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ under key K, addressed via session ids sids in $SID_{\mathcal{P}}$ and $SID_{\mathcal{V}}$, respectively, for two understood disjoint sets of integers $SID_{\mathcal{P}}, SID_{\mathcal{V}} \subset \mathbb{N}$. We remark that a session id sid characterizes an interaction between the adversary and an instance of $\mathcal{P}$ (or $\mathcal{V}$) and *not* (necessarily) between an instance of $\mathcal{P}$ and an instance of $\mathcal{V}$. Also, the same key K is shared accross all instance $sid \in SID_{\mathcal{P}} \cup SID_{\mathcal{V}}$. The global variables $\mathsf{state}[sid]$, $\mathsf{decision}[sid]$ and $\mathsf{done}[sid]$ maintain information associated with each sid, i.e., the state of the corresponding instance, whether it has accepted an interaction (in case $sid \in SID_{\mathcal{V}}$) or whether it has terminated. The game consists of m *phases*, each one of which involves a respective adversary $\mathcal{A}_i$, where $\mathcal{A}_i$ can pass on arbitrary state information to $\mathcal{A}_j$ for all $j > i$. In each phase, the corresponding adversary is granted access to a subset of the following oracles according to S_i:

- The *prover oracle* **P** accepts queries of the form (sid, msg) where $sid \in SID_{\mathcal{P}}$ and $msg \in \{0,1\}^*$. Upon such a query, it runs $\mathcal{P}(K, \mathsf{state}[sid], msg)$, obtaining output (σ', msg', y). It then sets $\mathsf{state}[sid]$ to σ', and if $y' = \bot$, returns msg' to the adversary; otherwise it returns (y', msg'). In the latter case, **P** does not accept any further queries of the form $(sid, *)$ until the end of the current phase.

– The *verifier oracle* $\mathbf{V}$ operates as $\mathbf{P}$, using $\mathcal{V}$ instead of $\mathcal{P}$. In addition, upon terminating, i.e., when returning $(d, \perp)$ for $d \in \{\mathsf{A}, \mathsf{R}\}$ after a query (sid, msg), it sets $\mathsf{decision}[sid] \leftarrow d$.
– The *transcript oracle* $\mathbf{T}$ samples a transcript $\mathrm{Tran} \xleftarrow{\$} (\mathcal{P} \leftrightarrow \mathcal{V})(K)$ and returns it.

Specifically, for $\mathsf{O} \in \{\mathsf{P}, \mathsf{V}, \mathsf{T}\}$, access to oracle $\mathbf{O} \in \{\mathbf{P}, \mathbf{V}, \mathbf{T}\}$ is given to $\mathcal{A}_i$ in phase i if and only if $\mathsf{O} \in S_i$. Abusing notation, we will write $\mathbf{S}_i$ for the set of *oracles* available at phase i. For instance if $S_1 = \{\mathsf{T}, \mathsf{P}\}$ then $\mathbf{S}_1 = \{\mathbf{T}, \mathbf{P}\}$. At the beginning of each phase, AUTH resets all global variables associated to all *sid*s. In this way, *sid*s can be reused in subsequent phases but do not maintain any state from previous ones. To address the randomized nature of $\mathcal{P}$ and $\mathcal{V}$, we assume that each oracle has access to a fresh randomness source and that oracles associated with different *sid*s (or with same *sid*s but accross different phases) use fresh random coins each time they are invoked.

In order to rule out trivial winning strategies for $\mathcal{A}$ when $\mathcal{P}$ is present in phase m, we use the notion of *matching conversations* from [7]. In particular, queries to $\mathbf{P}$ and $\mathbf{V}$ are assigned numbers in increasing order of occurrence via an auxiliary global variable ctr that measures relative time. A query $q = (sid, msg)$ to $\mathbf{P}$ (or $\mathbf{V}$), answered by msg', results in (sid, i, msg, msg') being added to T where i is the value of the global counter ctr at the time of the query and T is a global list that keeps track of the entire communication associated with all *sid*s. Let Π be an r-round authentication protocol and assume the communication is initiated by $\mathcal{P}$, i.e. $\mathcal{P}$ receives the message start. For a pair of *sid*s $(sid, \overline{sid}) \in SID_{\mathcal{P}} \times SID_{\mathcal{V}}$ consider the communication associated with each of them, $T[sid] = \{(sid, i_0, \mathsf{start}, msg'_0), (sid, i_2, msg_2, msg'_2), \ldots, (sid, i_{r-1}, msg_{r-1}, msg'_{r-1})\}$ and $T[\overline{sid}] = \{(\overline{sid}, \bar{i}_1, \overline{msg}_1, \overline{msg}'_1), (\overline{sid}, \bar{i}_3, \overline{msg}_3, \overline{msg}'_3), \ldots, (\overline{sid}, \bar{i}_r, \overline{msg}_r, \overline{msg}'_r)\}$. Following [7, 40], we say that $T[sid], T[\overline{sid}]$ are *matching* if
- $i_0 < \bar{i}_1 < i_2 < \ldots < i_{r-1} < \bar{i}_r$,
- for all *odd* i $(1 \le i \le r)$, $\overline{msg}_i = msg'_{i-1}$ and
- for all *even* i $(1 \le i \le r)$, $msg_i = \overline{msg}'_{i-1}$.

For $sid \in SID_{\mathcal{P}}$ and $\overline{sid} \in SID_{\mathcal{V}}$, the predicate $\mathsf{Matching}(T[sid], T[\overline{sid}])$, which returns true if and only if $T[sid], T[\overline{sid}]$ are matching, captures the concept of an interleaved communication between the instances corresponding to sid and $\overline{sid}$. The AUTH game finally returns `true` if $\mathcal{A}_m$ manages to make the verifier accept in phase m for some sid (i.e., $\mathsf{decision}[sid] = \mathsf{A}$ for some $sid \in SID_{\mathcal{V}}$), and additionally, there is no $sid' \in SID_{\mathcal{P}}$ such that $T[sid']$ and $T[sid]$ are matching.[7] It returns `false` otherwise.

For any adversary $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_m)$, we say that $\mathcal{A}$ makes $q_{\mathsf{P},i}$ queries to $\mathbf{P}$ in phase i if the number of *distinct* $sid \in SID_{\mathcal{P}}$ that appear across all queries of the form (sid, msg) during phase i are $q_{\mathsf{P},i}$. $q_{\mathsf{V},i}$ is defined similarly. Queries to $\mathbf{T}$ are not interactive and hence $q_{\mathsf{T},i}$ is precisely the number of calls to $\mathbf{T}$ during phase i. The $(S_1, \ldots, S_m)$-*auth advantage* of $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\Pi}^{(S_1,\ldots,S_m)\text{-auth}}(\mathcal{A}) = \Pr\left[\, (\mathrm{AUTH}_{\Pi}^{S_1,\ldots,S_m})^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] .$$

Moreover, for all positive t and $q_{\mathsf{T},i}, q_{\mathsf{P},i}, q_{\mathsf{V},i}$ (for all $i \in [m]$) we define

$$\mathbf{Adv}_{\Pi}^{(S_1,\ldots,S_m)\text{-auth}}(t, q_{\mathsf{T},1}, q_{\mathsf{P},1}, q_{\mathsf{V},1}, \ldots, q_{\mathsf{T},m}, q_{\mathsf{P},m}, q_{\mathsf{V},m}) = \max_{\mathcal{A}}\{\mathbf{Adv}_{\Pi}^{(S_1,\ldots,S_m)\text{-auth}}(\mathcal{A})\} .$$

The maximum here is over all adversaries $\mathcal{A}$ running in time t and making $q_{\mathsf{T},i}, q_{\mathsf{P},i}$ and $q_{\mathsf{V},i}$ queries to the corresponding oracles (during phase i) where, by definition, $q_{\mathsf{O},i} = 0$ if $\mathsf{O} \notin S_i$. (We will usually omit these quantities from the advantage measure but our notation will make unambiguous which oracles the query numbers correspond to.) Informally, we will say that a protocol Π is $(S_1, \ldots, S_m)$-secure (or enjoys $(S_1, \ldots, S_m)$-security) if for all *efficient* adversaries $\mathcal{A}$, $\mathbf{Adv}_{\Pi}^{(S_1,\ldots,S_m)\text{-auth}}(\mathcal{A})$ is small. For multiple-phase adversaries that get access to the same oracles alternately, we simplify the writing by adopting string-style notation. For instance, for a protocol Π, subsets $S, S' \subseteq \{\mathsf{T}, \mathsf{P}, \mathsf{V}\}$ and a 2ℓ-phase adversary $\mathcal{A}$,

[7] Note that, since all global variables of all *sid*s are reset at the beginning of phase m, if V is the only element of S_m, checking the existence of a matching $T[sid']$ for $sid' \in SID_{\mathcal{P}}$ is redundant.

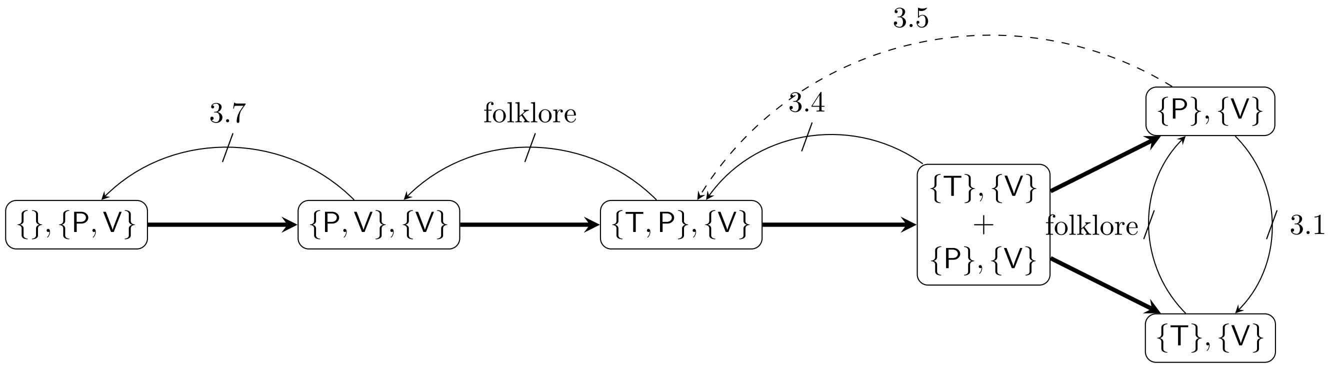


Figure 3: **Summary of relations among notions for two-phase adversaries.** Thick solid arrows indicate implications. Thin solid arrows with a slash depict separations. The number above each separation indicates the theorem that proves it. Finally, the dashed arrow indicates implication for Public Coin Verifier (PCV) protocols.

we write $\mathrm{AUTH}_{\Pi}^{(\{S\},\{S'\})^{\ell}}$ (instead of $\mathrm{AUTH}_{\Pi}^{(\{S\},\{S'\},\{S\},\{S'\},\ldots,\{S\},\{S'\})}$) for the corresponding security game. Likewise, we informally use the notation $(\{S\},\{S'\})^*$-security to indicate $(\{S\},\{S'\})^{\ell}$ security for *any* (polynomially bounded) $\ell \in \mathbb{N}$.

On winning in the last phase. We note that our framework only yields adversarial victory if the verifier is convinced in the last phase, whereas victory in earlier phases is not relevant. We could in fact extend our notion to incorporate multiple designated phases where an adversary can win. However, we will consider mostly games of the form $\mathrm{AUTH}_{\Pi}^{(S,\{\mathsf{V}\})^*}$, in which case it is not hard to see that checking for winning in the last verification phase and checking for winning in *all* even $\{\mathsf{V}\}$-phases are equivalent via a simple hybrid-argument.

Existing notions and extensions. Given the framework as described above, defining existing security notions for authentication protocols is rather straightforward. The vast majority of the definitions appearing in the literature consider two-phase adversaries. For instance, passive security is precisely $(\{\mathsf{T}\},\{\mathsf{V}\})$-auth-security , active security is $(\{\mathsf{P}\},\{\mathsf{V}\})$-auth security is active security, while $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-auth-security is man-in-the middle (MIM) security as used in the recent works (e.g., [23, 37, 33, 17]). Moreover, a stronger notion of MIM security was used by Vaudenay [40] and is equivalent to $(\{\mathsf{P},\mathsf{V}\})$-auth-security. This can also be seen as a special case of the notion of authenticity for mutual authentication used by Bellare and Rogaway [7].

3.2 Relations Among Security Notions

We now turn to discussing relations. We focus mostly on security against two-phase adversaries, as two-phase attacks have been the primary target of previous work. We note however that some of the relations (or separations) can be extended to hold for multiple-phase adversaries. Whenever this is the case, we explicitly state the result in its full generality from which the two-phase case can be derived as a corollary. Figure 3 summarizes our findings for two-phase notions. Some of them highlight surprising separations, one of them showing the existing definition of active security to be unsatisfactory, as well as implications that hold only for certain classes of protocols. To maintain consistency with 2-phase notation, for the most part of the current section, we use $(\{\},\{\mathsf{P},\mathsf{V}\})$ to denote strong MIM security. However, in other parts, we often use $(\{\mathsf{P},\mathsf{V}\})$ (omitting the first phase) interchangeably to mean the exact same security notion.

Implications. All implications are depicted with a solid thick arrow in Figure 3 and can be easily justified via the following observations: By definition, if $S_1, S_1', S_2, S_2 \subseteq \{\mathsf{T},\mathsf{P},\mathsf{V}\}$ such that $S_1' \subseteq S_1$ and $S_2' \subseteq S_2$, then (S_1, S_2)-security implies (S_1', S_2')-security. Also $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-security implies $(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})$-security since any adversary $\mathcal{A}$ can perfectly simulate the $\mathbf{T}$ oracle if given access to oracles $\mathbf{P}$ and $\mathbf{V}$. Indeed, for a $\mathbf{T}$ query, $\mathcal{A}$ simply forwards the replies of one oracle to the other perfectly simulating a full interaction. Finally, it is not hard to observe that having access to $\mathbf{P}$ and $\mathbf{V}$ during phase 2 gives an adversary *no less*[8] power than having access to the same oracles during phase 1. In particular, any $(\{\},\{\mathsf{P},\mathsf{V}\})$-adversary $\mathcal{A}$ can simulate a $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ as follows: $\mathcal{A}$ enters phase 2 directly and replies to all queries from $\mathcal{B}$ using its $\mathbf{P}, \mathbf{V}$ oracles. Along the simulation, $\mathcal{A}$ maintains a list $SID_{\mathcal{V}}^{(1)}$ that contains all $sid \in SID_{\mathcal{V}}$ that appear in queries (sid, msg) to $\mathbf{V}$ made by $\mathcal{B}_1$. When $\mathcal{B}$ enters (its) phase 2 then for every query (sid, msg) to $\mathbf{V}$ by $\mathcal{B}_2$, $\mathcal{A}$ queries (sid', msg) to its $\mathbf{V}$ oracle for some $sid' \notin SID_{\mathcal{V}}^{(1)}$, i.e., $\mathcal{A}$ never reuses an $sid \in SID_{\mathcal{V}}$ used in the first phase of $\mathcal{B}$'s attack. Notice that $\mathcal{A}$ simulates perfectly $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})}$ to $\mathcal{B}$. The crucial observation is that, by choosing *fresh sid*s to reply to $\mathcal{B}_2$'s queries, $\mathcal{A}$ ensures that for all pairs $(sid, sid') \in SID_{\mathcal{P}} \times SID_{\mathcal{V}} \setminus SID_{\mathcal{V}}^{(1)}$, $\mathsf{Matching}(T[sid], T[sid'])$ is `false`. Therefore, if $\mathcal{B}$ wins in $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})}$ so does $\mathcal{A}$ in $\mathrm{AUTH}_{\Pi}^{(\{\},\{\mathsf{P},\mathsf{V}\})}$. Interestingly, the ideas described above can be extended to show that in fact $(\{\mathsf{P},\mathsf{V}\})$-security is the strongest possible notion, that is $(\{\mathsf{P},\mathsf{V}\})$-security implies $(\{S_1,\ldots,S_m\})$-security for any $S_i \subseteq \{\mathsf{T},\mathsf{P},\mathsf{V}\}$.

On Active Security and the Necessity of the Transcript Oracle. Previous works suggest the use of $(\{\mathsf{P}\},\{\mathsf{V}\})$-security as an interesting and realistic security notion for practical secret-key authentication protocols, and call this notion *active security*. Many efficient schemes [30, 32, 33, 17, 28] are only proven to be $(\{\mathsf{P}\},\{\mathsf{V}\})$-secure. Quite surprisingly, we now show that this notion, by itself, is not a meaningful target: There exist protocols that are $(\{\mathsf{P}\},\{\mathsf{V}\})$-secure, yet they are not even passively secure. In fact, Theorem 3.1 describes an even stronger separation: There exist 3-round protocols that are $(\{\mathsf{P}\},\{\mathsf{V}\})^*$-secure but are not $(\{\mathsf{T}\},\{\mathsf{V}\})$-secure even against adversaries that make a *single* query to the transcript oracle.

Theorem 3.1. $[(\{\mathsf{P}\},\{\mathsf{V}\})^* \not\Rightarrow (\{\mathsf{T}\},\{\mathsf{V}\})]$ *For any* $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$*, there exists a three-round protocol* $\Pi = (\mathcal{K}, \mathcal{P}^F, \mathcal{V}^F)$ *such that*

- $\mathbf{Adv}_{\Pi}^{(\{\mathsf{T}\},\{\mathsf{V}\})\text{-auth}}(c,1,1) = 1$ *for a constant* $c > 0$*, while*
- *For all* $\ell \in \mathbb{N}$ *and all* $t, q_{\mathsf{P},1}, q_{\mathsf{V},2}, \ldots, q_{\mathsf{P},2\ell-1}, q_{\mathsf{V},2\ell} > 0$ [9]

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})^{2\ell}\text{-auth}}(t, q_{\mathsf{P},1}, \ldots, q_{\mathsf{V},2\ell}) \le \sum_{k=0}^{\ell-1} \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(t_k, q_{\mathsf{V}}^k, q_{\mathsf{P},2k+1}) + \sum_{k=0}^{\ell-1} \frac{q_{\mathsf{P},2k+1}(q_{\mathsf{P},2k+1} + q_{\mathsf{V}}^k)}{2^n} + \sum_{k=1}^{\ell-1} \mathbf{Adv}_F^{\mathsf{kr\text{-}cma}}(t_k, q_{\mathsf{V}}^k, q_{\mathsf{V},2k})$$

where $q_{\mathsf{V}}^k = \sum_{j=1}^k q_{\mathsf{V},2j}$ *and* $t_k = t + \mathcal{O}(\sum_{j=0}^k q_{\mathsf{P},2j+1})$ *for* $k \in [\ell]$.

Proof. Consider protocol Π as shown in Figure 4(a) where $K \xleftarrow{\$} \mathcal{K}$ is an k-bit string and $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ is a function. On the one hand, it is easy to see that Π is *not* $(\{\mathsf{T}\},\{\mathsf{V}\})$-secure even against adversaries making a single $\mathbf{T}$-query in phase 1 and a single $\mathbf{V}$-query in phase 2. Indeed,

[8] In fact, as we prove in Theorem 3.7, it gives *strictly more* power.

[9] The last term of the right hand side corresponds to the *key-recovery* advantage under chosen message attacks. Formally this is defined via a game KR-CMA proceeding similarly to SUF-CMA with the single difference that the adversary has access to a key-verification oracle **KVrfy** (instead of the standard **Vrfy** oracle), which, on input a string z from the keyspace of F returns 1 if and only if $z = K$. The advantage function is defined in a straightforward way, replacing the queries to **Vrfy** with queries to **KVrfy**. It is easy to show that $\mathbf{Adv}_F^{\mathsf{kr\text{-}cma}}(t, q_{\mathsf{T}}, q_{\mathsf{V}}) \le \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(t, q_{\mathsf{T}}, q_{\mathsf{V}})$.

any $\mathbf{T}$-query reveals K which can then be used, in the second phase, to make $\mathcal{V}$ accept (with a single interaction). Clearly, the resulting adversary runs in constant time and has advantage 1.

On the other hand, if F is a $\textsf{suf-cma}$-secure $\textsf{MAC}$, an adversary $\mathcal{A}$ is unlikely to make the verifier accept even if it can first repeatedly and alternately interact with instances of $\mathcal{P}$ and $\mathcal{V}$ (in isolation) for the following reason: When interacting with $\mathcal{P}$, $\mathcal{A}$ gets several fresh random r's for which it needs to guess $F_K(r)$ in order to learn the key. If $\mathcal{A}$ never guesses $F_K(r)$ for any of those r's, then $\mathcal{P}$ is of (almost) no use to $\mathcal{A}$ (from $\mathcal{A}$'s point of view, $\mathcal{P}$ only outputs random n-bit strings that carry no information about the key K). When given access to a verifier instance, $\mathcal{A}$ can get $F_K(r)$ for any r of its choice. But again, by the $\textsf{suf-cma}$-security of F, this interaction reveals very little *computationally* for K. Also, since $\mathcal{A}$ never gets access to $\mathcal{P}$ and $\mathcal{V}$ during the same phase, the evaluation of F_K even at chosen r's does not help $\mathcal{A}$ guess $F_K(r')$ on random r' in the next phase. Details follow.

Let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_{2\ell})$ be a 2ℓ-phase adversary that runs in time t, has access to $\mathbf{P}$ during odd phases $2k-1$ (making $q_{\mathsf{P},2k-1}$ queries) for all $k \in [\ell]$ and to $\mathbf{V}$ during even phases $2k$ (making $q_{\mathsf{V},2k}$ queries) for all $k \in [\ell]$. We define a sequence of $2\ell+1$ games as follows: G_i ($i \in \{0\} \cup [2\ell]$) runs in 2ℓ phases just like $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})^\ell}$. However $\mathbf{P}$ and $\mathbf{V}$ queries are replied differently: $\mathbf{P}$ queries that are issued during all odd phases *up to* phase i and have the form $(\mathit{sid}, \mathit{msg})$ with $\mathit{msg} \neq \mathsf{start}$ (these queries correspond to the second message in Π) are always replied by $\perp$. Similarly all terminating queries (that is, those that correspond to the third message of Π) to $\mathbf{V}$ issued during all even phases *up to* phase i return R.

Notice that by definition $G_0 = \mathrm{AUTH}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})^\ell}$. Also no adversary can win in $G_{2\ell}$ since by definition all interactions during the 2ℓ-th (final) phase are rejected. Therefore

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})^\ell\text{-auth}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \quad \text{and} \quad \Pr\left[\, G_{2\ell}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] = 0\,. \tag{2}$$

Claims 3.2 and 3.3 essentially assert that if F is $\textsf{suf-cma}$-secure then the probabilty $\mathcal{A}$ wins in G_i is not much larger than the that of winning in G_{i+1}. Due to a slightly different security reduction we treat the transition from G_i to G_{i+1} differently depending on whether i is even (Claim 3.2) or odd (Claim 3.3).

Claim 3.2. *For all $k \in \{0\} \cup [\ell-1]$, there exists an adversary $\mathcal{B}_k$ such that*

$$\Pr\left[\, G_{2k}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_{2k+1}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \leq \mathbf{Adv}_F^{\textsf{suf-cma}}(\mathcal{B}_k) + \frac{q_{\mathsf{P},2k+1}(q_{\mathsf{P},2k+1} + \sum_{j=1}^{k} q_{\mathsf{V},2j})}{2^n}\,. \tag{3}$$

Also, $\mathcal{B}_k$ makes $q_{\mathsf{TAG}}^k = \sum_{j=1}^{k} q_{\mathsf{V},2j}$ queries to its **Tag** *oracle, $q_{\mathsf{P},2k+1}$ queries to its* **Vrfy** *oracle and runs in time $t_k = t + \mathcal{O}(\sum_{j=0}^{k} q_{\mathsf{P},2j+1})$.*

Proof. (of Claim 3.2) G_{2k} and G_{2k+1} proceed identically during phases $1, 2, \ldots, 2k$ and differ only in the way $\mathbf{P}$-queries are replied during phase $2k+1$. Let BAD be a flag that is set to $\mathtt{true}$ (in both games) whenever at phase $2k+1$ a $\mathbf{P}$-query $(\mathit{sid}, \mathit{msg})$ with $\mathit{msg} \neq \mathsf{start}$ is such that $\mathit{msg} = F_K(\mathsf{state}[\mathit{sid}])$ (that is the adversary returns a valid tag on the challenge previously sent by the specific $\mathcal{P}$ instance corresponding to sid). After BAD is set to $\mathtt{true}$, G_{2k} returns K to $\mathcal{A}$ whereas G_{2k+1} returns $\perp$. G_{2k}, G_{2k+1} are identical-until-bad and hence

$$\Pr\left[\, G_{2k}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_{2k+1}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \leq \Pr\left[\, \mathsf{BAD} \,\right]$$

Now consider an adversary $\mathcal{B}_k$ (against $\mathrm{SUF\text{-}CMA}_F$) that simulates G_{2k+1} to $\mathcal{A}$. During phases $1, 3, \ldots, 2k-1$ if $\mathcal{A}$ makes a $\mathbf{P}$-query $(\mathit{sid}, \mathsf{start})$, then $\mathcal{B}_k$ simply samples $r \stackrel{\$}{\leftarrow} \{0,1\}^n$ and returns it to $\mathcal{A}$. If $\mathcal{A}$ makes a $(\mathit{sid}, \mathit{msg})$ query to $\mathbf{P}$ with $\mathit{msg} \neq \mathsf{start}$, then $\mathcal{B}_k$ returns $\perp$. During phases $2, 4, \ldots, 2k$ for every (non-terminating) query $(\mathit{sid}, \mathit{msg})$, $\mathcal{B}_k$ queries its **Tag** oracle with msg and returns the result to $\mathcal{A}$. Also, $\mathcal{B}_k$ returns R to every $(\mathit{sid}, \mathit{msg})$ $\mathbf{V}$-query for a terminating message msg. Finally, during phase $2k+1$, if $\mathcal{A}$ makes a $\mathbf{P}$-query $(\mathit{sid}, \mathsf{start})$, $\mathcal{B}_k$ samples $r \stackrel{\$}{\leftarrow} \{0,1\}^n$, sets $\mathsf{state}[\mathit{sid}]$ to r and returns r to $\mathcal{A}$. When $\mathcal{A}$ makes a query $(\mathit{sid}, \mathit{msg})$ (for $\mathit{msg} \neq \mathit{start}$) $\mathcal{B}_k$ sends $(\mathsf{state}[\mathit{sid}], \mathit{msg})$ as a candidate forgery to its **Vrfy** oracle and replies with R to $\mathcal{A}$. If **Vrfy** replies with 1, $\mathcal{B}_k$ aborts the simulation.

First, it is straightforward to verify that $\mathcal{B}_k$ simulates perfectly G_{2k+1} to $\mathcal{A}$ (until phase $2k+1$). Assume also, that during the simulation, the event $\mathsf{BAD} \leftarrow \mathtt{true}$ happens. Let $r_1, \ldots, r_{q_{\mathsf{P},2k+1}}$ be the messages sampled by $\mathcal{B}_k$ during **P**-queries of the form (sid, start) during phase $2k+1$. Define Col to be the event that at least one of $r_1, \ldots, r_{q_{\mathsf{P},2k+1}}$ has previously been part of a **V**-query (sid, msg) issued by $\mathcal{A}$ during phases $2, \ldots, 2k$. Clearly if Col does not happen, $\mathcal{B}_k$ has never invoked its **Tag** oracle in any of $r_1, \ldots, r_{q_{\mathsf{P},2k+1}}$ and therefore whenever BAD is $\mathtt{true}$, $\mathcal{B}_k$ wins in its SUF-CMA$_F$ game. We conclude that

$$\begin{aligned} \Pr\left[\, G_{2k}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_{2k+1}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] &\leq \Pr\left[\, \mathsf{BAD} \mid \neg\mathsf{Col} \,\right] + \Pr\left[\, \mathsf{Col} \,\right] \\ &\leq \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(\mathcal{B}_k) + \frac{q_{\mathsf{P},2k+1}(q_{\mathsf{P},2k+1} + \sum_{j=1}^{k} q_{\mathsf{V},2j})}{2^n} . \end{aligned}$$

Finally, for every **V** query from $\mathcal{A}$, $\mathcal{B}_k$ makes one query to its **Tag** oracle, for every **P** query during phases $1, 3, \ldots, 2k-1$, $\mathcal{B}_k$ needs only sample a random r whereas for every **P**-query during phase $2k+1$ $\mathcal{B}_k$ queries in addition its **Vrfy** oracle once. ☐

Claim 3.3. *For all $k \in [\ell]$, there exists an adversary $\mathcal{C}_k$ such that*

$$\Pr\left[\, G_{2k-1}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_{2k}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \leq \mathbf{Adv}_F^{\mathsf{kr\text{-}cma}}(\mathcal{C}_k) \qquad (4)$$

where $\mathcal{C}_k$ makes $q_{\mathsf{TAG}}^k = \sum_{j=1}^{k} q_{\mathsf{V},2j}$ queries to its **Tag** *oracle, $q_{\mathsf{V},2k}$ queries to its* **KVrfy** *oracle and runs in time $t_k = t + \mathcal{O}(\sum_{j=1}^{k} q_{\mathsf{P},2j-1})$.*

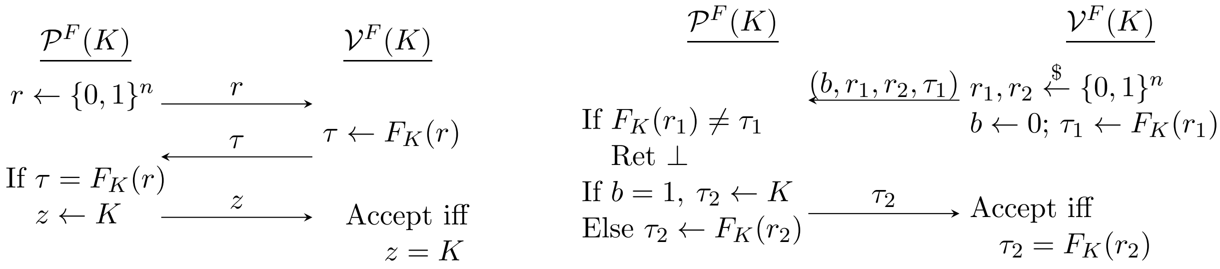
Proof. (of Claim 3.3) The proof is very similar to the proof of Claim 3.2. We only highlight the differences here. First notice that G_{2k-1} and G_{2k} are identical with respect to **P** queries. They only differ in the way **V**-queries during phase $2k$ are answered. In particular if a terminating **V**-query (sid, msg) is such that $msg = K$ (call this event BAD), G_{2k-1} returns A whereas G_{2k} returns R. It is not very hard to device an adversary $\mathcal{C}_k$ that perfectly simulates G_{2k} to $\mathcal{A}$ and wins whenever $\mathcal{A}$ causes the event BAD. All **P**-queries (during odd-numbered phases) can be simulated by $\mathcal{C}_k$ without the use of any of its oracles (on a query (sid, msg) to **P**, $\mathcal{C}_k$ returns a random n-bit string if $msg = \mathsf{start}$ and $\perp$ otherwise). For non-terminating **V**-queries (sid, msg), $\mathcal{C}_k$ uses it **Tag** oracle to reply, while for terminating queries, $\mathcal{C}_k$ simply returns $\perp$ (during phases $2, 4, \ldots, 2k-2$) or invokes its **KVrfy** oracle (during phase $2k$). Clearly if event BAD happens, $\mathcal{C}_k$ wins in its KR-CMA. Also for the simulation, $\mathcal{C}_k$ invokes its **Tag** oracle once per **V**-query and its **KVrfy** oracle once per **V**-query but only during phase $2k$. ☐

The proof of the theorem follows by combining equations (2), (3) and (4). ☐

The above makes it clear that despite its widespread usage in the existing literature, $(\{\mathsf{P}\}, \{\mathsf{V}\})$-security is not an appropriate measure of security against active attacks. For this reason, we advocate the use of $(\{\mathsf{T}, \mathsf{P}\}, \{\mathsf{V}\})$-security and $(\{\mathsf{T}, \mathsf{P}\}, \{\mathsf{V}\})^*$-security as more appropriate measures of active security. (We refer to the latter as *strong* active security, and it is not hard to show that it is in fact strictly stronger than $(\{\mathsf{T}, \mathsf{P}\}, \{\mathsf{V}\})$-security.) It is tempting, however, to disqualify this issue by demanding that the protocol is additionally passively secure, i.e. require *both* $(\{\mathsf{P}\}, \{\mathsf{V}\})$-security (or even $(\{\mathsf{P}\}, \{\mathsf{V}\})^*$-security) and $(\{\mathsf{T}\}, \{\mathsf{V}\})$-security. The following theorem addresses this issue providing a negative answer, showing the existence of protocols which are both $(\{\mathsf{P}\}, \{\mathsf{V}\})$-secure and $(\{\mathsf{T}\}, \{\mathsf{V}\})$-secure, yet not $(\{\mathsf{T}, \mathsf{P}\}, \{\mathsf{V}\})$-secure.

Theorem 3.4. $[(\{\mathsf{P}\}, \{\mathsf{V}\}) + (\{\mathsf{T}\}, \{\mathsf{V}\}) \not\Rightarrow (\{\mathsf{T}, \mathsf{P}\}, \{\mathsf{V}\})]$ *For any $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^m$, there exists a two-round protocol $\Pi = (\mathcal{K}, \mathcal{P}^F, \mathcal{V}^F)$ such that*

– $\mathbf{Adv}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}(c, 1, 1, 1) = 1$ *for a constant $c > 0$, while*



(a) protocol for $(\{\mathsf{P}\},\{\mathsf{V}\})^* \not\Rightarrow (\{\mathsf{T}\},\{\mathsf{V}\})$ (Thm. 3.1). If the prover's check fails, z is set to $\perp$.
(b) protocol for $(\{\mathsf{P}\},\{\mathsf{V}\}) + (\{\mathsf{T}\},\{\mathsf{V}\}) \not\Rightarrow (\{\mathsf{P},\mathsf{T}\},\{\mathsf{V}\})$ (Thm. 3.4)

Figure 4: Protocols for Theorems 3.1 and 3.4

– $\mathbf{Adv}_{\Pi}^{(\{\mathsf{T}\},\{\mathsf{V}\})\text{-auth}}(t, q_{\mathsf{T}}, q_{\mathsf{V}}) \leq \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(t', 2q_{\mathsf{T}} + q_{\mathsf{V}}, q_{\mathsf{V}}) + \frac{q_{\mathsf{V}}(2q_{\mathsf{T}}+q_{\mathsf{V}})}{2^n}$, *for all* $t, q_{\mathsf{T}}, q_{\mathsf{V}} > 0$ *where* $t' = t + \mathcal{O}(q_{\mathsf{V}} + q_{\mathsf{T}})$ *and*
– $\mathbf{Adv}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}(t, q_{\mathsf{T}}, q_{\mathsf{V}}) \leq \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(t_1, 0, q_{\mathsf{P}}) + \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(t_2, q_{\mathsf{V}}, q_{\mathsf{V}}) + \frac{q_{\mathsf{V}}^2}{2^n}$, *for all* $t, q_{\mathsf{T}}, q_{\mathsf{V}} > 0$, *where* $t_1 = t$ *and* $t_2 = t + \mathcal{O}(q_{\mathsf{V}})$.

Proof. Consider protocol Π shown in Figure 4(b) where F is a function $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$. Π is not $(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})$-secure: A simple adversary $\mathcal{A}$ first makes one $\mathbf{T}$-query to obtain $((0, r_1, r_2, \tau_1), \tau_2)$. It then makes a $\mathbf{P}$-query $(sid, (1, r_1, r_2, \tau_1))$ for some $sid \in SID_{\mathcal{P}}$, which reveals the secret key K. Finally, in phase 2, $\mathcal{A}$ makes a query (sid', start) to $\mathbf{V}$ (for some $sid' \in SID_{\mathcal{V}}$), obtaining $(0, r_1^*, r_2^*, \tau_1^*)$, and terminates by making a query $(sid', F(K, r_2^*))$ to $\mathbf{V}$, which must then accept. Clearly, $\mathcal{A}$ runs in constant time, makes 1 query to each of the $\mathbf{T}, \mathbf{P}$ and $\mathbf{V}$ oracles and makes $\mathbf{V}$ accept with probability 1.

However, we show that Π is passively secure assuming F is a $\mathsf{suf\text{-}cma}$-secure MAC: For every adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against $(\{\mathsf{T}\},\{\mathsf{V}\})$-security, we devise a $\mathsf{suf\text{-}cma}$ adversary $\mathcal{B}$, simulating $\text{AUTH}_{\Pi}^{(\{\mathsf{T}\},\{\mathsf{V}\})}$ to $\mathcal{A}$ as follows: Whenever $\mathcal{A}_1$ queries $\mathbf{T}$, $\mathcal{B}$ generates $r_1, r_2 \xleftarrow{\$} \{0,1\}^n$, and queries its $\mathbf{Tag}$ oracle on both r_1, r_2, receiving replies τ_1 and τ_2. It then returns the transcript consisting of $((0, r_1, r_2, \tau_1), \tau_2)$ back to $\mathcal{A}_1$. Whenever $\mathcal{A}_2$ in Phase 2 asks for a query (sid, start) to $\mathbf{V}$, $\mathcal{B}$ samples $r_{sid,1}, r_{sid,2} \xleftarrow{\$} \{0,1\}^n$, queries $r_{sid,1}$ to its $\mathbf{Tag}$ oracle, obtaining $\tau_{sid,1}$, and returns $(0, r_{sid,1}, r_{sid,2}, \tau_{sid,1})$ to $\mathcal{A}_2$. When $\mathcal{A}_2$ then inputs $(sid, \tau_{sid,2})$ to $\mathbf{V}$, $\mathcal{B}$ queries $(r_{sid,2}, \tau_{sid,2})$ to its $\mathbf{Vrfy}$ oracle. During the simulation, $\mathcal{B}$ makes 2 $\mathbf{Tag}$ queries for each $\mathbf{T}$ query by $\mathcal{A}_1$ and 1 $\mathbf{Tag}$ plus 1 $\mathbf{Vrfy}$ query for each $\mathbf{V}$ query by $\mathcal{A}_2$. Therefore $\mathcal{B}$ makes in total $2q_{\mathsf{T}} + q_{\mathsf{V}}$ $\mathbf{Tag}$ queries and q_{V} $\mathbf{Vrfy}$ queries to its oracles and runs in time $t' = t + \mathcal{O}(q_{\mathsf{V}} + q_{\mathsf{T}})$. It remains to bound $\mathcal{A}$'s advantage. Let BAD be the event that during the simulation of the entire phase 2, $\mathcal{B}$ samples $r_{sid,2}$ (as part of the reply to a $\mathbf{V}$-query (sid, start) made by $\mathcal{A}$) such that $r_{sid,2}$ has been previously input to a $\mathbf{Tag}$ query (made by $\mathcal{B}$ to its $\mathbf{Tag}$ oracle while simulating $\mathcal{A}$). For a single $r_{sid,2}$ this probability is upper bounded by $\frac{2q_{\mathsf{T}}+q_{\mathsf{V}}}{2^n}$. Taking the union bound across all q_{V} queries we obtain $\Pr[\,\mathsf{BAD}\,] \leq \frac{q_{\mathsf{V}}(2q_{\mathsf{T}}+q_{\mathsf{V}})}{2^n}$. Finally notice that, conditioned on BAD not happening, $\mathcal{B}$ wins in its SUF-CMA$_F$ game whenever $\mathcal{A}$ wins in $\text{AUTH}_{\Pi}^{(\{\mathsf{T}\},\{\mathsf{V}\})}$. Therefore

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{T}\},\{\mathsf{V}\})\text{-auth}}(\mathcal{A}) - \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(\mathcal{B}) \leq \Pr[\,\mathsf{BAD}\,] \leq \frac{q_{\mathsf{V}}(2q_{\mathsf{T}} + q_{\mathsf{V}})}{2^n}\,.$$

We finally prove that Π is $(\{\mathsf{P}\},\{\mathsf{V}\})$-secure. For that, we define two games G_0, G_1 and, throughout the proof, fix an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ running is t steps and making $q_{\mathsf{P}}, q_{\mathsf{V}}$ queries to $\mathbf{P}$ and $\mathbf{V}$ respectively. G_0 is precisely $\text{AUTH}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})}$ except from some internal bookkeeping within the $\mathbf{P}$ oracle. In particular, if an $(sid, b, r_1, r_2, \tau_1)$ query to $\mathbf{P}$ is such that $F_K(r_1) = \tau_1$, then a flag BAD_1 is set to

$\texttt{true}$. The rest of the execution is exactly as in $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})}$. Clearly, setting BAD_1 to $\texttt{true}$ does not affect the adversarial view and hence

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] . \tag{5}$$

G_1 is identical to G_0 except, when BAD_1 is set to $\texttt{true}$, $\mathbf{P}$ returns $\perp$. We claim that there exists a $\mathsf{suf\text{-}cma}$-adversary $\mathcal{C}_1$ such that

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] \leq \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(\mathcal{C}_1) . \tag{6}$$

In addition $\mathcal{C}_1$ makes 0 **Tag** and q_{P} **Vrfy** queries to its oracles and runs in time t. Notice that G_0, G_1 are equivalent-until-bad and therefore, by the fundamental lemma of game-playing,

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] \leq \Pr\left[\, \mathsf{BAD}_1 \,\right] .$$

The adversary $\mathcal{C}_1$ for the game SUF-CMA$_F$ simulates the *first* phase of G_0 to $\mathcal{A}_1$ as follows: On each $\mathbf{P}$ query $(\mathit{sid}, b, r_1, r_2, \tau_1)$ by $\mathcal{A}_1$, $\mathcal{C}_1$ simply queries its **Vrfy** oracle on input (r_1, τ_1). If **Vrfy** returns 1, $\mathcal{C}_1$ terminates (successfully) . Otherwise, it returns $\perp$ to $\mathcal{A}_1$. Clearly $\mathcal{C}_1$ simulates perfectly the first phase of G_0 to $\mathcal{A}_1$. Also BAD_1 is set to $\texttt{true}$ if and only if $\mathcal{C}_1$ forges, i.e., $\Pr\left[\, \mathsf{BAD} \,\right] = \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(\mathcal{C}_1)$. Finally, for every $\mathbf{P}$ query by $\mathcal{A}$, $\mathcal{C}_1$ makes a single verification query to its **Vrfy** oracle.

We finally claim that the problem of $\mathcal{A}$ winning in G_1 essentially reduces to the problem of forging F in SUF-CMA$_F$. In particular, we show that there exists a $\mathsf{suf\text{-}cma}$-adversary $\mathcal{C}_2$ such that

$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] \leq \mathbf{Adv}_F^{\mathsf{suf\text{-}cma}}(\mathcal{C}_2) + \frac{q_{\mathsf{V}}^2}{2^n} . \tag{7}$$

In addition $\mathcal{C}_2$ makes q_{V} **Tag** and q_{V} **Vrfy** queries to its oracles and runs in time $t + \mathcal{O}(q_{\mathsf{V}})$. $\mathcal{C}_2$ simulates $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as follows: For all $\mathbf{P}$ queries $(\mathit{sid}, b, r_1, r_2, \tau_2)$ by $\mathcal{A}_1$ in phase 1, $\mathcal{C}_2$ returns $\perp$. In phase 2, when $\mathcal{A}_2$ makes a query $(\mathit{sid}, \mathsf{start})$ to $\mathbf{V}$, $\mathcal{C}_2$ samples $r_{\mathit{sid},1}, r_{\mathit{sid},2} \xleftarrow{\$} \{0,1\}^n$, queries $r_{\mathit{sid},1}$ to its **Tag** oracle and upon getting $\tau_{\mathit{sid},1}$, returns $(0, r_{\mathit{sid},1}, r_{\mathit{sid},2}, \tau_{\mathit{sid},1})$ to $\mathcal{A}_2$. If $\mathcal{A}_2$ makes a query $(\mathit{sid}, \tau_{\mathit{sid},2})$ to $\mathbf{V}$, $\mathcal{C}_2$ queries $(r_{\mathit{sid},2}, \tau_{\mathit{sid},2})$ to its **Vrfy** oracle. During the simulation, $\mathcal{C}_2$ makes 1 **Tag** plus 1 **Vrfy** query for each $\mathbf{V}$ query by $\mathcal{A}_2$. Therefore $\mathcal{C}_2$ makes in total q_{V} **Tag** and q_{V} **Vrfy** queries and runs in time $t_2 = t + \mathcal{O}(q_{\mathsf{V}})$. In order to bound $\mathcal{C}_2$'s advantage, let BAD_2 be the event that during simulating phase 2 to $\mathcal{A}_2$, $\mathcal{C}_2$ samples $r_{\mathit{sid},2}$ (as part of the reply to a $\mathbf{V}$-query $(\mathit{sid}, \mathsf{start})$ made by $\mathcal{A}$) such that $r_{\mathit{sid},2}$ has been previously input to a **Tag** query. Notice that, $\mathcal{C}_2$ makes at most q_{V} queries to its **Tag** oracle and hence, for a single $r_{\mathit{sid},2}$ the probability of such a collision is upper bounded by $\frac{q_{\mathsf{V}}}{2^n}$. Taking the union bound across all q_{V} queries we obtain $\Pr\left[\, \mathsf{BAD}_2 \,\right] \leq \frac{q_{\mathsf{V}}^2}{2^n}$. Finally, conditioned on BAD_2 not happening, $\mathcal{C}_2$ wins in its SUF-CMA$_F$ game assuming $\mathcal{A}$ wins in $\mathrm{AUTH}^{(\{\mathsf{P}\},\{\mathsf{V}\})}$. The proof then follows from (5), (6) and (7). □

Extending to the case of $(\{\mathsf{P}\},\{\mathsf{V}\})^*$ security. One can extend the above result to show that there exists a protocol Π with the above properties, but which is additionally $(\{\mathsf{P}\},\{\mathsf{V}\})^*$-secure. In the following, let MAC be a $\mathsf{suf\text{-}cma}$-secure MAC with message space $\mathcal{M}$. The following protocol uses two keys K and K', where K is a MAC key, whereas K' is a random element of $\mathcal{M}$.

$\underline{\mathcal{P}^{\mathsf{MAC}}(K,K')}$		$\underline{\mathcal{V}^{\mathsf{MAC}}(K,K')}$
If $r_0 = K'$	$\xleftarrow{\quad r_0 \quad}$	$r_0 \xleftarrow{\$} \mathcal{M}$
$\quad r_1 \leftarrow K$		
Else $r_1 \xleftarrow{\$} \mathcal{M}$	$\xrightarrow{\quad r_1 \quad}$	
		$r_2 \xleftarrow{\$} \mathcal{M}$
		$\tau_2 \xleftarrow{\$} \mathsf{TAG}_K(r_1)$
If $\mathsf{VRFY}_K(r_1,\tau_1) = 1$	$\xleftarrow{\quad r_2, \tau_1 \quad}$	
$\tau_2 \xleftarrow{\$} \mathsf{TAG}_K(r_2)$		
$s_3 \leftarrow K'$	$\xrightarrow{\quad \tau_2, s_3 \quad}$	Accept iff $\mathsf{VRFY}_K(r_2,\tau_2) = 1$

We omit the rather tedious formal proof. Informally, passive security holds because, except in the unlikely even that $r_0 = K'$, no honest transcript reveals K. Therefore, the passive adversary is left with the task of finding a valid MAC tag for a fresh random r_2. Moreover, $(\{\mathsf{P}\},\{\mathsf{V}\})^*$ security holds because no adversary can even learn K' without either guessing at random or forging the MAC on a fresh random r_1. But the protocol is clearly insecure if given access to the transcript oracle and the prover concurrently, as an honest transcript reveals K' and which can then be used when interacting with the prover to acquire K.

We however now describe a fairly general class of protocols for which we prove that $(\{\mathsf{P}\},\{\mathsf{V}\})$-security does imply $(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})$-security. We say that a protocol Π is *Public-Coin Verifier* (PCV) if all (intermediate) messages sent from the verifier to the prover are chosen uniformly at random independently of the messages sent by the prover. Luckily, all existing protocols in the literature that follow the challenge-response (or the commit-challenge-response) paradigm and have been proven secure in the sense of achieving $(\{\mathsf{P}\},\{\mathsf{V}\})$-security are PCV (e.g. [30, 32, 33, 17, 28]). Theorem 3.5 essentially states that for PCV protocols, access to the $\mathbf{T}$ oracle (besides $\mathbf{P}$), does *not* add any more power to the adversary. Interestingly, the latter remains true even if the adversary runs in multiple phases getting access to either $\{\mathbf{T},\mathbf{P}\}$ or $\{\mathbf{V}\}$ alternately.

Theorem 3.5. [PCV : $(\{\mathsf{P}\},\{\mathsf{V}\}) \Rightarrow (\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^*$] *Let Π be any r-round Public-Coin Verifier authentication protocol. Then for all $\ell \in \mathbb{N}$ and $t, q_{\mathsf{T},1}, q_{\mathsf{P},1}, q_{\mathsf{V},2}, \ldots, q_{\mathsf{T},2\ell-1}, q_{\mathsf{P},2\ell-1}, q_{\mathsf{V},2\ell} > 0$,*

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^{\ell}\text{-auth}}(t, q_{\mathsf{T},1}, q_{\mathsf{P},1}, q_{\mathsf{V},2}, \ldots, q_{\mathsf{T},2\ell-1}, q_{\mathsf{P},2\ell-1}, q_{\mathsf{V},2\ell}) \le \sum_{k=0}^{\ell-1} \mathbf{Adv}_{\Pi}^{(\{\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}(t_k, q_{\mathsf{P}}^k, q_{\mathsf{V}}^k),$$

where $q_{\mathsf{P}}^k = \sum_{j=1}^{k+1}(q_{\mathsf{T},2j-1} + q_{\mathsf{P},2j-1})$, $q_{\mathsf{V}}^k = q_{\mathsf{V},2(k+1)}$ and $t_k = t + \mathcal{O}(r \cdot (\sum_{j=1}^{k+1} q_{\mathsf{T},2j-1} + \sum_{j=1}^{k-1} q_{\mathsf{V},2j}))$.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_{2\ell})$ be an adversary that makes $q_{\mathsf{T},1}, q_{\mathsf{P},1}, q_{\mathsf{V},2}, \ldots, q_{\mathsf{T},2\ell-1}, q_{\mathsf{P},2\ell-1}, q_{\mathsf{V},2\ell}$ queries to the corresponding oracles and runs in time t. We define a sequence of $\ell+1$ games as follows: G_k $k \in \{0\} \cup [\ell]$ runs in 2ℓ phases just like $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^{\ell}}$. Also, queries to $\mathbf{T}$ and $\mathbf{P}$ during odd-numbered phases are replied exactly as in $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^{\ell}}$. The only difference lies in the way terminating queries to $\mathbf{V}$ (those correspond to the r-th round of the protocol) are replied. In particular, in phases $2, 4, \ldots, 2k$ terminating queries to $\mathbf{V}$ are always answered with R. However, $\mathbf{V}$ queries in subsequent phases are replied as specified in $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^{\ell}}$. Notice that by definition $\mathrm{AUTH}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^{\ell}} = G_0$. Also no adversary can win in G_ℓ since by definition all interactions during the 2ℓ-th (final) phase are rejected. Therefore

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})^{\ell}\text{-auth}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \quad \text{and} \quad \Pr\left[\, G_\ell^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] = 0\,. \tag{8}$$

Consider two consecutive games G_k and G_{k+1} for any $k = 0, \ldots, \ell-1$. G_k and G_{k+1} proceed identically during phases $1, 2 \ldots, 2k+1$. In phase $2(k+1)$, if a terminating query (sid, msg) to $\mathbf{V}$ results in $\mathbf{V}$

accepting ($\mathsf{decision}[sid] = \mathsf{A}$), then both G_k and G_{k+1} set a flag BAD to $\mathtt{true}$. However, G_k returns A as the reply to the query whereas G_{k+1} returns R. That is the only difference between G_k and G_{k+1}. The following claim asserts that, if Π is $(\{\mathbf{P}\},\{\mathbf{V}\})$-secure then the probabilty $\mathcal{A}$ wins in G_k is not much larger than that of winning in G_{k+1}.

Claim 3.6. *There exists an adversary $\mathcal{B}_k$ such that*

$$\Pr\left[\, G_k^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_{k+1}^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \leq \mathbf{Adv}_{\Pi}^{(\{\mathbf{P}\},\{\mathbf{V}\})\text{-auth}}(\mathcal{B}_k)\ . \tag{9}$$

Also, $\mathcal{B}_k$ makes $q_{\mathsf{P}}^k = \sum_{j=0}^{k}(q_{\mathsf{T},2j+1} + q_{\mathsf{P},2j+1})$ queries to its $\mathbf{P}$ oracle and $q_{\mathsf{V},2k}$ to its $\mathbf{V}$ oracle and runs in time $t_k = t + \mathcal{O}(r \cdot (\sum_{j=0}^{k} q_{\mathsf{T},2j+1} + \sum_{j=0}^{k-1} q_{\mathsf{V},2j}))$.

Proof. (of Claim 3.6) Since G_k and G_{k+1} are identical-until-bad, by the fundamental lemma of game playing, we only need to bound the probability the flag BAD is set to $\mathtt{true}$ in game G_{k+1}. We do so by proving that the probabilty BAD is set to $\mathtt{true}$ equals the probability $\mathcal{B}_k$ wins in its $\mathrm{AUTH}_{\Pi}^{(\{P\},\{V\})\text{-auth}}$ game. At a high level, $\mathcal{B}_k$ simulates $\mathcal{A}$'s $\mathbf{T}$ queries during odd-numbered phases by using its own $\mathbf{P}$ oracle and the fact that Π is PCV. $\mathcal{B}_k$ simulates $\mathbf{V}$ queries during phases $2, 4, \ldots, 2k$ without using any oracles by simply sampling at random all the intermediate messages and returning always R (reject) as a reply to terminating queries. Finally $\mathcal{B}_k$ uses its $\mathbf{V}$ oracle for simulating phase $2(k+1)$ to $\mathcal{A}$. Details follow.

In the beginning of the simulation, $\mathcal{B}_k$ enters the first phase in its (2-phase) game where he has access only to $\mathbf{P}$. $\mathcal{B}_k$ simulates $\mathcal{A}$ as follows: It maintains a list L (initialized to $\emptyset$) containing all sids that it uses when simulating $\mathcal{A}$'s $\mathbf{T}$ and $\mathbf{P}$ queries during the first $2k+1$ phases of $\mathcal{A}$'s attack. On every $\mathbf{P}$-query (sid, start) by $\mathcal{A}$ where $sid \in SID_{\mathcal{P}}$, $\mathcal{B}_k$ simply queries its $\mathbf{P}$ oracle on (sid', msg) for some $sid' \notin L$, returns the answer to $\mathcal{A}$ and updates L by adding sid'. We assume that $\mathcal{B}_k$ maintains some map from $SID_{\mathcal{P}}$ to $SID_{\mathcal{P}}$ that associates the sids used in $\mathcal{A}$'s queries with the sid's that $\mathcal{B}_k$ uses when forwarding queries to its own $\mathbf{P}$ oracle. For every $\mathbf{T}$ query by $\mathcal{A}$, $\mathcal{B}_k$ produces a valid transcript by "engaging" to a full protocol execution with a prover (via $\mathbf{P}$ queries to its oracle). More formally, assume without loss of generality that Π is a r-round authentication protocol where $r = 2j$ is even, that is, the first message is sent from $\mathcal{V}$ to $\mathcal{P}$. $\mathcal{B}_k$ initializes $T \leftarrow \emptyset$ and selects a new $sid \in SID_{\mathcal{P}}$, that is, an sid that has not been previously used while simulating any $\mathbf{P}$ or $\mathbf{T}$ query issued by $\mathcal{A}$. In round 1 ($i = 1$), $\mathcal{B}_k$ simply samples msg_1 uniformly at random and adds msg_1 to T. It then queries its $\mathbf{P}$ oracle on (sid, msg_1) to get msg_2 back which it also adds to T. $\mathcal{B}_k$ proceeds in a similar fashion producing msg_{2j-1} by simply sampling them uniformly at random and msg_{2j} by querying its $\mathbf{P}$ oracle on (sid, msg_{2j-1}). After producing r messages, $\mathcal{B}_k$ returns $T = \{msg_1, \ldots, msg_r\}$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathbf{V}$ query (sid, msg) during an even-number phase $2i$ (with $i \leq k$), $\mathcal{B}_k$ replies as follows: If msg corresponds to an intermediate message in the protocol (that is a message different than the r-th), $\mathcal{B}_k$ simply samples a message for the appropriate distribution (recall that $\mathcal{B}_k$ can do so since Π is PCV) and returns it to $\mathcal{A}$. If msg is corresponds to the r-th message of Π, then $\mathcal{B}_k$ simply returns R to $\mathcal{A}$. The crucial observation is that $\mathcal{B}_k$ can perfectly simulate the first $2k+1$ phases of $\mathcal{A}$ without ever entering its second phase.

When $\mathcal{A}$ enters phase $2k+2$, then $\mathcal{B}_k$ enters phase 2 in its own game and uses its own $\mathbf{V}$ oracle to reply to $\mathcal{A}$'s queries. However all terminating queries by $\mathcal{A}$ are replied with R. Notice that $\mathcal{B}_k$ simulates perfectly G_{k+1} to $\mathcal{A}$ and that if the event $\mathsf{BAD} \leftarrow \mathtt{true}$ in phase $2(k+1)$ happens, then $\mathcal{B}_k$ wins in its $\mathrm{AUTH}_{\Pi}^{(\{P\},\{V\})\text{-auth}}$ game. Finally, $\mathcal{B}_k$ makes 1 query to its $\mathbf{P}$ oracle for each $\mathbf{P}$ and each $\mathbf{T}$ query by $\mathcal{A}$ (in any phase) and 1 query to its $\mathbf{V}$ oracle for each $\mathbf{V}$ query by $\mathcal{A}$ but only during phase $2(k+1)$. Also for a single $\mathbf{T}$ query by $\mathcal{A}$, $\mathcal{B}_k$ needs to sample $r/2$ new messages and the same holds for $\mathbf{V}$ queries by $\mathcal{A}$ during phases $2, 4, \ldots, 2k$. Therefore, $\mathcal{B}_k$ makes in total $\sum_{j=1}^{k+1}(q_{\mathsf{T},2j-1} + q_{\mathsf{P},2j-1})$ queries to its $\mathbf{P}$ oracle and $q_{\mathsf{V},2(k+1)}$ queries to its $\mathbf{V}$ oracle and runs in time $t_k = t + \mathcal{O}(r \cdot (\sum_{j=1}^{k+1} q_{\mathsf{T},2j-1} + \sum_{j=1}^{k-1} q_{\mathsf{V},2j}))$. $\square$

The proof of the theorem follows then easily by combining equations (8) and (9). □

On Man-In-the-Middle Security. Typically, the notion of MIM security considered in the literature is $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-security [33, 17]. Other works [7, 40] have also used (in a slightly different setting) $(\{\},\{\mathsf{P},\mathsf{V}\})$-security to measure resistance against MIM attacks. The following theorem shows that $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-security is a *strictly weaker* notion than $(\{\},\{\mathsf{P},\mathsf{V}\})$-security.

Theorem 3.7. $[(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\}) \not\Rightarrow (\{\},\{\mathsf{P},\mathsf{V}\})]$ *For any* $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$*, there exists a 2-round protocol* $\Pi = (\mathcal{K}, \mathcal{P}^F, \mathcal{V}^F)$ *such that*

- $\mathbf{Adv}_{\Pi}^{(\{\},\{\mathsf{P},\mathsf{V}\})\text{-auth}}(c,1,1) = 1$*, for a small constant* $c > 0$*, while*
- $\mathbf{Adv}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})\text{-auth}}(t, q_{\mathsf{P}}, q_{\mathsf{V},1}, q_{\mathsf{V},2}) \le \mathbf{Adv}_F^{\mathsf{prf}}(t', q_{\mathsf{P}} + q_{\mathsf{V},1} + q_{\mathsf{V},2}) + \frac{q_{\mathsf{V},2}(q_{\mathsf{P}}+q_{\mathsf{V},1}+q_{\mathsf{V},2}-1)}{2^n} + \frac{q_{\mathsf{V},2}}{2^m}$ *for all* $t, q_{\mathsf{P}}, q_{\mathsf{V},1}, q_{\mathsf{V},2}$*, where* $t' = t + \mathcal{O}(q_{\mathsf{P}} + q_{\mathsf{V},1} + q_{\mathsf{V},2})$*.*

Proof. Consider the 2-round protocol shown in Figure 6(a) where $K \xleftarrow{\$} \mathcal{K}$ is n k-bit string and $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ is a function. It is not hard to see that Π is not $(\{\},\{\mathsf{P},\mathsf{V}\})$-secure. Indeed, consider an adversary $\mathcal{A}$ that first queries $\mathbf{V}$ with (sid, start) to get r. $\mathcal{A}$ then queries $\mathbf{P}$ with (sid', r). Let $y = y' \,||\, 0$ the value returned. Then $\mathcal{A}$ queries $\mathbf{V}$ on $(sid, y' \,||\, 1)$. Clearly, $\mathcal{A}$ wins since, by the definition of the protocol, $\mathsf{pref}_m(y' \,||\, 1) = y' = F_K(r)$ and there is no matching conversation between $\mathbf{P}$ and $\mathbf{V}$ ($\mathbf{P}$ returns $y' \,||\, 0$ while $\mathcal{A}$ queries $\mathbf{V}$ on $y' \,||\, 1$). Therefore $\mathcal{A}$ wins with a single $\mathbf{P}$ and a single $\mathbf{V}$ query while running in a constant number of steps.

We now prove that Π is $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-secure. For the proof, fix an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ running in time t and making q_{P} and $q_{\mathsf{V},1}$, $q_{\mathsf{V},2}$ queries to its $\mathbf{P}$ and $\mathbf{V}$ oracles (in phases 1 and 2) respectively. We use the games shown in Figure 5 where, for compactness, we have omitted all checks for correct input format inside the code. We have also removed the condition for matching conversations, since $\mathbf{P}$ is not present in phase 2 of the attack. G_0 is exactly $\mathsf{AUTH}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})}$. By definition

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})\text{-auth}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] . \tag{10}$$

G_1 is similar to G_0 except that every call to F_K has been replaced by a call to a random function, i.e. a function that returns random m-bit strings when queried on fresh inputs while being consistent on previously queried inputs. We claim that there exists a prf-adversary $\mathcal{B}$ against F such that

$$\mathbf{Adv}_{\mathsf{F}}^{\mathsf{prf}}(\mathcal{B}) = \Pr\left[\, (\mathsf{PRF}^{\mathbf{F}})^{\mathcal{B}} \,\right] - \Pr\left[\, (\mathsf{PRF}^{\mathbf{R}})^{\mathcal{B}} \,\right] = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] . \tag{11}$$

$\mathcal{B}$ has access to $\mathbf{O}$ where $\mathbf{O}$ is either $\mathbf{F}$ or $\mathbf{R}$ and uses $\mathcal{A}$ as follows (we assume that, during the simulation, $\mathcal{B}$ performs all necessary checks and bookeeping and omit related details): in phase 1, on every $\mathbf{P}$ query (sid, msg) from $\mathcal{A}_1$, $\mathcal{B}$ queries $\mathbf{O}$ on msg and, upon receiving y', returns $y' \,||\, 0$ to $\mathcal{A}_1$. When $\mathcal{A}_1$ makes a (sid, msg) query to $\mathbf{V}$ in phase 1, if $msg = \mathsf{start}$, then $\mathcal{B}$ samples $r \xleftarrow{\$} \{0,1\}^n$ and sends it to $\mathcal{A}$. If $msg \neq \mathsf{start}$, $\mathcal{B}$ recovers r from $\mathsf{state}[id]$, parses msg as $y' \,||\, b$, queries $\mathbf{O}$ with r and accepts if and only if the result returned by $\mathbf{O}$ equals y'. $\mathbf{V}$ queries in phase 2 are answered the same way, except, if the value returned by $\mathbf{O}$ on input r matches y', $\mathcal{B}$ terminates and outputs 1. Otherwise, if $\mathcal{A}$ terminates without that happening, $\mathcal{B}$ terminates and outputs 0.

It is not hard to see that if $\mathbf{O} = \mathbf{F}$, then $\mathcal{B}$ simulates G_0 perfectly to $\mathcal{A}$ while if $\mathbf{O} = \mathbf{R}$, $\mathcal{B}$ simulates G_1. Also, notice that $\mathcal{B}$ outputs 1 when $\mathbf{O} = \mathbf{F}$ (resp. $\mathbf{O} = \mathbf{R}$) exactly when $\mathcal{A}$ wins G_0 (resp. G_1). This proves (11). Finally, the extra overhead of the simulation is constant per query. So $\mathcal{B}$ runs in $\mathcal{O}(q_{\mathsf{P}} + q_{\mathsf{V},1} + q_{\mathsf{V},2})$ more steps than $\mathcal{A}$.

It remains to bound $\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right]$. Let BAD be the event that during phase 2, a query (sid, start) to $\mathbf{V}$ returns a value r that has been either part of a (sid, r) query to $\mathbf{P}$ (in phase 1) or has been previously returned after an (sid, start) query to $\mathbf{V}$ (in either phase). Notice that the i-th $\mathbf{V}$ query ($i \in \{1, \ldots, q_{\mathsf{V},2}\}$)

procedure main: $//G_0, G_1$
$K \xleftarrow{\$} \mathcal{K}$
For all $x \in \{0,1\}^n$ do
  $V[x] \leftarrow \perp$
For all $sid \in \mathbb{N}$ do
  $\mathsf{state}[sid] = \epsilon$;
  $\mathsf{decision}[sid] = \perp$;
  $\mathsf{done}[sid] = \texttt{false}$

phase 1 : $\sigma_1 \xleftarrow{\$} \mathcal{A}_1^{\mathbf{P},\mathbf{V}}$
For all $sid \in \mathbb{N}$ do
  $\mathsf{state}[sid] = \epsilon$;
  $\mathsf{decision}[sid] = \perp$;
  $\mathsf{done}[sid] = \texttt{false}$

phase 2 : $\sigma \xleftarrow{\$} \mathcal{A}_2^{\mathbf{V}}(\sigma_1)$
Ret $\exists sid \in SID_{\mathcal{V}} : (\mathsf{decision}[sid] = \mathsf{A})$

oracle $\mathbf{P}(id, msg)$: $//G_0$
$y' \leftarrow F_K(msg)$
$y \leftarrow y' \,||\, 0$
$\mathsf{state}[id] \leftarrow msg \,||\, y$
$\mathsf{done}[id] \leftarrow \texttt{true}$
Ret y

oracle $\mathbf{V}(id, msg)$: $//G_0$
If $(msg = \mathsf{start})$
  $r \xleftarrow{\$} \{0,1\}^n$
  $\mathsf{state}[id] \leftarrow r$
  Ret r
Else
  $\mathsf{done}[sid] \leftarrow \texttt{true}$
  $y' \leftarrow \mathsf{pref}_m(msg)$
  $r \leftarrow \mathsf{state}[sid]$
  If $y' = F_K(r)$
    $\mathsf{decision}[id] \leftarrow \mathsf{A}$
    Ret A
  Ret R

oracle $\mathbf{P}(id, msg)$: $//G_1$
If $V[msg] = \perp$
  $V[msg] \xleftarrow{\$} \{0,1\}^m$
$y \leftarrow V[msg] \,||\, 0$
$\mathsf{state}[id] \leftarrow msg \,||\, y$
$\mathsf{done}[id] \leftarrow \texttt{true}$
Ret y

oracle $\mathbf{V}(id, msg)$: $//G_1$
If $(msg = \mathsf{start})$
  $r \xleftarrow{\$} \{0,1\}^n$; $\mathsf{state}[id] \leftarrow r$
  Ret r
Else
  $\mathsf{done}[id] \leftarrow \texttt{true}$
  $y' \leftarrow \mathsf{pref}_m(msg)$
  $r \leftarrow \mathsf{state}[sid]$
  If $V[r] = \perp$
    $V[r] \xleftarrow{\$} \{0,1\}^m$
  $z \leftarrow V[r]$
  If $y' = z$
    $\mathsf{decision}[sid] \leftarrow \mathsf{A}$
    Ret A
  Ret R

Figure 5: Sequence of games for the proof of Theorem 3.7

in phase 2 results in a *fresh* value r except with probability at most $(q_{\mathsf{P}} + q_{\mathsf{V},1} + i - 1)/2^n$. Using the union bound across all $q_{\mathsf{V},2}$ queries in phase 2, we obtain

$$\Pr[\,\mathsf{BAD}\,] \le \sum_{i=1}^{q_{\mathsf{V},2}} \frac{q_{\mathsf{P}} + q_{\mathsf{V},1} + i - 1}{2^n} \le \frac{q_{\mathsf{V},2}(q_{\mathsf{P}} + q_{\mathsf{V},1} + q_{\mathsf{V},2} - 1)}{2^n}\,.$$

Also, conditioned on $\neg\mathsf{BAD}$, every $(sid, y' \,||\, 0)$ query to $\mathbf{V}$ in phase 2, results on a freshly chosen $z \xleftarrow{\$} \{0,1\}^m$ and hence $y' \neq z$ except with probaility $1/2^m$. Taking the union bound across all $q_{\mathsf{V},2}$ queries to $\mathbf{V}$ in phase 2, we get

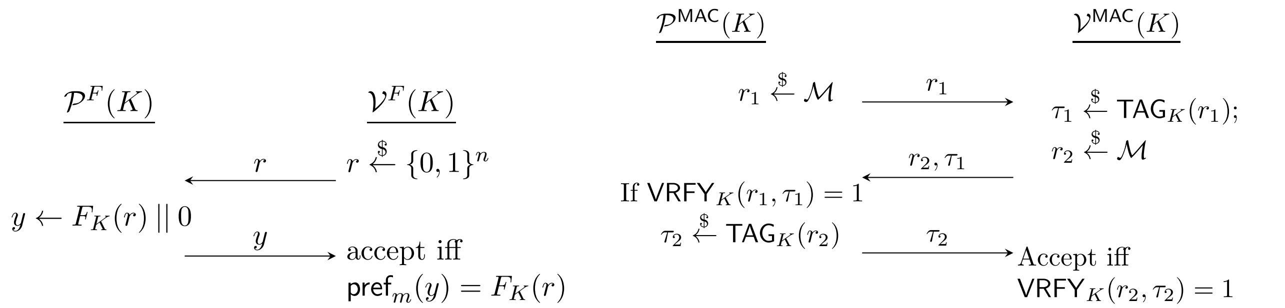
$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\middle|\, \neg\mathsf{BAD} \,\right] \le \frac{q_{\mathsf{V},2}}{2^m}$$

Therefore

$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] \le \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\middle|\, \neg\mathsf{BAD} \,\right] + \Pr[\,\mathsf{BAD}\,] \le \frac{q_{\mathsf{V},2}}{2^m} + \frac{q_{\mathsf{V},2}(q_{\mathsf{P}} + q_{\mathsf{V},1} + q_{\mathsf{V},2} - 1)}{2^n}\,. \quad (12)$$

The proof of Theorem 3.7 follows from (10), (11) and (12). □

One vs Multiple Verification Queries. Another commonly found folklore observation states that security for one verification query implies security for multiple verification queries (up to a linear in the number of verification queries decrease in the success probability). Once again, we show that this common belief is false. In fact, in Theorem 3.8, we show a much stronger statement: There exist three-round protocols which are secure against man-in-the-middle attacks, i.e. enjoy $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-security, when only one verification query is allowed in Phase 2, yet they are not even secure against a *verification-only* attack (i.e. $(\{\},\{\mathsf{V}\})$) merely consisting of *two* queries.



(a) 2-round protocol for $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\}) \not\Rightarrow (\{\},\{\mathsf{P},\mathsf{V}\})$ (Theorem 3.7)

(b) Protocol for 1 vs many verification queries separation (Thm. 3.8)

Figure 6: Protocols for Theorems 3.7 and 3.8

Theorem 3.8. [1 vs multiple verification queries] *Let* MAC *be a* suf-cma-*secure*[10] MAC *with message space* $\mathcal{M}$ *and completeness* 1. *Then, there exist* 3*-round protocols* $\Pi = (\mathcal{K}, \mathcal{P}^{\mathsf{MAC}}, \mathcal{V}^{\mathsf{MAC}})$ *such that*

- $\mathbf{Adv}_{\Pi}^{(\{\},\{\mathsf{V}\})\text{-auth}}(c, 2) = 1$, *for a small constant* $c > 0$, *while*
- $\mathbf{Adv}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})\text{-auth}}(t, q_{\mathsf{P}}, q_{\mathsf{V}}, 1) \le \mathbf{Adv}_{\mathsf{MAC}}^{\mathsf{suf\text{-}cma}}(t', q_{\mathsf{P}} + q_{\mathsf{V}} + 1, q_{\mathsf{P}} + q_{\mathsf{V}}) + \frac{q_{\mathsf{P}}+q_{\mathsf{V}}+1}{|\mathcal{M}|}$, *for all* $t, q_{\mathsf{P}}, q_{\mathsf{V}} > 0$, *where* $t' = t + \mathcal{O}(q_{\mathsf{P}} + q_{\mathsf{V}})$.

Proof. Let $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ be a suf-cma-secure mac with keyspace $\mathcal{K}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$. Consider the protocol Π as shown in Figure 6(b).

We first show that Π is insecure under a verification-only attack where the adversary can make 2 verification queries. Consider $\mathcal{A}$ that operates as follows: it first picks $r_1 \xleftarrow{\$} \mathcal{M}$ and queries $\mathbf{V}$ on (sid, r_1) (1^{st} query). Upon receiving r_2, τ_1, $\mathcal{A}$ simply makes a new $\mathbf{V}$-query (sid', r_2) where $sid' \neq sid$ (this is the 2^{nd} and final query by $\mathcal{A}$). Let r_3, τ_2 be the values returned. $\mathcal{A}$ then sends τ_2 as candidate for $\mathsf{TAG}_K(r_2)$, that is $\mathcal{A}$ makes a (sid, τ_2) query to $\mathbf{V}$. Notice that, by the perfect completeness of MAC, $\mathcal{A}$ brings the instance of $\mathcal{V}$ corresponding to id to accept, i.e. $\mathsf{decision}[id] = \mathsf{A}$.

We show however that Π is $(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})$-secure when a single verification query is allowed in phase 2. For that, consider an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ running in time t and making q_{P} and q_{V} queries to $\mathbf{P}$ and $\mathbf{V}$ respectively in phase 1 and a single verification query in phase 2. Let $G_0 = \mathsf{AUTH}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})\text{--auth}}$. By definition

$$\mathbf{Adv}_{\Pi}^{(\{\mathsf{P},\mathsf{V}\},\{\mathsf{V}\})\text{-auth}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] . \tag{13}$$

Now consider G_1 that differs only in the way the message in the second round of phase 2 is selected (in the description given above, this corresponds to message r_2 sent during the single $\mathbf{V}$ query of phase 2). G_1 maintains a list L initialized to $\emptyset$. For every $(sid, r_2 \,||\, \tau_1)$ query to $\mathbf{P}$ in phase 1, r_2 is added to L. Likewise, for every (sid, msg) query to $\mathbf{V}$ (where $\mathsf{state}[sid] = \varepsilon$) msg is added to L. In phase 2, the value r_1 from (id, r_1) (where $\mathsf{state}[sid] = \epsilon$ and sid is the unique $sid \in SID_{\mathcal{V}}$ queried in phase 2) is added to L. Finally, in phase 2, the unique query (id, msg) is answered by sampling $r^* \xleftarrow{\$} \mathcal{M} \setminus L$ (instead of $r^* \xleftarrow{\$} \mathcal{M}$ as per game G_0). Notice that $|L| \le q_{\mathsf{V}} + q_{\mathsf{P}} + 1$. Also, from an adversary's point of view, G_0 and G_1 are identical except from the distribution of r^*. It is not hard to see that for any $\mathcal{A}$ (even unbounded)

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \le \frac{|L|}{|\mathcal{M}|} \le \frac{q_{\mathsf{V}} + q_{\mathsf{P}} + 1}{|\mathcal{M}|} . \tag{14}$$

[10]Weaker notions of MACs are sufficient but we use suf-cma for ease of exposition.

We finally claim that the advantage of $\mathcal{A}$ in G_1 is upper bounded by the the advantage of some adversary $\mathcal{B}$ playing in game SUF-CMA. In particular, there exists adversary $\mathcal{B}$ such that

$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] \leq \mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{MAC}}(\mathcal{B}) \tag{15}$$

where $\mathcal{B}$ makes $q_{\mathsf{P}} + q_{\mathsf{V}} + 1$ **Tag** queries, $q_{\mathsf{P}} + q_{\mathsf{V}}$ **Vrfy** queries and the extra overhead of running $\mathcal{A}$ is $\mathcal{O}(q_{\mathsf{P}} + q_{\mathsf{V}})$ steps. $\mathcal{B}$ works as follows:[11] It first initiates a list $\mathrm{L} \leftarrow \emptyset$. In the first phase, on every (sid, start) query to $\mathbf{P}$, $\mathcal{B}$ picks $r_1 \xleftarrow{\$} \mathcal{M}$, sends it to $\mathcal{A}_1$ and updates $\mathsf{state}[sid]$ to r_1. On every (sid, msg) query (with $msg \neq \mathsf{start}$), $\mathcal{B}$ parses msg as $r_2 \,||\, \tau_1$, retrieves r_1 from $\mathsf{state}[sid]$ and queries its VRFY oracle on (r_1, τ_1). If the later query returns 1, then $\mathcal{B}$ queries its **Tag** oracle on r_2, sets $L \leftarrow L \cup \{r_2\}$ and returns the result (say τ_2) to $\mathcal{A}_1$. Replies to $\mathbf{V}$ queries during phase 1 proceed in a similar way. On a $\mathbf{V}$ query (sid, msg) from $\mathcal{A}_1$, $\mathcal{B}$ first checks that $\mathsf{state}[id] = \varepsilon$. If so, $\mathcal{B}$ sets $L \leftarrow L \cup \{msg\}$, queries its **Tag** oracle on msg to get τ_1, then samples $r_2 \xleftarrow{\$} \mathcal{M}$, returns (r_2, τ_1) to $\mathcal{A}_1$ and updates $\mathsf{state}[id]$ to $r_1 \,||\, \tau_1 \,||\, r_2$. If $\mathsf{state}[id] \neq \varepsilon$, then $\mathcal{B}$ recovers r_2 from $\mathsf{state}[id]$, queries its VRFY oracle on (r_2, msg) and, if VRFY returns 1, $\mathcal{B}$ sets $\mathsf{decision}[id] = \mathsf{A}$ and informs $\mathcal{A}_1$.

In phase 2, $\mathcal{A}_2$ can make only $\mathbf{V}$ queries and only for a *single id*. On such a query (id, msg), $\mathcal{B}$ checks again if $\mathsf{state}[id] = \varepsilon$ and if so, it queries its **Tag** oracle on msg and adds msg to L. Let τ_1 be the output of the **Tag** oracle. $\mathcal{B}$ then samples $r^* \xleftarrow{\$} \mathcal{M} \setminus L$, sends (r^*, τ_1) to $\mathcal{A}_2$ and updates $\mathsf{state}[id]$ to $r_1 \,||\, \,||\, \tau_1 \,||\, r_2$. If $\mathsf{state}[sid] \neq \varepsilon$, $\mathcal{B}$ retrieves r^* from $\mathsf{state}[sid]$, and outputs (r^*, msg) as the candidate forgery. This completes the description of the simulation.

For $\mathcal{B}$'s running time, notice that, in phase 1, for every $\mathbf{P}$ query from $\mathcal{A}$, $\mathcal{B}$ makes one **Tag** query and one **Vrfy** query and the same holds for every $\mathbf{V}$ query by $\mathcal{A}$. Finally, $\mathcal{B}$ makes a single **Tag** query in phase 2. Therefore, $\mathcal{B}$ makes $q_{\mathsf{T}} + q_{\mathsf{V}} + 1$ queries to **Tag** and $q_{\mathsf{T}} + q_{\mathsf{V}}$ queries to **Vrfy**. We conclude by analyzing $\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{MAC}}(\mathcal{B})$. It is straightforward to check that $\mathcal{B}$ simulates perfectly G_1. Also notice that at the end of the simulation, $\mathcal{B}$ outputs a candidate forgery (r^*, τ^*) for a message r^* that has never been an input in any previous **Tag** query (recall that L contains all such queries and $\mathcal{B}$ explicitly picks $r^* \notin L$). The above justify (15). The proof of Theorem 3.8 follows from (13), (14) and (15). □

Interestingly, the protocol used in the proof of Theorem 3.8 deviates from the (commit-)challenge-response paradigm. In particular, the message sent from the verifier to the prover (in round 2) depends on the shared secret. Theorem 3.9 essentially states that any authentication protocol Π for which security under multiple verification queries separates from security under a single verification query *should* involve key-dependent messages from the verifier to the prover. In other words, if all (intermediate) messages from $\mathcal{V}$ to $\mathcal{P}$ in Π are independent of K and $\mathcal{P}$'s messages, then multiple verification queries *do not* increase by much the success probability of an adversary in breaking the security of Π.

Theorem 3.9. [1 vs Many Verification Queries for Public-Coin Verifier Protocols] *Let* Π *be any Public-Coin Verifier* (PCV) r*-round authentication protocol with completeness 1. Then* $\forall\, S_1 \subseteq \{\mathsf{T}, \mathsf{P}, \mathsf{V}\}$ *and*[12] $S_2 \subseteq \{\mathsf{P}, \mathsf{V}\}$,

$$\mathbf{Adv}^{(S_1,S_2)\text{-auth}}_{\Pi}(t, q_{\mathsf{T},1}, q_{\mathsf{P},1}, q_{\mathsf{V},1}, q_{\mathsf{P},2}, q_{\mathsf{V},2}) \leq q_{\mathsf{V},2} \cdot \mathbf{Adv}^{(S_1,S_2)\text{-auth}}_{\Pi}(t, q_{\mathsf{T},1}, q_{\mathsf{P},1}, q_{\mathsf{V},1}, q_{\mathsf{P},2}, 1), \tag{16}$$

where $t' = t + q_{\mathsf{V},2} \cdot \mathcal{O}(r + q_{\mathsf{P},2})$.

[11] In order to keep the core steps of the simulation clean, in the description we omit details such as checking that the input is in the correct format or that the *id*s queried are valid.

[12] We assume without loss of generality that the $\mathbf{T}$ oracle is not present in phase 2. Since queries to $\mathbf{T}$ are not adaptive (in particular there is no input provided by the adversary), we may assume that the adversary makes all its queries to $\mathbf{T}$ in phase 1 without the security of the underlying protocol being affected.

Proof. Let $\mathsf{\Pi}$ be a Public-Coin Verifier authentication protocol. We will show that, for any $S_1, S_2 \subseteq \{\mathsf{T}, \mathsf{P}, \mathsf{V}\}$ and any adversary $\mathcal{A}$ against the (S_1, S_2)-security of $\mathsf{\Pi}$ making $q_{\mathsf{V},2}$ queries to $\mathbf{V}$ during phase 2, there exists an adversary $\mathcal{B}$ against the (S_1, S_2)-security of $\mathsf{\Pi}$ that makes a single query to $\mathbf{V}$ in phase 2 and succeeds with probability which is smaller than $\mathcal{A}$'s success probabilty by at most $q_{\mathsf{V},2}$. At a high level, $\mathcal{B}$ picks a random $sid \in SID_{\mathcal{V}}$ out of the $q_{\mathsf{V},2}$ *sid*s that $\mathcal{A}$ uses in its $\mathbf{V}$ queries and sets that as its target *sid* (the one to be used in the single $\mathbf{V}$-query $\mathcal{B}$ is allowed to make during phase 2). $\mathcal{B}$ simulates the answers to the queries corresponding to the rest of the *sid*s using the fact $\mathsf{\Pi}$ is PCV. However, in order for $\mathcal{B}$ to faithfully simulate $\mathrm{AUTH}_{\mathsf{\Pi}}^{(S_1,S_2)\text{-auth}}$ to $\mathcal{A}$, extra care needs to be taken when $\mathcal{A}$ makes a query (sid, msg) for a terminal message *msg* (that is the message corresponding to round r). More specifically, $\mathcal{B}$ should be able to detect (and reply accordingly in) the event of matching conversations between instances of $\mathcal{P}$ and $\mathcal{V}$. Details follow.

Without loss of generality we assume that the protocol is initiated by the verifier, i.e. *calV* gets a message (sid, start). (The proof can be easily adapted to the case where the protocol is initiated by the prover.) Also, as always, we assume that the last message is sent from the prover to the verifier. $\mathcal{B}$ simulates $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as follows: During phase 1, $\mathcal{B}$ simply forwards the queries from $\mathcal{A}$ to its corresponding oracles ($\mathcal{B}$ has access to the exact same oracles as $\mathcal{A}$ and can query them as many times as $\mathcal{A}$ during phase 1). After the end of phase 1, $\mathcal{B}$ picks at random $r \xleftarrow{\$} \{1, \ldots, q_{\mathsf{V},2}\}$ and initiates a set $I \leftarrow \emptyset$ and a string $sid^* \leftarrow \perp$. It also resets all global variables ($\mathsf{state}, \mathsf{decision}, \mathsf{done}$ etc.) and enters the simulation of phase 2. Queries by $\mathcal{A}$ to $\mathbf{P}$ are answered in the exact same way as in phase 1 (recall that $\mathcal{B}$ can make the same number of queries as $\mathcal{A}$ to its $\mathbf{P}$ oracle). Along the simulation, $\mathcal{B}$ also records the queries (sid, msg) to $\mathbf{P}$ for $sid \in SID_{\mathcal{P}}$ as well as the messages msg' returned as replies to these queries. Queries to $\mathbf{V}$ during phase 2 are answered differently: let (sid, start) be the starting query to $\mathcal{V}$ issued by $\mathcal{A}$. $\mathcal{B}$ first adds *sid* to I and then checks whether $r = |I|$ and if so, $\mathcal{B}$ sets $sid^* \leftarrow sid$ (this will be the single *sid* on which $\mathcal{B}$ will be querying its $\mathbf{V}$ oracle). To determine the reply to any query (sid, msg), we distinguish the following 2 cases:

1. $id = sid^*$: In that case $\mathcal{B}$ simply forwards the query to its $\mathbf{V}$ oracle and returns the result to $\mathcal{A}$. If *msg* corresponds to the last message sent from the prover to the verifier and $\mathcal{B}$'s $\mathbf{V}$ oracle returns A, then $\mathcal{B}$ terminates.

2. $id \neq sid^*$: If *msg* corresponds to an intermediate message (i.e. a message other than the last message sent from the prover to the verifier), then $\mathcal{B}$ simply samples msg' from the correct distribution (this is possible due to the fact that $\mathsf{\Pi}$ is PCV), returns the result to $\mathcal{A}$ and performs all necessary bookeeping. The trickiest part of the simulation is when *msg* is the last message sent to the verifier by the prover. In that case, $\mathcal{B}$ needs to be consistent and simulate perfectly $\mathbf{V}$. At a given time t, we say that an *sid* is *live* in phase i if $\mathcal{A}_i$ has previously made a query $(sid, *)$. For any query (sid, msg) to $\mathbf{V}$, where *msg* is a terminating message, $\mathcal{B}$ computes the predicate $\mathsf{Matching}(T[sid'], T[sid])$ for all live $sid' \in SID_{\mathcal{P}}$. If there exists $sid' \in SID_{\mathcal{P}}$ such that $\mathsf{Matching}(T[sid'], T[sid])$ is $\mathtt{true}$, then $\mathcal{B}$ returns A to $\mathcal{A}$ and sets $\mathsf{decision}[sid] \leftarrow \mathsf{A}$.[13] If no such sid' exists, $\mathcal{B}$ simply returns R.

Let F be the random variable taking the value of the first (chronologically) $id \in SID_{\mathcal{V}}$ such that: (a) $\mathsf{decision}[sid] = \mathsf{A}$ after a (sid, msg) query (for a terminal message *msg*) AND (b) for all $sid' \in SID_{\mathcal{P}}$ that are live up to that point $\neg\mathsf{Matching}(T[sid'], T[sid])$.[14] If no such *sid* exists, then F takes the special value $\perp$. Notice that, by definition $\Pr[\, F \neq \perp \,] = \mathbf{Adv}_{\mathsf{\Pi}}^{(S_1,S_2)\text{-auth}}(\mathcal{A})$. Also, if $\mathcal{B}$ has guessed F correctly, that

[13] Notice that, due to the fact that $\mathsf{Matching}(T[sid'], T[sid])$ is true for some $sid' \in SID_{\mathcal{P}}$, $\mathcal{A}$ does *not* necessarily win $\mathrm{AUTH}_{\mathsf{\Pi}}^{(S_1,S_2)\text{-auth}}$. However, by having $\mathcal{B}$ return A to $\mathcal{A}$ in such an event, we guarantee that $\mathcal{B}$ simulates $\mathbf{V}$ properly.

[14] The ability to simulate perfectly all queries before the terminal query corresponding to sid^* lies also in the heart of a similar argument by Bellare *et al.* [5]. In particularly, [5] show that (unlike $\mathsf{uf\text{-}cma}$-secure MACs), for $\mathsf{suf\text{-}cma}$-secure MACs, multiple verification queries do not help much.

Game UF-RMRC$_{\mathsf{MAC}}$	oracle Tag():	oracle Vrfy(m,τ):
procedure main:	$m \xleftarrow{\$} \mathcal{M}$	If $m \notin C$
$K \xleftarrow{\$} \mathsf{KGen}$	$\tau \leftarrow \mathsf{TAG}_K(m)$	Ret $\perp$
$\mathsf{Forge} \leftarrow \mathtt{false}$	Ret (m,τ)	$C \leftarrow C \setminus \{m\}$
$C \leftarrow \emptyset$	oracle Chal():	If $\mathsf{VRFY}_K(m,\tau) = 1$
Run $\mathcal{A}_{\mathsf{MAC}}^{\mathbf{Tag},\mathbf{Chal},\mathbf{Vrfy}}$	$m \xleftarrow{\$} \mathcal{M}$	$\mathsf{Forge} \leftarrow \mathtt{true}$
Ret Forge	$C \leftarrow C \cup \{m\}$	Ret 1
	Ret m	Ret 0

Game LPN$_{n,\eta}$	oracle Sample():
procedure main:	$\mathbf{a} \xleftarrow{\$} \mathbb{Z}_2^n$
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$	$e \leftarrow \mathsf{Ber}_\eta$
$d \leftarrow \mathcal{A}^{\mathbf{Sample}}$	Ret $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + e)$

Figure 7: Left: Game UF-RMRC$_{\mathsf{MAC}}$. Right: Game LPN$_{n,\eta}$.

is $F = sid^*$, then the simulation provided to $\mathcal{A}$ from $\mathcal{B}$ is perfect. Indeed, all queries to $\mathbf{P}$ are answered using $\mathcal{B}$'s $\mathbf{P}$ oracle. Also, the fact that Π is PCV allows $\mathcal{B}$ to answer correctly all queries of the form (sid, msg) to $\mathbf{V}$ for non-terminating messages. Finally, consider a (sid, msg) query to $\mathbf{V}$ for terminating msg. By definition of the random variable F, for a query (sid, msg), either their exists $sid' \in SID_{\mathcal{P}}$ such that $\mathsf{Matching}(T[sid'], T[sid])$ (which can be detected by $\mathcal{B}$ and answered properly) or the query should be rejected. In either case $\mathcal{B}$ provides the correct answer. Therefore

$$\mathbf{Adv}_{\Pi}^{(S_1,S_2)\text{-auth}}(\mathcal{B}) = \Pr\left[\, sid^* = F \wedge F \neq \perp \,\right] = \Pr\left[\, sid^* = F \;\middle|\; F \neq \perp \,\right] \cdot \Pr\left[\, F \neq \perp \,\right] = \frac{\mathbf{Adv}_{\Pi}^{(S_1,S_2)\text{-auth}}(\mathcal{A})}{q_{\mathsf{V},2}}$$

Finally, the extra overhead simulating $\mathcal{A}$ comes from the $\mathbf{V}$ queries in phase 2. For every such query, $\mathcal{B}$ needs to sample $\mathcal{O}(r)$ messages and check accross all live $sid \in SID_{\mathcal{P}}$ for matching conversations. Therefore, $\mathcal{B}$ runs in time $t' = t + q_{\mathsf{V},2} \cdot \mathcal{O}(r + q_{\mathsf{P},2})$. ☐

4 Generic Constructions of Authentication Protocols

In this section, we present generic constructions of authentication protocols based on weak variants of MACs, and which deviate from the challenge-response paradigm. We start by presenting a 3-round $(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})$-secure (i.e., actively secure) protocol in Section 4.1, for which we also present efficient instantiations from LPN and CDH. In Section 4.2, we devise a 2-round protocol that attains the strongest notion of MIM security, namely $(\{\},\{\mathsf{P},\mathsf{V}\})$-security. We also present an efficient instantiation based on qSDH.

4.1 Active Security from Random-Message-Random-Challenge-Secure MACs

For a MAC protocol $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$, we define unforgeability under *random message-random challenge* attacks (uf-rmrc) via the game UF-RMRC depicted on the left of Figure 7: **Tag** queries return pairs $(m, \mathsf{TAG}_K(m))$ for fresh *random* messages m. Moreover, **Vrfy** queries are only allowed if of the form (m,τ) for m previously output by the random challenge generator oracle **Chal**, and only a single verification query to **Vrfy** per valid challenge is allowed. For $t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}} > 0$, the uf-rmrc advantage function is defined as

$$\mathbf{Adv}_{\mathsf{MAC}}^{\mathsf{uf\text{-}rmrc}}(t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}}) = \max_{\mathcal{A}}\{\Pr\left[\, (\text{UF-RMRC}_{\mathsf{MAC}})^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right]\} ,$$

where the maximum is over all adversaries $\mathcal{A}$ running in time t and making q_{T}, q_{C} and q_{V} queries to **Tag**, **Chal** and **Vrfy**, respectively.

The DM protocol. Our new 3-round authentication protocol, $\mathsf{DM}[\mathsf{MAC}] = (\mathcal{K}, \mathcal{P}, \mathcal{V})$ (DM stands for Double Mac) proceeds as follows where K_1, K_2 are generated using KGen.

$\underline{\mathcal{P}(K_1, K_2)}$		$\underline{\mathcal{V}(K_1, K_2)}$
$r_1 \xleftarrow{\$} \mathcal{M}$	$\xrightarrow{\quad r_1 \quad}$	$\tau_1 \xleftarrow{\$} \mathsf{TAG}_{K_1}(r_1)$; $r_2 \xleftarrow{\$} \mathcal{M}$
If $\mathsf{VRFY}_{K_1}(r_1, \tau_1) = 1$	$\xleftarrow{\quad r_2, \tau_1 \quad}$	
$\tau_2 \xleftarrow{\$} \mathsf{TAG}_{K_2}(r_2)$	$\xrightarrow{\quad \tau_2 \quad}$	Accept iff $\mathsf{VRFY}_{K_2}(r_2, \tau_2) = 1$

The intuition behind the proof is fairly simple: Each prover instance commits to a value r_1, and hence, in order for the prover to do something useful for an active adversary, such as tagging an arbitrary message under K_2, the attacker must provide a valid tag for r_1 under K_1. Yet, the attacker can only obtain valid tags through the transcript oracle in the first phase, and the used r_1' values are very unlikely to collide with one of the values the prover instances commit to. Hence, with very high probability, the attacker never goes past the second round when interacting with the prover. The proof of the following theoremformalizes this intuition, but requires some care, mainly due to the interplay between the roles of the keys K_1 and K_2 in the reduction.

Theorem 4.1. [Security of DM] *For all $t, q_{\mathsf{T}}, q_{\mathsf{P}}, q_{\mathsf{V}} > 0$,*

$$\mathbf{Adv}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}_{\mathsf{DM}}(t, q_{\mathsf{T}}, q_{\mathsf{P}}, q_{\mathsf{V}}) \le \mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}}(t_1, q_{\mathsf{T}}, q_{\mathsf{P}}, q_{\mathsf{P}}) + \mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}}(t_2, q_{\mathsf{T}}, q_{\mathsf{V}}, q_{\mathsf{V}}) \qquad (17)$$

where $t_1 = t + \mathcal{O}(q_{\mathsf{T}} \cdot t_{\mathsf{TAG}})$, $t_2 = t + \mathcal{O}((q_{\mathsf{T}} + q_{\mathsf{V}}) \cdot t_{\mathsf{TAG}})$ and t_{TAG} is the time to evaluate a single tag.

Proof. The proof uses the games G_0 and G_1, whose main procedure and oracles are described in Figure 8. In order to avoid overloading our presentation, we omit the checks for correct input format in all games and assume that any input in incorrect format results in $\perp$. Also, throughout this proof, let us fix an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ making q_{T} queries to $\mathbf{T}$, q_{P} queries to $\mathbf{P}$, q_{V} queries to $\mathbf{V}$ and running in t steps.

Game G_0 is a compact representation of $\mathrm{AUTH}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})}_{\mathsf{DM}}$, without all unnecessary steps. In particular, because $S_2 = \{\mathsf{V}\}$, the "no matching conversation" condition trivially holds ($\mathbf{P}$ is not present during phase 2). Therefore,

$$\mathbf{Adv}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}_{\mathsf{DM}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] . \qquad (18)$$

Moreover, note that the game G_0, whenever a *valid* query $msg = r_2 \,||\, \tau_1$ is made to $\mathbf{P}$ in the second round, it sets the flag BAD if τ_1 is a valid tag for $\mathsf{state}[sid] = r_1$ sent in the first round, i.e., $\mathsf{VRFY}_{K_1}(\mathsf{state}[sid], \tau_1) = 1$. The second game, Game G_1, is identical to G_0, with the sole difference that no query (sid, msg) with $msg \neq \mathsf{start}$ made to $\mathbf{P}$ is accepted, i.e., they are all replied with $\perp$. The following claim bounds the difference between the probabilities of $\mathcal{A}$ winning games G_0 and G_1, respectively.

Claim 4.2. *There exists an adversary $\mathcal{B}$ such that*

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \le \mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}}(\mathcal{B}) . \qquad (19)$$

In particular, $\mathcal{B}$ makes q_{T} queries to **Tag** *and q_{P} queries to* **Chal** *and* **Vrfy***, and runs in time $t' = t + \mathcal{O}(q_{\mathsf{T}} \cdot t_{\mathsf{TAG}})$, where t_{TAG} is the time needed to evaluate* TAG.

```
procedure main:                                  //G_0, G_1
K_1, K_2 <-$ KGen
For all sid ∈ ℕ do
    state[sid] = ε; decision[sid] = ⊥
    done[sid] = false
σ_1 <-$ A_1^{P,T}                                // Phase 1
A_2^V(σ_1)                                       // Phase 2
Ret (∃sid ∈ SID_V : decision[sid] = A)

oracle V(sid, msg):                              // G_0, G_1
If (sid ∉ SID_V) ∨ done[sid] then
    Ret ⊥
If state[sid] = ϵ then                           // 2nd round
    τ_1 <-$ TAG_{K_1}(msg)
    state[sid] <-$ M
    Ret state[sid] || τ_1
Else                                             // decision
    done[sid] <- true
    If VRFY_{K_2}(state[sid], msg) = 1 then
        decision[sid] <- A
    Else
        decision[sid] <- R
    Ret decision[sid]

oracle T():                                      // G_0, G_1
r_1 <-$ M; τ_1 <-$ TAG_{K_1}(r_1)
r_2 <-$ M; τ_2 <-$ TAG_{K_2}(r_2)
Ret (r_1, (τ_1, r_2), τ_2)

oracle P(id, msg):                               // G_0, [G_1]
If (sid ∉ SID_P) ∨ done[sid] then
    Ret ⊥
If state[sid] = ε then                           // 1st round
    If msg ≠ start then
        Ret ⊥
    state[sid] <-$ M
    Ret state[sid]
Else                                             // 3rd round
    done[id] <- true
    r_2 || τ_1 <- msg
    If VRFY_{K_1}(state[id], τ_1) = 1 then
        BAD <- true
        τ_2 <-$ TAG_{K_2}(r_2)
        [τ_2 <- ⊥]
        Ret τ_2
    Ret ⊥.
```

Figure 8: **Games G_0 and G_1 used in the proof of Theorem 4.1.** Above, $a \,||\, b \leftarrow msg$ denotes the operation of parsing the string msg as the concatenation of the strings a and b of understood lengths.

Proof. First note that G_0 and G_1 are equivalent-until-bad. By the fundamental lemma of game playing,

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] \leq \Pr\left[\, G_1^{\mathcal{A}} \text{ sets } \mathsf{BAD} \,\right] \ .$$

We now construct the adversary $\mathcal{B}$ for the UF-RMRC$_{\mathsf{MAC}}$ game. The crucial observation here is that as long as we are only concerned about the probability of BAD being set, we only need to look at the first phase of the game, which will in particular avoid the reduction simulating the second phase. The adversary $\mathcal{B}$ simulates the interaction of the adversary $\mathcal{A}_1$ in the *first* phase of the game $\mathrm{AUTH}_{\mathsf{DM}}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})}$ as follows: First, it selects $K_2 \xleftarrow{\$} \mathsf{KGen}$. Upon receiving a $\mathbf{T}$ query from $\mathcal{A}_1$, $\mathcal{B}$ makes a query to its **Tag** oracle to get a pair r_1, τ_1 and also computes (r_2, τ_2) by sampling $r_2 \xleftarrow{\$} \mathcal{M}$ and setting $\tau_2 \xleftarrow{\$} \mathsf{TAG}_{K_2}(r_2)$ (recall that $\mathcal{B}$ can compute $\mathsf{TAG}_{K_2}(\cdot)$ using the K_2 it has chosen for the simulation). It then returns $(r_1, (\tau_1, r_2), \tau_2)$ as a transcript to $\mathcal{A}_1$. On every $\mathbf{P}$ query (sid, start), $\mathcal{B}$ makes a query to its **Chal** oracle, which returns a message message r_1. Then, $\mathcal{B}$ sends r_1 to $\mathcal{A}_1$ and upon receiving (sid, τ_1, r_2), for the same sid, $\mathcal{B}$ sends τ_1 as a forgery for r_1 (that is, $\mathcal{B}$ makes a query (r_1, τ_1) to its **Vrfy** oracle), but returns $\perp$ to $\mathcal{A}_1$. It is straightforward to verify that $\mathcal{B}$ simulates perfectly the first phase of G_1 to $\mathcal{A}_1$, and that the probability that BAD is set is exactly the probability that $\mathcal{B}$ forges. Finally, for the simulation, $\mathcal{B}$ calls its **Tag** oracle q_{T} times, and its **Chal** and **Vrfy** oracles, each, q_{P} times, and also needs to compute q_{T} tags by itself. ☐

To conclude the proof, we reduce the problem of $\mathcal{A}$ winning the game G_1 to forging MAC in the game UF-RMRC$_{\mathsf{MAC}}$. Specifically, we build an adversary $\mathcal{C}$ such that

$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] = \Pr\left[\, \text{UF-RMRC}_{\mathsf{MAC}}^{\mathcal{C}} \Rightarrow \texttt{true} \,\right] \ . \tag{20}$$

The adversary $\mathcal{C}$ simulates an interaction of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with the game G_1 as follows: It first chooses $K_1 \xleftarrow{\$} \mathsf{KGen}$. When $\mathcal{A}_1$ makes a query to $\mathbf{T}$, $\mathcal{C}$ first generates $r_1 \xleftarrow{\$} \mathcal{M}$ and $\tau_1 \xleftarrow{\$} \mathsf{TAG}_{K_1}(r_1)$, then samples a pair (r_2, τ_2) by querying its own **Tag** oracle and finally returns $(r_1, (r_2, \tau_1), \tau_2)$ to $\mathcal{A}$. Moreover, every query (sid, msg) to $\mathbf{P}$ is replied as follows: If $msg = \mathsf{start}$, then $\mathcal{C}$ simply samples $r \xleftarrow{\$} \mathcal{M}$ and returns it to $\mathcal{A}_1$. If $msg = r' \,||\, \tau'$, $\mathcal{C}$ replies with $\perp$. Finally, whenever $\mathcal{A}_2$ makes a query (sid, r_1) to $\mathbf{V}$, $\mathcal{C}$ queries its **Chal** oracle, obtaining a value r_2. It then samples $\tau_1 \xleftarrow{\$} \mathsf{TAG}_{K_1}(r_1)$, and returns $r_2 \,||\, \tau_1$. If $\mathcal{A}_2$ queries $\mathbf{V}$ again for the same sid, with a value τ_2, then $\mathcal{C}$ submits (r_2, τ_2) to **Vrfy**, and returns the outcome to $\mathcal{A}_2$. It is not hard to see that the probability that $\mathcal{C}$ forges is exactly the probability that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins the game G_1. Also, $\mathcal{C}$ has running time $t_2 = t + \mathcal{O}((t_{\mathsf{T}} + t_{\mathsf{V}}) \cdot t_{\mathsf{TAG}})$, and makes q_{T} queries to **Tag** and q_{V} queries to **Chal** and **Vrfy**. □

It is worth mentioning that the security of DM is based on a very weak assumption. Previous generic constructions require either a much stronger MAC allowing for chosen-message queries, and giving a challenge-response protocol directly, or a weak PRF [17], which is a strictly stronger assumption, as a weak PRF yields a (deterministic) $\mathsf{uf\text{-}rmrc}$-secure MAC. Also, in contrast to the weak-PRF based protocol of [17], our proof avoids rewinding, hence yielding an essentially tight reduction.

Instantiation from LPN. We instantiate DM using the Learning Parity with Noise (LPN) assumption. (A further instantiation using Ring-LWE is presented in Appendix A.) In the game $\mathrm{LPN}_{n,\eta}$ (shown on the right of Figure 7), the **Sample** oracle, given a secret $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$, returns pairs $(\mathbf{a}, \mathbf{as} + e)$ for a random $\mathbf{a} \in \mathbb{Z}_2^n$ and $e \xleftarrow{\$} \mathsf{Ber}_\eta$ upon each invocation, where $\mathbf{as}$ denotes scalar product in $\mathbb{Z}_2$. The (decisional) LPN is the problem of distinguishing $\mathrm{LPN}_{n,\eta}$ from $\mathrm{LPN}_{n,1/2}$. For $t, q > 0$, we define the lpn advantage function as

$$\mathbf{Adv}^{\mathsf{lpn}}_{n,\eta}(t, q) = \max\left\{\Pr\left[\,\mathrm{LPN}^{\mathcal{A}}_{n,\eta} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{LPN}^{\mathcal{A}}_{n,1/2} \Rightarrow 1\,\right]\right\} \tag{21}$$

where the maximum is taken over all adversaries $\mathcal{A}$ running in time t and making q queries to the **Sample** oracle.

We now define $\mathsf{MAC}_{\mathrm{LPN}} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ with keyspace $\mathcal{K} = \mathbb{Z}_2^n$, message space $\mathcal{M} = \mathbb{Z}_2^{m \times n}$ and tag space $\mathcal{T} = \mathbb{Z}_2^m$, parametrized by constants η, η' such that $0 < \eta < \eta' < 1/2$:

$\underline{\mathsf{KGen}:}$	$\underline{\mathsf{TAG}(\mathbf{s}, \mathbf{A}):}$	$\underline{\mathsf{VRFY}(\mathbf{s}, \mathbf{A}, \mathbf{t}):}$
Ret $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$.	$\mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\eta^m$; Ret $\mathbf{t} = \mathbf{As} + \mathbf{e}$.	If $\mathsf{hw}(\mathbf{t} - \mathbf{As}) < \eta' \cdot m$ then Ret 1 else Ret 0.

The expected hamming weight of the vector $\mathbf{t} - \mathbf{As}$ is $\eta \cdot m$. Therefore the completeness error of $\mathsf{MAC}_{\mathrm{LPN}}$ can be upper bounded by the Chernoff bound (1) as

$$\epsilon_c = \Pr\left[\,\mathsf{hw}(\mathbf{e}) > \eta' \cdot m\,\right] \leq 2^{-\mathrm{D}(\eta' \,||\, \eta)m}\ . \tag{22}$$

The following lemma states that $\mathsf{MAC}_{\mathrm{LPN}}$ is $\mathsf{uf\text{-}rmrc}$-secure assuming LPN is hard. Its proof uses ideas similar to the ones used in the proof that the HB protocol is secure against passive attacks [32]. Similar ideas are also implicit in the LPN-based randomized weak PRF construction by Applebaum *et al* [1].

Lemma 4.3. **[Security of $\mathsf{MAC}_{\mathrm{LPN}}$]** *Let $\bar{\eta} = \eta + \eta' - 2\eta\eta'$ and η'' such that*[15] $0 < \bar{\eta} < \eta'' < 1/2$. *Then, for all $t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}} > 0$,*

$$\mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}_{\mathrm{LPN}}}(t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}}) \leq \mathbf{Adv}^{\mathsf{lpn}}_{n,\eta}(t', q) + q_{\mathsf{V}} \cdot \left(2^{-\mathrm{D}(\eta'' \,||\, \bar{\eta})m} + 2^{-(1-\mathrm{H}_2(\eta''))m}\right)\ ,$$

where $t' = t + \mathcal{O}(q_{\mathsf{C}})$ and $q = (q_{\mathsf{T}} + q_{\mathsf{C}}) \cdot m$.

[15] For constants $\eta, \eta' \in (0, 1/2)$, a constant η'' within that range always exists.

procedure main: $//G_0 - G_4$
Let $\bar{\eta} = \eta + \eta' - 2\eta\eta'$
$\eta'' \in (\bar{\eta}, 1/2)$
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$
$\mathsf{Forge} \leftarrow \mathtt{false}$
$C \leftarrow \emptyset$
Run $\mathcal{A}^{\mathbf{Tag},\mathbf{Chal},\mathbf{Vrfy}}$
Ret Forge

oracle Chal(): $//G_0 - G_4$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n}$
$C \leftarrow C \cup \{\mathbf{A}\}$
Ret $\mathbf{A}$

oracle Vrfy$(\mathbf{A}^*, \mathbf{t}^*)$: $//G_0$
If $\mathbf{A}^* \notin C$
  Ret $\perp$
$C \leftarrow C \setminus \{\mathbf{A}^*\}$
$\mathbf{e}^* \leftarrow \mathsf{Ber}_\eta^m$
If $\mathsf{hw}(\mathbf{t}^* - \mathbf{A}^*\mathbf{s}) \le \eta' m$
  $\mathsf{Forge} \leftarrow \mathtt{true}$
  If $\mathsf{hw}(\mathbf{t}^* - \mathbf{A}^*\mathbf{s} - \mathbf{e}^*) > \eta'' m$
    $\mathsf{BAD} \leftarrow \mathtt{true}$
  Ret 1
Ret 0

oracle Tag(): $//G_0 - G_3$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n}$
$\mathbf{e} \leftarrow \mathsf{Ber}_\eta^m$
$\mathbf{t} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
Ret $(\mathbf{A}, \mathbf{t})$

oracle Vrfy$(\mathbf{A}^*, \mathbf{t}^*)$: $//G_1$, $\boxed{G_2}$
If $\mathbf{A}^* \notin C$
  Ret $\perp$
$C \leftarrow C \setminus \{\mathbf{A}^*\}$
$\mathbf{e}^* \leftarrow \mathsf{Ber}_\eta^m$
If $\mathsf{hw}(\mathbf{t}^* - \mathbf{A}^*\mathbf{s}) \le \eta' m$
  $\mathsf{Forge} \leftarrow \mathtt{true}$
  If $\mathsf{hw}(\mathbf{t}^* - \mathbf{A}^*\mathbf{s} - \mathbf{e}^*) > \eta'' m$
    $\mathsf{BAD} \leftarrow \mathtt{true}$
    $\boxed{\mathsf{Forge} \leftarrow \mathtt{false}}$
    $\boxed{\text{Ret } 0}$
  Ret 1
If $\mathsf{hw}(\mathbf{t}^* - \mathbf{A}^*\mathbf{s} - \mathbf{e}^*) \le \eta'' m$
  $\mathsf{Forge} \leftarrow \mathtt{true}$
  Ret 1
Ret 0

oracle Vrfy$(\mathbf{A}^*, \mathbf{t}^*)$: $//G_3$
If $\mathbf{A}^* \notin C$
  Ret $\perp$
$C \leftarrow C \setminus \{\mathbf{A}^*\}$
$\mathbf{e}^* \leftarrow \mathsf{Ber}_\eta^m$
If $\mathsf{hw}(\mathbf{t}^* - \mathbf{A}^*\mathbf{s} - \mathbf{e}^*) \le \eta'' m$
  $\mathsf{Forge} \leftarrow \mathtt{true}$
  Ret 1
Ret 0

oracle Tag(): $//G_4$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n}$
$\mathbf{t} \xleftarrow{\$} \mathbb{Z}_2^m$
Ret $(\mathbf{A}, \mathbf{t})$

oracle Vrfy$(\mathbf{A}^*, \mathbf{t}^*)$: $//G_4$
If $\mathbf{A} \notin C$
  Ret $\perp$
$C \leftarrow C \setminus \{\mathbf{A}^*\}$
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^m$
if $\mathsf{hw}(\mathbf{t}^* - \mathbf{r}) \le \eta'' \cdot m$
  $\mathsf{Forge} \leftarrow \mathtt{true}$
  Ret 1
Ret 0

Figure 9: Sequence of games for the proof of Lemma 4.3

Proof. We use the sequence of games whose main procedure and oracles are shown in Figure 9. Throughout the proof, we fix an adversary $\mathcal{A}$ making q_T, q_C and q_V queries to **Tag**, **Chal** and **Vrfy** respectively. Game G_0 is equivalent to UF-RMRC$_{\mathsf{MAC}_{\mathrm{LPN}}}$. All extra commands in the code of the **Vrfy** oracle serve as internal bookkeeping and do not affect adversary's view. Therefore

$$\mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}_{\mathrm{LPN}}}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \;. \tag{23}$$

Game G_1 is identical to G_0 except in the way queries to **Vrfy** are answered. However, whenever Forge is set to $\mathtt{true}$ in G_0, so is in G_1 and thus

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \le \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \;. \tag{24}$$

Games G_2, G_1 are clearly equivalent-until-bad. Below, we compute the probabilty that the $\mathsf{BAD} \leftarrow \mathtt{true}$ happens in G_2. Let $\mathbf{y} = \mathbf{t}^* - \mathbf{A}^*\mathbf{s}$ and $\mathbf{y}' = \mathbf{t}^* - \mathbf{A}^*\mathbf{s} - \mathbf{e}^*$. Consider a single query $(\mathbf{A}^*, \mathbf{t}^*)$ to **Vrfy**. Then

$$\Pr\left[\, \mathsf{BAD} \leftarrow \mathtt{true} \,\right] = \Pr\left[\, (\mathsf{hw}(\mathbf{y}') > \eta'' m) \wedge (\mathsf{hw}(\mathbf{y}) \le \eta' m) \,\right] \le \Pr\left[\, \mathsf{hw}(\mathbf{y}') > \eta'' m \,\middle|\, \mathsf{hw}(\mathbf{y}) \le \eta' m \,\right] \;.$$

Each coordinate y_i' of $\mathbf{y}'$ is an independent random variable with $\mathop{\mathbb{E}}_{e_i^*}[\, y_i' \,] = (1-2\eta)y_i + \eta$. Therefore (since $\mathsf{hw}(\mathbf{y}) \le \eta' m$)

$$\mathop{\mathbb{E}}_{\mathbf{e}^*}\left[\, \mathsf{hw}(\mathbf{y}') \,\right] = (1-2\eta)\mathsf{hw}(\mathbf{y}) + \eta \cdot m \le (\eta + \eta' - 2\eta\eta') \cdot m = \bar{\eta} \cdot m \;.$$

Since $\eta'' > \bar{\eta}$, by applying Chernoff bound, we get $\Pr[\,\mathsf{hw}(\mathbf{y}') > \eta'' m\,] \leq 2^{-\mathrm{D}(\eta'' \,\|\, \bar{\eta})m}$. Using the fundamental lemma of game playing and the union bound across q_{V} queries to **Vrfy** we get that for every $\mathcal{A}$ (even unbounded)

$$\Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] - \Pr\left[\,G_2^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] \leq \Pr[\,\mathsf{BAD} \leftarrow \mathtt{true}\,] \leq q_{\mathsf{V}} \cdot 2^{-\mathrm{D}(\eta'' \,\|\, \bar{\eta})m} \,. \tag{25}$$

G_3 is essentially a compact rewriting of G_2. The view of any adversary $\mathcal{A}$ (even unbounded) is exactly the same in both games. Therefore

$$\Pr\left[\,G_2^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] = \Pr\left[\,G_3^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] \,. \tag{26}$$

G_4 differs from G_3 with respect to both **Tag** and **Vrfy** oracles. We claim that there exists an adversary $\mathcal{B}$ against LPN such that

$$\Pr\left[\,G_3^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] - \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] = \mathbf{Adv}_{n,\eta}^{\mathsf{lpn}}(\mathcal{B}) \,. \tag{27}$$

$\mathcal{B}$ maintains a set C (initialized to $\emptyset$) that contains all messages $\mathbf{A}$ for which $\mathcal{A}$ is allowed to query the oracle **Vrfy** and replies to $\mathcal{A}$'s queries as follows: For each query to **Tag** by $\mathcal{A}$, $\mathcal{B}$ makes m queries to its **Sample** oracle. Let $\{(\mathbf{a}_i, z_i)\}_{i\in[m]}$ be the samples returned. $\mathcal{B}$ then sends $(\mathbf{A}, \mathbf{z})$ to $\mathcal{A}$ where $\mathbf{A}$ is an $m \times n$ matrix with the i-th row being $\mathbf{a}_i$ and $\mathbf{z} = (z_1, \ldots, z_m)^T$. On each **Chal** query by $\mathcal{A}$, $\mathcal{B}$ gets m more samples $(\mathbf{A}^*, \mathbf{z}^*)$ and returns $\mathbf{A}^*$ to $\mathcal{A}$. At the same time, $\mathcal{B}$ adds $\mathbf{A}^*$ to C along with $\mathbf{z}^*$. On a $(\mathbf{A}^*, \mathbf{t}^*)$ query to **Vrfy**, $\mathcal{B}$ first checks that $\mathbf{A}^* \in C$ and if so, recovers the vector $\mathbf{z}^*$ that corresponds to $\mathbf{A}^*$. It then checks whether $\mathsf{hw}(\mathbf{t}^* - \mathbf{z}^*) \leq \eta'' m$ and if so, it outputs 1 and terminates. Otherwise it returns 0 to $\mathcal{A}$, removes $\mathbf{A}^*$ and $\mathbf{z}^*$ from C and resumes the simulation. Clearly, if $\mathcal{B}$ is playing in game $\mathrm{LPN}_{n,\eta}$, then it simulates G_3 perfectly to $\mathcal{A}$ whereas if it is playing in game $\mathrm{LPN}_{n,1/2}$, then it simulates G_4 perfectly to $\mathcal{A}$. Thus

$$\Pr\left[\,G_3^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] - \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] = \Pr\left[\,\mathrm{LPN}_{n,\eta}^{\mathcal{B}} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{LPN}_{n,1/2}^{\mathcal{B}} \Rightarrow 1\,\right] = \mathbf{Adv}_{n,\eta}^{\mathsf{lpn}}(\mathcal{B}) \,.$$

Moreover, for each **Tag** and each **Chal** query by $\mathcal{A}$, $\mathcal{B}$ makes m queries to its **Sample** oracle. Hence, $\mathcal{B}$ makes $(q_{\mathsf{T}} + q_{\mathsf{C}})m$ queries in total to its oracle and runs in time $t + \mathcal{O}(q_{\mathsf{C}})$.
Finally, the view of $\mathcal{A}$ in G_4 is completely independent of $\mathbf{s}$. Consider again a single query $(\mathbf{A}^*, \mathbf{t}^*)$ to **Vrfy**. It is straightforward to verify that $\mathbf{t}^* - \mathbf{r}$ is uniform and random over $\mathbb{Z}_2^m$. Therefore the probabilty a verification query causes **Vrfy** to return 1 is

$$2^{-m} \sum_{i=0}^{\lfloor \eta'' m \rfloor} \binom{m}{i} \leq 2^{-(1-\mathrm{H}_2(\eta''))m} \,.$$

Using the union bound,

$$\Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathtt{true}\,\right] \leq q_{\mathsf{V}} \cdot 2^{-(1-\mathrm{H}_2(\eta''))m} \,. \tag{28}$$

Proof then follows combining (23), (24), (25), (26), (27) and (28). □

Instantiating DM with $\mathsf{MAC}_{\mathrm{LPN}}$ yields a protocol $\mathsf{DM}_{\mathrm{LPN}}$ whose secret key consists of two vectors $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}_2^n$. The prover first selects a random matrix $\mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_2^{m\times n}$ and sends it to the verifier. The verifier then selects another matrix $\mathbf{A}_2 \xleftarrow{\$} \mathbb{Z}_2^{m\times n}$ and a noise vector $\mathbf{e}_1 \xleftarrow{\$} \mathsf{Ber}_\eta^m$, and sends $(\mathbf{A}_2, \mathbf{A}_1\mathbf{s}_1 + \mathbf{e}_1)$ to the prover. Upon receiving a pair $(\mathbf{A}_2, \mathbf{z}_1)$, the prover checks whether $\mathsf{hw}(\mathbf{z}_1 - \mathbf{A}_1\mathbf{s}_1) \leq \eta' \cdot m$, and if so, samples $\mathbf{e}_2 \xleftarrow{\$} \mathsf{Ber}_\eta^m$, and sends $\mathbf{A}_2\mathbf{s}_2 + \mathbf{e}_2$ back to the verifier. Finally, the verifier, on input $\mathbf{z}_2$, accepts iff $\mathsf{hw}(\mathbf{z}_2 - \mathbf{A}_2\mathbf{s}_2) < \eta' \cdot m$.

Protocol	#rounds	Complexity			LPN size	Security bound
		keysize	Communication	Computation		
HB$^+$ [30]	3	$2n$	$2nm+n$	$\Theta(n\cdot m)$	n	$q_{\mathsf{V}}\cdot\sqrt{\epsilon}$
KP$^+$ [33]	2	$\geq 4.2n$	$\geq 2.1nm$	$\Theta(n\cdot m)$	n	$q_{\mathsf{V}}\cdot\epsilon$
This work	3	$2n$	$2nm+2n$	$\Theta(n\cdot m)$	n	ϵ

Table 1: **(Asymptotic) comparison of known** LPN**-based active secure protocols.** Here, n is the secret-size for the underlying LPN problem and ϵ is the assumed hardness of LPN given $q = (q_{\mathsf{P}}+q_{\mathsf{V}}+q_{\mathsf{T}})m$ samples.

The overall advantage of our $\mathsf{DM}_{\mathrm{LPN}}$ protocol can be computed, combining (17) and Lemma 4.3, as

$$\begin{aligned}\mathbf{Adv}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}_{\mathsf{DM}_{\mathrm{LPN}}}(t,q_{\mathsf{T}},q_{\mathsf{P}},q_{\mathsf{V}}) &\leq \mathbf{Adv}^{\mathsf{lpn}}_{n,\eta}(t_1,q_1)+\mathbf{Adv}^{\mathsf{lpn}}_{n,\eta}(t_2,q_2)\\ &+ (q_{\mathsf{P}}+q_{\mathsf{V}})\left[2^{-\mathrm{D}(\eta''\,\|\,\bar{\eta})m}+2^{-(1-\mathrm{H}_2(\eta''))m}\right]\end{aligned} \tag{29}$$

where $q_1=(q_{\mathsf{T}}+q_{\mathsf{P}})m$, $q_2=(q_{\mathsf{T}}+q_{\mathsf{V}})m$, $t_1=t+\mathcal{O}(q_{\mathsf{T}}+q_{\mathsf{P}})\cdot t_{\mathsf{TAG}}$, $t_2=t+\mathcal{O}(q_{\mathsf{T}}+q_{\mathsf{V}})\cdot t_{\mathsf{TAG}}$ and t_{TAG} is the time to compute a single LPN mac. It is easy to see that this bound is superior to the one of HB$^+$ [30, 32], due to their use of rewinding, which results in a loose security reduction. Comparing with KP$^+$ [33] is more complicated. For that, we use the bound provided in their security reduction [33, Thm. 1]. Moreover, we use Theorems 3.5 and 3.9 to adapt their security bound to the case where both transcript and *multiple* verification queries are allowed. When the keysize of KP$^+$ is 2ℓ, then the overall bound can be computed as

$$\mathbf{Adv}^{(\{\mathsf{T},\mathsf{P}\},\{\mathsf{V}\})\text{-auth}}_{\mathrm{KP}^+}(t,q_{\mathsf{T}},q_{\mathsf{P}},q_{\mathsf{V}}) \leq q_{\mathsf{V}}\cdot\mathbf{Adv}^{\mathsf{lpn}}_{d,\eta}(t',q)+q_{\mathsf{V}}\left[\frac{(q_{\mathsf{P}}+q_{\mathsf{T}})m}{2^{g+1}}+(q_{\mathsf{P}}+q_{\mathsf{T}})2^{-c_1\cdot\ell}+2^{-c_2\cdot m}\right] \tag{30}$$

where $t'=t+\mathcal{O}(q_{\mathsf{P}}+q_{\mathsf{T}})$, $q=(q_{\mathsf{P}}+q_{\mathsf{T}})m$, c_1,c_2 are constants, and d,g are parameters such that $d+g\leq \ell/2.1$. Also, for keysize 2ℓ, KP$^+$ has communication complexity $2\ell+m\ell+m$. Notice that the security of KP$^+$ (with keysize 2ℓ) is based on the hardness of LPN with secret size $d<\ell/2.1$. In contrast, $\mathsf{DM}_{\mathrm{LPN}}$ with keysize $2n$ relies on the hardness of LPN with secret size n. Moreover, too small values for g affect negatively the security of KP$^+$ and in practice one might have to choose $g=d<\ell/4.2$. This means that, even in the most optimistic case ($\ell=2.1d$), for the same security level, i.e. LPN with the same secret size, $\mathsf{DM}_{\mathrm{LPN}}$ requires a substantially smaller key than KP$^+$ and incurs lower communication complexity.

The comparison with both HB$^+$ and KP$^+$ is even more in our favor when multiple verification queries are considered. Indeed, the bounds for HB$^+$ and KP$^+$ increase linearly with the number of verification queries.

Based on the above analysis, Table 1 provides an asymptotic comparison of our LPN-based authentication protocol with HB$^+$ and KP$^+$. In the table, we have used LPN with fixed secret size n as the underlying hardness assumption across all three protocols.

A CDH**-based scheme.** For a cyclic group $\mathbb{G}$ with generator g, and an adversary $\mathcal{A}$, the CDH advantage function is $\mathbf{Adv}^{\mathsf{cdh}}_{\mathbb{G},g}(t)=\max\left\{\Pr\left[\,x,y\xleftarrow{\$}\mathbb{Z}_{|\mathbb{G}|}\;:\;\mathcal{A}(g,g^x,g^y)=g^{xy}\,\right]\right\}$, where the maximum is over all adversaries $\mathcal{A}$ running in time t. For a group $\mathbb{G}$, $\mathsf{MAC}_{\mathrm{CDH}}=(\mathsf{KGen},\mathsf{TAG},\mathsf{VRFY})$ has keyspace $\mathcal{K}=\mathbb{Z}_{|\mathbb{G}|}$, message space $\mathcal{M}=\mathbb{G}$ and tag space $\mathcal{T}=\mathbb{G}$ and is defined as follows:

$\underline{\mathsf{KGen}:}$	$\underline{\mathsf{TAG}(K,m):}$	$\underline{\mathsf{VRFY}(K,m,h):}$
Ret $K\xleftarrow{\$}\mathbb{Z}_{|\mathbb{G}|}$	Ret $h\leftarrow m^K$	If $h=m^K$ then Ret 1 else Ret 0.

It is straightforward to verify that $\mathsf{MAC}_{\text{CDH}}$ has completeness 1. Lemma 4.4 asserts that $\mathsf{MAC}_{\text{CDH}}$ is uf-rmrc-secure assuming CDH is hard.

Lemma 4.4. [Security of $\mathsf{MAC}_{\text{CDH}}$] *For all $t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}} > 0$,*

$$\mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}_{\text{CDH}}}(t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}}) \leq q_{\mathsf{C}} \cdot \mathbf{Adv}^{\mathsf{cdh}}_{\mathbb{G},g}(t'),$$

where $t' = t + \mathcal{O}(q_{\mathsf{T}} + q_{\mathsf{C}}) \cdot t_{exp}$ and t_{exp} is the cost of one exponentiation in $\mathbb{G}$.

Proof. Fix an adversary $\mathcal{A}$ against the uf-rmrc-security of $\mathsf{MAC}_{\text{CDH}}$ that runs in time t and makes $q_{\mathsf{T}}, q_{\mathsf{C}}$ and q_{V} queries to **Tag**, **Chal** and **Vrfy** oracles respectively. We construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to find a solution to CDH. $\mathcal{B}$ gets as input a triple $(g, h_1 = g^x, h_2 = g^y) \in \mathbb{G}^3$ where x, y are chosen uniformly at random from $\mathbb{Z}_{|\mathbb{G}|}$. It first samples $z \xleftarrow{\$} \{1, \ldots, q_{\mathsf{C}}\}$ and initializes a list C to $\emptyset$ (this will contain all challenges that $\mathcal{A}$ can use in its verification queries). On every **Tag** query by $\mathcal{A}$, $\mathcal{B}$ samples $r \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}$ and returns (g^r, h_1^r) to $\mathcal{A}$. On the i-th query to **Chal**, $\mathcal{B}$ replies as follows: If $i \neq z$ then $\mathcal{B}$ samples $r_i \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}$, returns g^{r_i} to $\mathcal{A}$ and then adds $g_i = g^{r_i}$ along with $h_i = h_1^{r_i}$ to C. If $i = z$, $\mathcal{B}$ sends h_2 to $\mathcal{A}$. Finally, if $\mathcal{A}$ makes a query (g^*, h^*) to **Vrfy**, $\mathcal{B}$ first checks if $g^* \in C$ and rejects if not. Otherwise, if $g^* = h_2$, $\mathcal{B}$ returns h^* to its oracle as the candidate value for g^{xy} and halts. If $g^* = g_i \neq h_2$, $\mathcal{B}$ recovers $h_i = h_1^{r_i}$ from C and returns 1 if and only if $h^* = h_i$.

Notice that, since the pairs $(g^r, h_1^r) = (g^r, (g^r)^x)$ are distributed exactly as (m, τ) pairs from $\mathsf{MAC}_{\text{CDH}}$ (with $m = g^r$ and $K = x$), $\mathcal{B}$ provides a perfect simulation to $\mathcal{A}$. It remains to compute the advantage of $\mathcal{B}$. Let F be a random variable that takes as value the index of the **Chal** query made by $\mathcal{A}$ that results in the *first* **Vrfy** query that returns 1 ($F = 0$ if no **Vrfy** query returns 1). Notice, that if $z = F$ and $F \neq 0$, then $\mathcal{B}$ wins in its CDH game. Indeed, in that case $h^* = g_z^x = h_2^x = g^{xy}$. Therefore

$$\mathbf{Adv}^{\mathsf{cdh}}_{\mathbb{G},g}(\mathcal{B}) \geq \Pr[\, z = F \wedge F \neq 0 \,] = \Pr\left[\, z = F \,\middle|\, F \neq 0 \,\right] \cdot \Pr[\, F \neq 0 \,] = \frac{\mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}_{\text{CDH}}}(\mathcal{A})}{q_{\mathsf{C}}}\,.$$

Finally, for the simulation, $\mathcal{B}$ makes 2 exponentiations for each **Tag** and each **Chal** query by $\mathcal{A}$. ☐

The resulting authentication protocol proceeds as follows: The secret key consists of two elements $s_1, s_2 \in \mathbb{Z}_{|\mathbb{G}|}$. The prover selects a random element $r_1 \xleftarrow{\$} \mathbb{G}$, and sends it to the verifier. The verifier, on input r_1, samples a random element $r_2 \xleftarrow{\$} \mathbb{G}$, and sends the pair $(r_2, r_1^{s_1})$ to the prover. On input (r_2, z_1), the prover checks whether $z_1 = r_1^{s_1}$, and if so, sends $r_2^{s_2}$ back to the verifier. Finally, the verifier, on input z_2, accepts iff $z_2 = r_2^{s_2}$.

The execution of the protocol requires two exponentiations for each of the prover and the verifier. Moreover, the overall communication complexity amounts to 4 group elements. We note that the only alternative construction based on CDH is due to Dodis *et al* [17], which is much less efficient, requiring in total 8 exponentiations in $\mathbb{G}$, but achieves a stronger notion of security. In the same paper, the authors present a (strongly) actively secure protocol that requires the same number of exponentiations as ours but only exchanges 3 groups elements. Yet, its security relies on the stronger DDH assumption. Moreover, their proof of security uses rewinding techniques leading to a looser security reduction.

4.2 A Generic Construction of a 2-Round MIM-Secure Protocol

In this section, we present a generic construction of a 2-round MIM-secure authentication protocol, which is $(\{\}, \{\mathsf{P}, \mathsf{V}\})$-secure, i.e. satisfies the strongest notion of MIM security. The protocol $\mathsf{MM}[\mathsf{MAC}] = (\mathcal{K}, \mathcal{P}, \mathcal{V})$ (MM stands for <u>M</u>irror-<u>M</u>ac), using $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ is as follows, where K_1, K_2 are generated by KGen:

Game SUF-RMCC$_{\mathsf{MAC}}$

procedure main:	**oracle Tag**():	**oracle ReTag**(m):	**oracle Vrfy**(m,τ):
$K \xleftarrow{\$} \mathsf{KGen}$	$m \xleftarrow{\$} \mathcal{M}$	If $m \notin M$	If $\mathsf{VRFY}_K(m,\tau) = 1$
$\mathsf{Forge} \leftarrow \mathtt{false}$	$M \leftarrow M \cup \{m\}$	Ret $\perp$	If $(m,\tau) \notin S$
$S \leftarrow \emptyset$; $M \leftarrow \emptyset$	$\tau \xleftarrow{\$} \mathsf{TAG}_K(m)$	$\tau \xleftarrow{\$} \mathsf{TAG}_K(m)$	$\mathsf{Forge} \leftarrow \mathtt{true}$
Run $\mathcal{A}_{\mathsf{MAC}}^{\mathbf{Tag},\mathbf{Vrfy}}$	$S \leftarrow S \cup \{(m,\tau)\}$	$S \leftarrow S \cup \{(m,\tau)\}$	Ret 1
Ret Forge	Ret (m,τ)	Ret τ	Ret 0

Figure 10: **Pseudocode description of Game** SUF-RMCC$_{\mathsf{MAC}}$

$\mathcal{P}(K_1,K_2)$		$\mathcal{V}(K_1,K_2)$
	$\xleftarrow{\quad r,\tau_1 \quad}$	$r \xleftarrow{\$} \mathcal{M}$;
If $\mathsf{VRFY}_{K_1}(r,\tau_1) = 1$		$\tau_1 \leftarrow \mathsf{TAG}_{K_1}(r)$;
$\tau_2 \leftarrow \mathsf{TAG}_{K_2}(r)$	$\xrightarrow{\quad \tau_2 \quad}$	Accept iff
		$\mathsf{VRFY}_{K_2}(r,\tau_2) = 1$

The main idea here is that if MAC is a sufficiently strong MAC, no adversary can even get to input to the prover values which have not been output by the verifier, hence forcing the attacker to behave correctly. The catch is, however, that a secure MAC gives a challenge-response secure protocol in the first place, so the question is how far can we *weaken* the assumption on the MAC so that the protocol remains secure? It turns out that it suffices to require MAC to be what we call *strongly unforgeable under random-message-chosen-challenge attacks* (suf-rmcc), a notion which is weaker than suf-cma-security in the sense that the adversary gets to see tags for messages that are random rather than of its choosing.

Formally, suf-rmcc-security is defined via the game SUF-RMCC shown in Figure 10. Queries to **Tag** oracle return pairs $(m, \mathsf{TAG}_K(m))$ for fresh random messages m not controlled by the adversary. However, once a message m has been sampled during a **Tag** query, the adversary can ask for multiple tags for m. This is modeled by giving the adversary access to a special oracle **ReTag**, that accepts an input $m \in \mathcal{M}$ and returns $\mathsf{TAG}_K(m)$ *only if* m has been previously returned after a **Tag** query (otherwise **ReTag** returns $\perp$). Finally the adversary gets access to a verification oracle **Vrfy** which it can invoke on inputs (m,τ) of its choosing. The suf-rmcc advantage function of a MAC for integers $t, q_{\mathsf{T}}, q_{\mathsf{RT}}$, and q_{V} is defined as

$$\mathbf{Adv}_{\mathsf{MAC}}^{\mathsf{suf\text{-}rmcc}}(t, q_{\mathsf{T}}, q_{\mathsf{RT}}, q_{\mathsf{V}}) = \max_{\mathcal{A}} \left\{ \Pr\left[(\text{SUF-RMCC}_{\mathsf{MAC}})^{\mathcal{A}} \Rightarrow \mathtt{true} \right] \right\}$$

where the maximum is over all adversaries $\mathcal{A}$ making q_{T}, q_{RT} and q_{V} queries to **Tag**, **ReTag** and **Vrfy** oracles respectively and running in time t.

One vs. multiple verification queries. Requiring *strong* unforgeability (i.e. the fact that a pair (m,τ) is considered a forgery even if (m,τ') with $\tau' \neq \tau$ has previously been output by a tag query) is sufficient to ensure that the advantage of an adversary making multiple verification queries grows only linearly in the number of verification queries. We remark that this is in contrast with (weak) unforgeability under chosen message attacks (uf-cma) as first noticed by Bellare et al. [5]. Lemma 4.5 formalizes the aforementioned property for suf-rmcc-secure MACs. The proof is essentially identical to the proof of [5, Theorem 5.1] and hence omitted.

Lemma 4.5. [1 vs. multiple ver. queries for suf-rmcc-secure MACs] *Let* $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ *be a* MAC. *Then for all positive integers* t, q_{T}, q_{RT} *and* q_{V}

$$\mathbf{Adv}_{\mathsf{MAC}}^{\mathsf{suf\text{-}rmcc}}(t, q_{\mathsf{T}}, q_{\mathsf{RT}}, q_{\mathsf{V}}) \leq q_{\mathsf{V}} \cdot \mathbf{Adv}_{\mathsf{MAC}}^{\mathsf{suf\text{-}rmcc}}(t, q_{\mathsf{T}}, q_{\mathsf{RT}}, 1) \ .$$

The following theorem summarizes the concrete security of MM.

Theorem 4.6. **[Security of** MM**]** *Let* $\mathcal{M}$ *be the message space of* $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$. *Then for all* $t, q_{\mathsf{P}}, q_{\mathsf{V}} > 0$,

$$\mathbf{Adv}^{(\{\},\{\mathsf{P},\mathsf{V}\})\text{-auth}}_{\mathsf{MM}[\mathsf{MAC}]}(t, q_{\mathsf{P}}, q_{\mathsf{V}}) \leq \mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}}(t_1, q_{\mathsf{V}}, 0, q_{\mathsf{P}}) + (q_{\mathsf{V}}+1)\mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}}(t_2, q_{\mathsf{V}}, q_{\mathsf{P}}, q_{\mathsf{V}}) + \frac{q_{\mathsf{V}}^3}{2|\mathcal{M}|}\ ,$$

where $t_1 = t + \mathcal{O}(t_{\mathsf{KGen}} + q_{\mathsf{P}} \cdot t_{\mathsf{TAG}} + q_{\mathsf{V}} \cdot t_{\mathsf{VRFY}})$ *and* $t_2 = t + \mathcal{O}(t_{\mathsf{KGen}} + q_{\mathsf{P}} \cdot t_{\mathsf{VRFY}} + q_{\mathsf{V}} \cdot t_{\mathsf{TAG}})$.

Proof. For the proof, fix an adversary $\mathcal{A}$ against MM that makes q_{P} (resp. q_{V}) queries to **P** (resp. **V**) and runs in time t. The proof relies on games G_0 and G_1 shown in Figure 11 to simplify exposition. Game G_0 is a stripped down version of game $\mathrm{AUTH}^{(\{\},\{\mathsf{P},\mathsf{V}\})}_{\Pi}$, where all unnecessary notation has been removed. Therefore,

$$\mathbf{Adv}^{(\{\},\{\mathsf{P},\mathsf{V}\})\text{-auth}}_{\Pi}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right]\ . \tag{31}$$

In addition, G_0 keeps track of pairs (r, τ_1) generated by verifier instances upon invocation from the adversary via **V** queries, adding them to a set S. The game G_0 is such that whenever the prover oracle **P** is invoked on a message $r \,||\, \tau_1$ such that $\mathsf{VRFY}_{K_1}(r, \tau_1) = 1$ and $(r, \tau_1) \notin S$ (i.e., a corresponding message-tag pair was *not* previously returned as a reply to a **V**-query $(\mathit{sid}, \mathsf{start})$), the flag BAD is set. Game G_1 is identical to G_0 except that if the flag BAD is set in the **P** queries, then it returns $\perp$. The following claim shows that the probability of any adversary $\mathcal{A}$ winning in Game G_0 cannot be much bigger than the corresponding probability in G_1, upper bounding it via the suf-rmcc-security of MAC.

Claim 4.7. *There exists adversary* $\mathcal{B}$ *such that*

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \leq \mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}}(\mathcal{B}), \tag{32}$$

where $\mathcal{B}$ *makes* q_{V}, *0 and* q_{P} *queries to its* **Tag**, **ReTag** *and* **Vrfy** *oracles respectively and has running time* $t_1 = t + \mathcal{O}(t_{\mathsf{KGen}} + q_{\mathsf{P}} \cdot t_{\mathsf{TAG}} + q_{\mathsf{V}} \cdot t_{\mathsf{VRFY}})$.

Proof. (of Claim 4.7.) Notice that G_0 and G_1 are identical until the flag BAD is set to `true`. Therefore, by the fundamental lemma of game playing

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] - \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right] \leq \Pr\left[\, G_1^{\mathcal{A}} \text{ sets } \mathsf{BAD} \,\right]\ .$$

The event $\mathsf{BAD} \leftarrow \mathtt{true}$ happens if the condition $\mathsf{VRFY}_{K_1}(r, \tau_1) = 1$ is satisfied in G_1 within a **P** query, yet $(r, \tau_1) \notin S$. This event can be reduced fairly directly to a forgery in the suf-rmcc-security game. The suf-rmcc adversary $\mathcal{B}$ simulates game G_1 to $\mathcal{A}$ as follows: First it picks $K_2 \xleftarrow{\$} \mathsf{KGen}$. On every **V**-query $(\mathit{sid}, \mathsf{start})$ by $\mathcal{A}$, $\mathcal{B}$ gets a pair (r, τ_1) by invoking its **Tag** oracle in game $\mathrm{SUF\text{-}RMCC}_{\mathsf{MAC}}$ and then returns (r, τ_1) to $\mathcal{A}$. For every **V**-query $(\mathit{sid}, \mathit{msg})$ with $\mathit{msg} \neq \mathsf{start}$, $\mathcal{B}$ uses K_2 (recall that K_2 was chosen by $\mathcal{B}$) to check whether $\mathsf{VRFY}_{K_2}(\mathsf{state}[\mathit{sid}], \mathit{msg}) = 1$ and returns the answer to $\mathcal{A}$. Likewise, **P** queries are handled so that $\mathsf{VRFY}_{K_1}(r, \tau_1)$ is checked by invoking the **Vrfy** oracle from the underlying $\mathrm{SUF\text{-}RMCC}_{\mathsf{MAC}}$ game, whereas tag generation with respect to K_2 can be done, since $\mathcal{B}$ knows K_2. It is not hard to see that $\mathcal{B}$ is simulating perfectly G_1 to $\mathcal{A}$. Also, $\mathcal{A}$ provoking $\mathsf{BAD} \leftarrow \mathtt{true}$ results precisely to $\mathcal{B}$ provoking a forgery (and hence winning) its $\mathrm{SUF\text{-}RMCC}_{\mathsf{MAC}}$ game.

Finally, for every **P** query by $\mathcal{A}$, $\mathcal{B}$ makes 1 query to its **Vrfy** oracle and computes a tag by itself, whereas for every **V** query by $\mathcal{A}$, $\mathcal{B}$ makes 1 query to its **Tag** oracle and has to compute VRFY once. □

Now, to conclude the proof, it remains to upper bound the probability $\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \,\right]$ via the forgery probability in the game $\mathrm{SUF\text{-}RMCC}_{\mathsf{MAC}}$. Let $(\mathit{sid}^*, \tau_2^*)$ $(\mathit{sid}^* \in \mathit{SID}_{\mathcal{V}})$ be the *first* **V**-query which makes $\mathcal{A}$ win in G_1 and assume that this query was made at (relative) time i_k. This query is associated

procedure main: $//G_0, G_1$
$K_1, K_2 \xleftarrow{\$} \mathsf{KGen}$; $S \leftarrow \emptyset$;
$T \leftarrow \emptyset$; $i \leftarrow 0$
For all $sid \in \mathbb{N}$ do
  $\mathsf{state}[sid] = \varepsilon$;
  $\mathsf{decision}[sid] = \perp$;
  $\mathsf{done}[sid] = \mathtt{false}$
$\mathcal{A}^{\mathbf{P},\mathbf{V}}$
Ret $\exists sid \in SID_{\mathcal{V}}$: $(\mathsf{decision}[sid] = \mathsf{A})$
  $\wedge$ $(\forall sid' \in SID_{\mathcal{P}} : \neg\mathsf{Matching}(T[sid'], T[sid]))$
 	
oracle P(sid, msg): // G_0, $\boxed{G_1}$
If $(\mathsf{done}[sid] = \mathtt{true})$ then Ret $\perp$
$i \leftarrow i + 1$
$r \mid\mid \tau_1 \leftarrow msg$
If $\mathsf{VRFY}_{K_1}(r, \tau_1) = 1$ then
  $\tau_2 \xleftarrow{\$} \mathsf{TAG}_{K_2}(r)$
  If $(r, \tau_1) \notin S$ then
    $\mathsf{BAD} \leftarrow \mathtt{true}$
    $\boxed{\tau_2 \leftarrow \perp}$
  $\mathsf{done}[id] \leftarrow \mathtt{true}$
  $T[sid] \leftarrow T[sid] \cup \{(sid, i, r \mid\mid \tau_1, \tau_2)\}$
  Ret τ_2
Ret $\perp$
	oracle V(sid, msg): // G_0, G_1
	If $(\mathsf{done}[sid] = \mathtt{true})$ then Ret $\perp$
	$i \leftarrow i + 1$
	If $\mathsf{state}[sid] = \varepsilon$ then // 1st round
	  If $msg \neq \mathsf{start}$ then Ret $\perp$
	  $\mathsf{state}[sid] \xleftarrow{\$} \mathcal{M}$
	  $\tau_1 \xleftarrow{\$} \mathsf{TAG}_{K_1}(\mathsf{state}[sid])$
	  $S \leftarrow S \cup \{(\mathsf{state}[sid], \tau_1)\}$
	  $T \leftarrow T \cup \{(sid, i, \mathsf{start}, r \mid\mid \tau_1)\}$
	  Ret $(\mathsf{state}[sid], \tau_1)$
	Else // final decision
	  $\mathsf{done}[sid] \leftarrow \mathtt{true}$
	  If $\mathsf{VRFY}_{K_2}(\mathsf{state}[sid], msg) = 1$ then
	    $\mathsf{decision}[sid] \leftarrow \mathsf{A}$
	  Else $\mathsf{decision}[sid] \leftarrow \mathsf{R}$
	  $T \leftarrow T \cup \{(sid, i, msg, \mathsf{decision}[sid]\}$
	  Ret $\mathsf{decision}[sid]$

Figure 11: **Games G_0, G_1 used in the proof of Theorem 4.6.**

with a **V**-query (sid^*, start) made at time $i_0 < i_k$. Let $r^* \mid\mid \tau_1^*$ be the reply to (sid^*, start). These 2 queries have resulted to entries $(sid^*, i_0, \mathsf{start}, r^* \mid\mid \tau_1^*)$ and $(sid^*, i_k, \tau_2^*, \mathsf{A})$ in the global list T that keeps track of all the queries and replies. By definition of winning, we know that: (a) $\mathsf{VRFY}_{K_2}(r^*, \tau_2^*) = 1$ and (b) $\forall$ $sid \in SID_{\mathcal{P}}$ and $\forall i \in \mathbb{N}$ with $i_0 < i < i_k$ if $(sid, i, msg, msg') \in T$ then $(msg, msg') \neq (r^* \mid\mid \tau_1^*, \tau_2^*)$ ((b) stems from the requirement that the verifier instance corresponding to sid^* has not engaged to a matching conversation with any of the prover instances). We consider the following two cases:

(i) There exists a **P**-query $(sid, r^* \mid\mid \tau_1^*)$ at time i $(i_0 < i < i_k)$. Call this event E_1. By the no matching conversation requirement any such query must have returned $\tau_2' \neq \tau_2^*$ as a reply. We build an adverary $\mathcal{C}$ against SUF-RMCC$_{\mathsf{MAC}}$ (with underlying key K_2) with success probability not much smaller than $\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \;\wedge\; \mathsf{E}_1 \,\right]$. $\mathcal{C}$ maintains two sets[16] S_1, S_2 (initialized to $\emptyset$) for bookkeeping and operates as follows: It first selects $K_1 \xleftarrow{\$} \mathsf{KGen}$, and then simulates the game G_1 to $\mathcal{A}$. On a query (sid, start) to **V**, $\mathcal{C}$ invokes its **Tag** oracle obtaining (r, τ_2), sets $\mathsf{state}[sid] \leftarrow r$, computes $\tau_1 \xleftarrow{\$} \mathsf{TAG}_{K_1}(r)$ (recall that $\mathcal{C}$ has chosen K_1 itself), returns τ_1 to $\mathcal{A}$ and finally adds (r, τ_1) to S_1 and (r, τ_2) to S_2. On every **V**-query (sid, msg) $(msg \neq \mathsf{start})$, $\mathcal{C}$ invokes its **Vrfy** oracle on $(\mathsf{state}[sid], msg)$. If **Vrfy** returns 1 and $(\mathsf{state}[sid], msg) \notin S_2$ then $\mathcal{C}$ aborts (in such a case $(\mathsf{state}[sid], msg)$ is a valid forgery and hence $\mathcal{C}$ wins

[16]Roughly, S_1 and S_2 keep track of all message-tag pairs that have been produced by TAG_{K_1} and TAG_{K_2} respectively during the simulation.

its SUF-RMCC$_{\mathsf{MAC}}$ game). Otherwise, $\mathcal{C}$ simply returns the reply to $\mathcal{A}$. Moreover, when $\mathcal{A}$ makes a **P**-query $(sid, r' \mid\mid \tau_1')$, $\mathcal{C}$ first checks (using its own K_1) whether $\mathsf{VRFY}_{K_1}(r', \tau_1') = 1$ and returns $\perp$ if this is not the case. It also returns $\perp$ if $(r', \tau_1') \notin S_1$. Otherwise, $\mathcal{C}$ operates as follows: If r' has not appeared before in any **P**-query, $\mathcal{C}$, recovers the entry (r', τ_2') from S_2 and returns τ_2' to $\mathcal{A}$ (notice that, since $(r', \tau_1') \in S_1$, by simulation, r' must have been part of a reply $(r', *)$ from $\mathcal{C}$'s **Tag** oracle and hence S_2 must contains such an entry (r', τ_2')). If r' has appeared before as part of a **P**-query, $\mathcal{C}$ simply uses its **ReTag** oracle to get τ_2'', returns it to $\mathcal{A}$ and adds (r', τ_2'') to S_2 (again, $(r', \tau_1') \in S_1$ implies that r' must have been previously output by $\mathcal{C}$'s **Tag** oracle and therefore $\mathcal{C}$ is entitled to ask its **ReTag** oracle on r'.) It is not hard to see that $\mathcal{C}$ simulates perfectly G_1 to $\mathcal{A}$. Also $\mathcal{C}$ makes (at most) 1 **ReTag** query and 1 VRFY computation for each **P**-query by $\mathcal{A}$ and 1 query to each of its **Tag** and **Vrfy** oracles and 1 TAG computation for each **V**-query by $\mathcal{A}$.
It remains to compute $\mathcal{C}$'s probability in winning SUF-RMCC$_{\mathsf{MAC}}$. Let Col be the event that, during the simulation, there exists (at least) two distinct queries to **Tag** that returned $(r, *)$ for the same value r. Assume that Col does *not* happen during the simulation. Then S_1 contains at most one pair $(r, \tau)\ \forall\, r \in \mathcal{M}$. In particular, at time i_0, (r^*, τ_1^*) is added to S_1 and that is the only pair of the form $(r^*, *)$ ever added to S_1. Therefore the only **P**-queries that involve r^* and *do not* return $\perp$, should necessarily be of the form $(sid, r^* \mid\mid \tau_1^*)$ and must happen at relative time $i > i_0$. Each such query returns $\tau_2' \neq \tau_2^*$ (by the no matching conversation requirement) and results in a pair (r^*, τ_2') added to S_2. This means that $(r^*, \tau_2^*) \notin S_2$ and therefore (r^*, τ_2^*) is a valid forgery in $\mathcal{C}$'s SUF-RMCC$_{\mathsf{MAC}}$ game. To conclude

$$\begin{aligned}\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true}\ \wedge\ \mathsf{E}_1 \,\right] &\leq \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true}\ \wedge\ \mathsf{E}_1 \wedge \neg\mathsf{Col} \,\right] + \Pr\left[\, \mathsf{Col} \,\right] \\ &\leq \Pr\left[\, \text{SUF-RMCC}_{\mathsf{MAC}}^{\mathcal{C}} \Rightarrow \mathtt{true} \,\right] + \frac{q_{\mathsf{V}}(q_{\mathsf{V}}-1)}{2|\mathcal{M}|}\,. \end{aligned} \tag{33}$$

(ii) All **P**-queries (sid, msg) at relative time i $(i_0 < i < i_k)$ are such that $msg \neq r^* \mid\mid \tau_1^*$. Call this event E_2. We further decompose E_2 into $\mathsf{E}_2^1 \vee \mathsf{E}_2^2 \vee \ldots \vee \mathsf{E}_2^{q_{\mathsf{V}}}$ where E_2^j is the event that (sid^*, start) was the j-th $\mathcal{V}$ instance queried by $\mathcal{A}$. (Stated differently, before making the query (sid^*, start) to **V**, $\mathcal{A}$ has made queries (sid, start) to **V** for exactly $j-1$ *distinct* $sid \in SID_{\mathcal{V}}$.) We show below that for each $j \in \{1, \ldots, q_{\mathsf{V}}\}$, there exists adversary $\mathcal{C}_j$ against SUF-RMCC$_{\mathsf{MAC}}$ with success probability approximately equal to $\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true}\ \wedge\ \mathsf{E}_2^j \,\right]$. $\mathcal{C}_j$ simulates G_1 to $\mathcal{A}$ precisely as in case (i) with one important difference: When $\mathcal{A}$ makes a **V**-query (sid, start) for the j-th *new distinct* $sid \in SID_{\mathcal{V}}$, $\mathcal{C}_i$ generates $r^* \xleftarrow{\$} \mathcal{M}$ by itself (without invoking its **Tag** oracle). Let τ_1^* be the reply to such a query (τ_1^* is computed similarly to case (i)). The rest of the simulation is the same. Notice that conditioned on Col not happening, (r^*, τ_1^*) is the only entry in S_1 that involves r^* and, by hypothesis, $r^* \mid\mid \tau_1^*$ is never queried to **P**. Therefore S_2 contains no pair $(r^*, *)$ which in turn implies that (r^*, τ_2^*) is a valid forgery for $\mathcal{C}_j$ when participating in its SUF-RMCC$_{\mathsf{MAC}}$ game. That is,

$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true}\ \wedge\ \mathsf{E}_2^j \,\right] \leq \Pr\left[\, \text{SUF-RMCC}_{\mathsf{MAC}}^{\mathcal{C}_j} \Rightarrow \mathtt{true} \,\right] + \frac{q_{\mathsf{V}}(q_{\mathsf{V}}-1)}{2|\mathcal{M}|}$$

where $\mathcal{C}_j$ runs in the same time and makes the same number of oracle queries as in case (i). Summing up, we obtain

$$\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \wedge \mathsf{E}_2 \,\right] = \sum_{j=1}^{q_{\mathsf{V}}} \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \mathtt{true} \wedge \mathsf{E}_2^j \,\right] \leq q_{\mathsf{V}} \mathbf{Adv}_{\mathsf{MAC}}^{\mathsf{suf\text{-}rmcc}}(t, q_{\mathsf{V}}, q_{\mathsf{P}}, q_{\mathsf{V}}) + \frac{q_{\mathsf{V}}^2(q_{\mathsf{V}}-1)}{2|\mathcal{M}|} \tag{34}$$

Overall, $\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right]$ is upper bounded using (33) and (34) as

$$\begin{aligned}\Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \,\right] &= \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \wedge \mathsf{E}_1 \,\right] + \Pr\left[\, G_1^{\mathcal{A}} \Rightarrow \texttt{true} \wedge \mathsf{E}_2 \,\right] \\ &\leq (q_{\mathsf{V}}+1)\mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}}(t_2, q_{\mathsf{V}}, q_{\mathsf{P}}, q_{\mathsf{V}}) + \frac{q_{\mathsf{V}}^3}{2|\mathcal{M}|} \qquad (35)\end{aligned}$$

The proof then follows from (31), (32) and (35). □

An instantiation based on qSDH. We present an instantiation of our MM protocol based on the hardness of the q-Strong Diffie-Hellman (qSDH) problem introduced by Boneh and Boyen [11]. For a cyclic group $\mathbb{G}$ of prime order p, qSDH is the problem of computing a pair $(c, g^{1/(c+x)})$ (for some $c \in \mathbb{Z}_p$ of the adversary's choice) given $(g, g^x, g^{x^2}, \ldots, g^{x^q})$, for a given random generator g. Formally, for any adversary $\mathcal{A}$, we define the qSDH advantage function (parametrized by q) as

$$\mathbf{Adv}^{\mathsf{q\text{-}sdh}}_{\mathbb{G},q}(t) = \max_{\mathcal{A}} \left\{ \Pr\left[\, g \xleftarrow{\$} \mathbb{G},\ x \xleftarrow{\$} \mathbb{Z}_p \ :\ \mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}) = (c, g^{1/(x+c)}) \,\right] \right\} ,$$

where the maximum is taken over all adversaries running in t steps. We devise a qSDH-based suf-rmcc-secure MAC $\mathsf{MAC}_{\mathrm{qSDH}} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ with message space $\mathcal{M} = \mathbb{Z}_p$ and tag space $\mathcal{T} = \mathbb{G}$, which is reminiscent of the weakly-secure signature scheme of [11], and defined via the following:

<u>KGen :</u>	<u>$\mathsf{TAG}((g,K),m)$:</u>	<u>$\mathsf{VRFY}((g,K),m,h)$:</u>
Ret $(g,K) \xleftarrow{\$} \mathbb{G} \times \mathbb{Z}_p$.	Return $h = g^{\frac{1}{m+K}} \in \mathbb{G}$	If $h = g^{1/(m+K)}$ then Ret 1 else Ret 0.

Clearly, $\mathsf{MAC}_{\mathrm{qSDH}}$ has completeness 1. The following lemma states that $\mathsf{MAC}_{\mathrm{qSDH}}$ is also suf-rmcc-secure under the assumption that qSDH in $\mathbb{G}$ is hard. The proof of Lemma 4.8 follows closely the proof from [11, Lemma 9] and is given in Appendix B. Interestingly, unlike the case of signatures [11], we do *not* need pairings to prove suf-rmcc security.

Lemma 4.8. **[Security of** $\mathsf{MAC}_{\mathrm{qSDH}}$**]** *For all* t, q_{T}, q_{RT} *and* $q_{\mathsf{V}} > 0$,

$$\mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}_{\mathrm{qSDH}}}(t, q_{\mathsf{T}}, q_{\mathsf{RT}}, q_{\mathsf{V}}) \leq q_{\mathsf{V}} \cdot \mathbf{Adv}^{\mathsf{q\text{-}sdh}}_{\mathbb{G},q_{\mathsf{T}}}(t'),$$

where $t' = t + \mathcal{O}(q_{\mathsf{T}}^2 \cdot t_{\mathsf{exp}})$ *and* t_{exp} *is the cost of a single exponentiation in* $\mathbb{G}$.

Let us now instantiate MM with $\mathsf{MAC}_{\mathrm{qSDH}}$: The secret key of the scheme consists of two random integers $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}$, and two random generators $g_1, g_2 \xleftarrow{\$} \mathbb{G}$. In the first round, the verifier picks $r \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}$, and sends $(r, g_1^{\frac{1}{s_1+r}})$ to the prover. The prover, given (r, τ_1) checks whether $g_1^{\frac{1}{s_1+r}} = \tau_1$, and if so, sends back $g_2^{\frac{1}{s_2+r}}$. Finally, the verifier, upon receiving τ_2, accepts iff $\tau_2 = g_2^{\frac{1}{s_2+r}}$.

Regarding complexity, our qSDH-based protocol needs two exponentiations by the prover and two exponentiations by the verifier. Overall, two group elements and one integer modulo $|\mathbb{G}|$ are exchanged. The communication and computational complexities are comparable to the recent MIM-secure challenge-response scheme by Dodis *et al* [17] based on Gap-CDH, assuming equal group size. The key size is also comparable. Of course, we stress that a fair comparison should take into account the hardness of Gap-CDH and qSDH, but very little is known about them being easier than standard CDH (qSDH was studied in [14]). In any case, we find our approach a promising alternative, which may pave the way to further instantiations based on q-strong type assumptions.

References

[1] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *CRYPTO*, pages 595–618, 2009.

[2] Gildas Avoine. Adversarial Model for Radio Frequency Identification. *IACR Cryptology ePrint Archive*, 2005:49, 2005.

[3] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. A Framework for Analyzing RFID Distance Bounding Protocols. *Journal of Computer Security*, 19(2):289–317, April 2011.

[4] Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing Time Complexity in RFID Systems. In *Selected Areas in Cryptography*, pages 291–306, 2005.

[5] Mihir Bellare, Oded Goldreich, and Anton Mityagin. The Power of Verification Queries in Message Authentication and Authenticated Encryption. Cryptology ePrint Archive, Report 2004/309, 2004.

[6] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *EUROCRYPT*, pages 139–155, 2000.

[7] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *CRYPTO*, pages 232–249, 1993.

[8] Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *STOC*, pages 57–66, 1995.

[9] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT*, pages 409–426, 2006.

[10] Daniel J. Bernstein and Tanja Lange. Never Trust a Bunny. Cryptology ePrint Archive, Report 2012/355, 2012.

[11] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *J. Cryptology*, 21(2):149–177, 2008.

[12] Stefan Brands and David Chaum. Distance-Bounding Protocols (Extended Abstract). In *EUROCRYPT93, Lecture Notes in Computer Science 765*, pages 344–359. Springer-Verlag, 1993.

[13] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In *SecPerU*, pages 28–33, 2006.

[14] Jung Hee Cheon. Security Analysis of the Strong Diffie-Hellman Problem. In *EUROCRYPT*, pages 1–11, 2006.

[15] Ivan Damgård and Michael Østergaard Pedersen. RFID Security: Tradeoffs between Security and Efficiency. In *CT-RSA*, pages 318–332, 2008.

[16] Tassos Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In *SecureComm*, pages 59–66, 2005.

[17] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message Authentication, Revisited. In *EUROCRYPT*, 2012.

[18] Danny Dolev and Andrew Chi-Chih Yao. On the Security of Public Key Protocols (Extended Abstract). In *FOCS*, pages 350–357, 1981.

[19] Dang N. Duc and Kwangjo Kim. Securing HB^+ Against GRS Man-in-the-Middle Attack. In *SCIS*, 2007.

[20] Ulrich Duerholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A Formal Approach to Distance-Bounding RFID Protocols. Cryptology ePrint Archive, Report 2011/321, 2011.

[21] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-Knowledge Proofs of Identity. *J. Cryptology*, 1(2):77–94, 1988.

[22] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. 263:186–194, 1986.

[23] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. Good Variants of HB^+ Are Hard to Find. In *Financial Cryptography*, pages 156–170, 2008.

[24] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. $HB^\#$: Increasing the Security and Efficiency of HB^+. In *EUROCRYPT*, pages 361–378, 2008.

[25] Henri Gilbert, Matthew J. B. Robshaw, and Hervé Sibert. An Active Attack Against HB+ - A Provably Secure Lightweight Authentication Protocol. *IACR Cryptology ePrint Archive*, 2005:237, 2005.

[26] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[27] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, 1988.

[28] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An Efficient Authentication Protocol Based on Ring-LPN. In *FSE*, 2012.

[29] Nicholas J. Hopper and Manuel Blum. Secure Human Identification Protocols. In *ASIACRYPT*, pages 52–66, 2001.

[30] Ari Juels and Stephen A. Weis. Authenticating Pervasive Devices with Human Protocols. In *CRYPTO*, pages 293–308, 2005.

[31] Ari Juels and Stephen A. Weis. Defining Strong Privacy for RFID. *IACR Cryptology ePrint Archive*, 2006:137, 2006.

[32] Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and Concurrent Security of the HB and HB^+ Protocols. *J. Cryptology*, 23(3):402–421, 2010.

[33] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient Authentication from Hard Learning Problems. In *EUROCRYPT*, pages 7–26, 2011.

[34] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT*, pages 1–23, 2010.

[35] Jorge Munilla and Alberto Peinado. HB-MP: A Further Step in the HB-Family of Lightweight Authentication Protocols. *Computer Networks*, 51(9):2262–2267, 2007.

[36] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *CRYPTO*, pages 31–53, 1992.

[37] Khaled Ouafi, Raphael Overbeck, and Serge Vaudenay. On the Security of HB# Against a Man-in-the-Middle Attack. In *ASIACRYPT*, pages 108–124, 2008.

[38] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[39] Jacques Stern. A New Paradigm for Public Key Identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.

[40] Serge Vaudenay. On Privacy Models for RFID. In *ASIACRYPT*, pages 68–87, 2007.

A An RLWE-based MAC

For an integer n[17] and a prime p let $\mathsf{R}_p = \mathbb{Z}_p[x]/\langle x^n + 1\rangle$ be the quotient ring of polynomials modulo the ideal $\langle p, x^n + 1\rangle$. Let also χ be a distribution with support $[\chi] = \mathsf{R}_p$. $\mathrm{RLWE}_{n,p,\chi}$ is the problem of distinguishing between several samples of the form $(\mathsf{a}, \mathsf{a} \cdot \mathsf{s} + \mathsf{e})$ (where s is a secret element from R_p, $\mathsf{a} \xleftarrow{\$} \mathsf{R}_p$ and $\mathsf{e} \leftarrow \chi$) and random samples $(\mathsf{a}, \mathsf{u}) \xleftarrow{\$} \mathsf{R}_p \times \mathsf{R}_p$. Figure 12 provides a formal definition of RLWE.

Game $\mathrm{RLWE}_{n,p,\chi}$	**oracle Sample**():
procedure main:	$\mathsf{a} \xleftarrow{\$} \mathsf{R}_p$
$\mathsf{s} \xleftarrow{\$} \mathsf{R}_p$	$\mathsf{e} \leftarrow \chi$
$d \leftarrow \mathcal{A}^{\mathbf{Sample}}$	Ret $(\mathsf{a}, \mathsf{a} \cdot \mathsf{s} + \mathsf{e})$

Figure 12: Pseudocode description for game $\mathrm{RLWE}_{n,p,\chi}$. $\mathcal{A}$ has access to **Sample** and at the end of the game outputs a value $d \in \{0, 1\}$.

Similar to LPN, the advantage function for $\mathrm{RLWE}_{n,p,\chi}$ is defined as

$$\mathbf{Adv}^{\mathsf{rlwe}}_{n,p,\chi}(t, q) = \max_{\mathcal{A}} \left\{ \Pr\left[\, \mathrm{RLWE}^{\mathcal{A}}_{n,p,\chi} \Rightarrow 1 \,\right] - \Pr\left[\, \mathrm{RLWE}^{\mathcal{A}}_{n,p,\mathcal{U}(\mathsf{R}_p)} \Rightarrow 1 \,\right] \right\} \tag{36}$$

where $\mathcal{U}(\mathsf{R}_p)$ is the uniform distribution over R_p and the maximum is over all adversaries $\mathcal{A}$ receiving q samples and running in time t.

We describe a $\mathsf{uf\text{-}rmrc}$-secure MAC based on the hardness of RLWE. We first present the construction for arbitrary distribution χ over R_p. For a polynomial a, we use $\|\mathsf{a}\|_2$ to denote the l_2 (Euclidean) norm of a under the standard coefficient embedding. $\mathsf{MAC}_{\mathrm{RLWE}} = (\mathsf{KGen}, \mathsf{TAG}, \mathsf{VRFY})$ has keyspace, message space and tag space $\mathcal{K} = \mathcal{M} = \mathcal{T} = \mathsf{R}_p$ and is defined as follows:

Parameters: $n = n(\kappa)$, prime p, distribution χ with $[\chi] = \mathsf{R}_p$, $X \in \mathbb{R}^+$ such that $\Pr_{\mathsf{e} \leftarrow \chi}[\, \|\mathsf{e}\|_2 > X \,]$ is small.

$\mathsf{KGen}(1^\kappa)$: Pick $\mathsf{s} \xleftarrow{\$} \mathsf{R}_p$.

$\mathsf{TAG}(\mathsf{s}, \mathsf{a})$: Sample $\mathsf{e} \leftarrow \chi$; return $\mathsf{y} = \mathsf{a} \cdot \mathsf{s} + \mathsf{e}$.

$\mathsf{VRFY}(\mathsf{s}, \mathsf{a}, \mathsf{t})$: Return 1 iff $\|\mathsf{t} - \mathsf{a} \cdot \mathsf{s}\|_2 < X$.

[17] For security, n is typically chosen to be a power of 2.

Lemma A.1 states the security of $\mathsf{MAC}_{\mathrm{RLWE}}$ for a specific distribution χ and bound X. We omit the proof since it is essentially identical to the proof of Lemma 4.3.

Lemma A.1. **[Security of $\mathsf{MAC}_{\mathrm{RLWE}}$]** *For prime p, let $\tilde{p} = \lfloor \sqrt{p}/2 \rfloor$, $\chi = \mathcal{U}(\mathbb{Z}_{\tilde{p}}^n)$ and $X = \sqrt{n}\tilde{p}/2$. Consider also the $\mathsf{MAC}_{\mathrm{RLWE}}$ as defined above. Then for all positive integers $t, q_{\mathsf{T}}, q_{\mathsf{C}}$ and q_{V}*

$$\mathbf{Adv}^{\mathsf{uf\text{-}rmrc}}_{\mathsf{MAC}_{\mathrm{RLWE}}}(t, q_{\mathsf{T}}, q_{\mathsf{C}}, q_{\mathsf{V}}) \leq \mathbf{Adv}^{\mathsf{rlwe}}_{n,p,\chi}(t', q) + q_{\mathsf{V}} \cdot 2^{-0.2n}$$

where $t' = t + \mathcal{O}(q_{\mathsf{C}} + q_{\mathsf{V}})$ and $q = q_{\mathsf{T}} + q_{\mathsf{C}}$. Also, $\mathsf{MAC}_{\mathrm{RLWE}}$ has completeness 1, i.e. $\epsilon_c = 0$.

B Proof of Lemma 4.8 (Security of $\mathsf{MAC}_{\mathrm{qSDH}}$)

Let $\mathcal{A}$ be an adversary against the $\mathsf{suf\text{-}rmcc}$-security of $\mathsf{MAC}_{\mathrm{qSDH}}$ that runs in t steps making q_{T} queries to **Tag**, q_{RT} queries to **ReTag** and a single verification query.[18] We will show that there exists an adversary $\mathcal{B}$ that uses $\mathcal{A}$ and solves qSDH (with $q = q_{\mathsf{T}}$) with advantage not much smaller than the advantage of $\mathcal{A}$.

$\mathcal{B}$ maintains a list M (initialized to $\emptyset$) for bookeeping and upon receiving as input $(g, h_1 = g^x, h_2 = g^{x^2}, \ldots, h_q = g^{x^q})$, uses $\mathcal{A}$ as follows: first it picks $m_1, m_2, \ldots, m_q$ uniformly at random from $\mathbb{Z}_p$ and computes the polynomial $\mathsf{a}(X) = \Pi_{i=1}^q (X + m_i)$. Let $\mathsf{a}(X) = a_0 + \ldots + a_{q-1}X^{q-1} + X^q$ be the power expansion of a. $\mathcal{B}$ also picks $z \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}$. On the i-th **Tag** query by $\mathcal{A}$ ($i \in \{1, \ldots, q\}$), $\mathcal{B}$ computes the values

$$\mathsf{a}^{(i)}(X) = \frac{\mathsf{a}(X)}{X + m_i} = \sum_{j=0}^{q-1} a_j^{(i)} \cdot X^j \quad \text{and} \quad \tau_i \leftarrow \prod_{j=0}^{q-1} h_j^{a_j^{(i)} \cdot z},$$

sends (m_1, τ_i) to $\mathcal{A}$ and also adds (m_i, τ_i) to M. Whenever $\mathcal{A}$ makes a query m to **ReTag**, $\mathcal{B}$ first checks whether there exists an entry (m, τ) in M (for any τ) and if not returns $\perp$. Otherwise, it returns the τ that appears in (m, τ) to $\mathcal{A}$ (notice that $\mathsf{MAC}_{\mathrm{qSDH}}$ is deterministic and hence if such an entry exists, then it is unique). Let (m^*, τ^*) by the (single) **Vrfy** query (forgery attempt) by $\mathcal{A}$. Since $\mathsf{TAG}_{\mathrm{qSDH}}$ is deterministic, we may assume without loss of generality that $m^* \notin \{m_1, \ldots, m_q\}$. On input such a pair, $\mathcal{B}$ first computes (using Eucledean division) $\mathsf{u}(X) = u_0 + u_1 \cdot X + \ldots + u_{q-1} \cdot X^{q-1}$ and v_0 such that $\mathsf{a}(X) = \mathsf{u}(X)(X + m^*) + v_0$. It finally returns

$$(m^*, h^*) \quad \text{where} \quad h^* = \left((\tau^*)^{1/z} \prod_{i=0}^{q-1} (h_i)^{-u_i} \right)^{1/v_0}$$

as the candidate solution to the qSDH instance.[19]

First notice that the output to the **Tag** queries have the correct distribution where $K = x$ is randomly distributed in $\mathbb{Z}_p$ and $\tilde{g} = g^{z \cdot \mathsf{a}(x)}$ is a random generator of $\mathbb{G}$. Indeed

$$\tau_i = \prod_{j=0}^{q-1} h_j^{a_j^{(i)} \cdot z} = \prod_{j=0}^{q-1} (g^{x^j})^{a_j^{(i)} \cdot z} = \left(g^{z \cdot \mathsf{a}(x)} \right)^{\frac{1}{x + m_i}} = \tilde{g}^{\frac{1}{x + m_i}} .$$

Also, assume that (m^*, τ^*) is a valid forgery, i.e. $\tau^* = \tilde{g}^{\frac{1}{x + m^*}}$. Then

$$h^* = \left((\tau^*)^{1/z} \prod_{i=0}^{q-1} (h_i)^{-u_i} \right)^{1/v_0} = \left(\left(\tilde{g}^{\frac{1}{x + m^*}} \right)^{1/z} g^{-\mathsf{u}(x)} \right)^{1/v_0} = \left(g^{\frac{\mathsf{a}(x)}{x + m^*}} \cdot g^{-\mathsf{u}(x)} \right)^{1/v_0} = g^{\frac{1}{x + m^*}}$$

[18]Without loss of generality we assume that $\mathcal{A}$ makes exactly q_{T} queries to **Tag**.

[19]Notice that h^* is well defined since $z \neq 0$ and $v_0 \neq 0$ (because $m^* \notin \{m_1, \ldots, m_q\}$) and hence $(X + m^*) \not| \mathsf{a}(X)$.

which implies that $\mathcal{B}$ solves its qSDH instance whenever $\mathcal{A}$ returns a valid forgery. Therefore

$$\mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}_{\mathrm{qSDH}}}(t, q_{\mathsf{T}}, q_{\mathsf{RT}}, q_{\mathsf{V}}) \leq q_{\mathsf{V}} \cdot \mathbf{Adv}^{\mathsf{suf\text{-}rmcc}}_{\mathsf{MAC}_{\mathrm{qSDH}}}(t, q_{\mathsf{T}}, q_{\mathsf{RT}}, 1) \leq q_{\mathsf{V}} \cdot \mathbf{Adv}^{\mathsf{q\text{-}sdh}}_{\mathbb{G}, q_{\mathsf{T}}}(t')$$

where for the first inequality we used Lemma 4.5
Finally $\mathcal{B}$ needs $\mathcal{O}(q_{\mathsf{T}}^2)$ time to compute the polynomial expansions of $\mathsf{a}, \mathsf{a}^{(1)}, \ldots, \mathsf{a}^{(q)}$ and $q_{\mathsf{T}} \cdot t_{\mathsf{exp}}$ steps per TAG query to compute τ_i where t_{exp} is the cost of a single exponentiation in $\mathbb{G}$. Therefore $\mathcal{B}$ runs in $t' = t + \mathcal{O}(q_{\mathsf{T}}^2 \cdot t_{\mathsf{exp}})$ steps.